

# COMP9120

Week 2: Conceptual Modelling

Semester 2, 2016

Based on material by Dr. Bryn Jeffries



THE UNIVERSITY OF  
SYDNEY

- › Assignment 1 was released on Monday
  - Let your tutor know once you have formed a group, and your tutor will assign you a number you can use to sign-up with online.
  
- › First set of homework questions released tomorrow
  - *due in 1 week from tomorrow – unlimited attempts - will include questions from last week + this week*
  
- › “Grok Learning” – the SQL Challenge replacement for practice questions now available

## 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.**

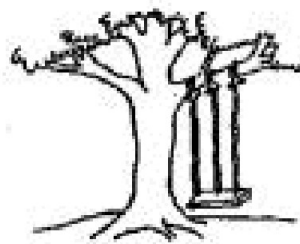
---

- › **Introduction to Conceptual Data Modelling**
- › **Entity Relationship Model**
  - **Notation and usage**
  - **Entity and Relationship types, attributes**
  - **Key, participation and cardinality constraints**
  - **Weak entities, IsA hierarchies, aggregation**

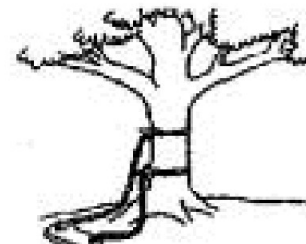
- › What is a conceptual (or semantic) model?
- › What is the purpose of a conceptual model?



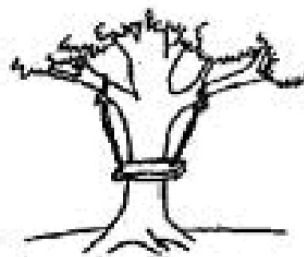
**As proposed  
by the project  
sponsor.**



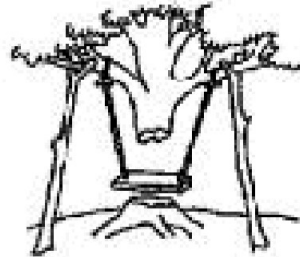
**As specified  
in the project  
request.**



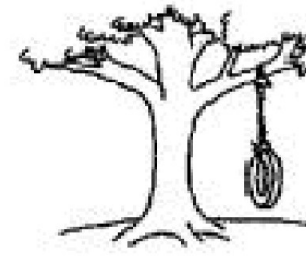
**As designed  
by the senior  
architect.**



**As produced  
by the  
engineers.**



**As installed at  
the user's  
site.**



**What the  
customer  
really wanted.**

## 1. Requirements Analysis

- Understand...
  - ▶ what data is 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

---

## 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

- Identify User who plays what Roles, in what Workflows, & secure ...
  - ▶ What roles are played by different system entities in system processes, and what permissions should be given to these roles?

- › Goal: Specification of database schema
- › Methodology:
  - **Conceptual Design:** A technique for understanding and capturing business information requirements graphically
    - depicts the how we can describe the data associated with a real world in the context of a business problem, in terms of types of objects and their relationships.
    - Convert conceptual data model design to DDL (Logical Design)
- › Conceptual Data Model Design does *not* imply how data is implemented, created, modified, used, or deleted.
  - Works as communication vehicle
  - Facilitate planning, operation & maintenance of various data resources
- › An conceptual data model is *model & database independent*

- › Entity-Relationship Model (ERM)
- › Object-oriented Data Models
  - Unified Modelling Language (UML)
  - OMG, Booch, ...
- › Etc.
  - Object Role Modelling (ORM)
  - Semantic Object Model (SOM)
  - Semantic Data Models (SDM)



- › Data model for conceptual data model design:  
High-level *graphical representation* of what data needs to be contained in the system.
  - Used to interpret, specify, and document database systems.
  - Tools: ERwin, Sybase Power Designer, ...
  - E-R diagram templates for drawing tools, e.g., Microsoft Visio
  
- › First designed by Peter Chen in 1976
  - Several variations have since appeared
  - Here: **enhanced** or **extended E-R model**

## Entity Relationship Model (Cont'd)

- › A data modelling approach that depicts the associations among different categories of data within a business or information system.
  - 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* or *business rules* that hold?
- › A database 'schema' in the ER Model is represented pictorially (*ER diagrams*).
  - We can convert an ER diagram into a logical (e.g., relational) schema.
- › It is about what data needs to be stored
  - It does **not** imply how data is created, modified, used, or deleted.

- › **Entity**: a person, place, object, event, or concept about which you want to gather and store data.
  - it must be distinguishable from other entities
  - Example: John Doe, unit COMP9120, account 4711
  
- › **Entity Type** (also: **entity set**): a collection of entities that share common properties or characteristics
  - Example: students, courses, accounts
  - Rectangle represent entity type
  
- › **Attribute**: describes one aspect of an entity type
  - Example: people have *names* and *addresses*
  - depicted by an ellipses

- › An Entity Type is described by a set of attributes
  - Descriptive properties possessed by all members of an entity type
  - Example: Person has ID, Name, Address, Hobbies
- › **Domain**: possible values of an attribute
  - In contrast to relational model values can be complex / set-oriented!
    - **Simple** and **composite** attributes.
    - **Single-valued** and **multi-valued** attributes
  - Example see next slide
- › **Key**: minimal set of attributes that uniquely identifies an entity in the set (several such candidate keys possible)
  - One chosen as **Primary Key** (PK) => depicted by underlining attr.
- › **Entity Schema**: entity type name, attributes (+domains), PK

# Graphical Representation in E-R Diagram

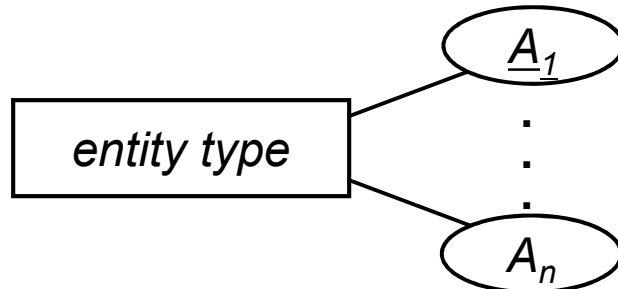
## Symbols:

- ▶ Entity Types represented by a rectangle

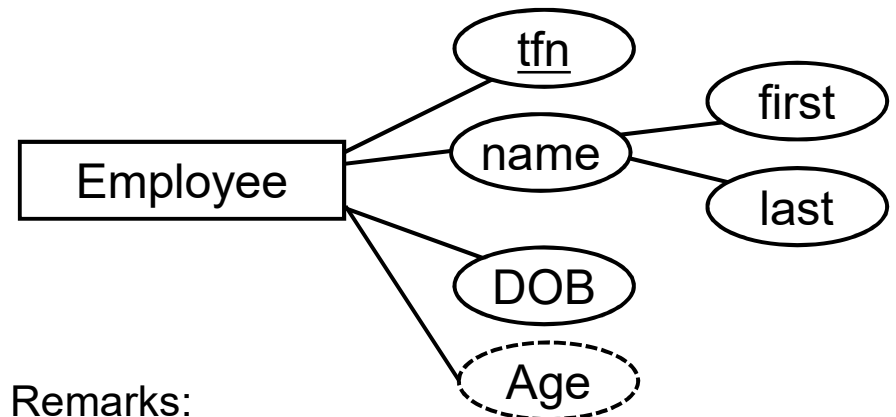
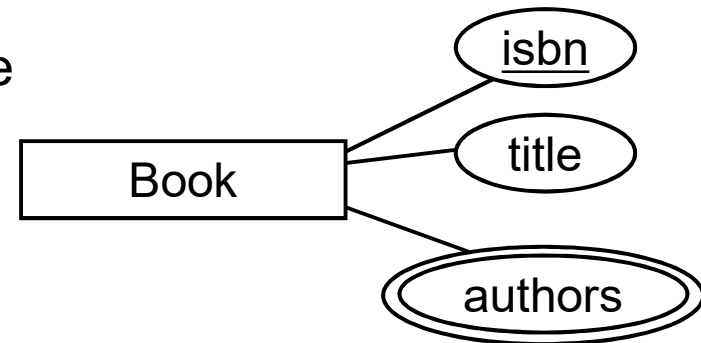


- ▶ Attributes depicted by ellipses

- ▶ Double ellipses for multi-valued attributes
- ▶ Keys are underlined



## Examples:



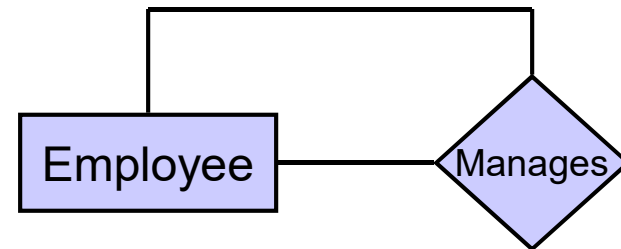
### Remarks:

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

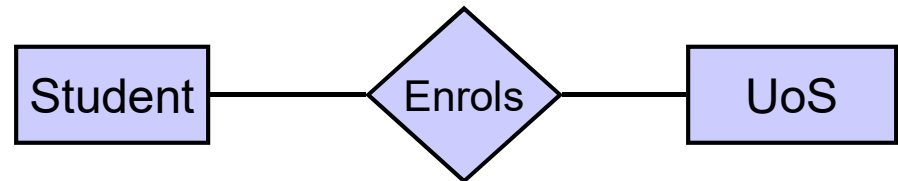
- › **Relationship**: relates two or more entities
  - Example: John *is enrolled in* INFO2120
  
- › **Relationship Type (R.ship Set)**: set of similar relationships
  - Formally: a relation among  $n \geq 2$  entities, each from entity sets:  
 $\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$
  - Example: **student** (entity type) related to **UnitOfStudy** (entity type) by **EnrolledIn** (relationship type).
  
- › Distinction:
  - relation (relational model) - set of tuples
  - relationship (E-R Model) – describes relationship between entities
  - Both entity sets and relationship sets (E-R model) may be represented as relations (in the relational model)

› Degree of a Relationship:  
# of entity types involved

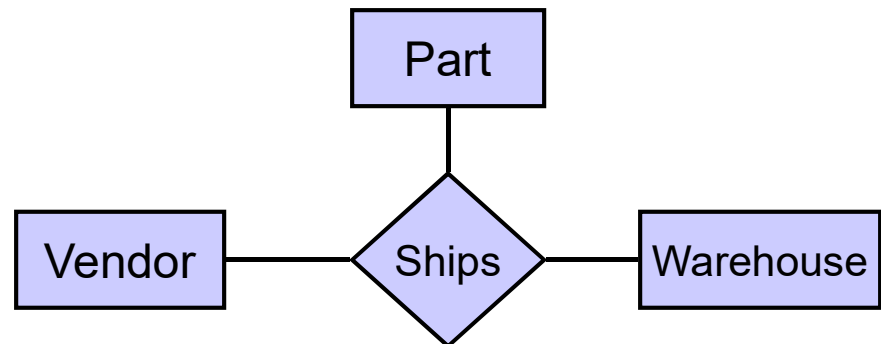
- Unary Relationship (Recursive)



- Binary Relationship (~80%)



- Ternary Relationship



## › Relationship-Attribute

Relationships can also have additional properties

- E.g., John enrolls in INFO2120 *in* the first semester 2010
- John and INFO2120 are related
- 2010sem1 describes the relationship - value of the *Semester* attribute of the **EnrolledIn** relationship set

## › Relationship-Role

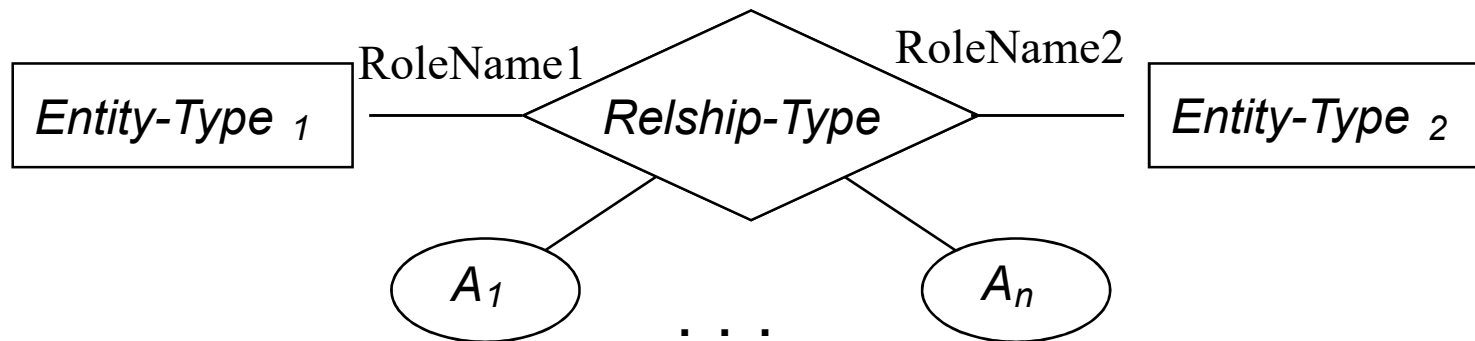
Each participating entity can be named with an explicit role.

- E.g. John is value of *Student* role, INFO2120 value of *Subject* role
- useful for relationship that relate elements of same entity type
- Example: **Supervises( Employee:Manager, Employee )**



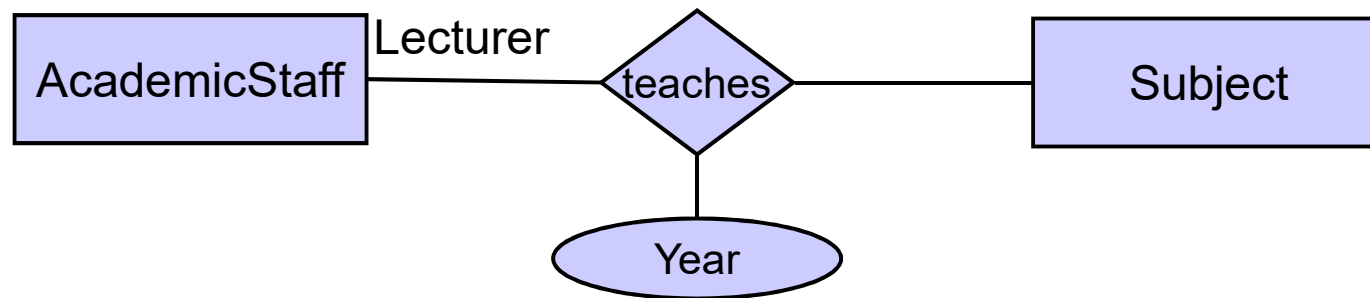
# Graphical Representation of Relationships in E-R Diagrams

Symbol:



- ▶ Diamonds represent relationship types
- ▶ Lines link attributes to entity types and entity types to relationship types.
- ▶ Roles are edges labeled with role names

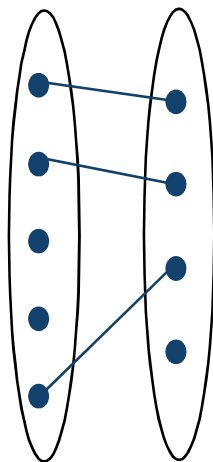
Example



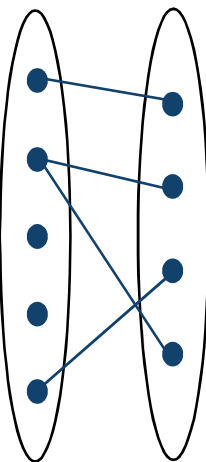
- › The combination of the primary keys of the participating entity types forms a super key of a relationship.
  - Example:  
(student\_Id, UoS\_number) is the super key of *Enrols*
  - One must consider the mapping cardinality of the relationship when deciding what are the candidate keys
  
- › Relationship Set Schema:
  - Relationship name
  - Role names (or: names of participating entity sets)
  - Relationship attributes and their types
  - key

# Multiplicity of Relationships

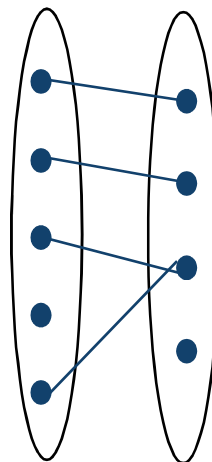
- › Consider **Works\_In**: An employee can work in many departments; a department can have many employees.
- › In contrast, each department has at most one manager
- › We find examples of each style of relationship
  - Think carefully about both directions:
  - To how many instances of B can a given instance of A be related?
  - To how many instances of A can a given instance of B be related?
- › Warning: natural language can be confusing
  - Many-to-1 means each A is related to *at most* 1 of B



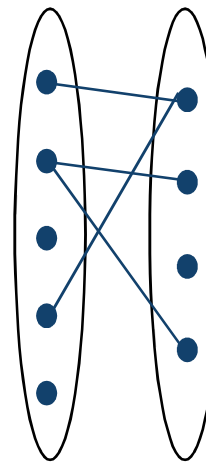
1-to-1



1-to Many



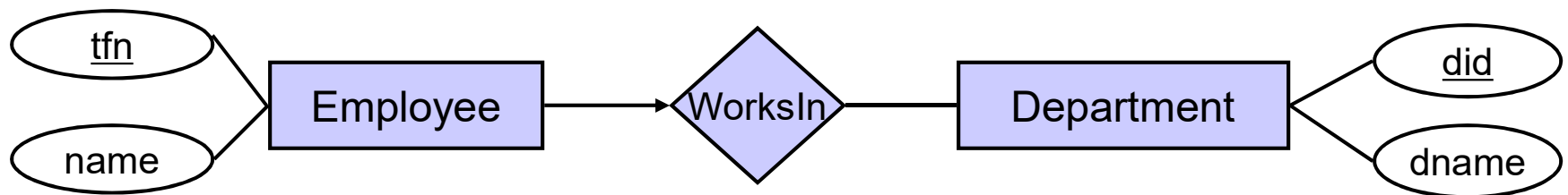
Many-to-1



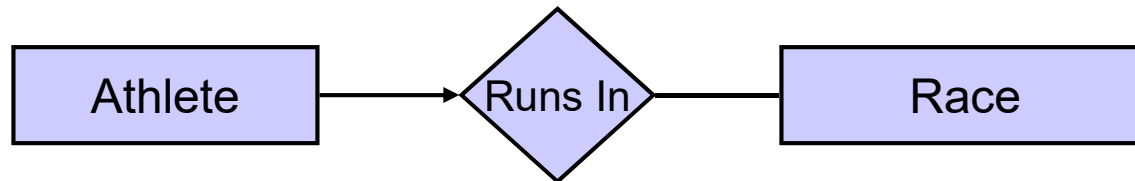
Many-to-Many

Multiplicities  
are depicted in  
E-R diagrams as  
constraints...  
(see next slides)

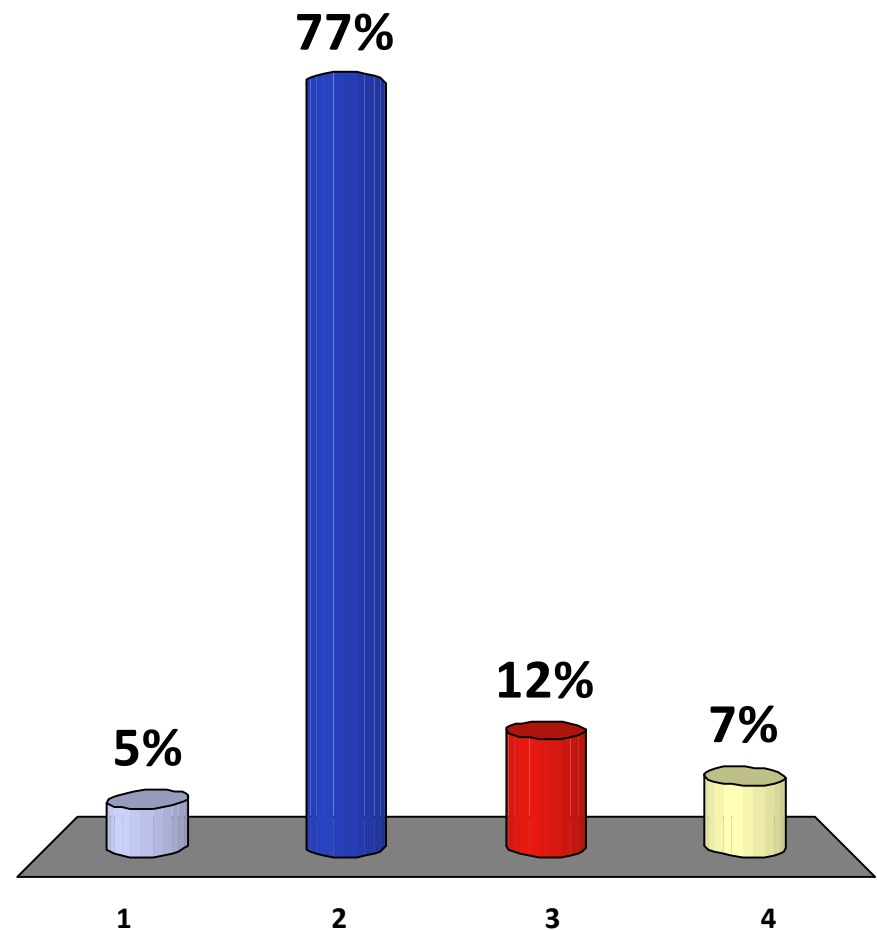
- › If, for a particular participant entity type, each entity participates in at most one instance of a relationship, the corresponding role is a key of relationship type
  - E.g., *Employee* role is unique in *WorksIn*
    - but there may be employees who are working in no department at all
    - also called: **many-to-one** or **N:1 relationship**
- › Representation in E-R diagram: arrow
- › Example: An employee works in at most one department.



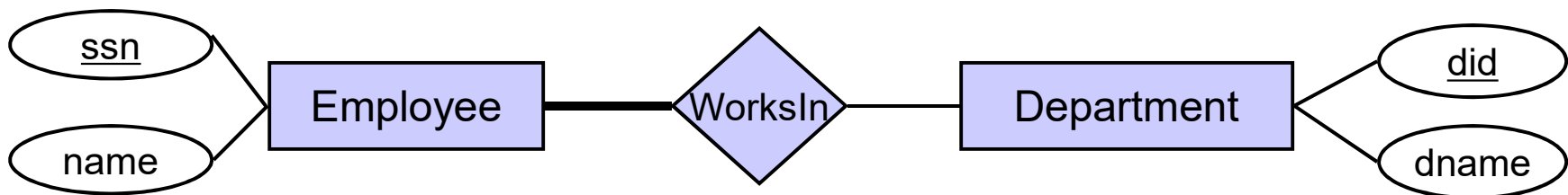
## Which is the correct interpretation?



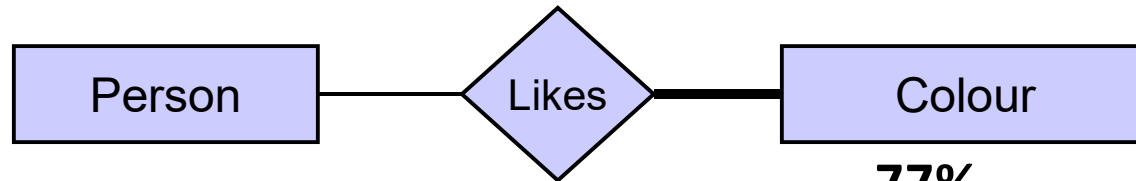
1. Athletes can run in races
2. Each athlete can run in at most one race
3. Every athlete runs in at least one race
4. Each race must have at least one athlete



- › If every entity participates in at least one instance of a relationship, a *participation constraint* holds:
  - also called a **total participation** of  $E$  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

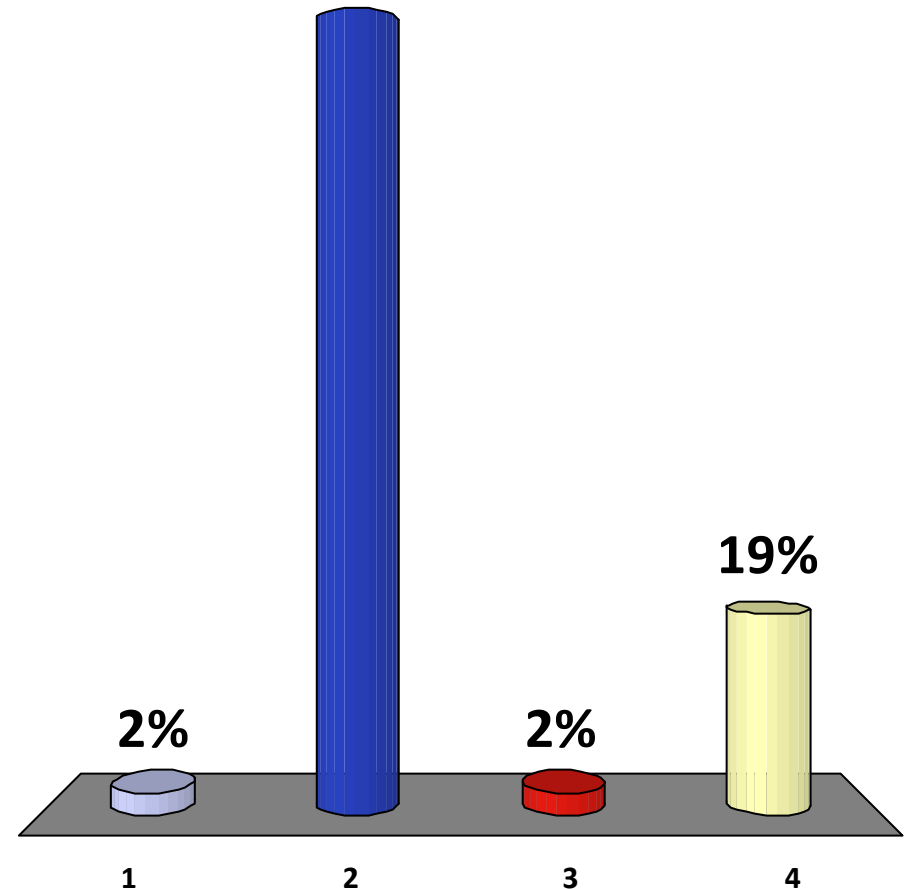


## Which is the correct interpretation?



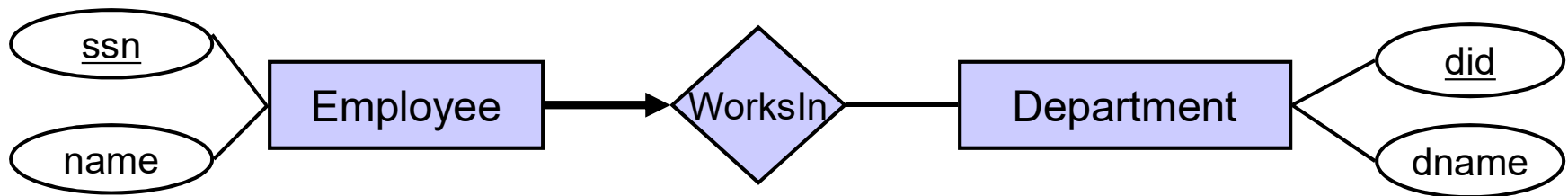
77%

1. Every person must like exactly one colour
2. Each colour must be liked by at least one person
3. Each person must like a different colour
4. Each person must like at least one colour



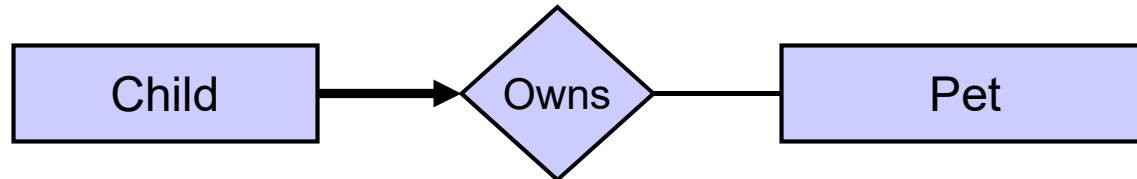
## Participation and Key Constraint

- › If every entity participates in exactly one relationship, 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

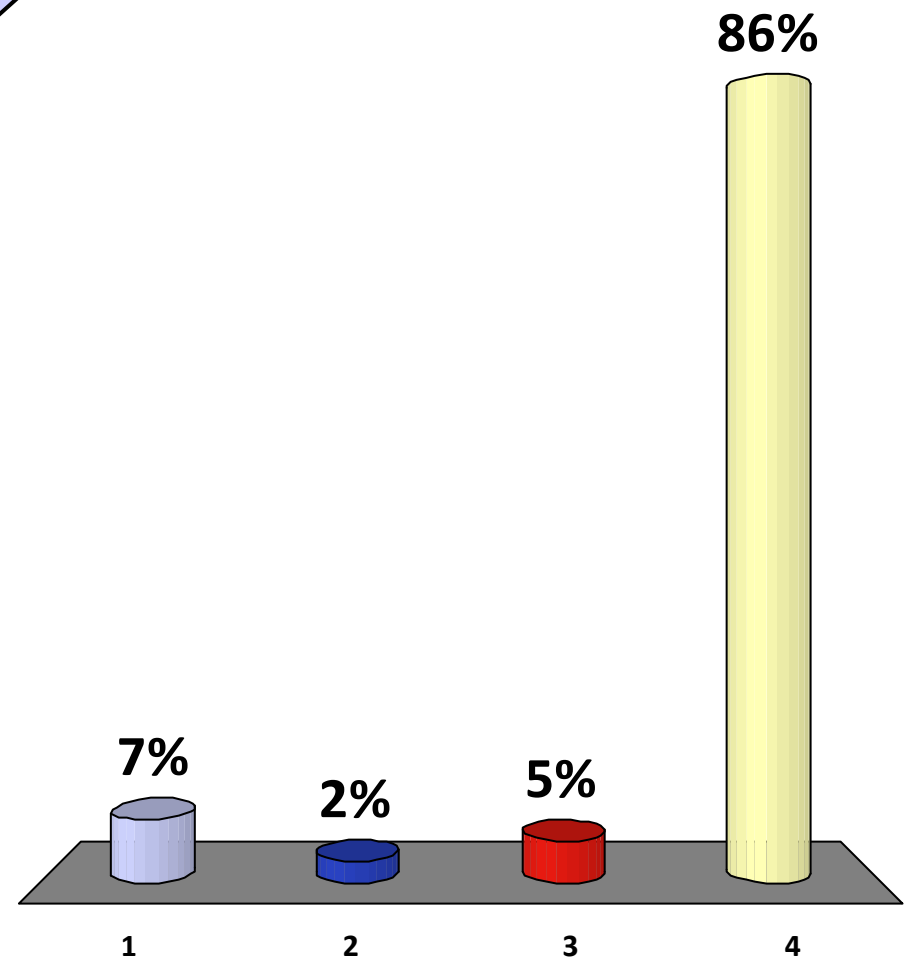




## Which is the correct interpretation?

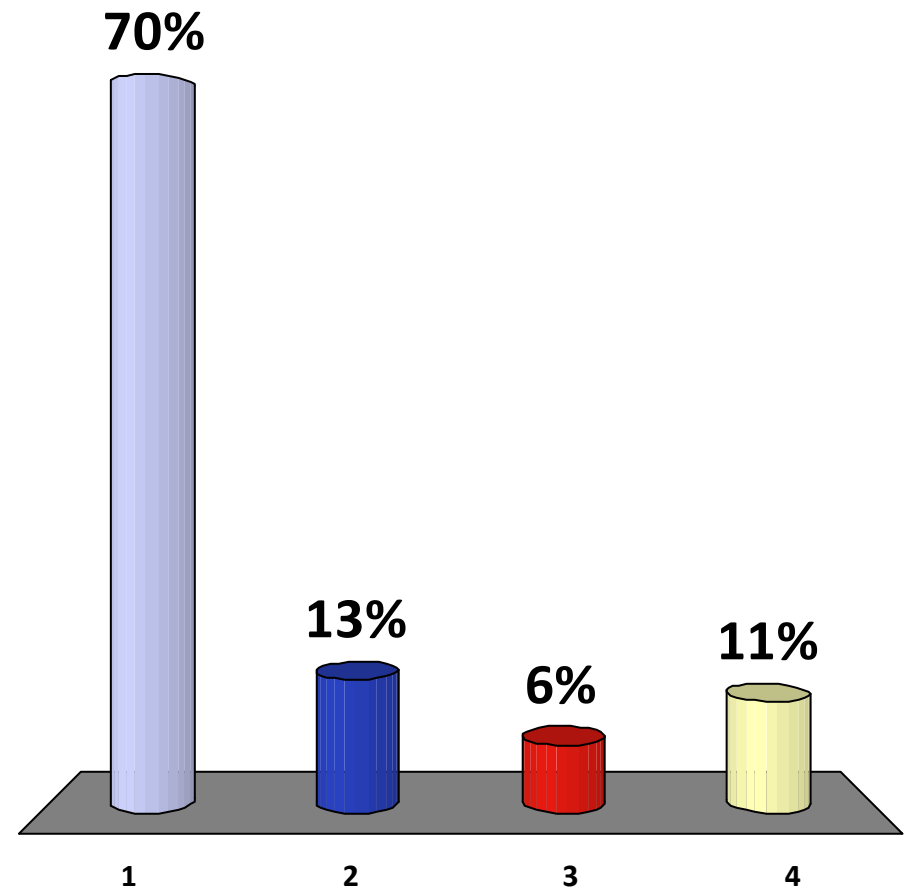
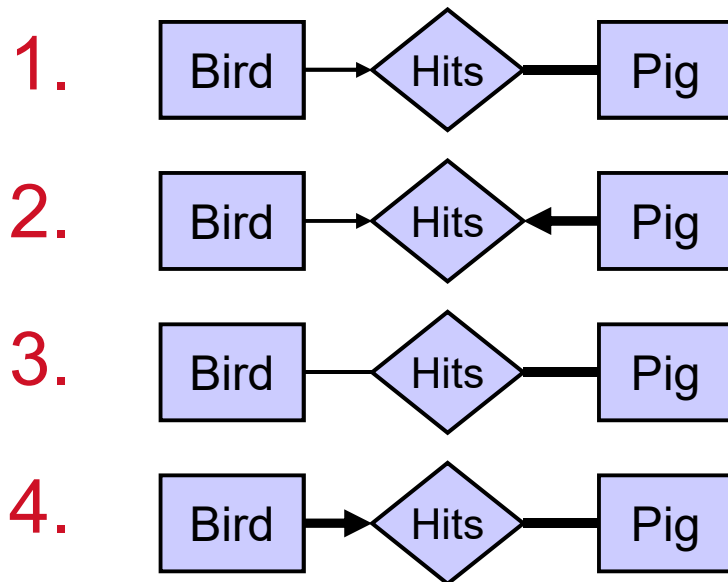


1. Each pet is owned by exactly one child
2. Each child can own at most one pet
3. Each child must own at least one pet
4. Every child must own exactly one pet

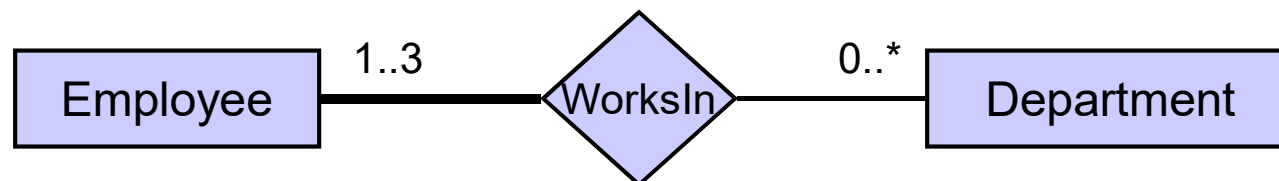


## Which is the correct model?

“Each bird hits at most one pig. Every pig must be hit by at least one bird.”



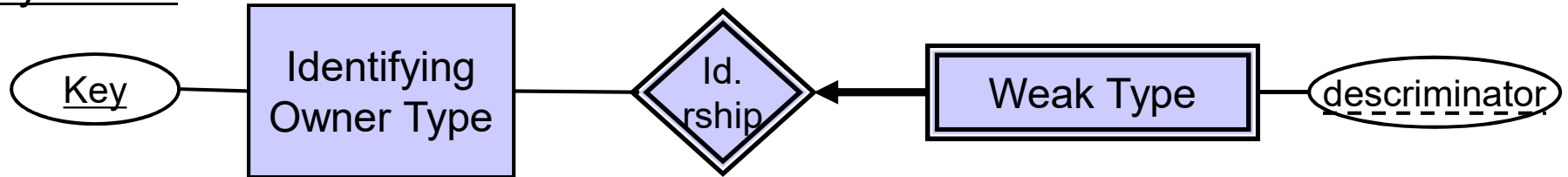
- › Generalisation of key and participation constraints
- › A **cardinality constraint** for the participation of an entity set E in a relationship R specifies how often an entity of set E participates in R at least (minimum cardinality) and at most (maximum cardinality).
  - In an ER-diagram we annotate the edge between an entity type 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").
- › Example: Every employee works in 1 to 3 departments.



- › **Weak entity type**: An entity type that does not have a self-contained primary key.
  - Can be seen as an **exclusive ‘part-of’ relationship**
  - 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
  - Example:  
*payment of a loan*
- › The **discriminator** (or **partial key**) of a weak entity type is the set of attributes that distinguishes 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.

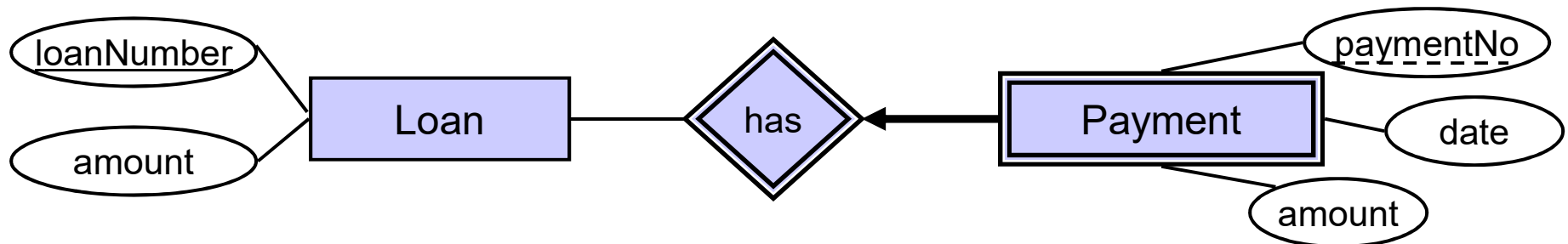
# Representation of Weak Entity Types

## Symbols:



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

## Example:

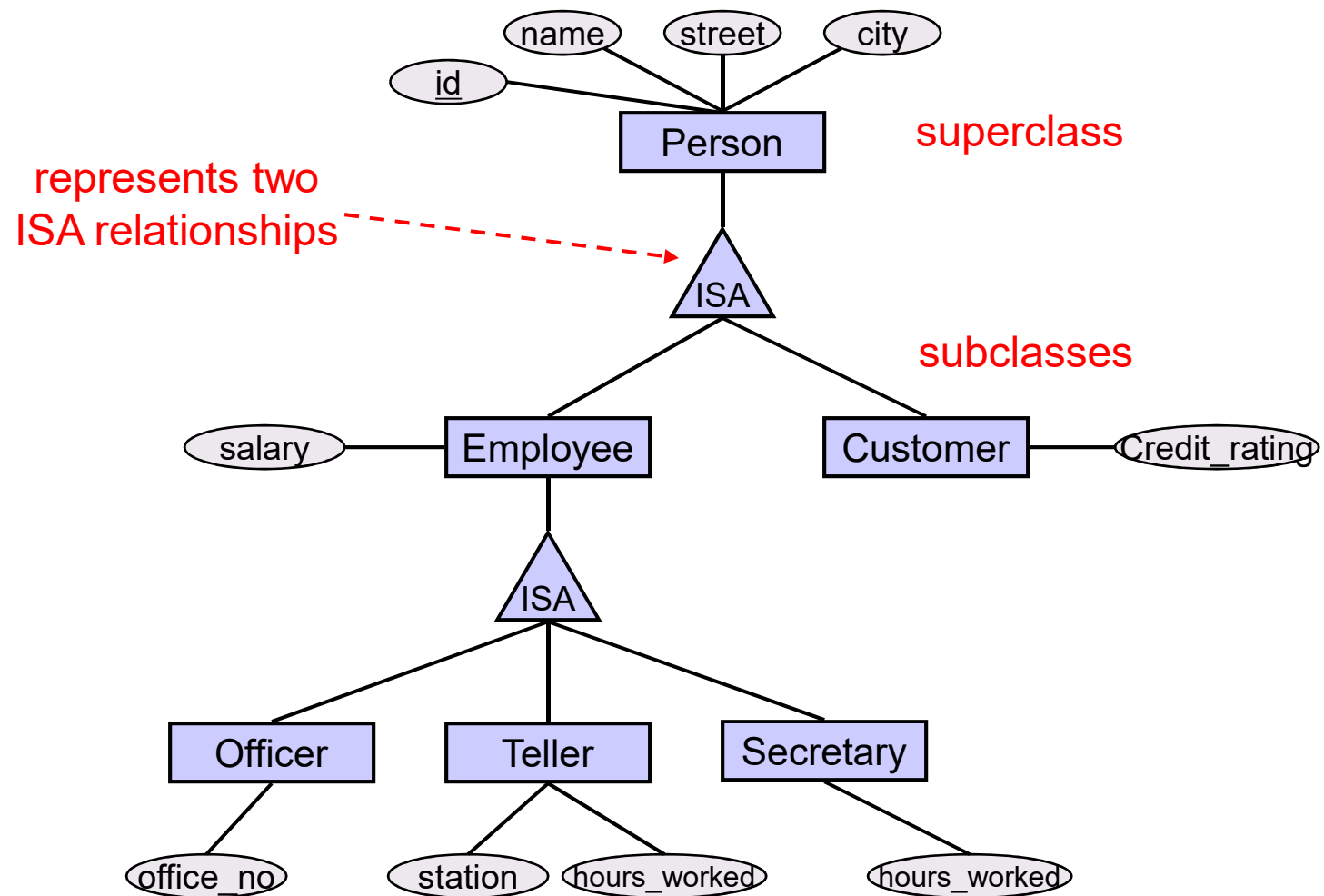


- ▶ paymentNumber: discriminator of the payment entity type
- ▶ Primary key for payment: (loanNumber, paymentNumber)

- › ER model in its original form did not support
  - SPECIALIZATION/ GENERALIZATION
  - ABSTRACTIONS ('aggregation')
  
- › This led to development of 'Enhanced' ER model
  - Includes all modeling concepts of basic ER
  - Additionally some object-oriented concepts: subclasses/superclasses, specialization/generalization, categories, attribute inheritance
  - The resulting model is sometimes called the enhanced-ER or Extended ER (E2R or EER) model
    - used to model applications more completely and accurately if needed
  
- › If we will talk about E-R model, we always mean EER model

- › Arranging of entity types in a type hierarchy.
  - Determine entity types whose set of properties are actual a subset of another entity type.
- › Definition **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  $e$ ")
- › One says 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

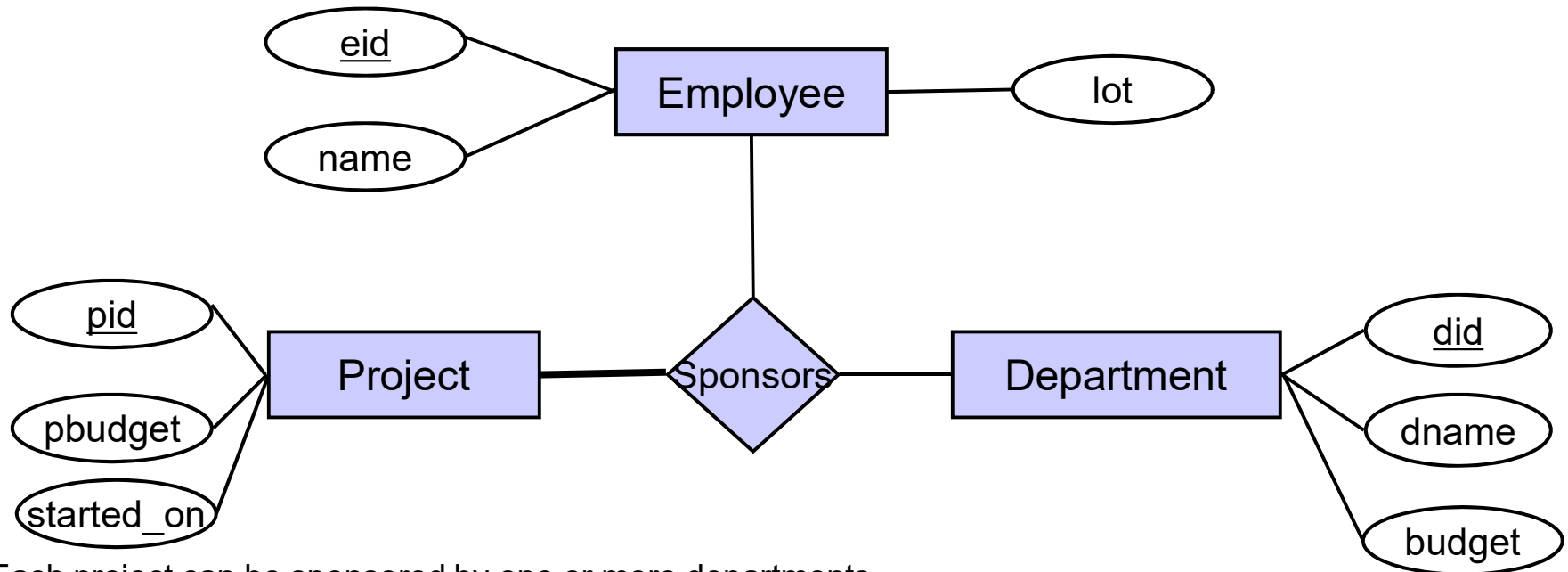
# Superclass / Subclass Example





- › 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

› 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
- Problems?

(Example adapted from Ramakrishnan & Gherke, DBMS, 3<sup>rd</sup> edition)

- › Relationship sets *Sponsors* really tries to model two relationships
  - It tries to model the fact that departments *sponsor* projects, but also that sponsorships can be *monitored*.
  - What if we wanted to add different attributes for each of these relationships?
    - It is more meaningful, and our model can communicate more information if we add such required attributes on the correct relationship.
  
- › We can convey relationships between an entity set and another relationship type via *aggregation*
  - Treat relationship as an abstract entity – surround this with a dashed box (see next slide)
  - Allows relationships between relationships
  - Abstraction of relationship into new entity – the one surrounded by the dashed box

## E-R Diagram With Aggregation

- › To convey a relationship between the *sponsors* relationship type and an 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, 3<sup>rd</sup> edition) - Note use of plural entity type names; we use singular

## › The Database Design Process

- An understanding of the general database design process and the roles of conceptual and logical data modelling

## › Conceptual Data Modelling using the E-R Model

- Understanding and experience with conceptual data modelling using the entity-relationship model:
  - Basic Constructs: Entity, Attributes (single, composite, multivalued, derived), Relationships, Cardinality Constraints
  - Advanced Concepts: Weak Entities, Inheritance, Aggregation



!!! IMPORTANT !!!

- › Which notation should I use for labs/assessments?
  - Use the notation just outlined in these slides

- › Ramakrishnan/Gehrke (3rd edition )
  - Chapter 2 (online quiz released tomorrow, due next week will include questions relevant to this content)
- › 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

### › Readings:

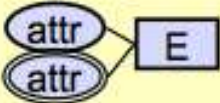


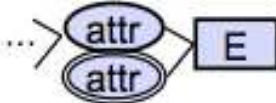
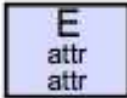





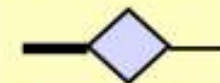


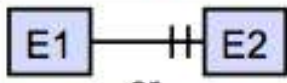
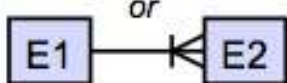



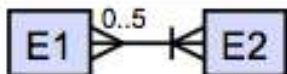





- Ramakrishnan/Gehrke Chapter 3 plus Chapter §1.5
- Kifer/Bernstein/Lewis Chapter 3
- Ullman/Widom Chapter 2.1 - 2.3, Section 7.1 and Chapter 8.1-8.2

### › Attempt homework to direct study

### › Keywords:

- Cardinality, degree
- logical schema
- Relation, tuples/records,
- candidate key, primary key, foreign key
- Mapping ER diagrams to relations
- NULL values



	Kifer / Bernstein / Lewis	Ramakrishnan / Gehrke	Ullman / Widom	Korth / Silberschatz / Sudarshan	Hoffer / Prescott "Crows-Foot"
<b>Entity Name</b>	Entity Type	Entity Set (plural names)	Entity Set (plural names)	Entity Set	Entity Type
<b>Attributes</b>	 <p>only atomic; single- set-valued</p>	 <p>only atomic &amp; single valued</p>	 <p>only single valued (but mention variants with structs &amp; sets)</p>	 <p>single- set-valued composite attr. derived attributes</p>	 <p>single- set-valued composite attr. derived attributes</p>
<b>Key Constraints (for 1-many relationship)</b>	 <p>(arrow from N-side to diamond)</p>	 <p>(arrow from N-side to diamond)</p>	 <p>(arrow from diamond to 1-side)</p>	 <p>(arrow from diamond to 1-side)</p>	 <p>(no diamond; tick on 1-side, crow's foot on many side)</p>
<b>Participation Constraints</b>	 <p>(thick line on total participation side)</p>	 <p>(thick line on total participation side)</p>	n/a	 <p>(double line - total participation side)</p>	 <p>or</p> 
<b>Cardinality Constraints</b>	 <p>min..max notation</p>	n/a	 <p>limit constraint</p>	 <p>min..max notation</p>	 <p>on opposite side!</p>
<b>Roles</b>	yes	yes	yes	yes	yes
<b>Weak Entity (&amp; identifying rel.ship)</b>					
<b>ISA</b>	