

# ISYS2120: Data & Information Management

## Week 2: Conceptual Data Design

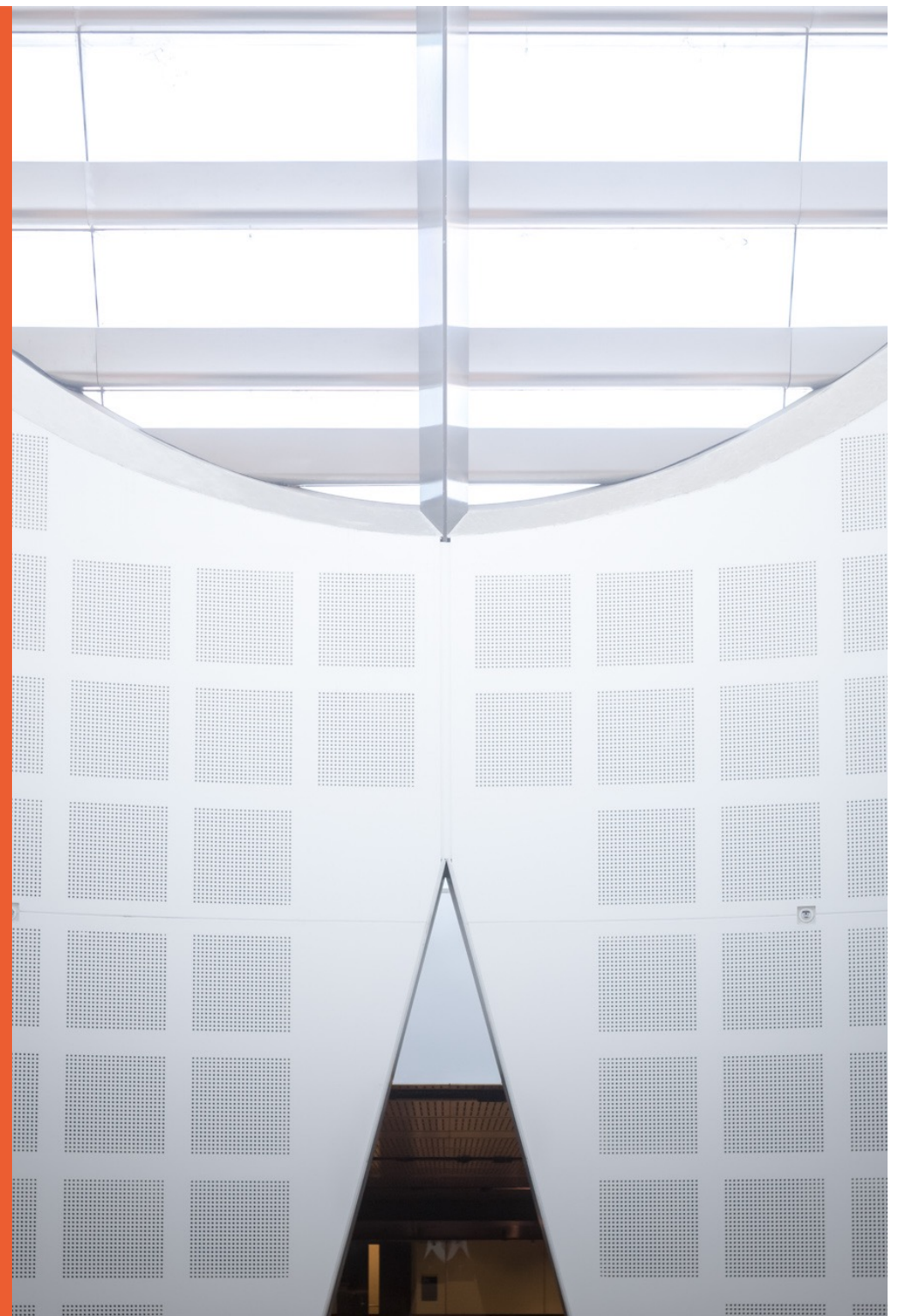
**Presented by**

**Alan Fekete**

- › Based on slides from Kifer/Bernstein/Lewis (2006) “Database Systems” and from Ramakrishnan/Gehrke (2003) “Database Management Systems”, and including material from Ullman, Fekete and Röhms.



THE UNIVERSITY OF  
**SYDNEY**



## 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.**

# Agenda

- Review of Relational Data
- Graphical notation for a relational schema
- Choices for capturing the same information
- Process of Database Design
- Entity-Relationship Conceptual Model
  - ▶ And diagrammatic notation

# Recall: Relational Approach to data

- All data is seen by users as tables of related, uniformly structured information
  - ▶ No duplicate rows, order is not important
  - ▶ Each entry is simple: integer, string, etc
  - ▶ Matching values in different tables indicate connections

Supplier:

SupplD Name Phone

8703	Heinz	0293514287
8731	Edgell	0378301294
8927	Kraft	0299412020
9031	CSR	0720977632

Product:

ProdID Descr SupplD

29012	Peas with Mint, 400g	8731
30086	Peas and Carrots, 450g	8703
31773	Salted Peanuts, 500g	8731



# Relational Schema Notations

- **Supplier(SupplID: int, Name:string, Phone: string);  
Product(ProdID:int, Descr:string, SupplID:int)**
- Alternatively: sometimes leave out the datatype information (implicit)
  - ▶ **Supplier(SupplID, Name, Phone); Product(ProdID, Descr, SupplID)**
- A visual notation

Supplier		
SupplID	Name	Phone

Product		
ProdID	Descr	SupplID

# Relational Schema Notations

- **Supplier(SupplID: int, Name:string, Phone: string);  
Product(ProdID:int, Descr:string, SupplID:int)**
- The visual notation can also be shown vertically

<b>Supplier</b>
SupplID
Name
Phone

<b>Product</b>
ProdID
Descr
SupplID

# Constraints

- Show a column (or set of columns) that is unique among the instances of a table, by underlining those column names
  - ▶ This column (or combination of columns) can serve as an identifier, and other tables can include this identifier as a column (or set of columns) to make connections to the given table
    - The referring column may have a different name, though often it is the same as the identifier)
  - ▶ Show an arrow from the referring columns/table, to the one where the values are found as identifier

Supplier		
<u>SupplD</u>	Name	Phone

SupplD is identifier column for Supplier table

SupplD in Product table  
Reference the value of Suppl identifier in Supplierble

Product		
<u>ProdID</u>	Descr	SupplD

ProdID is identifier column for Product table

# Confusion

- The term “data model” is used in different ways!
- A model of the structure of a particular database (ie the schema for that database)
- OR, the kind of modelling approach that is used for the schema (eg “relational model”, “document model”, etc)



# Lots of choices

- The same information about the world, can be captured in relational databases with different structure (schema)
- Here is a different schema
- **Supplier(SupplID: int, Name:string, Phone: string);  
Product(ProdID:int, Descr:string, SupplID:int);  
SupplierProduct(ProdID:int, SupplID:int)**
- The same information, in this schema

29012	8731
30086	8703
31773	8731

8703	Heinz	0293514287
8731	Edgell	0378301294
8927	Kraft	0299412020
9031	CSR	0720977632

29012	Peas with Mint, 400g
30086	Peas and Carrots, 450g
31773	Salted peanuts, 500g

# A challenge

- Different schema can have advantages and disadvantages
- But some schema have real mistakes
  - ▶ Eg a schema that can't capture some important data in the domain
  - ▶ Eg a schema that makes it hard to express important constraints about the domain
- How can we choose a schema that allows a database to capture the facts about the domain, and do it well?
- We will teach a process to obtain a schema for the database, and then to evaluate it and perhaps improve it
  - ▶ Over several weeks, including Asst1 and Asst2
- Be prepared: some students find it hard to deal with a task where there is no single clear correct answer, but still some answers are not good ones!

# Database design - an example of design process

- Design is a fundamental skill in all engineering fields
- Find some way to do things, that meet the needs of the stakeholders, and is possible within technical restrictions (eg using materials that are available)
  - ▶ And among the ways that are acceptable and possible, find one that is desirable (eg lower cost than others, less environmental damage than others)
- Learn to do design by
  - ▶ Looking at designs others have done, and thinking about the choices that were made
  - ▶ Practicing the process
    - Getting feedback from others, especially more experienced designers (“design review”)

# Design and construction – How to make an idea a reality?

Not examinable

*Let's build a house...*



Source: [the-self-build-guide.co.uk](http://the-self-build-guide.co.uk)

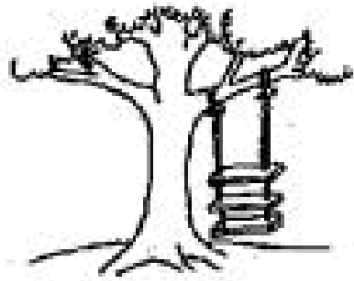


Source: [www.bg-properties.info](http://www.bg-properties.info)

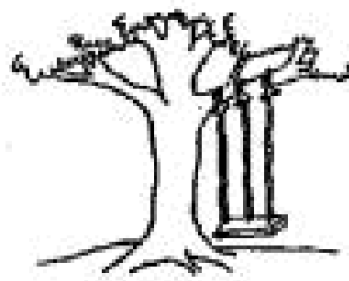


# There are some pitfalls though...

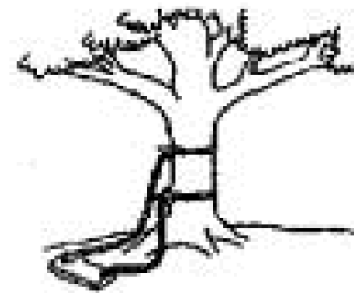
Not examinable



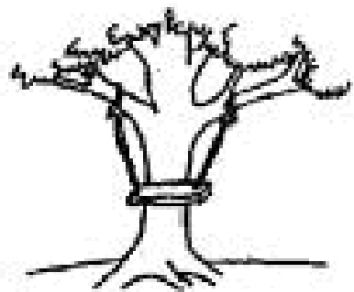
**As proposed  
by the project  
sponsor.**



**As specified  
in the project  
request.**



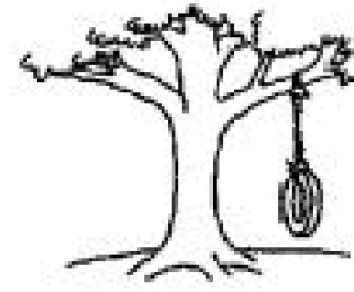
**As designed  
by the senior  
architect.**



**As produced  
by the  
engineers.**



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



**What the  
customer  
really wanted.**

Courtesy: Christopher Alexander, "The Oregon Experiment", 1975.

# Database Design Sequence

## Requirements Analysis

- Understand...
    - ▶ what data is to be stored
    - ▶ what applications must be built
    - ▶ what operations are most frequent
- 

## Conceptual Design

- Develop...
    - ▶ high-level description of the data closely matching how users think of the data
    - ▶ for communication with stakeholders
- 

## Logical Design

- Convert...
    - ▶ conceptual design into a relational database schema (logical)
- 

## Physical Design

- Convert...
  - ▶ logical schema into a physical schema (storage choices) for a specific DBMS and tuned for workload



# Conceptual Data Model

- Goal: Specification of database schema
- Methodology:
  - ▶ **Conceptual Design:** A technique for understanding and capturing business information requirements graphically
    - depicts the associations among different categories of data within a business or information system.
  - ▶ Later one converts conceptual database design to logical schema and express that in SQL DDL
- Conceptual Database 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 independent of *kind-of-data-model, and of DBMS technology*

# Variety of Conceptual Data Models

Not examinable

- Entity-Relationship Model (ERM)
- Object-oriented Data Models
  - ▶ Unified Modelling Language (UML)
  - ▶ ...
- Other approaches.
  - ▶ Object Role Modelling (ORM)
  - ▶ Semantic Object Model (SOM)
  - ▶ Semantic Data Models (SDM)
  - ▶ KL-ONE etc.



# History of Entity Relationship Model

Not examinable

- First designed by Peter Chen in 1976
  - ▶ Several variations have since appeared
  - ▶ Here: **enhanced** or **extended E-R model**
  
- Data model for conceptual database 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

# Entity Relationship Model

- A data modeling 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 relational schema.
  - ▶ Unfortunately, different books, classes and tools use different pictorial notations!
    - In isys2120, be careful to follow what is in the slides for isys2120
- It is about what data needs to be stored
  - ▶ It does **not** imply how data is created, modified, used, or deleted.



# Entities

- **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 COMP5138, 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
  - ▶ Note: entity sets need not to be disjoint (e.g. person who is manager)
- **Attribute**: describes one aspect of an entity type
  - ▶ Example: people have *names* and *addresses*
  - ▶ depicted by an ellipses

# Entity Type

- 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.
  - ▶ Be careful: a key (including the primary key), can consist of a collection of several columns together
- **Entity Schema**: entity type name, attributes (+domains), PK

# Graphical Representation in E-R Diagram

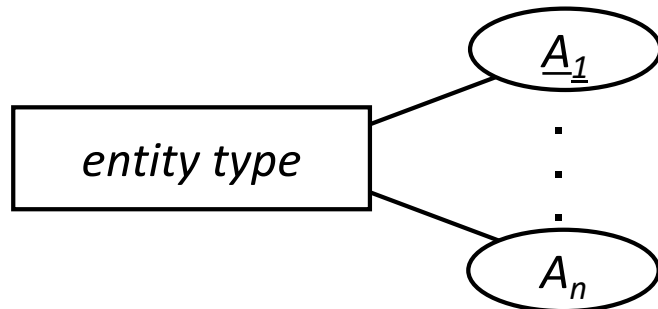
## Symbols:

- ▶ Entity Types represented by a rectangle

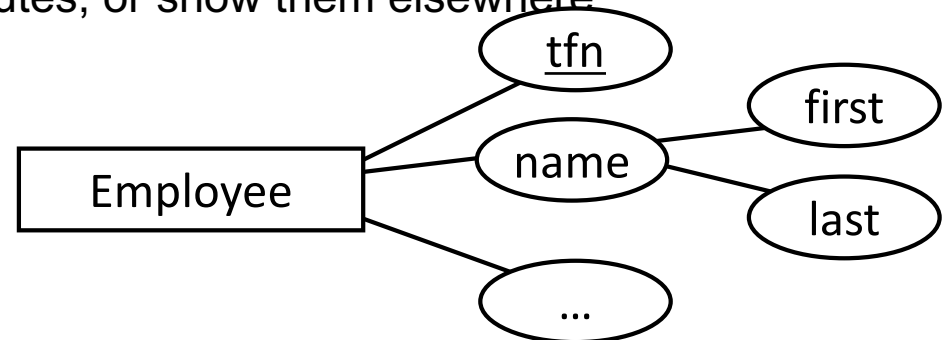
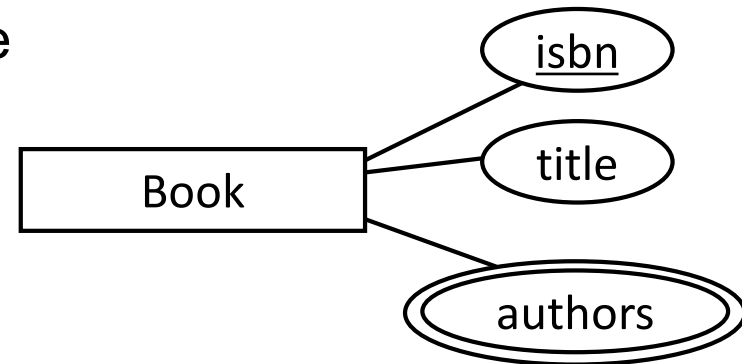


- ▶ Attributes depicted by ellipses

- ▶ Double ellipses for multi-valued attributes
- ▶ Keys are underlined
- ▶ High-level description may leave out attributes, or show them elsewhere



## Examples:



## Remarks:

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

# Relationships

## ■ **Relationship**: relates two or more entities

- ▶ number of entities is also known as the **degree** of the relationship
- ▶ Example: John *is enrolled in* ISYS2120

## ■ **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 **Subject** (entity type) by **EnrolledIn** (relationship type).
  - This **EnrolledIn** relationship type involves two entity types, so we call it a binary relationship type
    - In practice, most relationship types we care about are binary

# Warnings

- A relationship in an ER conceptual model must not be confused with relation in relational model (which means set of tuples)
  - ▶ The information about a relationship may be captured in a relation in a relational schema which we eventually produce for the database
  - ▶ The information about an entity type may also be captured in a relation in a relational schema which we eventually produce for the database

# Relationship Attributes & Roles

## ■ Relationship-Attribute

Relationships can also have additional properties

- ▶ E.g., John enrolls in ISYS2120 *in* the Second Semester 2018
- ▶ John and ISYS2120 are related
- ▶ 2018sem2 describes the relationship - value of the *Semester* attribute of the **EnrolledIn** relationship set

## ■ Relationship-Role

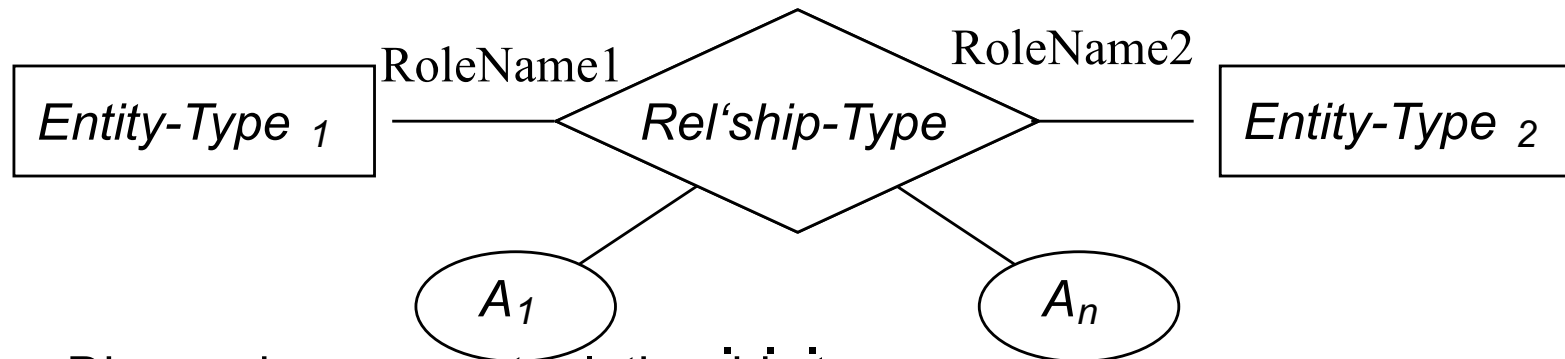
Each participating entity can be named with an explicit role.

- ▶ E.g. John is value of *Student* role, ISYS2120 value of *Subject* role
- ▶ useful for relationship that relate elements of same entity type
- ▶ Example: **Supervises ( Employee:Manager, Employee )**



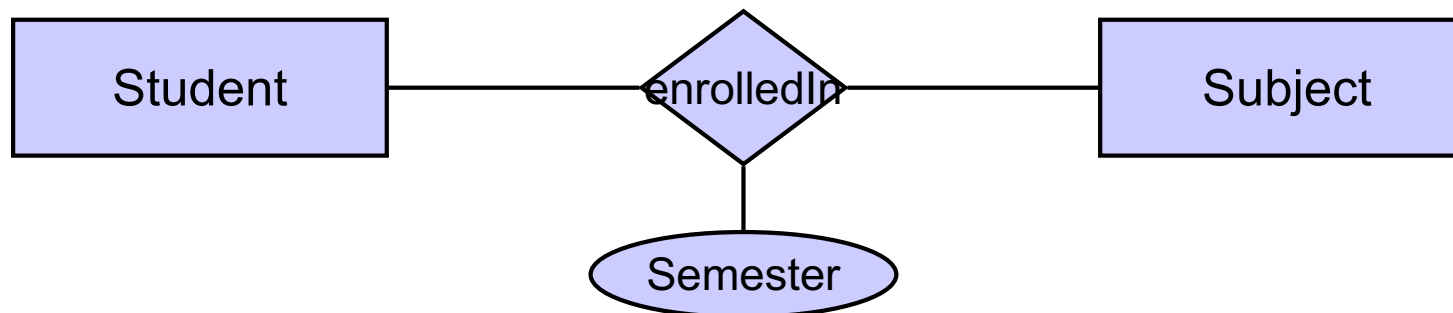
# Graphical Representation of Relationships in E-R Diagrams

Symbol:



- ▶ Diamonds represent relationship types
- ▶ Lines link attributes to entity types and entity types to relationship types.
- ▶ Roles are edges labelled with role names
  - ▶ role label is often omitted when same as entity name

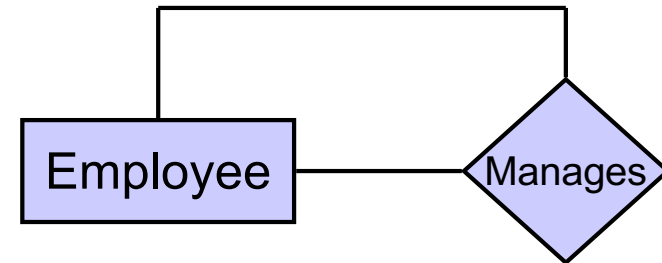
Example



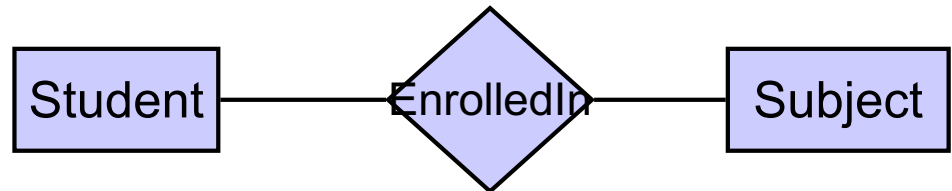
# Relationship Degree

- Degree of a Relationship:  
# of entity types involved

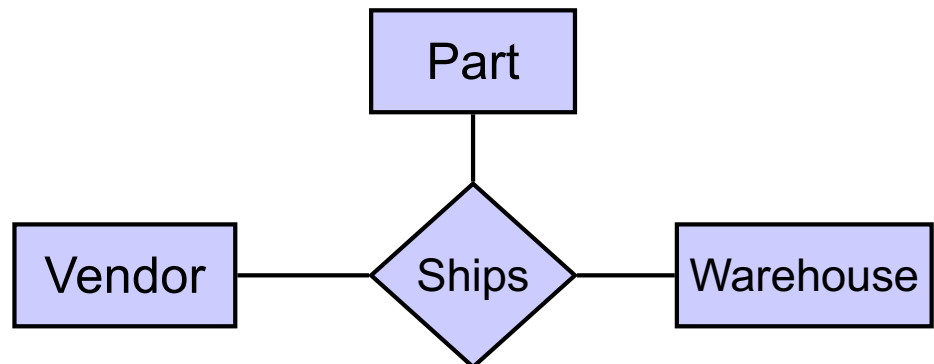
- ▶ Unary Relationship (Recursive)



- ▶ Binary Relationship (most common kind of relationship)

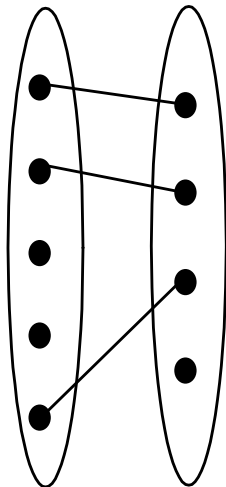


- ▶ Ternary Relationship  
(three entity types involved)

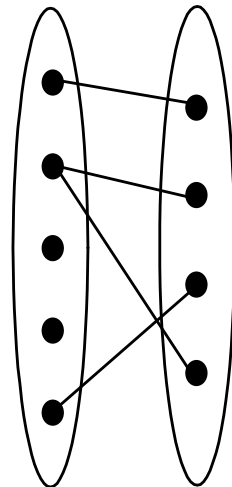


# Multiplicity of Relationships

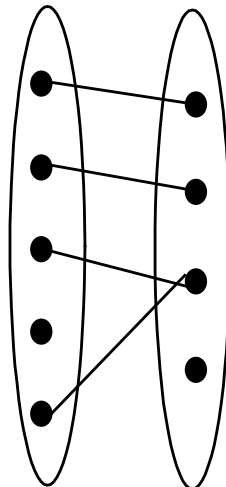
- Consider **Works\_In**: An employee can work in many departments; a department can have many employees.
  - ▶ In contrast, each department has at most one manager
- We find examples of each style of relationship
  - ▶ Think carefully about both directions:
  - ▶ To how many instances of B can a given 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



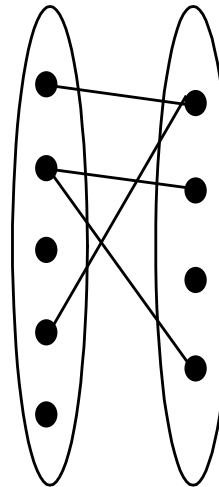
1-to-1



1-to Many



Many-to-1



Many-to-Many

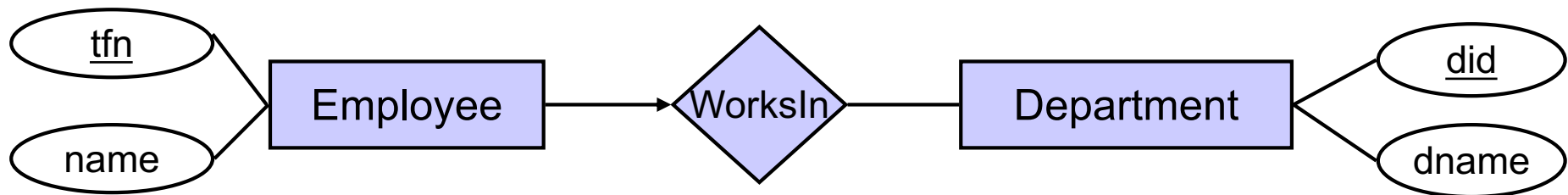
Multiplicities  
are depicted in  
E-R diagrams as  
constraints...  
(see next slides)

# Model comes from domain knowledge

- Consider **Works\_In**: we said “An employee can work in many departments; a department can have many employees.”
- This is not true for every enterprise! There may be a policy that assigns each employee to only one department.
- A conceptual data model should indicate whatever we know about the organisation’s business rules
  - ▶ These become constraints that say “every valid database state must have certain characteristics”
  - ▶ These can be captured in a relational design, and then the DBMS can enforce them
- Multiplicity and other constraints are not intrinsic but must be decided in each case, from discussion with the stakeholders
- In ISYS2120, our focus is on how we document the constraint (whatever it is in a particular situation)
  - ▶ and how this impacts on the database schema and contents

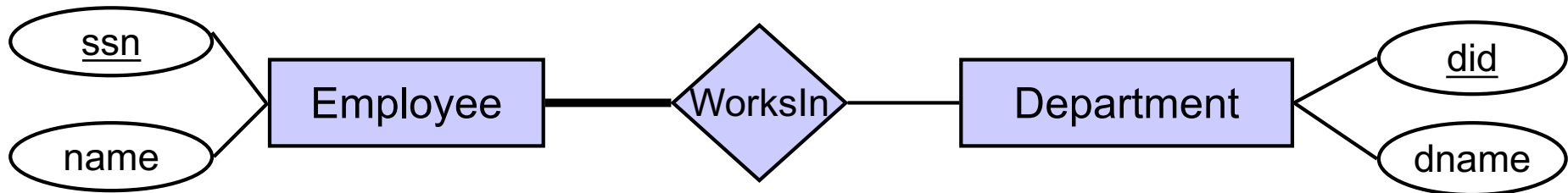
# Key Constraints

- If, for a particular participant entity type, each entity participates in at most one relationship, the corresponding role is a key of relationship type
  - ▶ E.g., *Employee* role is unique in *WorksIn*
  - ▶ there may be many employees who are working in a department
  - ▶ also called: **many-to-one** or **N:1 relationship**
- Representation in E-R diagram: arrow from key side to the relationship diamond
- Example: An employee works in at most one department.



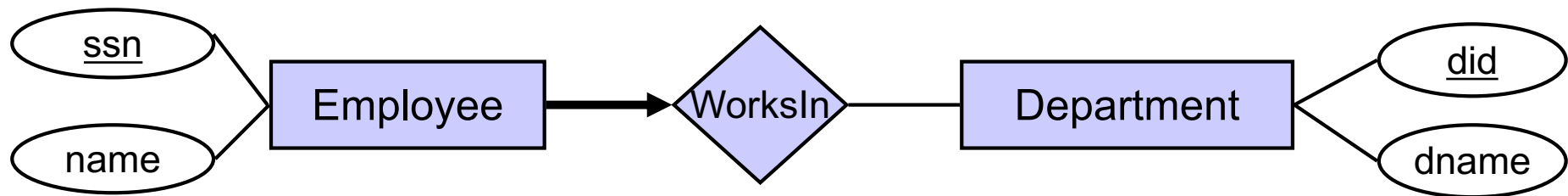
# Participation Constraint

- If every entity participates in at least one relationship, a *participation constraint* holds:
  - ▶ Total participation: each entity  $e \in E$  must participate in a relationship, it cannot exist without that participation (total participation aka existence dependency).
  - ▶ Partial participation: default; each entity  $e \in E$  can participate in a relationship.
- Representation in E-R diagram: thick line from entity type that must participate to relationship diamond
- Example: every employee works in at least one department



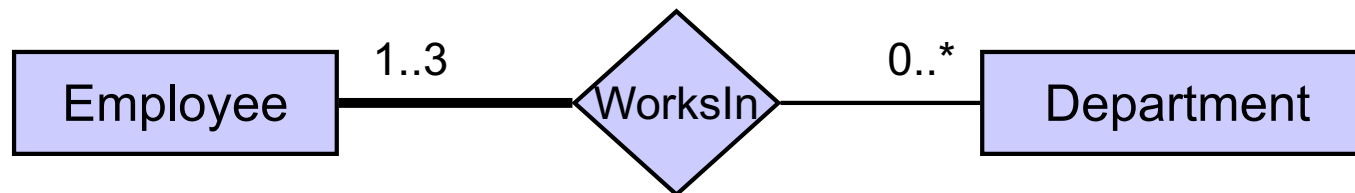
# Participation and Key Constraint

- 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



# 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 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.



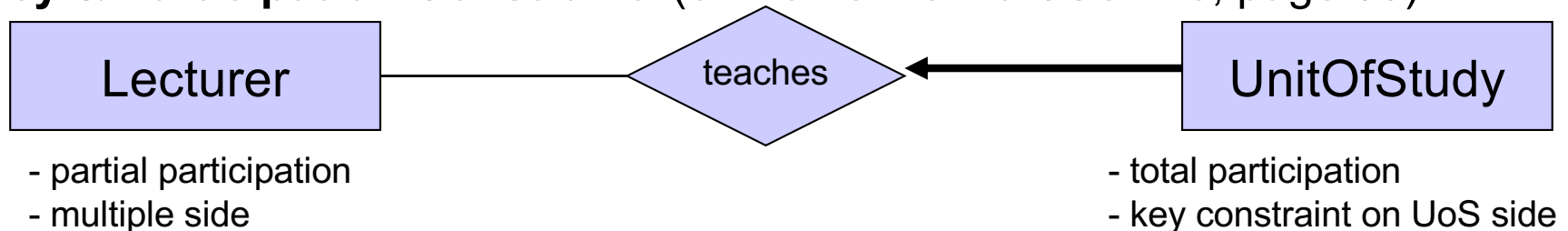


# Comparison of Notations

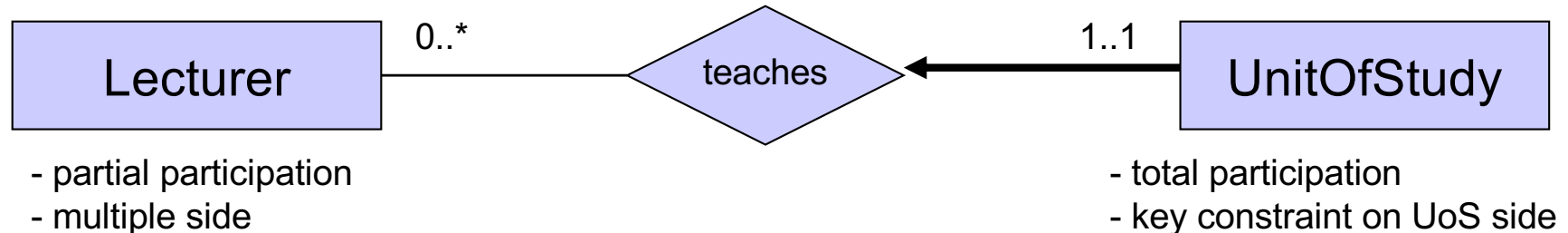
“A lecturer can teach several subjects.”

“Every subject is taught by exactly one lecturer.”

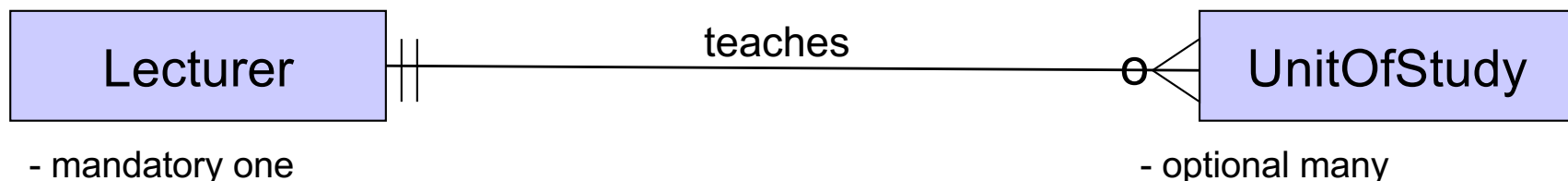
**With Key & Participation Constraint:** (cf. Ramakrishnan/Gehrke, page 33)



**With Cardinality Constraints:** (cf. Kifer/Bernstein/Lewis, pages 76/77 and 82/83)



**“Crow’ s-foot” notation:** (used in many tools)



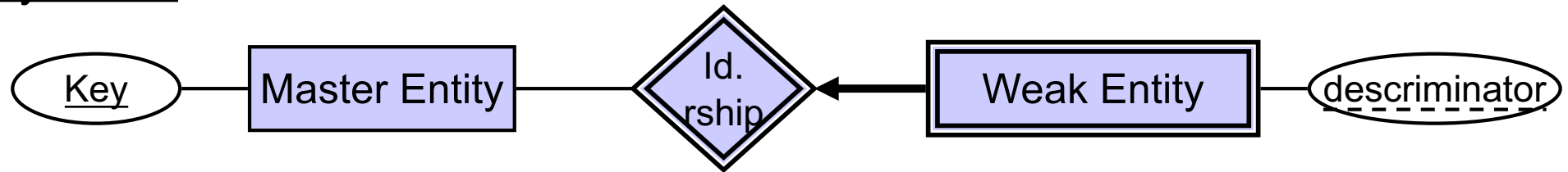
**Be aware** that the Crow’ s foot notation puts 1 or many symbol on other end, as compared to our notation! UML is like Crow’s foot in this aspect.

# Weak Entities

- **Weak entity type**: An entity type that does not have a primary key.
  - ▶ Can be seen as an **exclusive ‘part-of’ relationship**
  - ▶ Its existence depends on the existence of one or more *identifying entity types*
  - ▶ it must relate to the identifying entity set via a total, one-to-many *identifying relationship type* from the identifying to the weak entity set
  - ▶ Examples:  
*child* from parents, *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.

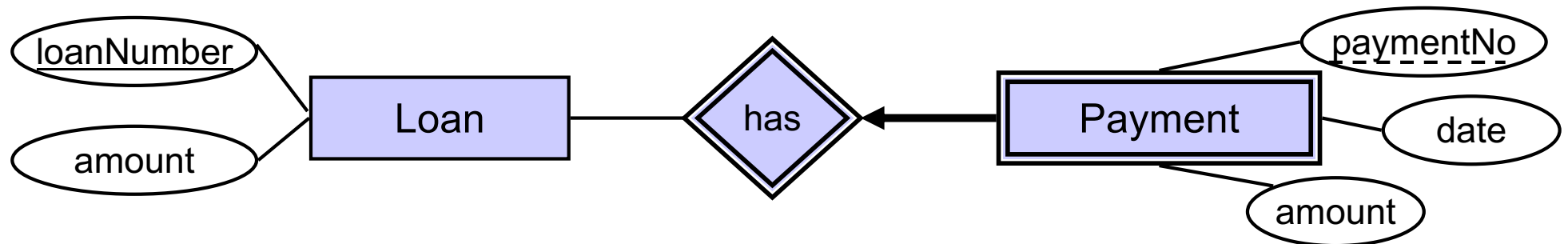
# Representation of Weak Entity Types

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)

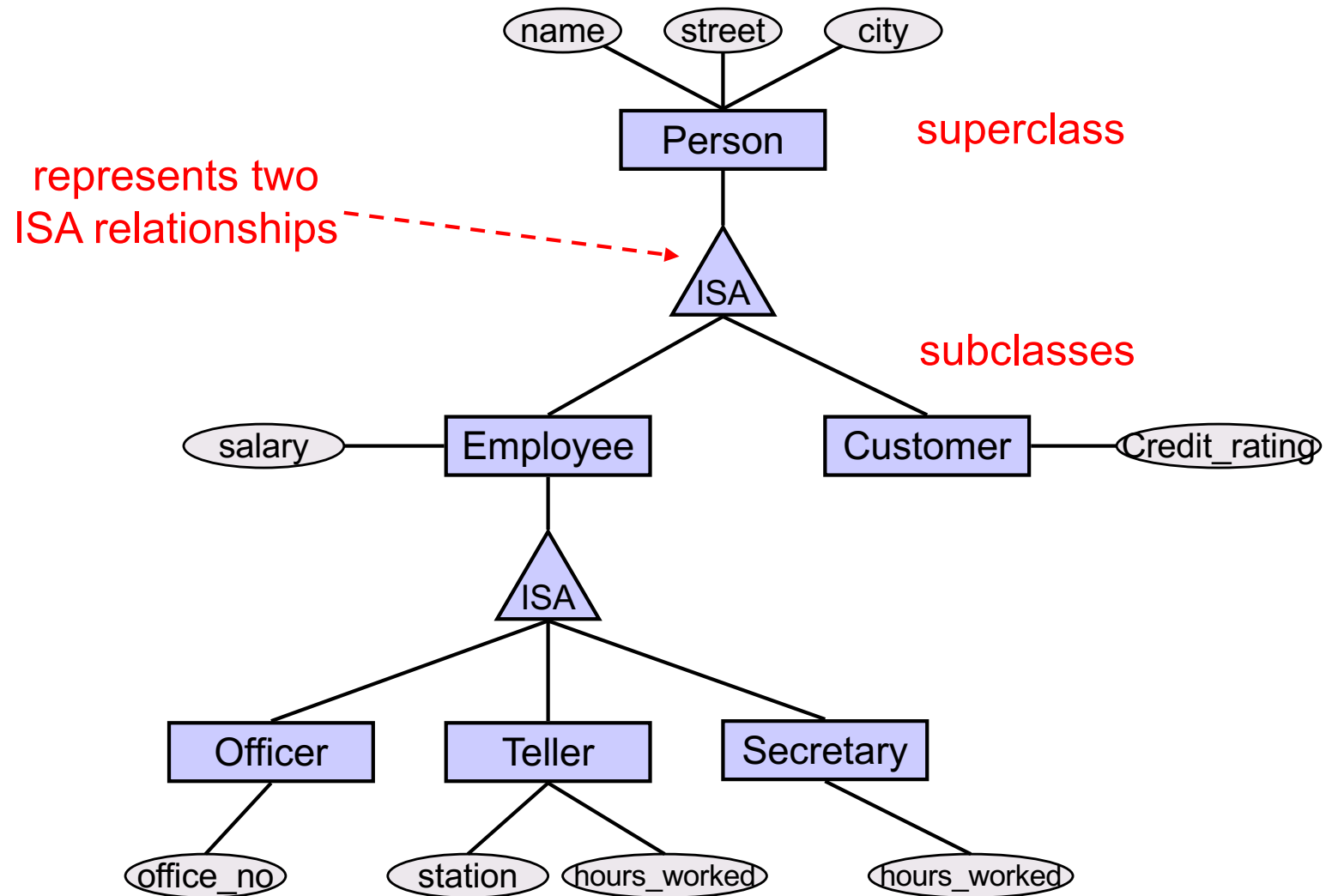
# Enhanced E-R Model

- 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
  - ▶ Additional 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

# Generalisation / Specialisation

- 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 entity type inherits all the attributes and relationship participations of its supertype.
- Depicted by a triangle component labeled IsA

# Superclass/Subclass Example



# 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**

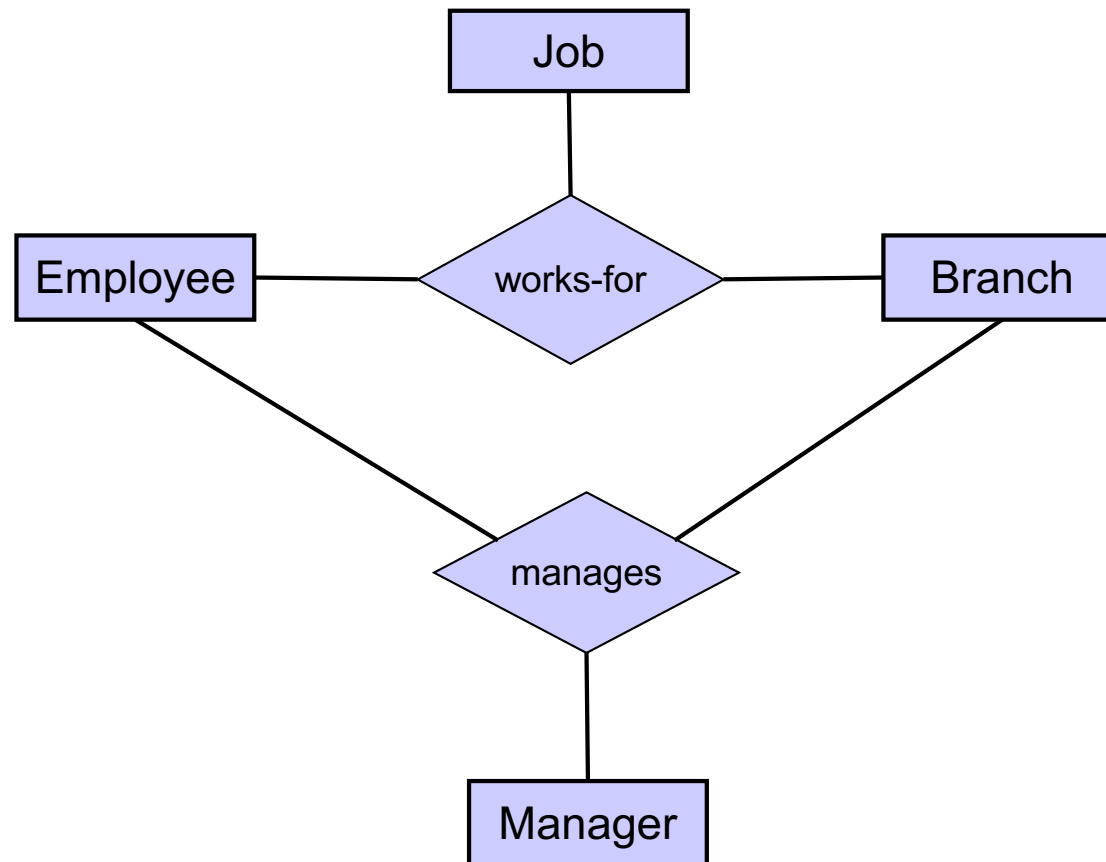
- an entity must belong to one of the lower-level entity sets
    - Denoted with a thick line between the ISA-triangle and the superclass

- ▶ **Partial** (the default)

- an entity need not belong to one of the lower-level entity sets

# Aggregation

- Consider a ternary relationship *works-on*
- Suppose we want to record managers for *tasks* performed by an employee at a branch.
- Like this?



Source:  
Silberschatz/Korth/Sudarshan:  
*Database System Concepts*, 2002.

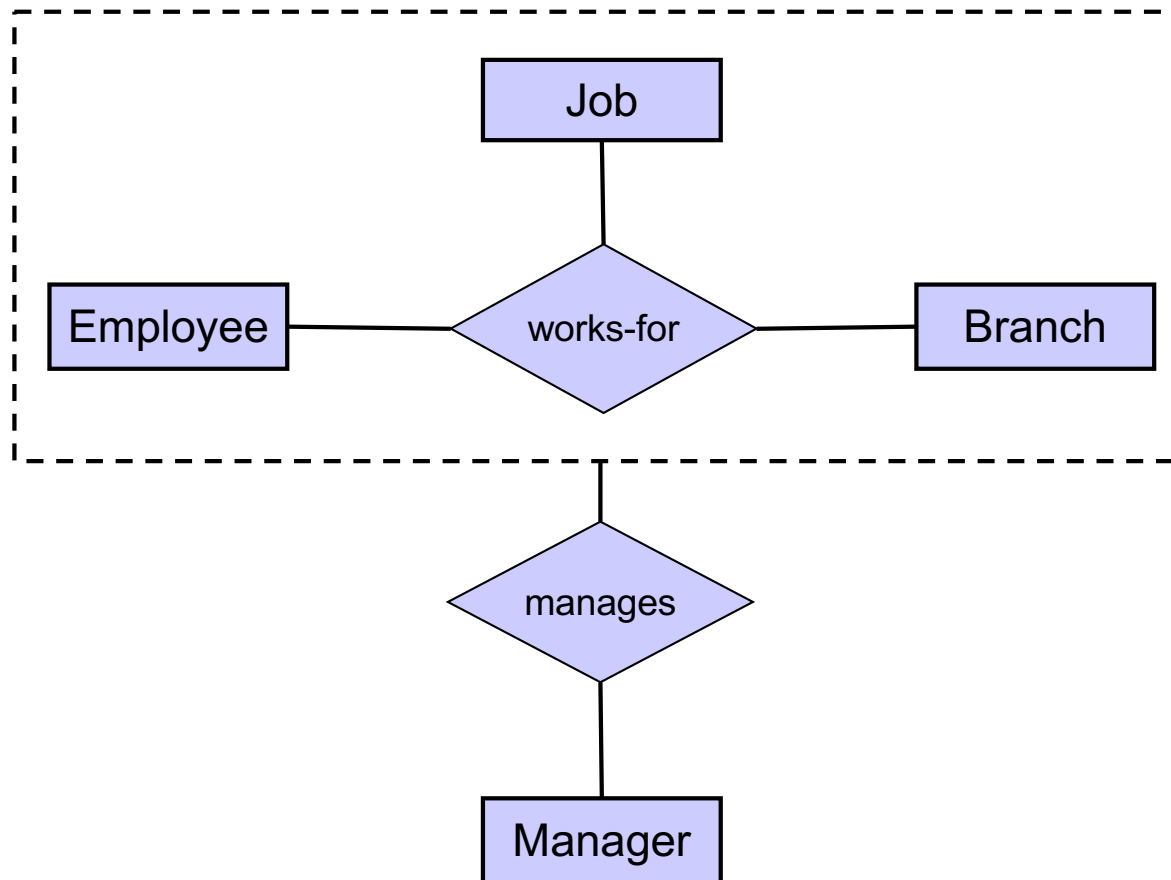


# Aggregation (Cont.)

- Relationship sets *works-on* and *manages* represent overlapping information
  - ▶ Every *manages* relationship corresponds to a *works-on* relationship
  - ▶ However, some *works-on* relationships may not correspond to any *manages* relationships
    - So we can't discard the *works-on* relationship
- Eliminate this redundancy via **aggregation**
  - ▶ Aggregation allows an relationship set to be treated as an (abstract) entity set for the purpose of participating in another relationships
  - ▶ Allows relationships between relationships
  - ▶ Abstraction of relationship into new entity

# E-R Diagram With Aggregation

- Without introducing redundancy, the following diagram represents:
  - ▶ An employee works on a particular job at a particular branch
  - ▶ An employee, branch, job combination may have an associated manager

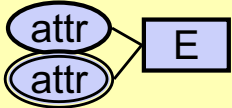
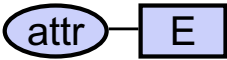
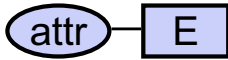
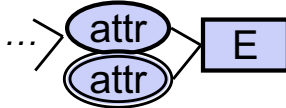
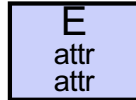
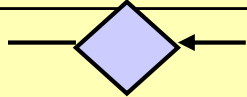
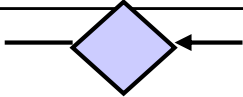
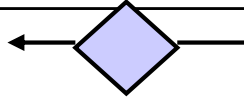
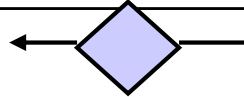
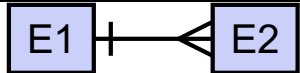
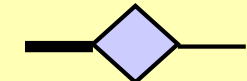
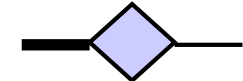
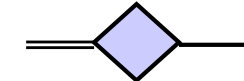
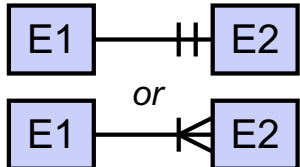
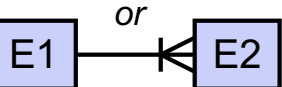
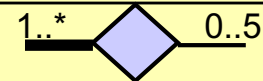
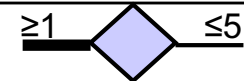
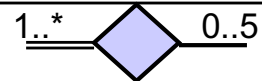
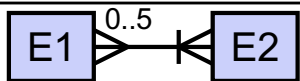
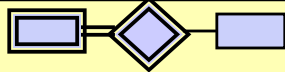



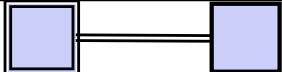


Source:  
Silberschatz/Korth/Sudarshan:  
*Database System Concepts*, 2002.

# References

- Kifer/Bernstein/Lewis (2nd edition)
  - ▶ Chapter 4
  - ▶ *Lecture uses this ER-D notation and naming scheme; book also covers UML*
- Ramakrishnan/Gehrke (3rd edition - the 'Cow' book)
  - ▶ Chapter 2
  - ▶ *similar ER-D notation with few differences (cf. appendix); incl. case study Internet Shop;*
- Ullman/Widom (3rd edition)
  - ▶ Chapter 4
  - ▶ *only basic ER-D with a few differences in its notation;*
- Silberschatz/Korth/Sudarshan (6th edition - 'sailing boat')
  - ▶ Chapter 7
  - ▶ *standard ER-D notation with most details*

# Appendix: Notation Comparison

	Kifer / Bernstein / Lewis (ISYS2120)	Ramakrishnan / Gehrke	Ullman / Widom	Korth/Silberschatz / Sudarshan	“Crows-Foot”
Entity Name	Entity Type	Entity Set (plural names)	Entity Set (plural names)	Entity Set	Entity Type
Attributes	 only atomic; single- set-valued	 only atomic & single valued	 only single valued (but mention variants with structs & sets)	 single- set-valued composite attr. derived attributes	 single- set-valued composite attr. derived attributes
Key Constraints (1-many relationship)	 (arrow from N-side to diamond)	 (arrow from N- side to diamond)	 (arrow from diamond to 1-side)	 (arrow from diamond to 1-side)	 (no diamond; tick on 1-side, crow's foot on many side)
Participation Constraints	 (thick line on total participation side)	 (thick line on total participation side)	n/a	 (double line - total participation side)	 or 
Cardinality Constraints	 min..max notation	n/a	 limit constraint	 min..max notation	 on opposite side!
Roles	yes	yes	yes	yes	yes
Weak Entity (& identifying rel.ship)					
ISA	