Conceptual Database Design

1. Conceptual Database Design

Appl   ⟶   E-R   ⟶   Relational DB

Database Design

Entity-Relationship Model

- The Entity-Relationship (ER) model is a high-level conceptual data model (Chen in 1966).

- ER is used mainly as a design tool.

Entity-Relationship Model_(cont)

- *Entity type*: Group of object with the same properties

- *Entity*: member of an entity type - analogous to an object.

- *Attribute*: a property of object

- *Relationship*: among objects
  - ER can model "n-way" relationship,
  - ER models a relationship and its inverse by a single relationship.

1.1 Entity and Attributes

- *Entities* represent things in the real world.

- *Attributes* describe properties of entities.

- Attributes may be

    – simple(atomic) e.g. sex = 'Female', or

    – composite e.g. name consists of title (Dr), Initials (C.C.), family name (Chen).

1.1 Entity and Attributes(cont)

- Each entity has values for each attribute.

- Attributes may be

  - *single-valued* e.g. student number, name, or

  - *multivalued* e.g. keywords = neural networks, computer graphics, databases.

1.1 Entity and Attributes(cont)

- Each simple attribute has a *value set (domain)*: the set of possible values for that attribute.

- In a composite attribute $A = (A_1, \ldots, A_n)$, suppose that $V_1, \ldots, V_n$ are the domains of $A_1, \ldots, A_n$.

- The domain V of A is $V_1 \times \ldots \times V_n$.

- Mathematically, an attribute A of an entity type E is a function

$$A : E \rightarrow \wp(V) \,.$$

- where V is the domain of A, and $\wp(V)$ is the power set of V

- For single-valued attributes, $A(e)$ must be a singleton.

1.1 Entity and Attributes(cont)

- An attribute can have a null value if, for example:

  – there is no suitable value e.g. a student may have no interests: keywords = NULL

  – the true value is not known e.g. the marriage date of a person is not known: marriage date = NULL.

- A derived attribute is one whose value can be derived from other attributes and entities. e.g. number of students.

1.1 Entity and Attributes(cont)

- An *entity* type is a set of entities with the same attributes.

- It is described by an *entity* schema: a name and a list of attributes.

- The set of individual entity *instances* at a particular moment in time is called an extension of the entity type.

1.1 Entity and Attributes(cont)

Schema (Intension)	RESEACHER Name, Payroll_no, No_of_students, Keywords	DEPARTMENT Name
Instances (Extension)	(Dr C.C. Chen, 230-0013, 3, Neural Networks) (Dr R. Wilkinson, 231-0091, 1, Databases)	Computer Science Psychology Management

1.1 Entity and Attributes(cont)

- An entity type usually has a *key*: a set of attributes that uniquely identifies an entity. For example:
    - {payroll number} is a key of RESEARCHER,
    - {name} is a key of DEPARTMENT.
- There may be more than one possible key.
- An important constraint is the key constraint: in any extension of the entity type, there cannot be two entities having the same values for their key attributes.

1.1 Entity and Attributes(cont)

- We can describe schemata with composite attributes using ()'s and with multi-valued attributes using {}'s. e.g.

1.1 Entity and Attributes(cont)

CAR Registration(Registration No, State), Make, Model, Year, {Colour}
((ARQ) 595, Vic), Datsun, 120Y, 1972, {green}) ((8HR) 696, WA), Mazda, 929, 1979, {grey, black})

1.1 Entity and Attributes(cont)

- Entities and their attributes can also be described with Entity-Relationship Diagrams (ERDs). e.g.
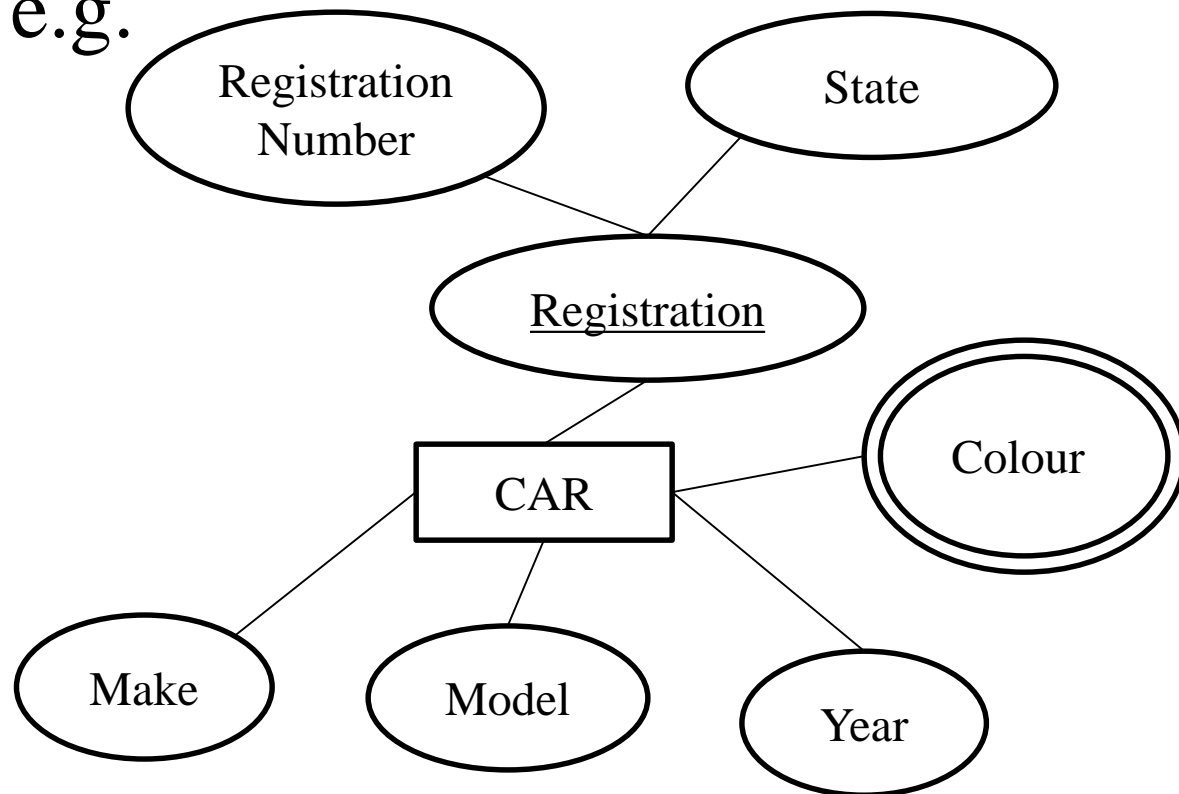
1.2 Relationships

- A *relationship* represents an association between things.

- A *relationship* type R among n entity types $E_1, \ldots, E_n$ is a set of associations among entities from these types.

- Mathematically, a relationship type R among entity types $E_1, \ldots, E_n$ is a subset of $E_1 \times \ldots \times E_n$.

- Each instance $r = (e_1, \ldots, e_n)$ in R is a relationship.

1.2 Relationships(cont)

- We say that $E_1, \ldots, E_n$ participate in R.

- Similarly if $r = (e_1, \ldots, e_n)$ is an instance of R, we say that each e_i participates in r.

- The *degree* of R is the number of participating entity types. For example,

  – ENROLMENT could be a ternary (degree 3) relationship between RESEARCHER, STUDENT and COURSE.

- We can illustrate this using an occurrence diagram:

RESEARCHER

Wilkinson

Chen

STUDENT

Chen

Juliff

Gledhill

Lee

ENROLMENT

COURSE

PhD

MSc
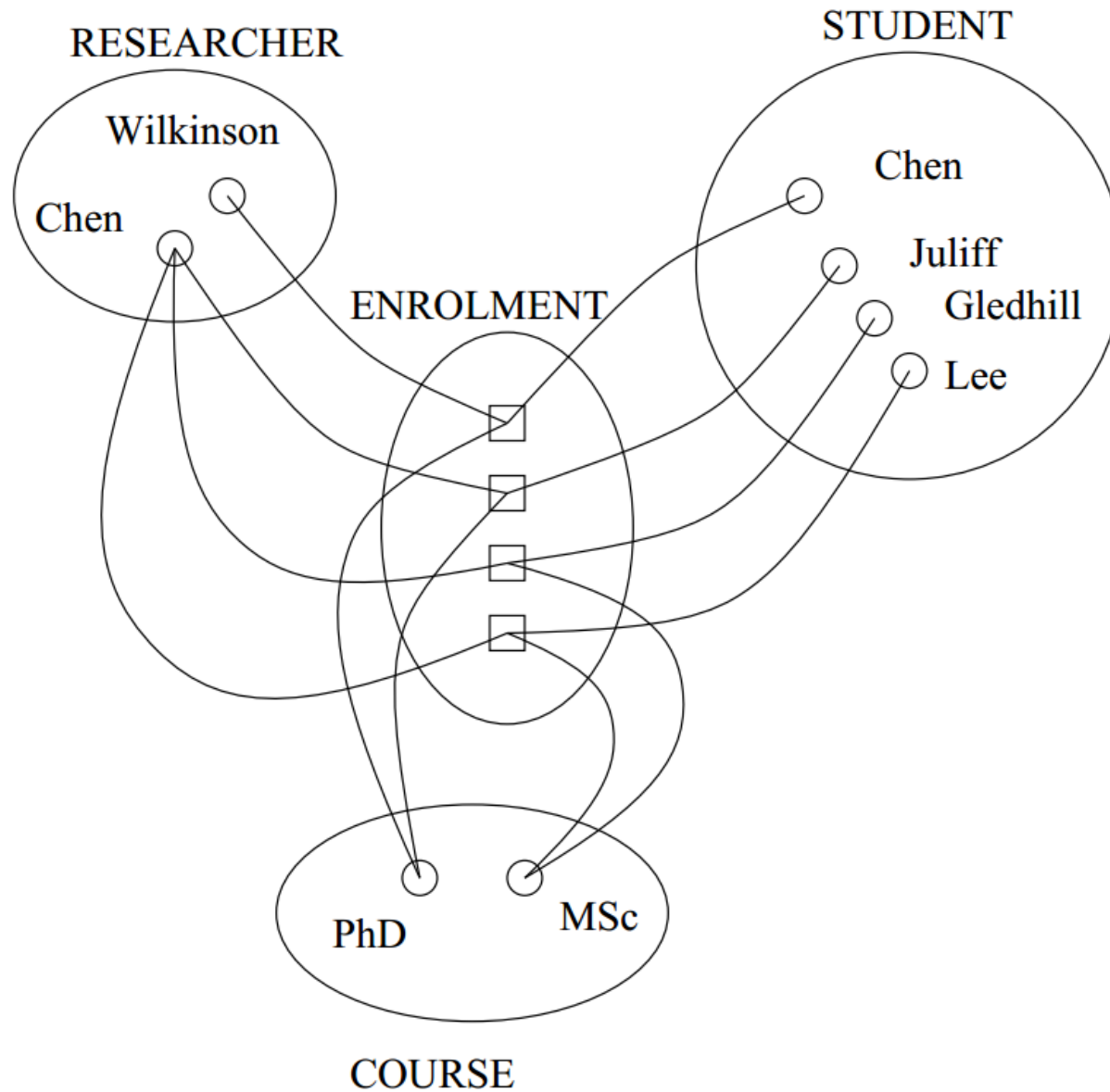
1.2 Relationships(cont)

- Entities and their relationships can also be represented using Entity-Relationship diagrams:
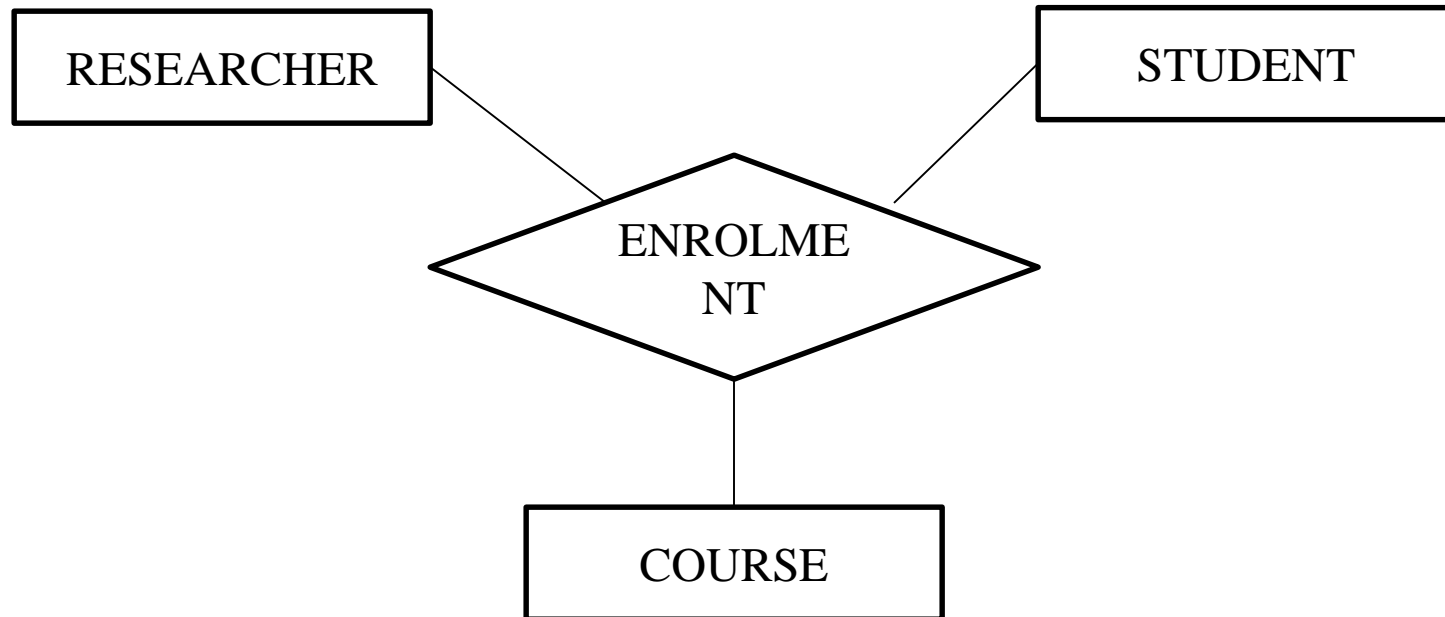
1.2 Relationships(cont)

- Each entity type that participates in a relationship plays a particular *role* in the relationship.

- An entity type can play
  - different roles in different relationships, or
  - more than one role in a relationship.

- A role name can be used to distinguish these.

- For example, ENROLMENT could be a relationship between PERSON(as researcher), PERSON(as student) and COURSE as in the diagram below:
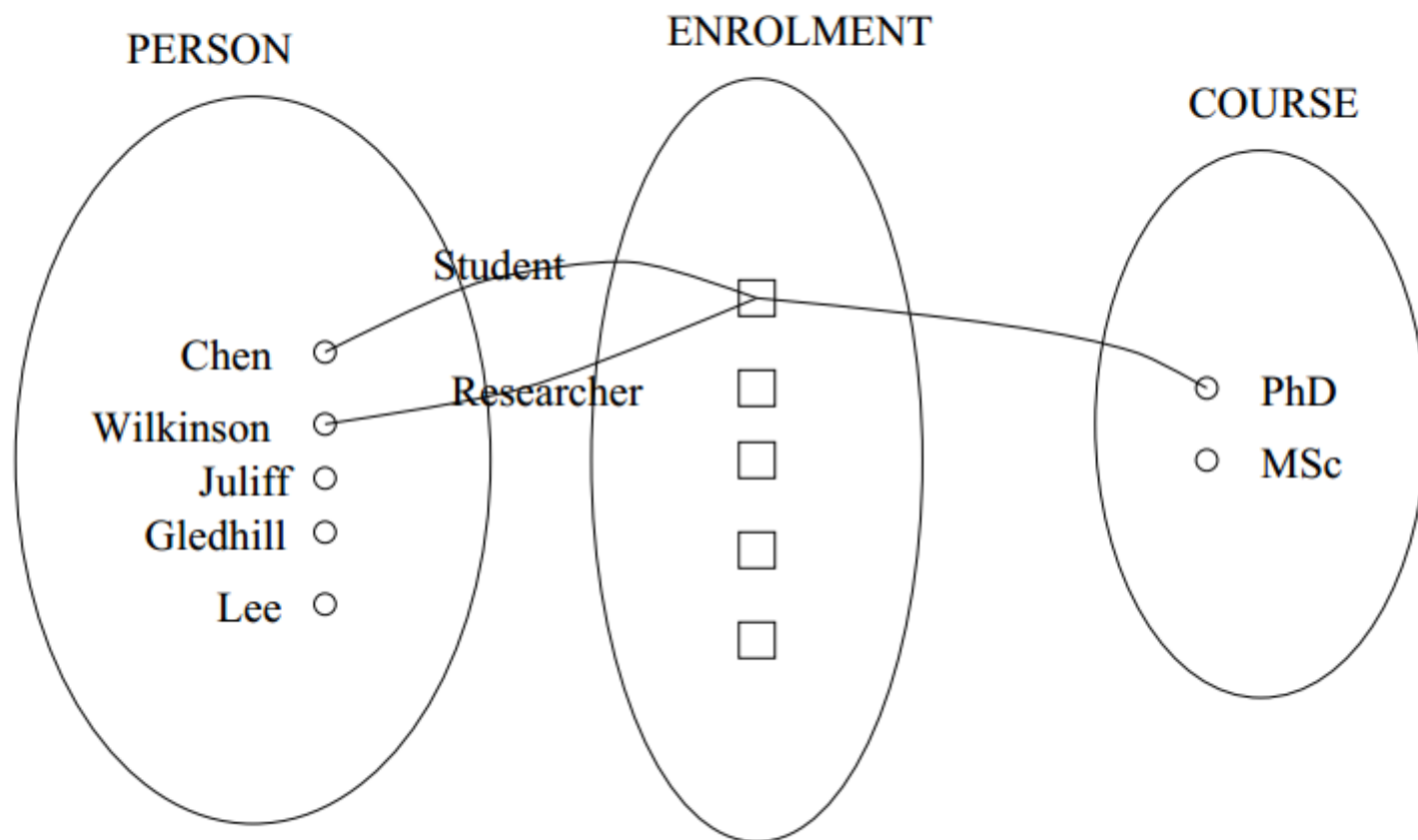
1.2 Relationships(cont)

- Or, using an ERD:
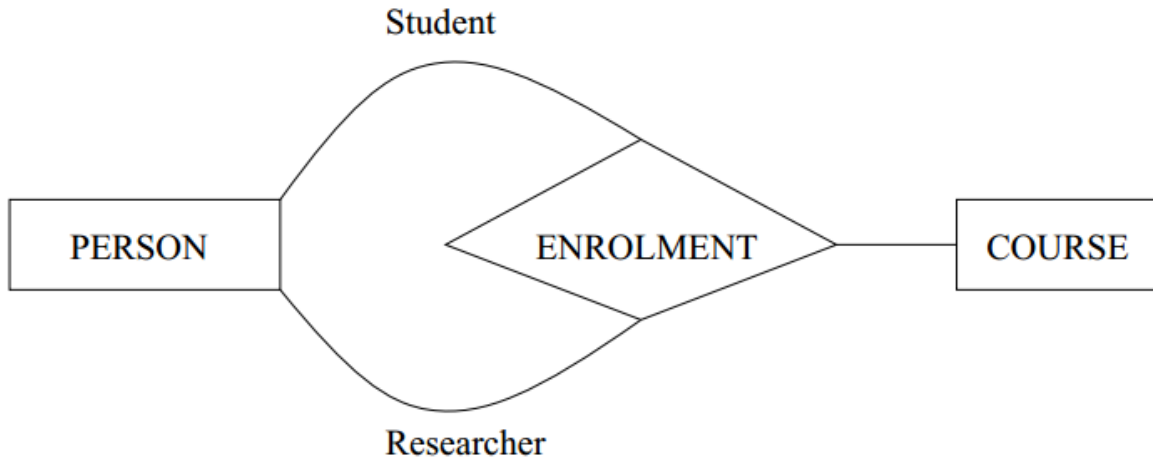
- This is called a recursive relationship.

1.3 Weak entity types

- Some entity types do not have a key of their own.

- Such entity types are called weak entity types.

- Entities of a weak entity type can be identified by a partial key and by being related to another entity type - *owner.*

- The relationship type between a weak entity type to its owner is the *identifying relationship* of the weak entity type.

1.3 Weak entity types(cont)

- For example, a TAX PAYER entity may be related to several DEPENDENT, identified by their names.

- In this example, DEPENDENT is called a weak entity, {Name} is a partial key for it. The identifying relationship between DEPENDENT and TAX PAYER is IS DEPENDENT OF. TAX PAYER is said to *own* DEPENDENT.

1.4 Constraints on relationship types

- Relationship types usually have certain constraints that limit the possible combinations of entities participating in relationship instances.

- They should reflect the correct factors

- *Cardinality ratio constraint*: specifies the number of relationship instances an entity can participate in.

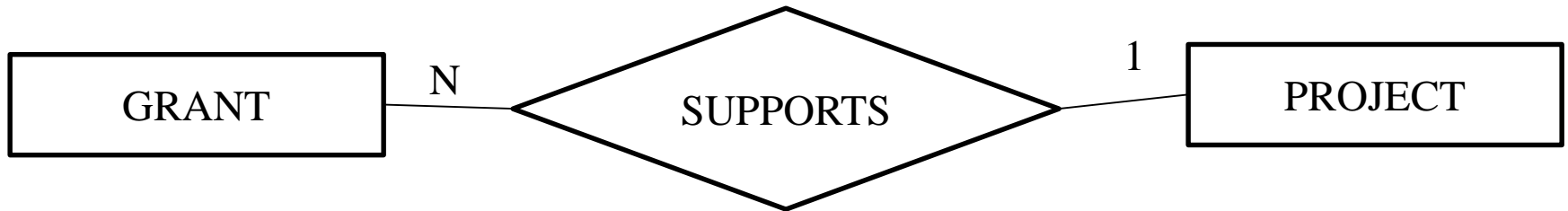- Example: A research grant supports only one research project, but a research project may be supported by many grants. PROJECT:GRANT is a 1 : N relationship.

1.4 Constraints on relationship types(cont)

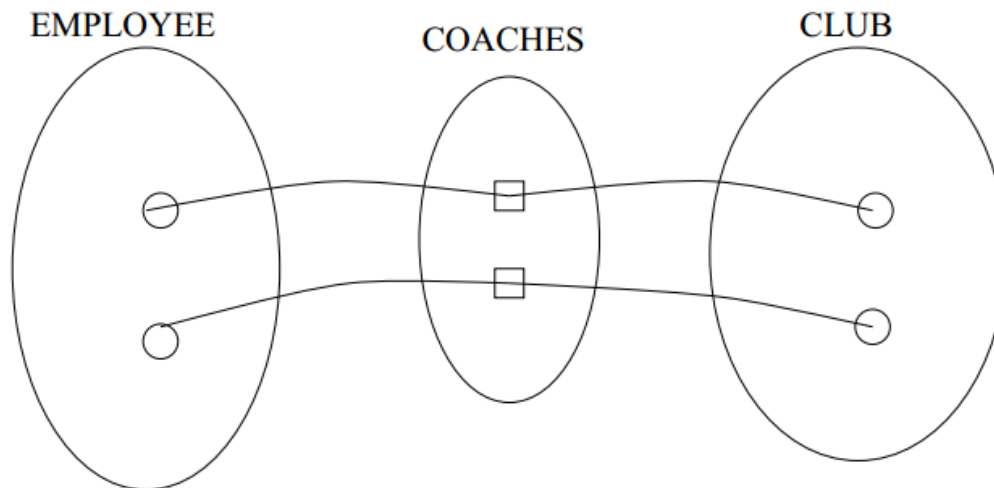- This is illustrated in the occurrence diagram below:
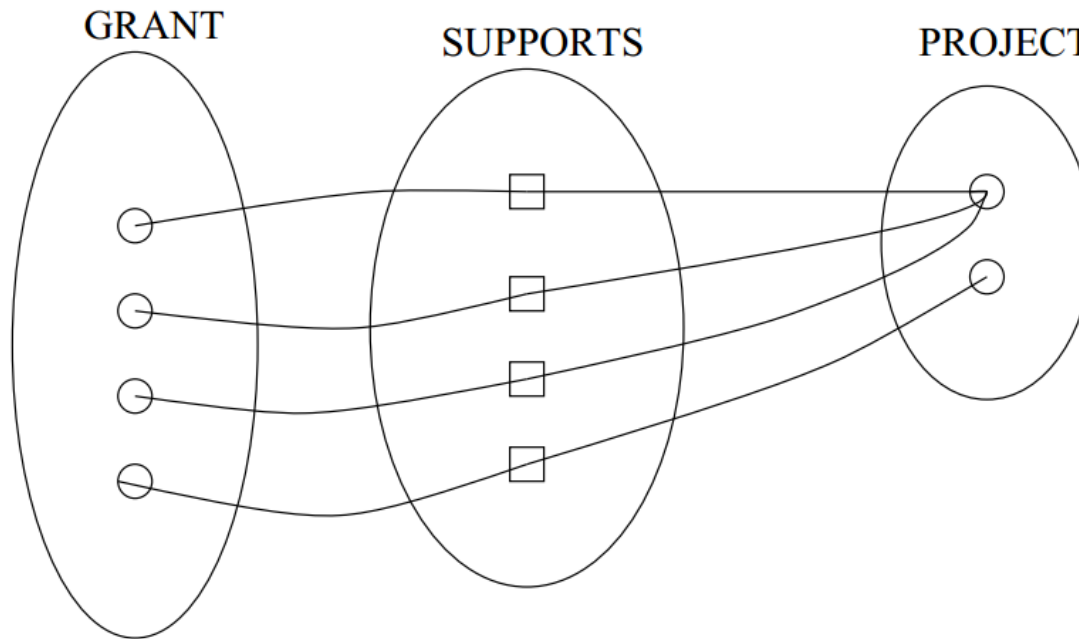
1.4 Constraints on relationship types(cont)

- We can also show this in an ERD:

```
┌──────────┐   N              1   ┌──────────┐
│  GRANT   │───────< SUPPORTS >───│ PROJECT  │
└──────────┘                      └──────────┘
```

1.4 Constraints on relationship types(cont)

- Example: Consider a database of AFL (here substitute your favourite team sport) statistics. The relationship of head coaches to clubs is an example of a 1 : 1 relationship.
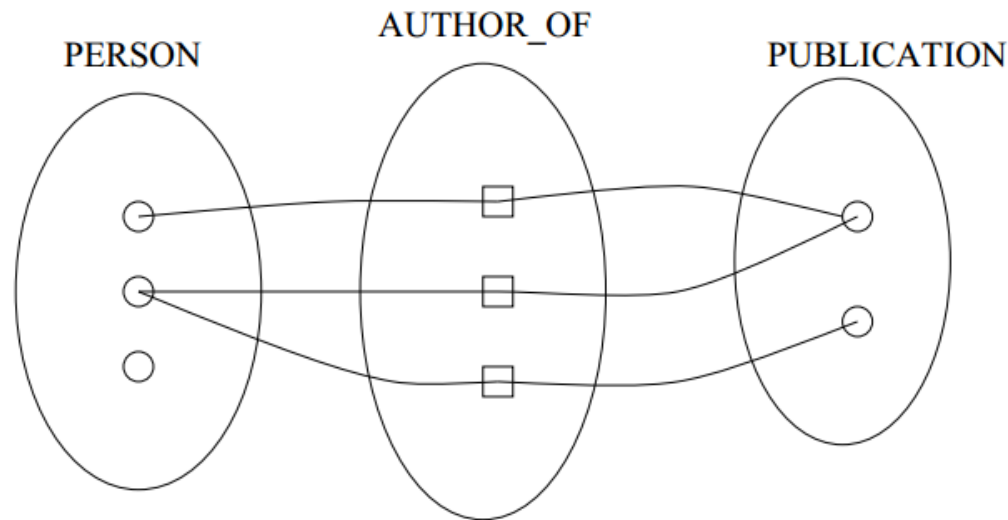
1.4 Constraints on relationship types(cont)

- With an ERD:

| EMPLOYEE | 1 — COACHES — 1 | CLUB |

1.4 Constraints on relationship types(cont)

- Example: An example of an N : M relationship is authorship of publications:

1.4 Constraints on relationship types(cont)

- The equivalent ERD:

PERSON — N — AUTHOR_OF — M — PUBLICATION

1.2.4 Constraints on relationship types(cont)

- Another kind of constraint that can be represented using the ER model is a

  – *Participation constraint*: participation of an entity in a relationship can be:

    - *total*: every entity must participate e.g. every publication has an author.

    - *partial*: not necessarily total. e.g. not every person has publications.

# 1.4 Constraints on relationship types(cont)

- This can be shown with an ERD like the one below:

PERSON —N— AUTHOR_OF —M— PUBLICATION

# 1.5 Attributes of relationship types

- Relationship types can have attributes – for example,

  – a researcher may work on several projects. The fraction of her time devoted to a particular project could be an attribute of the WORKS ON relationship type.

- This can be shown in an ERD as below:

1.5 Attributes of relationship types(cont)
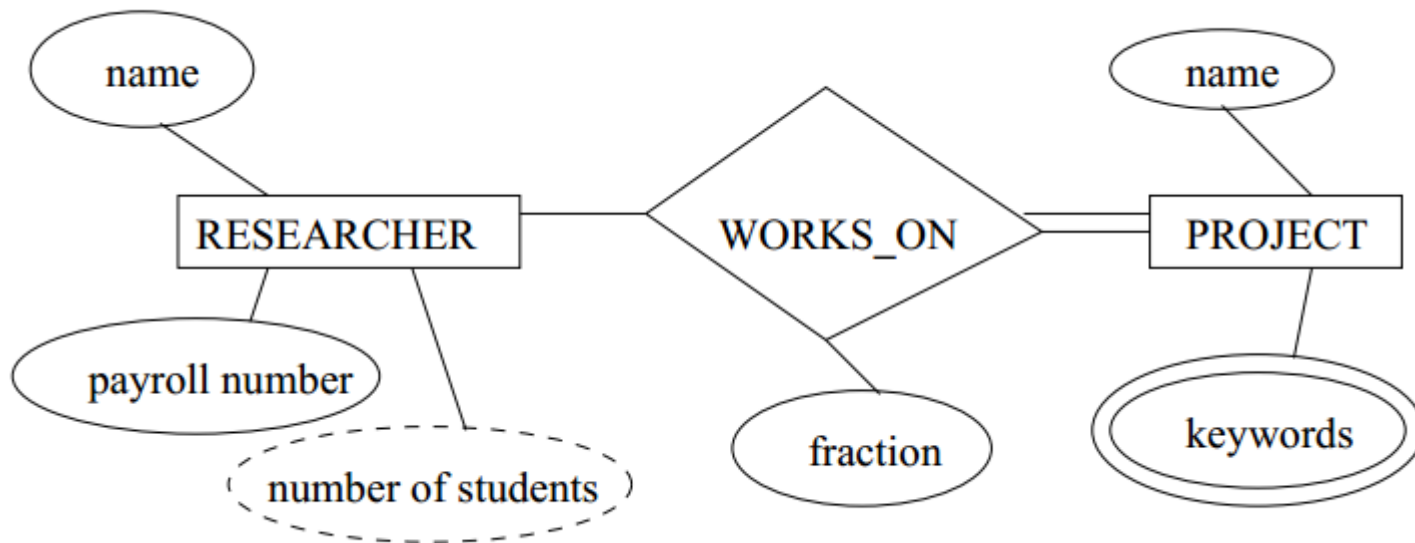
1.5 Attributes of relationship types(cont)

- The notation used for ERDs is summarised in Elmasre/Navathe Figure 3.15.
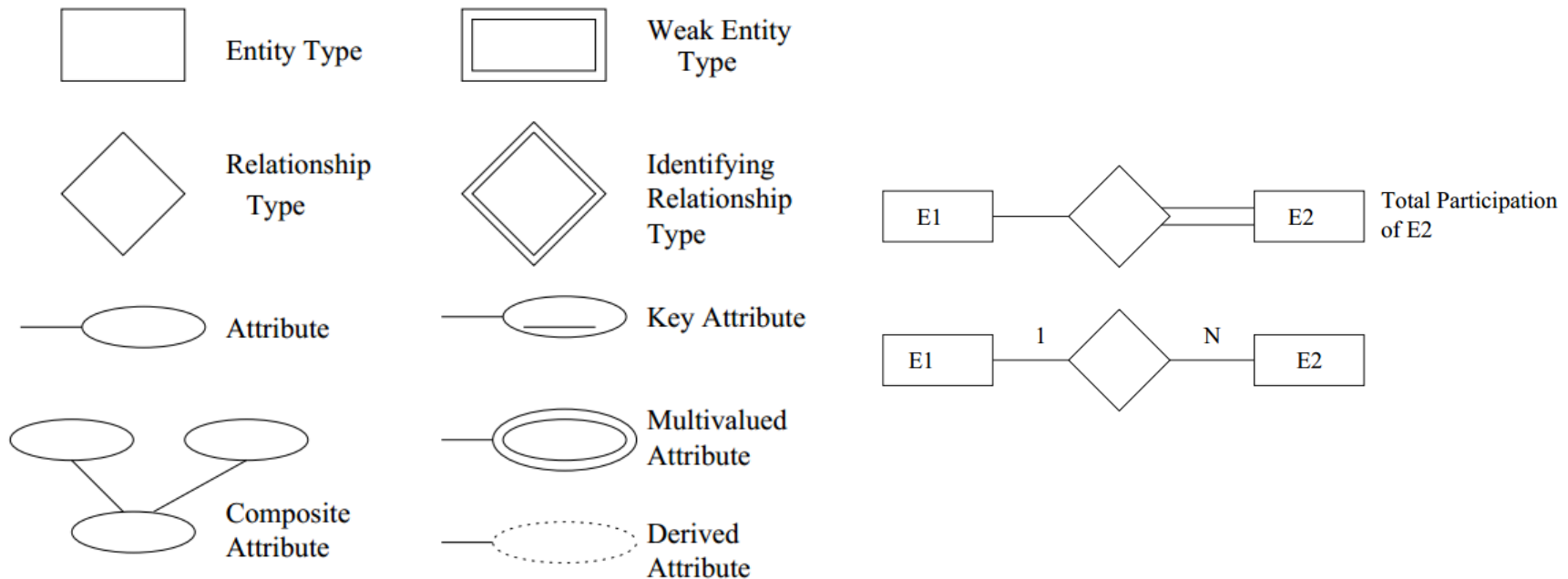
# 1.6 Enhanced ER (EER) model

- Designers must use additionally modelling concepts to

  - represent the requirements from applications as accurately and explicitly as possible.

1.6 Enhanced ER (EER) model(cont)

- There are many extensions to the ER model. We will look at one:

  - *Specialisation*: the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization.

  - *Generalisation*: a reverse process of specialisation.

- A subclass inherits all the attributes of the superclasses.

# 1.6 Enhanced ER (EER) model(cont)

- A specialisation involves the following aspects:

    - Define a set of subclasses of an entity type.

    - Associate additional specific attributes with each subclass.

    - Establish additional specific relationship types between each subclass and other entity types, or other subclasses.

- A subclass may have multiple superclasses.

- A specialisation:

    - may be either total or partial; and
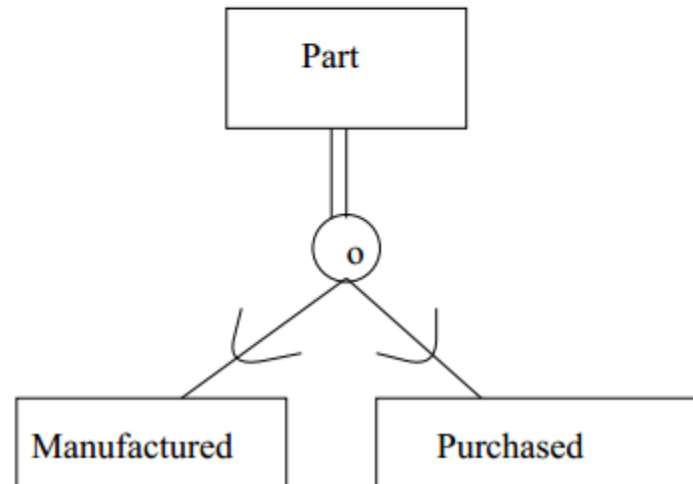
    - may be either disjoint or overlapping.
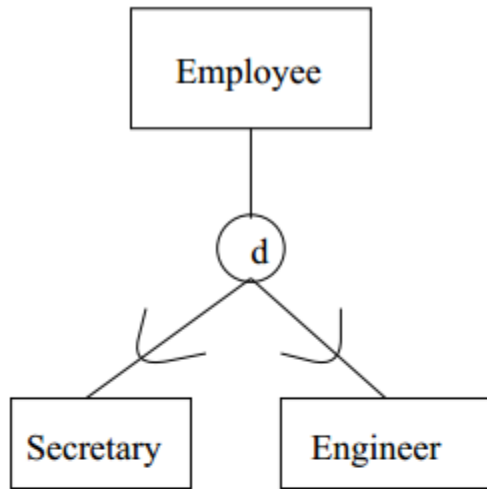
Partial disjoint
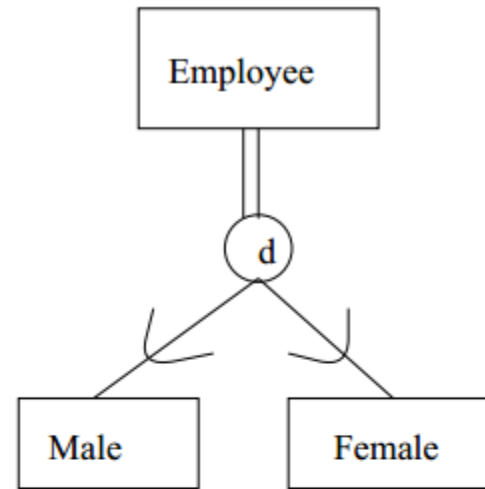
Total disjoint
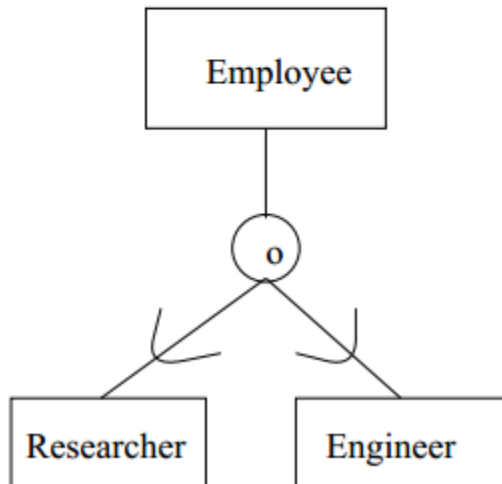
Partial Overlapping

Total Ovelapping

1.7 Design Principles

- Faithfulness: reflect reality.

- Avoid redundancy.

- Picking the right kind of element.