COMP9120

Week 2: Conceptual Database Design

Semester 1, 2025

Remember to form group and put that information in Canvas

68% have already formed a group

Professor Athman Bouguettaya
School of Computer Science

THE UNIVERSITY OF SYDNEY

Contents developed by Dr Lijun Chang

Warming up!

Let’s menti!

Acknowledgement of Country

I would like to acknowledge the Traditional Owners of Australia and recognise their continuing connection to land, water and culture. I am currently on the land of the Gadigal people of the Eora nation and pay my respects to their Elders, past, present and emerging.

Copyright warning

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

BREAKING NEWS!!

LIVE HELP ON ED: *STARTING NEXT WEEK!*

To access the **weekly schedule**, please go to Canvas:

https://canvas.sydney.edu.au/courses/63042/modules

under Modules, under NEW! LIVE ED SCHEDULE S1 2025

How: The indicated tutor (e.g., Dipankar) will be on Ed during the hours mentioned in the schedule to answer your questions **live**!

Week	Weekday	FROM	TO	Name
3	TUE	6:00 pm	8:00 pm	Dipankar
3	THUR	11:00 am	1:00 pm	Dipankar
4	TUE	6:00 pm	8:00 pm	Dipankar
4	THUR	11:00 am	1:00 pm	Dipankar
5	TUE	6:00 pm	8:00 pm	Dipankar
5	WED	6:00 pm	8:00 pm	Dipankar
5	THUR	11:00 am	1:00 pm	Dipankar
6	TUE	6:00 pm	8:00 pm	Dipankar
6	WED	6:00 pm	8:00 pm	Dipankar
6	THUR	11:00 am	1:00 pm	Dipankar
6	WED	6:00 pm	8:00 pm	Abbey
6	THUR	11:00 am	12:00 pm	Abbey
7	TUE	6:00 pm	8:00 pm	Dipankar
7	THUR	11:00 am	1:00 pm	Dipankar
8	TUE	6:00 pm	8:00 pm	Dipankar
8	THUR	11:00 am	1:00 pm	Dipankar
9	TUE	6:00 pm	8:00 pm	Dipankar
9	WED	6:00 pm	8:00 pm	Dipankar
9	THUR	11:00 am	1:00 pm	Dipankar
10	TUE	6:00 pm	8:00 pm	Dipankar
10	WED	6:00 pm	8:00 pm	Dipankar
10	THUR	11:00 am	1:00 pm	Dipankar
10	WED	6:00 pm	8:00 pm	Abbey
11	THUR	11:00 am	12:00 pm	Abbey
11	TUE	6:00 pm	8:00 pm	Dipankar
11	WED	6:00 pm	8:00 pm	Dipankar
11	THUR	11:00 am	1:00 pm	Dipankar
11	WED	6:00 pm	8:00 pm	Abbey
11	THUR	11:00 am	12:00 pm	Abbey
12	TUE	6:00 pm	8:00 pm	Dipankar
12	THUR	11:00 am	1:00 pm	Dipankar
13	TUE	6:00 pm	8:00 pm	Dipankar
13	WED	6:00 pm	8:00 pm	Dipankar
13	THUR	11:00 am	1:00 pm	Dipankar

THE UNIVERSITY OF SYDNEY

Outline

- Introduction to Conceptual Database Design

- Entity Relationship Model
  - Notation and usage
  - Entity and Relationship types, attributes
  - Key, participation and cardinality constraints
  - Weak entities, IsA hierarchies, aggregation

Conceptual Database Design

- **Purpose** of conceptual database design
  - **Agree** on the **structure of the database** before deciding on a particular implementation: Let us look at Tree swing example to see the issues that may arise:

How the customer explained it

https://medium.com/@thx2001r/the-project-cartoon-root-cause-5e82e404ec8a

Database Design Sequence

1. Requirements Analysis

- Understand…
  - what data needs to be stored
  - what applications must be built
  - what operations are most frequent

2. Conceptual Design

- Develop…
  - high-level description of the data closely matching how users think of the data
  - Works as communication vehicle

Today

3. Logical Design

- Convert…
  - conceptual design into a logical database schema

4. Schema Refinement

- Refine…
  - Identify problems in current schema & refine

5. Physical Design

- Convert…
  - logical schema into a physical schema for a specific DBMS and tuned for app.

6. App & Security Design

- Determine who plays what roles, in what workflows, with security settings …
  - What roles are played by different system entities in system processes, and what permissions should be given to these roles?

Conceptual Database Design

- What is our goal?
  - Specification of the database schema
- **Conceptual Database Design**: A framework for understanding and capturing business information requirements *graphically*
- It does *not* include how data is implemented, created, modified, used, or deleted.
  - Works as communication vehicle *between* technical people *and* non-technical people
  - Facilitates planning, operation & maintenance of various data resources
- Usually, this is the role of the *System Analyst*

Entity Relationship Model

- First designed by **Peter Chen** in 1976
  - Several variations/versions have since appeared
  - The *conceptual database model* we will use in this class**: Enhanced** or **Extended E-R model (EER)**

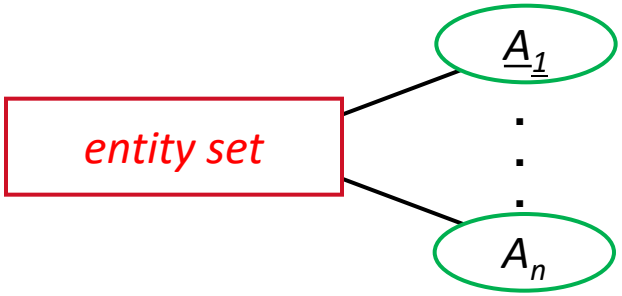
- Definition: A (conceptual) *data modelling* approach is a *visual representation* that depicts the *associations* among different *categories of data* within a Universe of Discourse – UoD (e.g., enterprise).
  - What are the ***entities*** and ***relationships*** in the UoD (e.g., enterprise)?
  - What information do we need to store in the database about these entities and relationships?
  - What are the *business rules* (represented by *integrity constraints)* that should always hold (i.e., be *true*)?

- A database 'schema' in the ER Model is represented pictorially (***ER Diagrams – ERD*** as a shorthand).
  - We can always convert an ER diagram (ERD) into a logical (e.g., relational) schema.

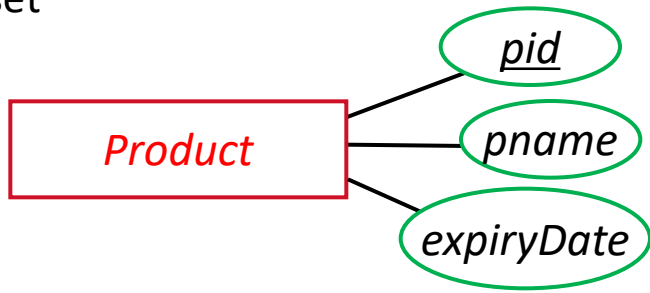
Entities & Entity set

- **Entity**: represents an *individual* object from the UoD (such as a University): a specific person, place, object, event, etc.
  - it must be *distinguishable* from *other entities*
  - Example: *John Doe*, unit *COMP9120*, bank account *4711*

- **Entity Type** (also called **entity set**): a *collection* of entities that share *common properties* or characteristics
  - Example: *students*, *courses*, *bank accounts*
  - In E-R, a *Rectangle* represents an entity set



- **Attribute**: describes one *aspect* (i.e., a description) of an entity set
  - Descriptive *properties* possessed *by all* members of an entity set
  - Example: product have *pid*, *name* and *expiryDate*
  - Depicted by an *ellipse*



Entity Schema

- ***Domain***: all possible *values* of an attribute
  - Values may be *complex / set-oriented which is,* as we will see, contrary to the relational model.
    - **Simple** and **composite** attributes. E.g. (color vs. date)
    - **Single-valued** and **multi-valued** attributes (studentID vs spokenLanguages)

- ***Key***: *minimal* set of attributes, called a *candidate key*, that *uniquely* identifies an entity in the set (may have more than one candidate key)
  - Example: each student is *uniquely* identified by the student *ID*.
  - The *selected key* among *candidate keys* is called **Primary Key** (PK) => depicted by underlining the attributes forming the key. **Only** *one primary key* at all times!

- ***Entity Schema***: *entity set name*, *attributes*, their *domains*, and *PK*

Identify Keys

- Which one would you choose as a primary key?
  - Staff(staff_id, name, email, phone_number, address, dateOfBirth)

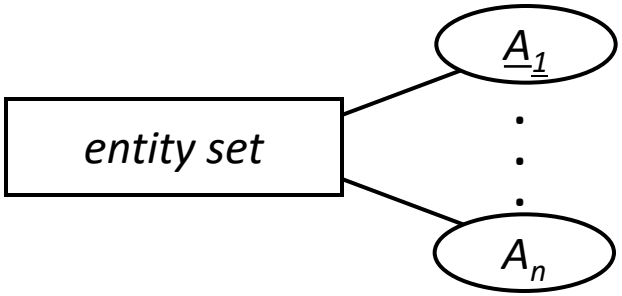
- Which one would you choose as a primary key?
  - House(house_number, street_name, city, state, build_in_year, selling_price)

Graphical Representation in E-R Diagram

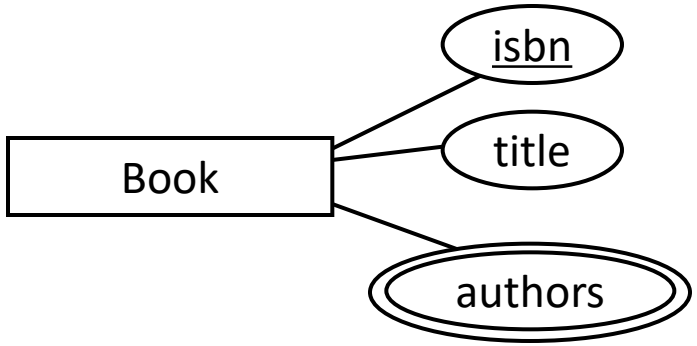
Symbols:

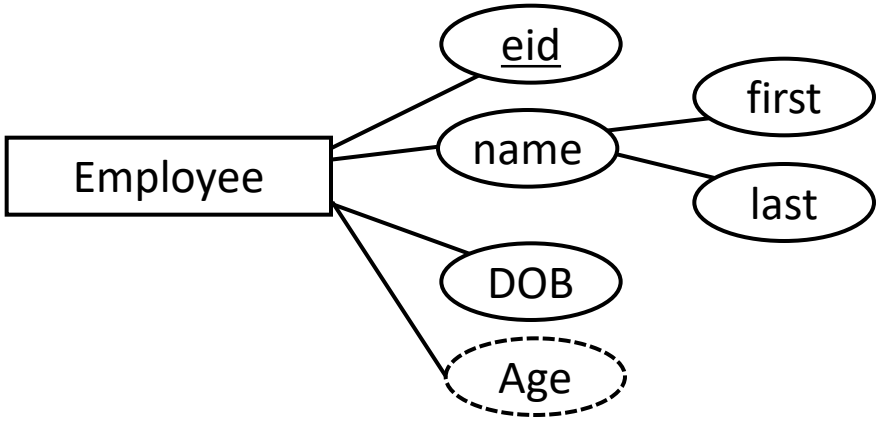
- Entity Sets represented by a rectangle

- Attributes depicted by ellipses
  - Keys are underlined
  - Double ellipses for multi-valued attributes



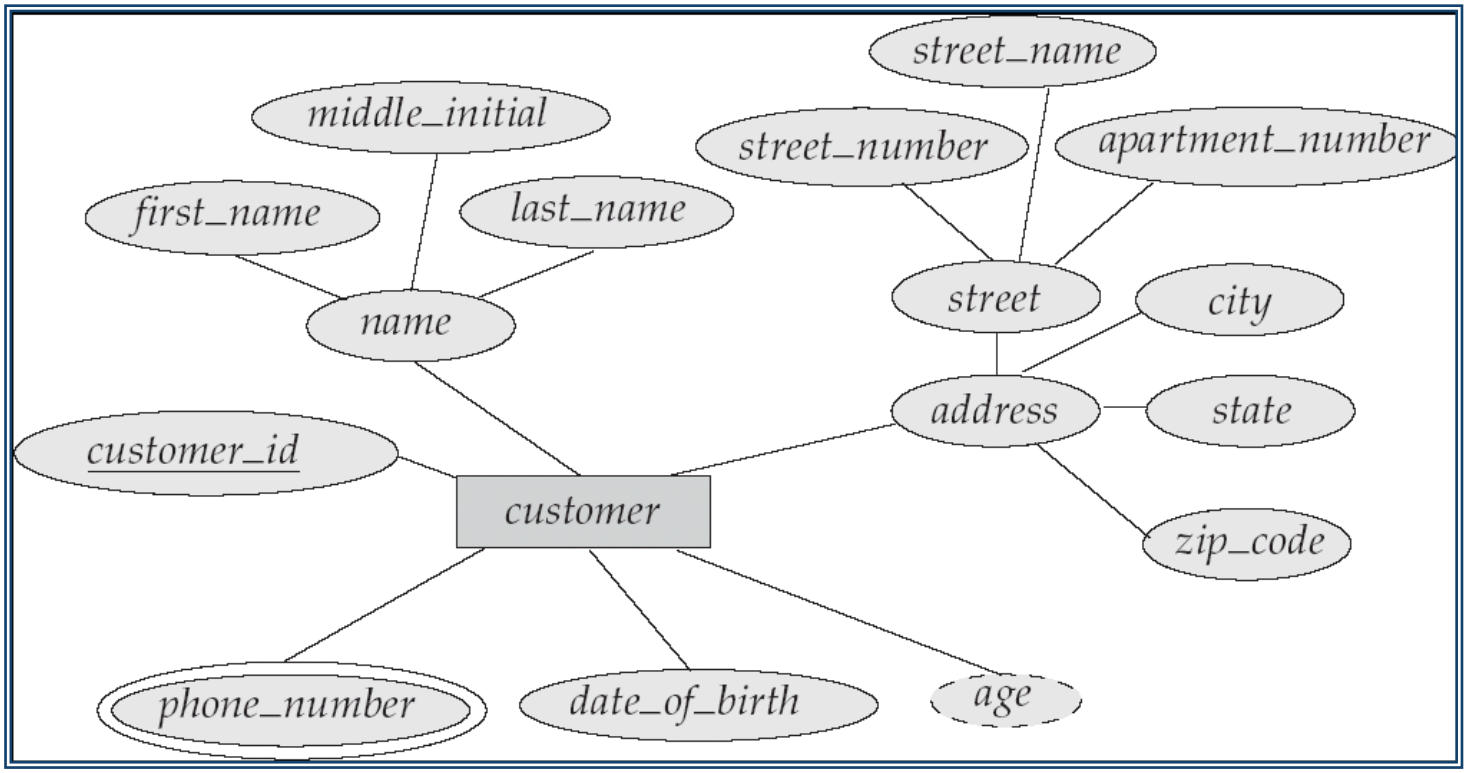
Examples:





Note:
Book.authors is a *multi-valued attribute*;
Employee.name is a *composite attribute*.
Employee.Age is a *derived attribute*

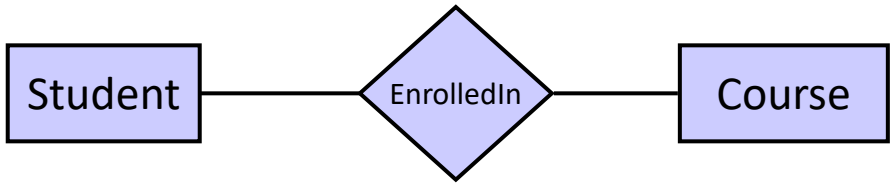
Entity Set


street_name
middle_initial
street_number
apartment_number
first_name
last_name
street
city
name
address
state
customer_id
customer
zip_code
phone_number
date_of_birth
age

Relationships

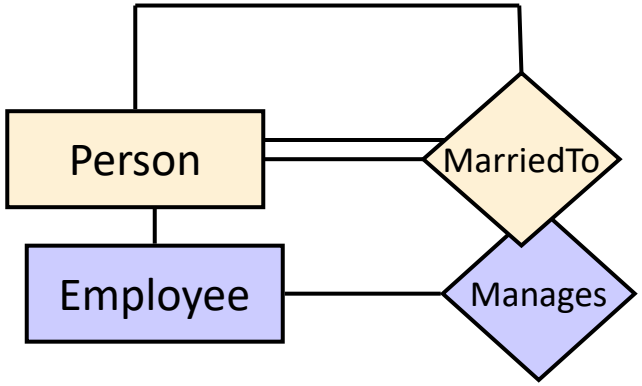
- **Relationship**: relates *two* or *more* entities
  - Example: John *is enrolled in* COMP9120

- **Relationship Type** (also called **Relationship Set**): set of similar relationships
  - Formally: a relation among $n \geq 2$ entities, each entity from an entity set is defined as:
    $\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$
  - Example: **Student** (entity set) is related to **Course** (entity set) by **EnrolledIn** (relationship type).
  - *Diamond* shape represents the *relationship type*

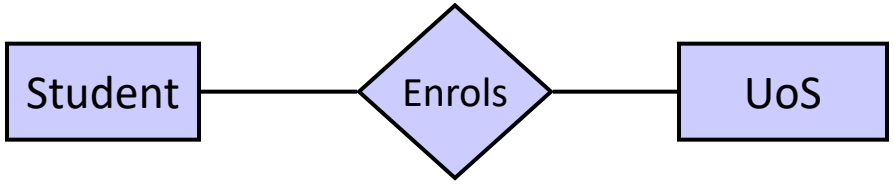


Relationship Degree

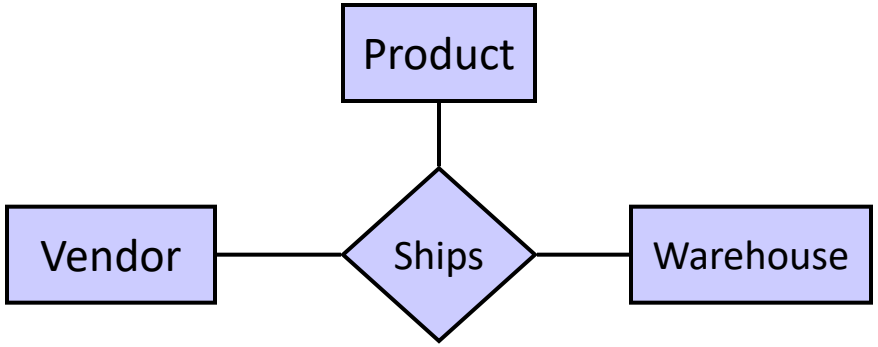
- *Degree* of a Relationship: *number* of entity types involved



- Unary Relationship (Recursive)



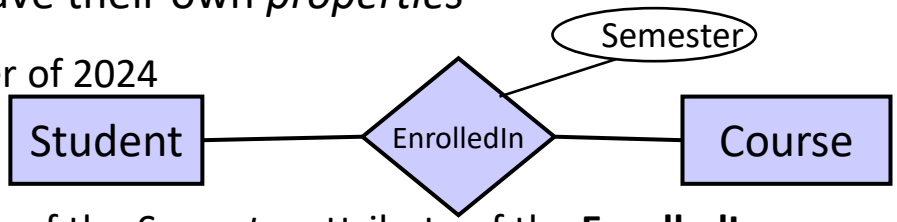
- Binary Relationship



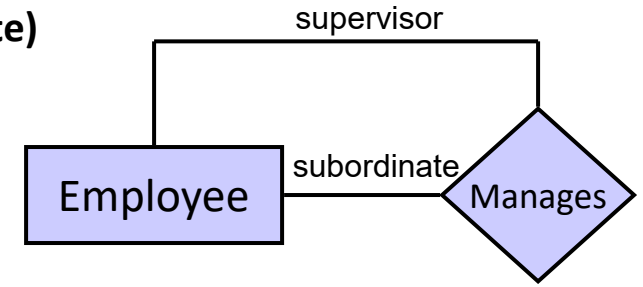
- Ternary Relationship

Relationship Attributes and Roles

- **Relationship Attributes:** *Relationships* can also have their own *properties*
  - E.g., John enrols in COMP9120 *in* the second semester of 2024
  - John and COMP9120 are related *for a limited time*
  - 2024sem2 describes this *temporal* relationship - value of the *Semester* attribute of the **EnrolledIn** relationship set



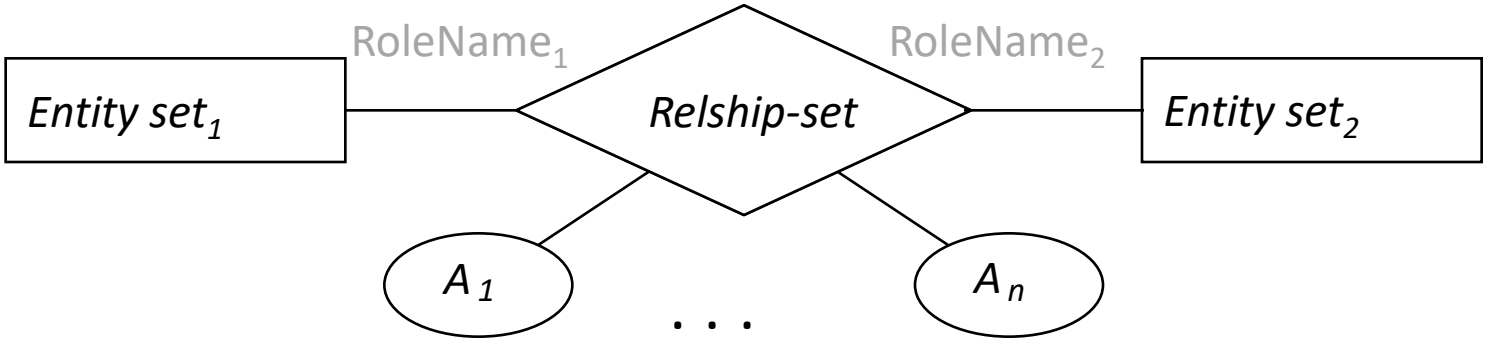
- **Relationship Role:** Each participating entity can be named with an explicit *role*
  - The "supervisor" and "subordinate" labels are called *roles*. They indicate how Employee entities interact via the *Manages* relationship
  - Useful for relationships that relate elements of the same entity type
  - Example: **Manages(Employee: supervisor, Employee: subordinate)**



- **Relationship Type Schema**:
  - Relationship name
  - Role names (or names of participating entity sets) – this is optional
  - Relationship attributes and their domains

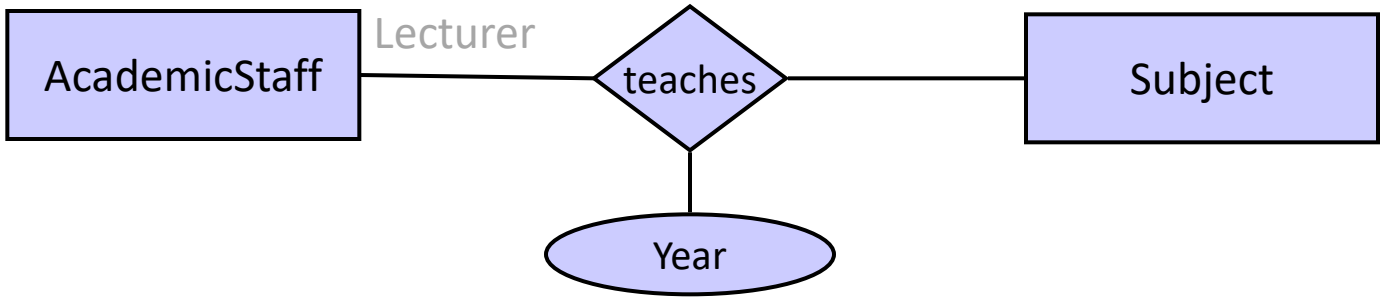
Graphical Representation of Relationships in ERD

Symbol:



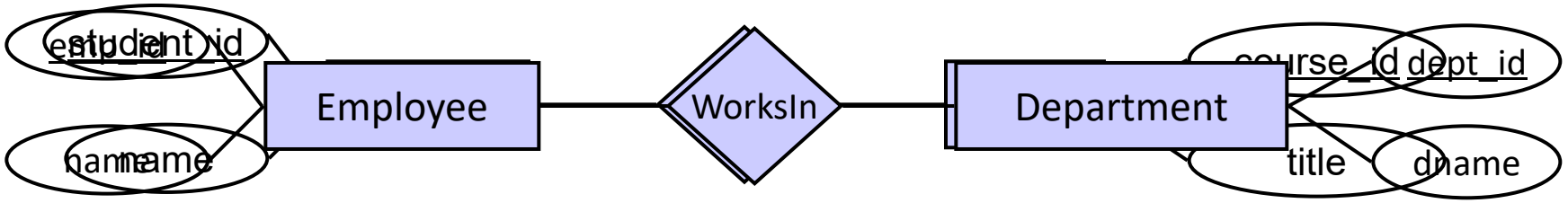
- *Diamonds* represent relationship set
- *Lines* link attributes to entity/relationship sets and entity sets to relationship set.
- *Roles* are labels over edges (lines), labelled with role names

Example



Key of Relationship Set

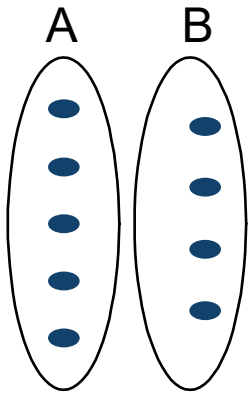
- The combination of the primary keys of the participating entity sets forms a **superkey** (superset of a key) of an (entity) relationship.
  - Example: (student_id, course_id) is the superkey of *Enrols*

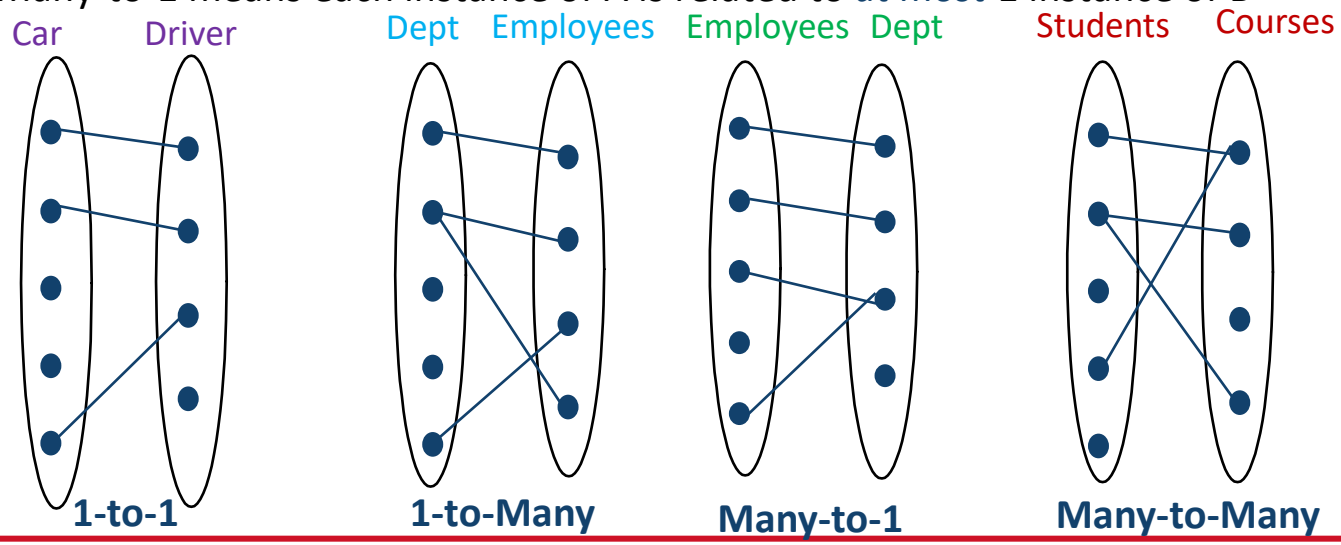


  - We must consider the ***mapping cardinality of the relationship*** when deciding what the ***candidate key*** *(minimal set* of attributes*)* is:
    - Consider **WorksIn**: *An employee* can work in *many* departments; *a department* can have *many* employees.
    - In contrast, *each department* has at *most one* manager
  - **We do <u>not</u> represent the *keys* of the *relationship set* in E-R diagram**

Mapping Cardinality of Relationships

- We present examples of each style of relationship.
  - Given a relationship between two entity sets **A** and **B**, we need to consider *both directions* of the relationship:
  - How many instances of **B** can a given instance of **A** be related to?
  - How many instances of **A** can a given instance of B be related to?
- Answer: ***1 to 1, 1 to N, N to 1, N to M*** are the possible **cardinalities** of a relationship between entities
- Beware: the natural language formulation may confuse you!
  - Many-to-1 means each instance of A is related to *at most* 1 instance of B

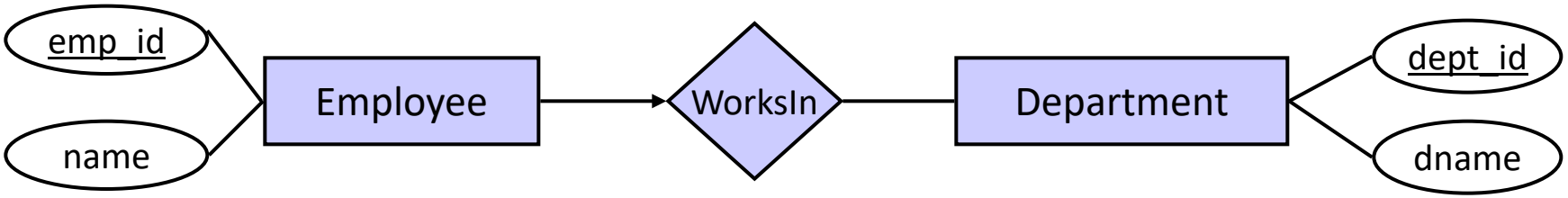




Cardinalities are depicted in E-R diagrams (ERD) as *constraints.*

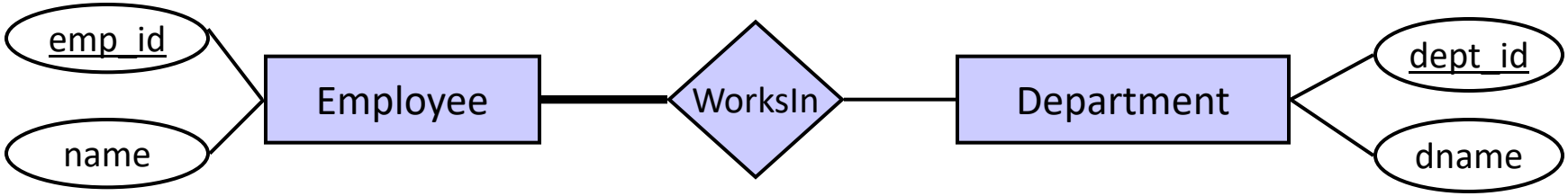
Key Constraints

- If for a particular entity set, each entity participates in *at most one* instance of a relationship, the *key* of the entity set is the (candidate) key of the relationship type
  - E.g., *Employee* key (emp_id) is also *unique* in *WorksIn*
    - called: **many-to-one, also** denoted **N:1 relationship**
- Representation in E-R diagram: ***arrow***
- Example: *An employee* works in at *most one* department.



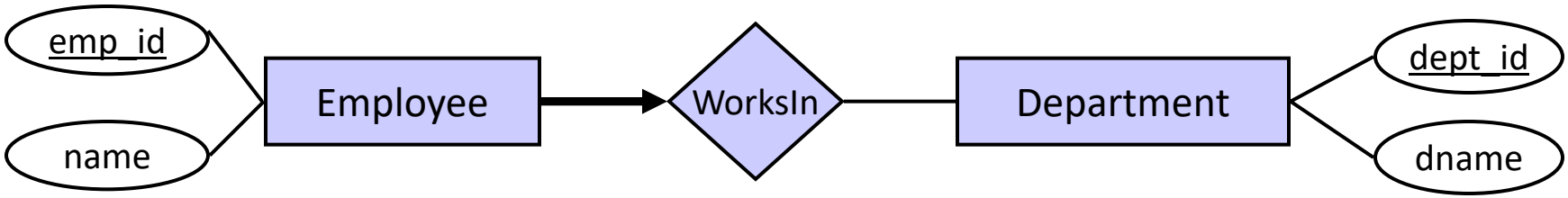
Participation Constraint

- If *every* entity of an entity type participates in *at least one* instance of a relationship, a ***participation*** *constraint* holds, i.e., it is true:
  - also called a **total participation** of entity *E* in *R*
    - A participation that is *not total* is said to be **partial**
- Representation in E-R diagram: ***thick line***
- Example: *Every* employee works in at *least* one department



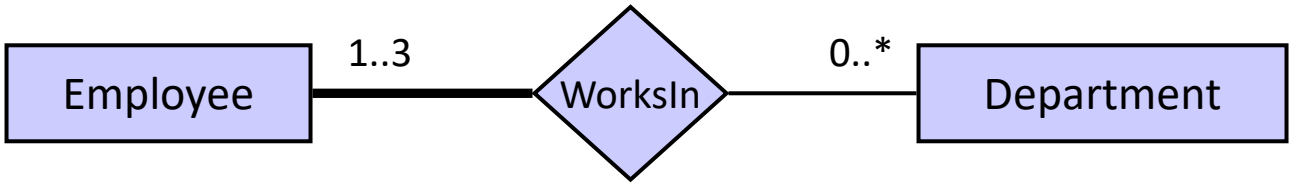
Participation and Key Constraint

- If every entity participates in *exactly one* relationship, then both a **participation** and a **key constraint** hold.
- Representation in E-R diagrams: ***thick arrow***
- Example: Every employee works in exactly one department
  - Again: N:1 relationship



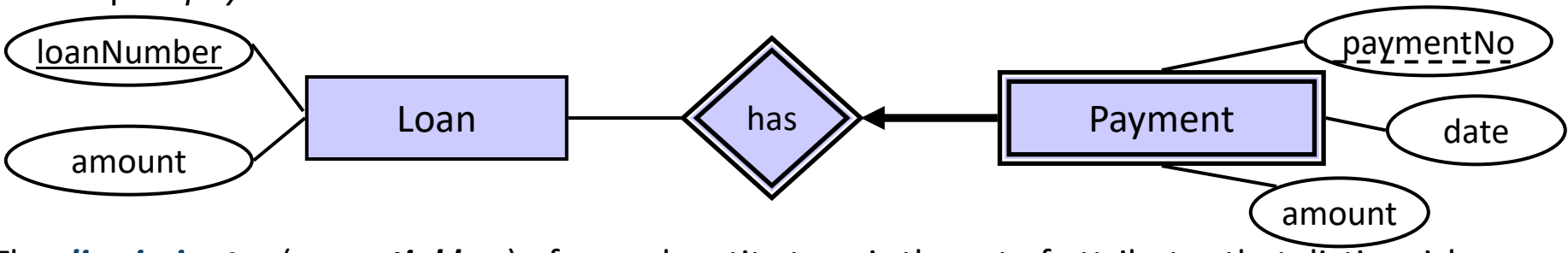
Cardinality Constraints

- ***Generalisation*** of ***key*** and ***participation*** constraints
- A **cardinality constraint** for the participation of an entity set E in a relationship R specifies **how often** an entity of set E participates in R: **at least** (*minimum* cardinality) and **at most** (*maximum* cardinality).
  - In an ER-diagram, we annotate the edge between an entity set E and relationship R with min..max, where *min* is the minimum cardinality and *max* the maximum cardinality. If no maximal cardinality is specified, we set '*' as *max* number ("don't care" or no upper limit).
- Example: Every employee works in *at least* 1 department and *at most* 3 departments.



Weak Entities

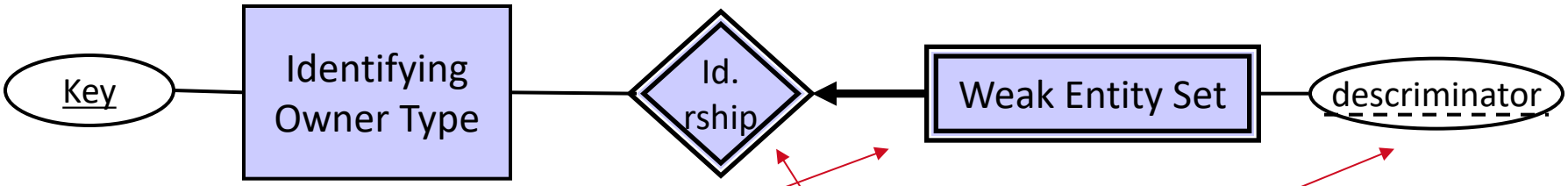
- ***Weak entity type***: An entity type that does ***not*** have a self-contained *primary key*.
  - Its existence **depends** on the existence of an ***identifying owner*** entity
  - The weak entity ***must***:
    - relate to the identifying owner entity set via a **one-to-many** *identifying relationship type* **from** the *identifying owner* entity set to the *weak* entity set
    - have **total** participation in the identifying relationship type
  - Can be seen as an **exclusive 'part-of' relationship**
  - Example: *payment* of a loan



- The ***discriminator*** (or ***partial key***) of a weak entity type is the set of attributes that distinguishes them among all the entities of a weak entity type related to the *same owning entity*.
- The *primary key* of a *weak entity type* is formed by the ***primary key*** of the strong entity type(s) on which the weak entity type is existence dependent, ***plus*** the weak entity type's ***discriminator***.

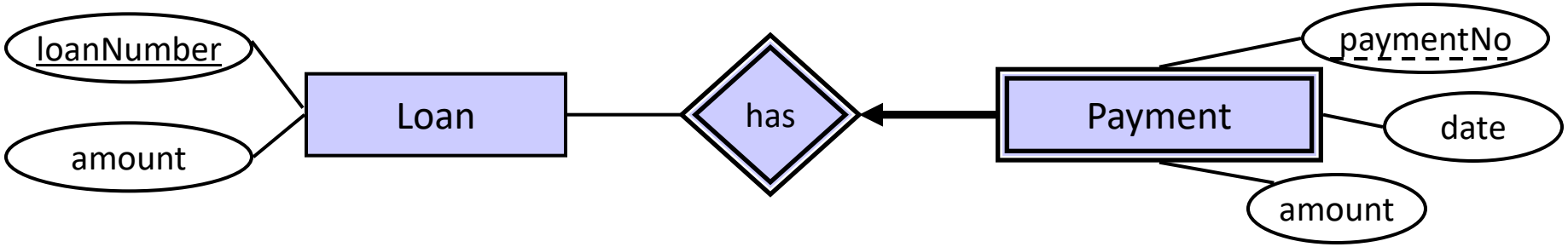
Representation of Weak Entity Types

Symbols:



- We depict a weak entity set by double rectangles.
- Identifying relationship depicted using a double diamond
- underline the discriminator of a weak entity set with a dashed line

Example:



- paymentNumber: discriminator of the Payment entity set
- Primary key for Payment: (loanNumber, paymentNumber)

Let’s take a break!

It’s time to play our menti game!

Enhanced E-R Model

- ER model in its original form did not support
  - SPECIALIZATION/ GENERALIZATION
  - ABSTRACTIONS ('aggregation')

- This led to the development of an 'Enhanced' ER model:
  - Includes all modelling concepts of basic ER
  - Some additional object-oriented concepts: subclasses/superclasses, specialization/generalization, categories, attribute inheritance
  - The resulting model is called the Enhanced-ER or Extended ER (E2R or EER) model
    - used to model applications more completely and accurately

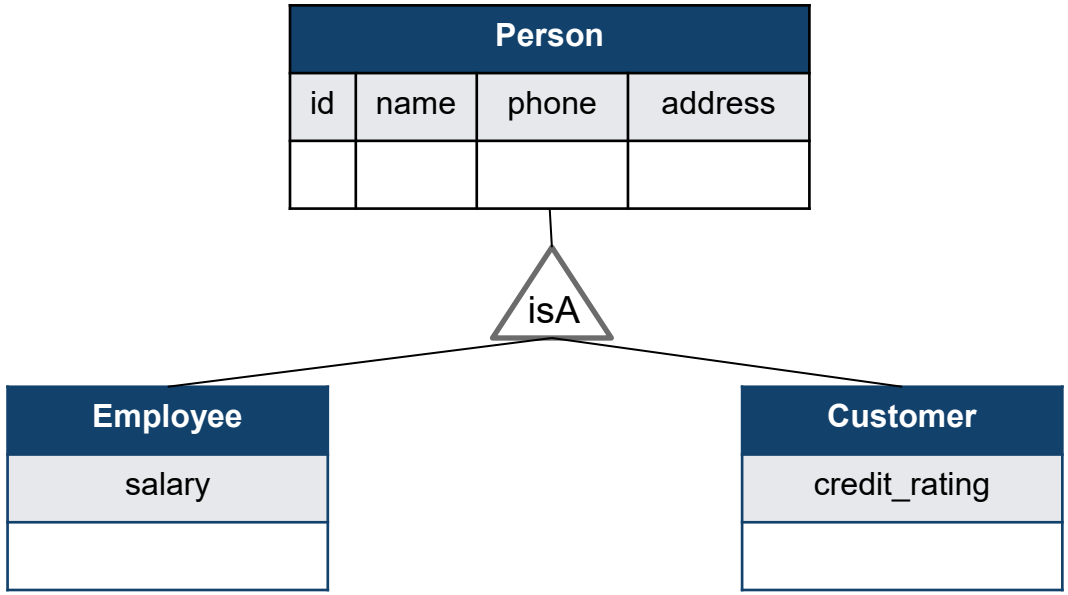
- When we talk about E-R model, we always mean EER model

Generalisation/ Specialisation

- Arranging entity sets in a type hierarchy.
  - Determine entity sets whose set of *properties* are a **subset** of another entity set

Employee				
id	name	phone	address	salary

Customer				
id	name	phone	address	credit_rating



Generalisation/ Specialisation

- Arranging entity sets in a type hierarchy.
  - Determine entity sets whose set of properties are a subset of another entity set

- Definition of **Generalisation** / **Specialisation** / **Inheritance**:
  Two entity types *E* and *F* are in an **IsA-relationship** ("*F* is a *E*"), if

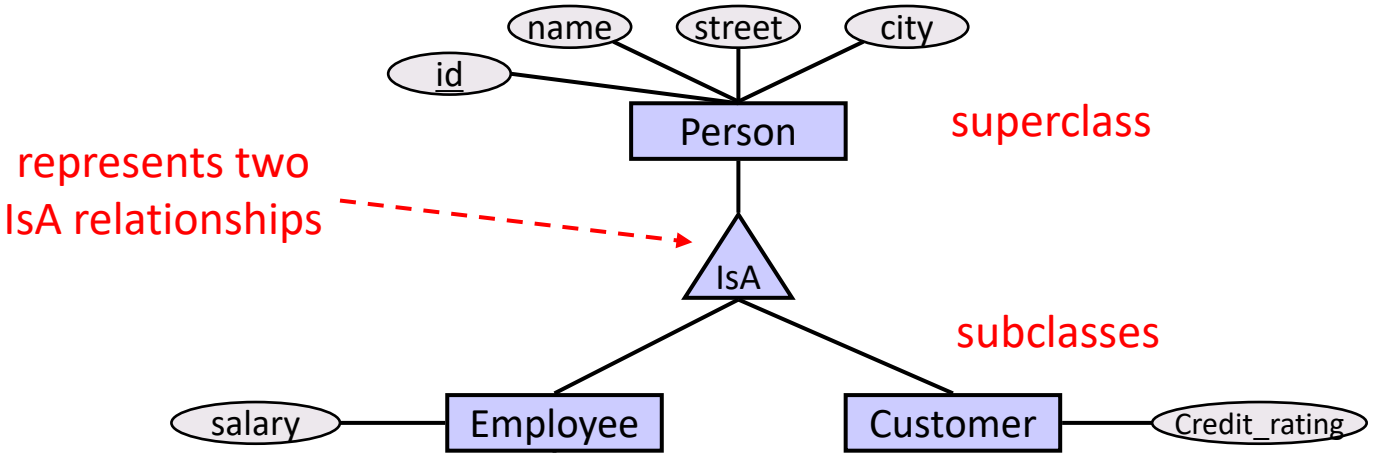
  (1) the set of attributes of *F* is a *superset* of the set of attributes of *E*, and

  (2) the entity set *F* is a *subset* of the entity set of *E* ("each *f* is an *e*")

- We say that *F* is a *specialisation* of *E* (*F* is **subclass**) and *E* is a *generalisation* of *F* (*E* is **superclass**).
  - Example: **Student** is a subclass of **Person**
- **Attribute inheritance** – a lower-level (subclass) entity type *inherits* all the attributes and relationship participations of its supertype.

- Depicted by a *triangle* component labelled *IsA*

Superclass / Subclass Example



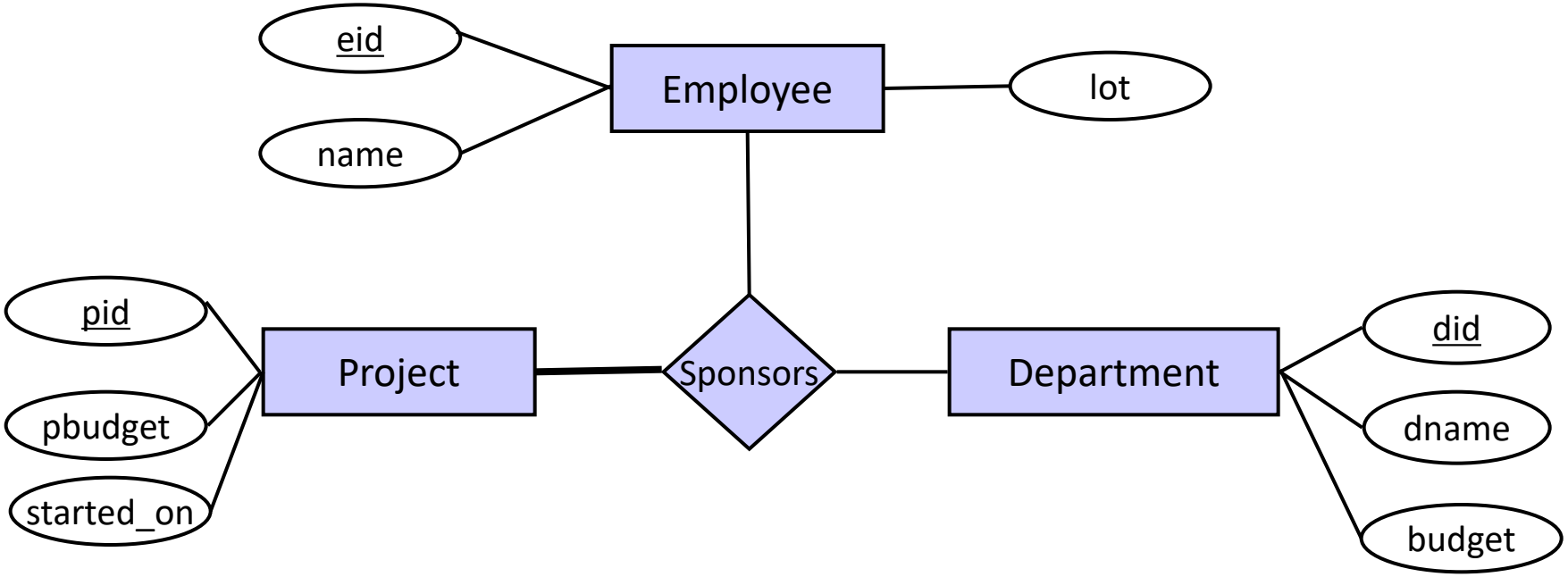
- Specialisation stems from a single entity set. It emphasizes differences among entities within the set by creating distinct lower-level entities
- Generalization proceeds from the recognition that a number of entity sets share some common features

Constraints on IsA Hierarchies

- We can specify *overlap* and *covering* constraints for ISA hierarchies:
- ***Overlap Constraints***
  - **Disjoint**
    - an entity can belong to *only one* lower-level entity set
    - Noted in E-R diagram by writing *disjoint* next to the IsA triangle
  - **Overlapping** (the default - *opposite to Ramakrishnan/Gehrke book*)
    - an entity can belong to *more than one* lower-level entity set
- ***Covering Constraints***
  - **Total**
    - an entity must belong to one of the lower-level entity sets
    - Denoted with a thick line between the IsA-triangle and the superclass
  - **Partial** (the default)
    - an entity need not belong to one of the lower-level entity sets

Aggregation

› Consider a *ternary* relationship *Sponsors*

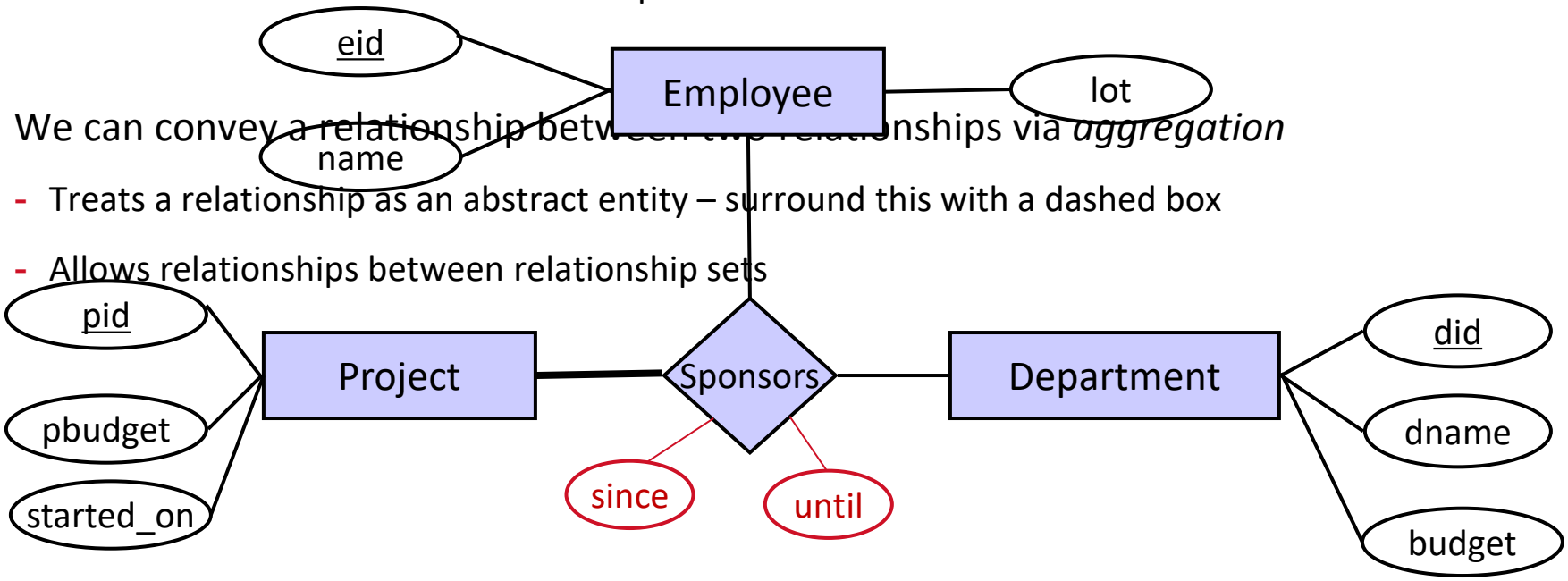


- Each project can be sponsored by one or more departments
- A department can appoint one or more employees to monitor a sponsorship
- What are the issues with this representation?

(Example adapted from Ramakrishnan & Gherke, DBMS, 3rd edition)

Aggregation (Cont'd)

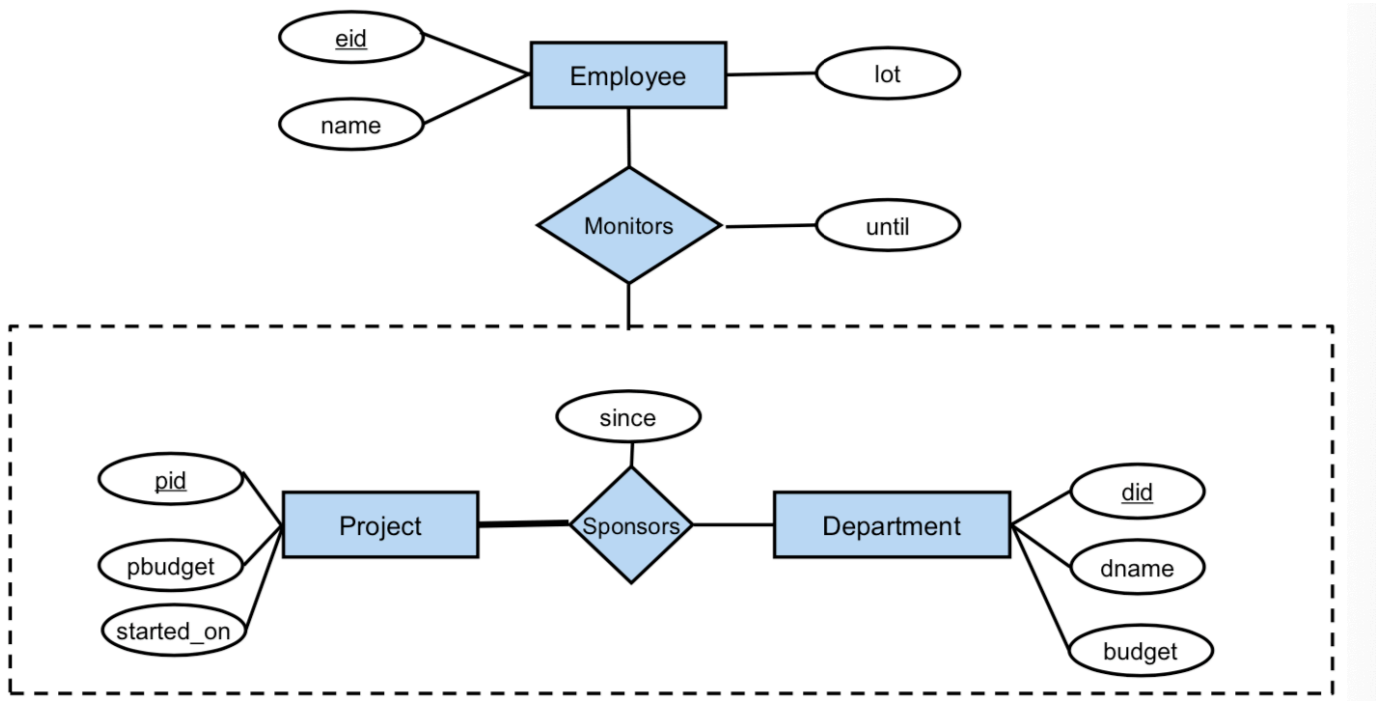
- Relationship set *Sponsors is* really trying to model *two relationships*
  - It is trying to model the fact that departments *sponsor* projects, but also these sponsorships are *monitored.*
  - What if we want to add different attributes for each of these relationships, e.g., attribute "*since*" to the sponsorship and until attribute "*until*" to the monitoring?
    - It would be more meaningful. However, we need to ensure our model adds such required attributes on the correct relationship.



- We can convey a relationship between two relationships via *aggregation*
  - Treats a relationship as an abstract entity – surround this with a dashed box
  - Allows relationships between relationship sets

E-R Diagram With Aggregation

- **Aggregation:** It conveys a **relationship between two relationships**: in our example, it conveys a relationship between the *sponsors* relationship type and the Employee entity type: we introduce aggregation via a *Monitors* relationship type.
  - Allows us to model that *sponsorships* start at a given time and that employees are assigned to *monitor* a *sponsorship* until a given time.



(Example from Ramakrishnan & Gherke, DBMS, 3rd edition)

This Week We Have Learned…

- **Conceptual database design using the E-R Model**
  - Understanding and experience with conceptual database design using the entity-relationship model:
    - Basic Constructs: Entity, Attributes (single, composite, multivalued, derived), Relationships, Cardinality Constraints
    - Advanced Concepts: Weak Entities, Inheritance, Aggregation

THE UNIVERSITY OF SYDNEY

IMPORTANT

- Many variations of ER diagrams are in use, and there is no widely accepted standard.

- **Which notation should I use for assessments like quiz, final exam etc?**
  - **Must use** the notation just outlined in this lecture

References

- Ramakrishnan/Gehrke (3rd edition)
  - **Chapter 2**

- Kifer/Bernstein/Lewis (2nd edition)
  - Chapter 4
- Ullman/Widom (3rd edition)
  - Chapter 4
- Silberschatz/Korth/Sudarshan (5th edition - ‘sailing boat’)
  - Chapter 6
- Elmasri/Navathe (5th edition)
  - Chapters 3 and 4

Next Week: Relational Model

- Readings:
  - **Ramakrishnan/Gehrke, Chapter 3 plus Chapter 1.5**
  - Kifer/Bernstein/Lewis, Chapter 3
  - Ullman/Widom, Chapter 2.1 - 2.3, Section 7.1 and Chapter 8.1-8.2

See you next week!

Remember to form group and put that information in Canvas

THE UNIVERSITY OF SYDNEY