

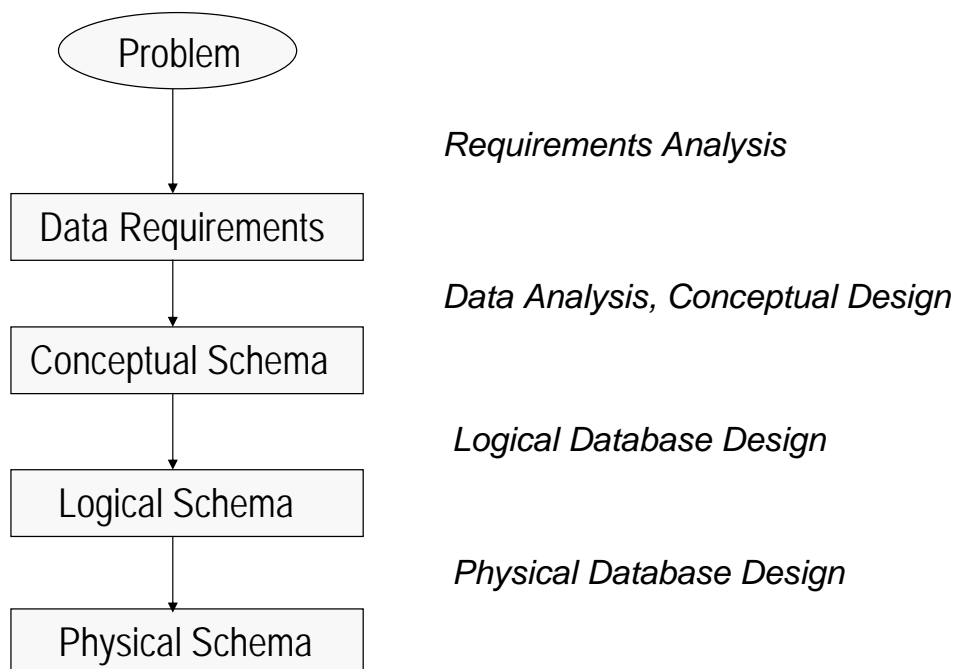
# ***Introduction to Database Systems***

## **Conceptual Modeling**

Werner Nutt

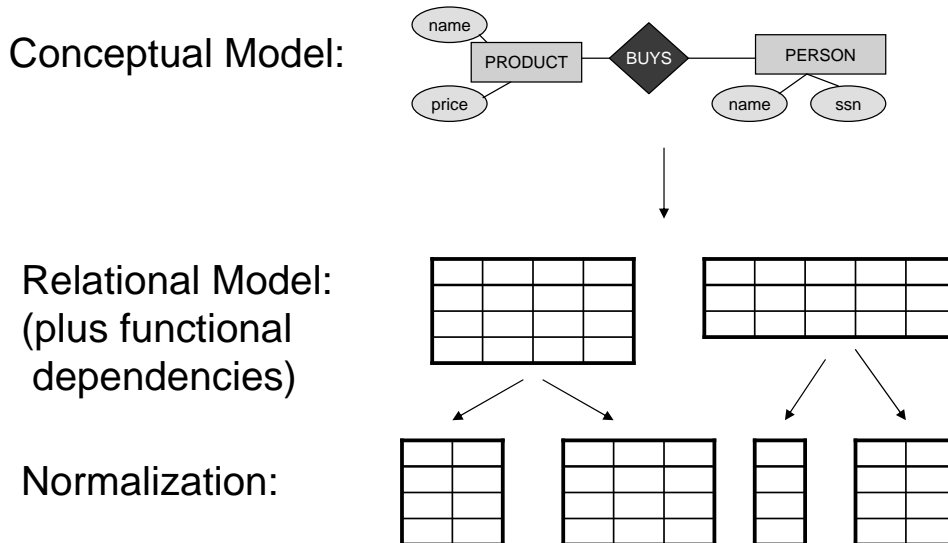
1

### **Database Design Goes Through Stages**



2

# Conceptual and Logical Design



3

## Toy Example: University Database

Example queries we may want to ask:

- Which are the given and the family name of the student with student number 5432?
- How many students are enrolled for the course “Introduction to Databases”?
- For which courses is Georg Egger enrolled?
- Which machine equipment is used for Introduction to Databases?

4

## More Example Queries

- At which school and for which course is Georg Egger enrolled?
- For which students does Prof. Rossi act as a tutor and at which times does he meet with his students?
- Is there a professor who is tutoring students that do not attend any of the professor's courses?

5

## University Data Requirements

- A student has a name, which consists of a given name and a family name, and a student ID. Each student is uniquely identified by his/her student ID.
- A course has a subject and a course ID. For each course, we want to record the number of students taking that course and the type of equipment being used for the course. A course is uniquely identified by its course ID.
- A student can be enrolled in an arbitrary number of courses, and an arbitrary number of students can be enrolled in a course. For each course in which they are enrolled students receive a lab mark and an exam mark.
- A course cannot exist if there is no student enrolled in it.
- A school is distinguished by the honour's degree that it awards. We also want to record to which faculty a school belongs. A student is registered with at most one school, while a school can have an arbitrary number of students.

6

## University Data Requirements (cntd.)

- A student is also registered for a year of study. An year of study is identified by a number between 1 and 4. A student is registered for only one year of study, but each cohort can have many students.
- For each member of staff we want to record their name and their room number. A member of staff is identified by the combination of these two pieces of data. Staff are appraised by other staff. A member of staff has no more than one appraiser.
- Students can be allocated to a member of staff as their tutor. A student can have no more than one tutor. The tutor and the student agree upon a time slot for regular meetings.
- For each year of study, there is one member of staff who acts as the year tutor. A member of staff can only be responsible for one year of study. Students can be registered for a year of study.
- Courses are taught by members of staff. A course can have several teachers, and a staff member can teach several courses.

7

## Conceptual Design with the ER Model

The questions to ask:

- What are the *entities* (= objects, individuals) in the organization?
- Which *relationships* exist among the entities?
- What information (= *attributes*) do we want to store about these entities and relationships?
- What are the business rules of the organization?
- Which integrity constraints do arise from them?

The answers are represented in an  
Entity Relationship Diagram (ER diagram)

8

# Entities and Entity Sets/Types

**Entity:** An object distinguishable from other objects  
(e.g., an employee)

- An entity is described by a set of attributes.

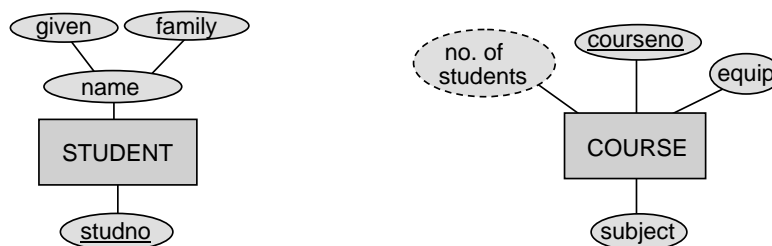
*Examples of entities?*

*Examples of things that are not entities?*

**Entity Set/Entity Type:** A collection of similar entities  
(e.g., all employees)

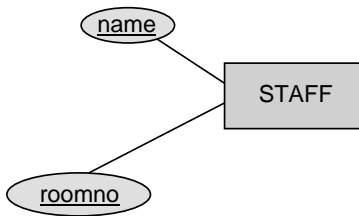
- All entities in an entity set have the same set of *attributes*.
- Each attribute has a *domain*.
- Each entity set has a *key*  
(i.e., one or more attributes whose values uniquely identify an entity) <sup>9</sup>

## Graphical Representation of Entity Sets



- Entity Sets are drawn as rectangles
- Attributes are drawn using ovals
- Simple attributes contain atomic values
- Composite attributes combine two or more attributes
- Derived attributes are indicated by dashed lines
- The attributes making up the key are underlined

# Composite Keys



- Some entities cannot be uniquely identified by the values of a single attribute ...
  - ... but may be identified by the combination of two or more attribute values
- ➔ several attributes together make up a compound key

11

# Relationships and Relationship Sets/Types

**Relationship:** An association between two or more entities (e.g., “Joe Smith” is “enrolled” in “CS123”)

- Relationships may have attributes

*Examples of relationships?*

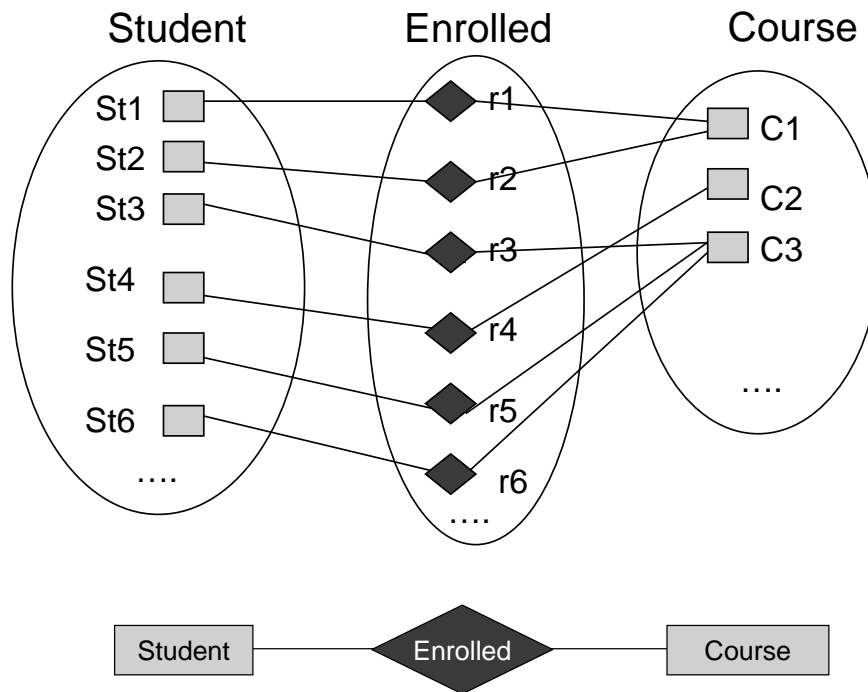
**Relationship Set/Type:** A collection of similar relationships

- An  $n$ -ary relationship type relates  $n$  entity types  $E_1, \dots, E_n$
- Each relationship involves  $n$  entities  $e_1 \in E_1, \dots, e_n \in E_n$

*Examples of relationship sets?*

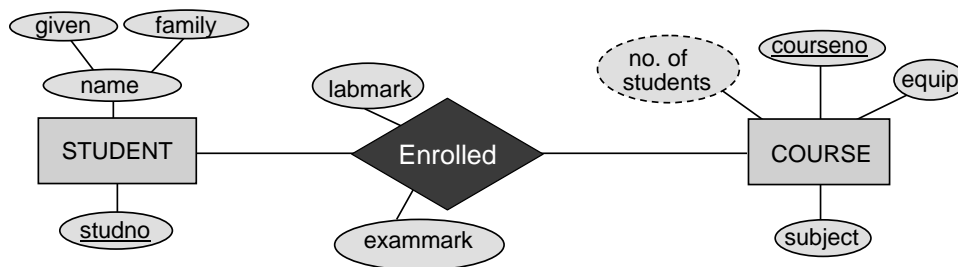
12

# An Instance of a Relationship Type



13

# Graphical Representation of Relationship Types



- Relationship sets are drawn as diamonds

*How many labmarks can a student have?*

14

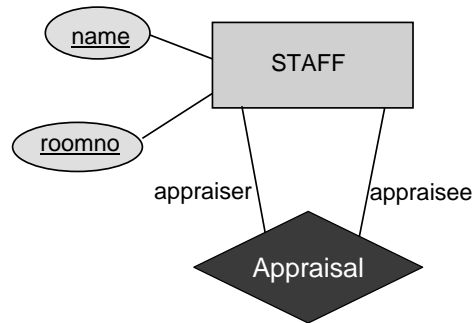
# Roles and Recursive Relationships

An entity type can

- participate in several relationship sets

and

- participate more than once in one relationship set (taking on different “roles”)

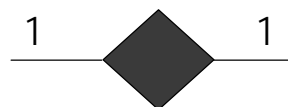
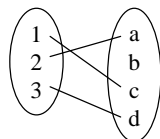


*Which are other examples of recursive relationships?*

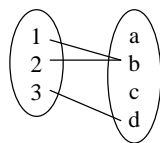
15

# Multiplicity (cardinality) of Relationship Types

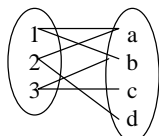
- one-one:



- many-one:



- many-many:



*Sometimes the letters  $m$ ,  $n$  are used to indicate the “many” side of relationships.*

16

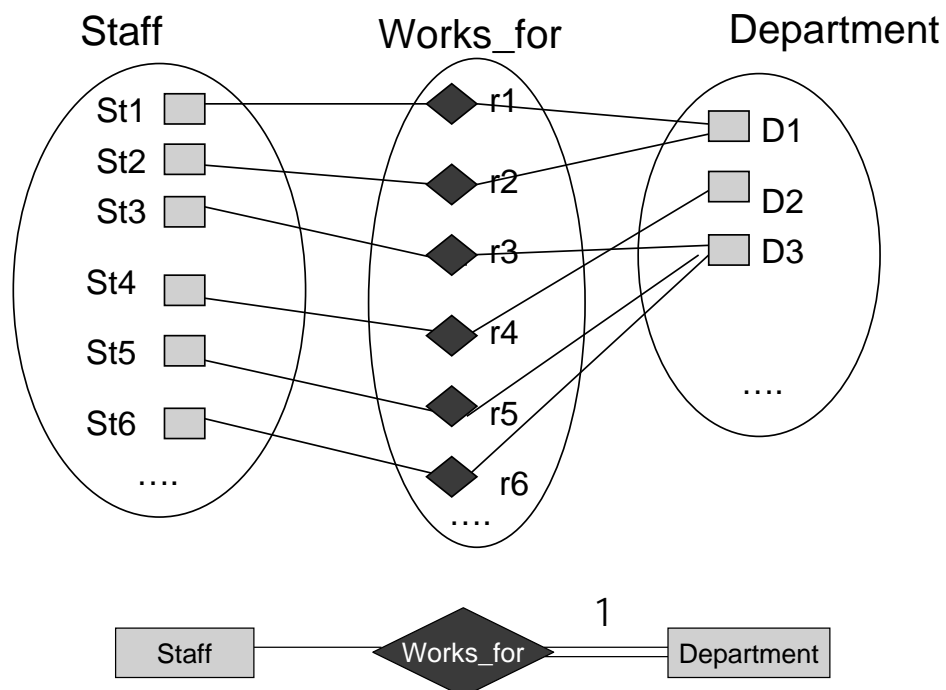


# Participation Constraints

- Participation constraints specify whether or not an entity must participate in a relationship set
- When there is no participation constraint, it is possible that an entity will not participate in a relationship set
- When there is a participation constraint, the entity must participate *at least once*
- Participation constraints are drawn using a *double line* from the entity set to the relationship set

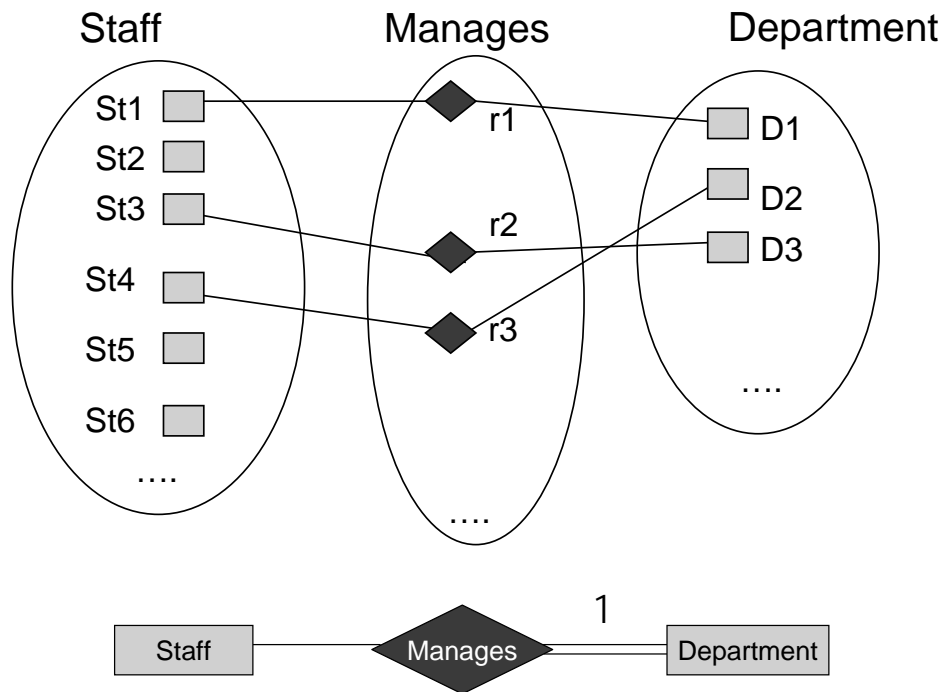
17

# Mandatory Participation



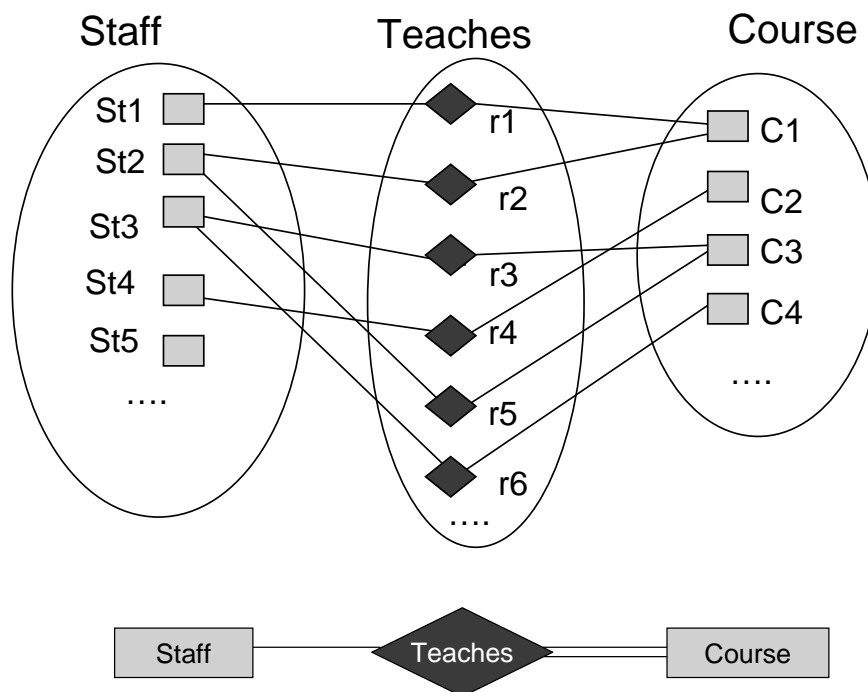
18

# Optional and Mandatory Participation



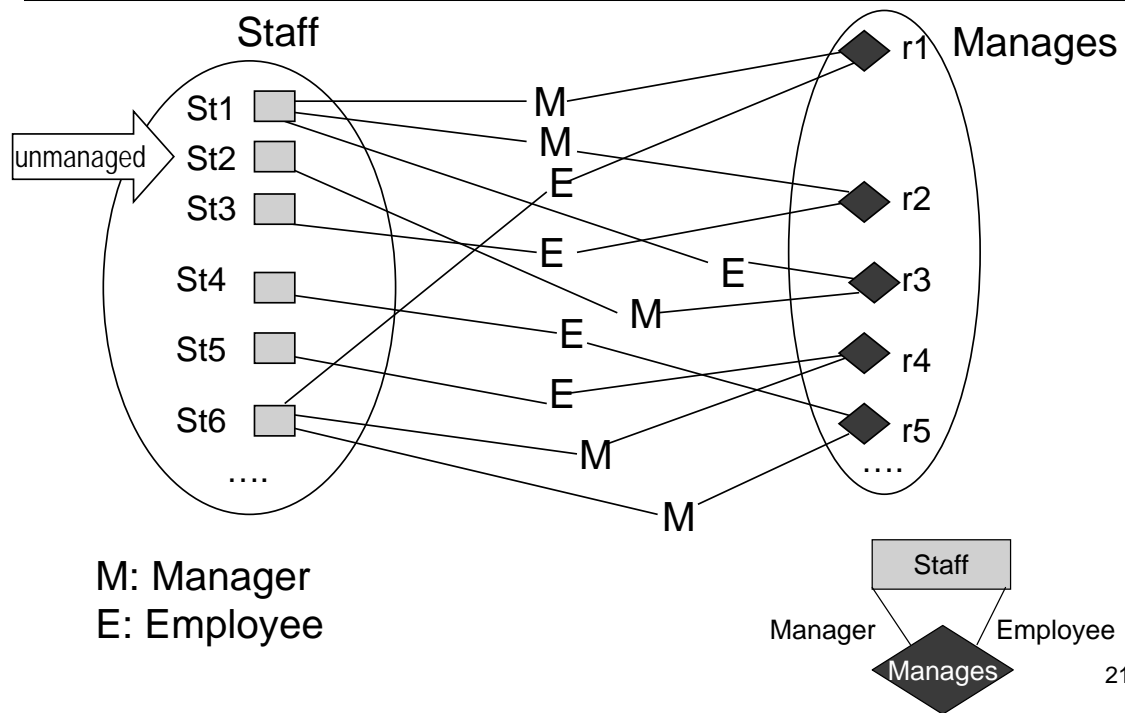
19

# Many:many Relationship Type with Optional and Mandatory Participation



20

## Recursive Relationship Type with Optional Participation



## Summary: Properties of Relationship Types

### Degree

- The number of participating entity types

### Cardinality ratios

- The number of instances of each of the participating entity types which can partake in a single instance of the relationship type:

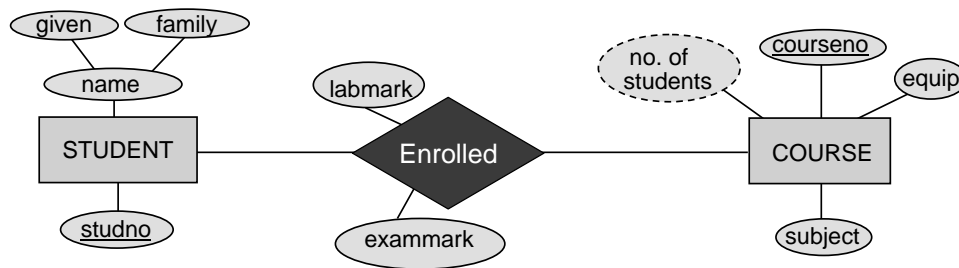
*1:1, 1:many, many:1, many:many*

### Participation

- Whether an entity instance has to participate in a relationship instance
- Represented with a double line

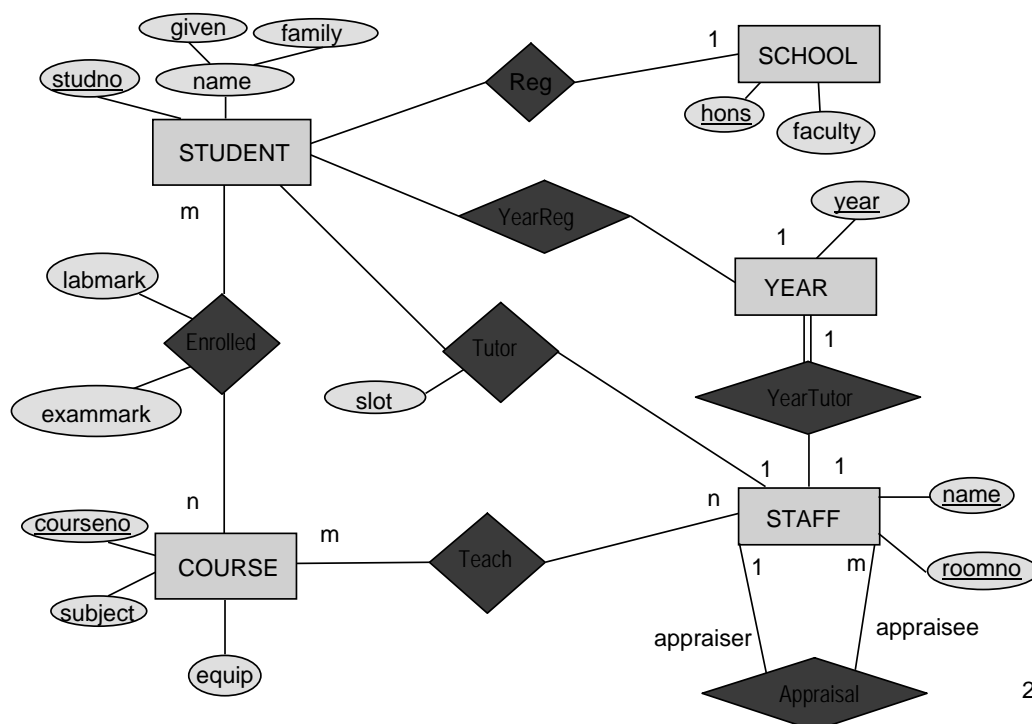
# Attributes in ER Modeling

- For every attribute we define
  - *Domain or data type*
  - *Format, i.e., composite or atomic*
  - whether it is derived
- Every entity type must have as *key* an attribute or a set of attributes



23

# ER Model of the University DB



24

## Exercise: Supervision of PhD Students

A database needs to be developed that keeps track of PhD students:

- For each student store the name and matriculation number. Matriculation numbers are unique.
- Each student has exactly one address. An address consists of street, town and post code, and is uniquely identified by this information.
- For each lecturer store the name, staff ID and office number. Staff ID's are unique.
- Each student has exactly one supervisor. A staff member may supervise a number of students.
- The date when supervision began also needs to be stored.

25

## Exercise: Supervision of PhD Students

- For each research topic store the title and a short description. Titles are unique.
- Each student can be supervised in only one research topic, though topics that are currently not assigned also need to be stored in the database.

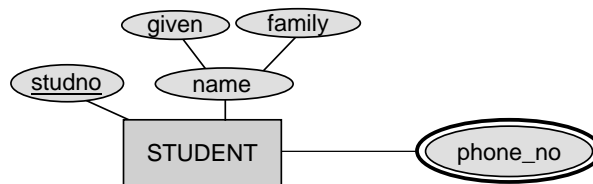
Task:

Design an entity relationship diagram that covers the requirements above. Do not forget to include cardinality and participation constraints.

26

## Multivalued Attributes

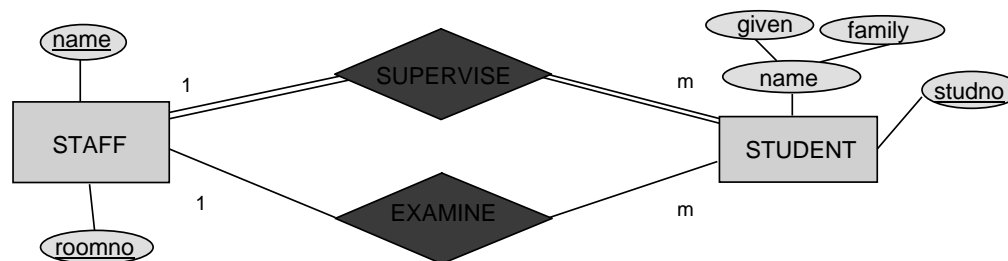
- Students often have more than one phone number (home, student hall, mobile)
- There is no additional information, other than the number, we need to store
- This captured by a multivalued attribute
- Notation: double-lined oval



27

## Roles in Non-recursive Relationships

Also in non-recursive relationships we may annotate relationship links with the roles that entities play in the relationship

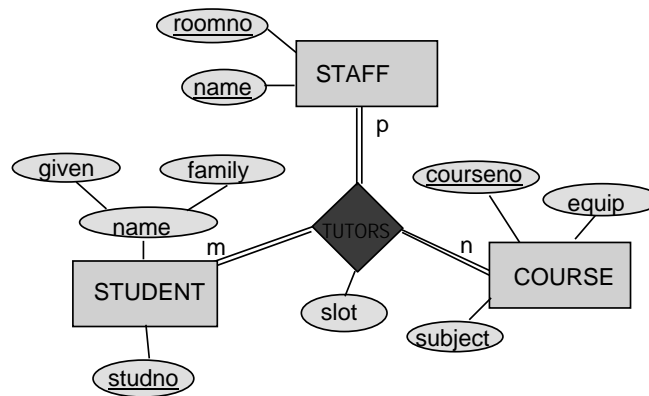


*What would be appropriate roles in this example?*

28

## Multiway (non-binary) Relationship

Relationships can involve more than two entity types...



29

## Constraints: Definition

- A constraint is an assertion about the database that must be true at all times
- Constraints are part of the database schema

30

## Modeling Constraints

Finding constraints is part of the modeling process. They reflect facts that hold in the world or business rules of an organization.

Examples:

Keys: *codice fiscale* uniquely identifies a person

Single-value constraints: a person can have only one father

Referential integrity constraints: if you work for a company, it must exist in the database

Domain constraints: peoples' ages are between 0 and 150

Cardinality constraints: at most 100 students enroll in a course

31

## Keys

A key is a set of attributes that uniquely identify an object or entity:

Person: social-security-number (U.S.)  
national insurance number (U.K.)  
codice fiscale (Italy)  
name  
name + address  
name + address + dob

*(Why not "age"?)*

Perfect keys are often hard to find, so organizations usually invent something.

*Do invented keys (e.g., `course_id`) have drawbacks?* 32



## Variants of Keys

- Multi-attribute (composite) keys:
  - E.g. name + address
- Multiple keys:
  - E.g. social-security-number, name + address

33

## Existence Constraints

Sometimes, the existence of an entity of type X  
depends on the existence of an entity of type Y:

Examples:

- Book chapters presume the existence of a book
- Tracks on a CD presume the existence of the CD
- Orders depend on the existence of a customer

We call Y the *dominating* entity type and  
X the *subordinate* type

⇒

strong and weak entities

34

# Strong and Weak Entities

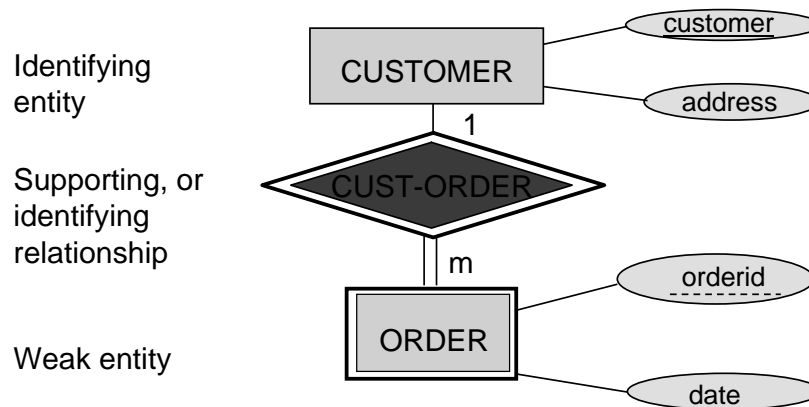
Dominating and subordinate types are modeled as

- Entities

(also “strong”, or “identifying” entities)

and

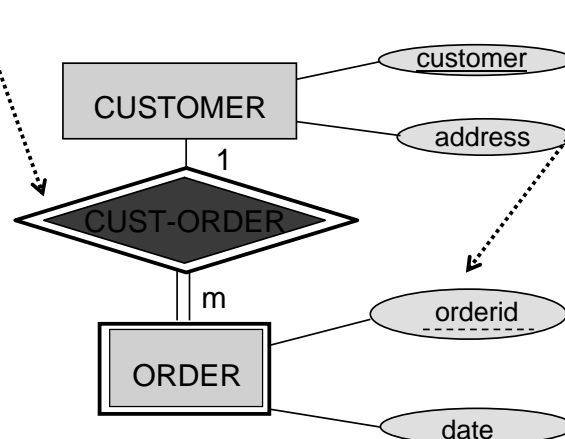
- Weak entities



35

# Strong and Weak Entities (Identifier Dependency)

- A *strong* entity type has an identifying primary key
- A *weak* entity’s key comes not (completely) from its own attributes, but from the keys of one or more entities to which it is linked by a *supporting many-one relationship*
- A *weak* entity type does not have a primary key but does have a *discriminator*



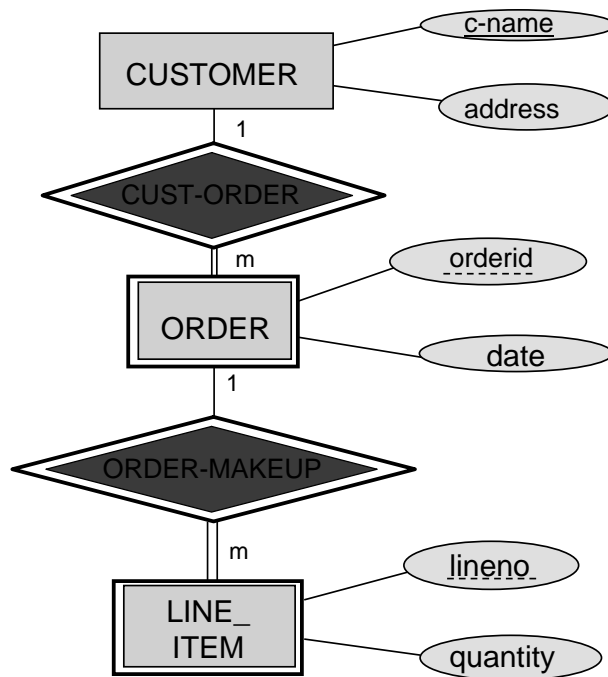
36

## Weak Entities May Depend on Other Weak Entities

- Strong entity type
- Identifying entity for ORDER
- Identifying entity for LINE\_ITEM

- Weak entity
- Identifying entity for LINE\_ITEM

- Weak entity type



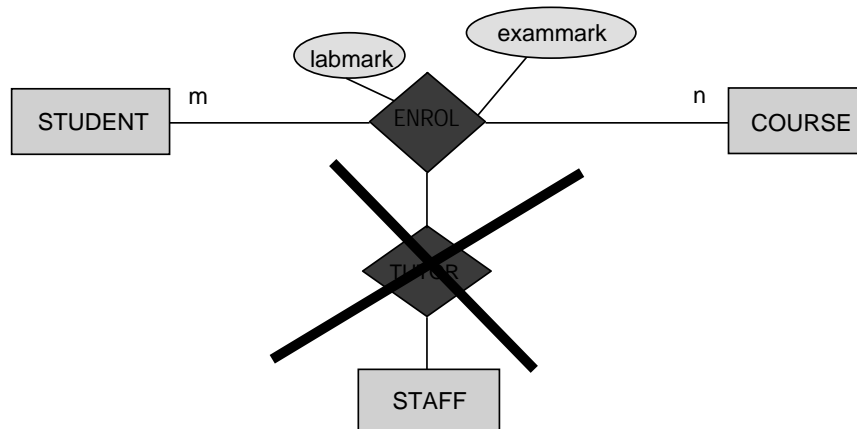
37

## Turning Relationships into Entities

Relationship types are less natural if

- the relationships have many attributes, or
- we want to model a relationship with that relationship type

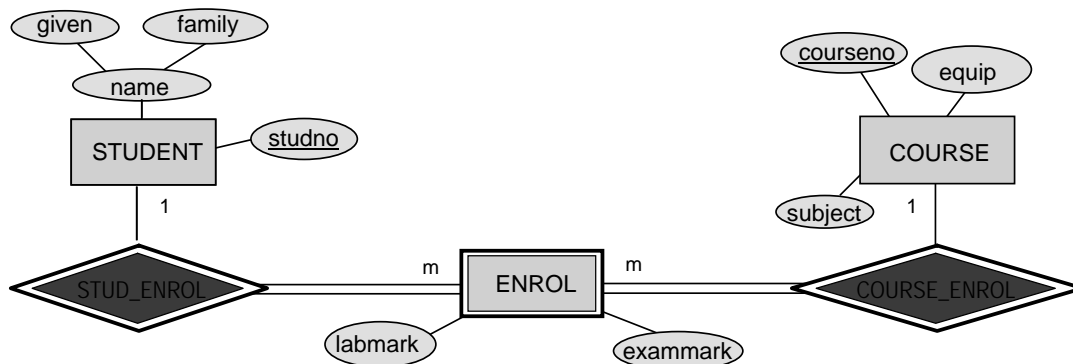
**Example:**



38

# Association Entity Types

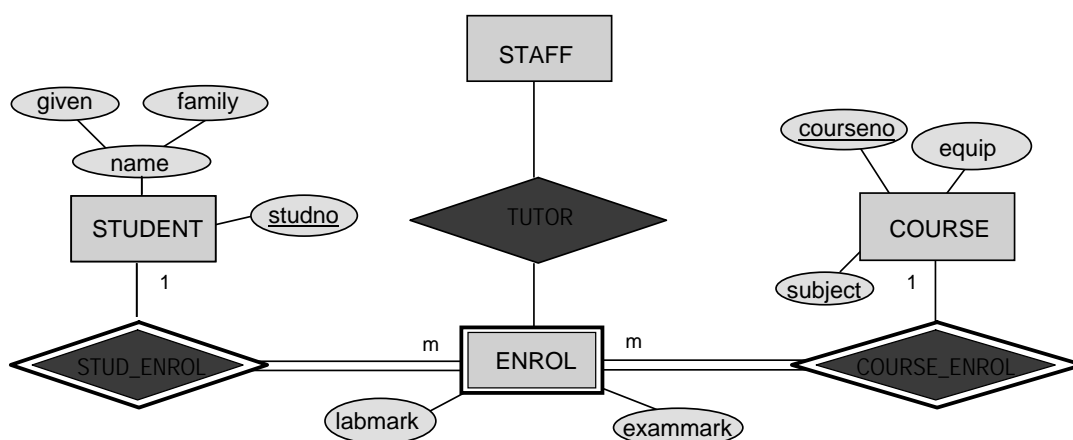
An entity type that represents a relationship type:



- The association entity type is a subordinate type
- The participating entity types become dominating types
- Attributes of the relationship become attributes of the entity
- Relationships with the dominating entity types are many-one and mandatory

39

## How Does This Solve Our Problem?



Association entity types can participate in any relationship type

40

# Design Principles for ER Modeling

There are usually several ways to model a real world concept, e.g.:

- entity vs. attribute
- entity vs. relationship
- binary vs. ternary relationships, etc.

Design choices can have an impact on

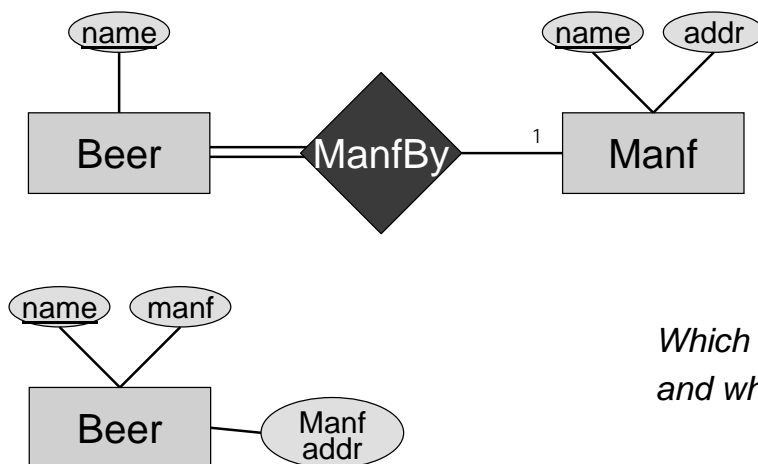
- redundancies among the data that we store
- integrity constraints captured by the database structure

41

## “Don’t Say the Same Thing More Than Once”

Redundancy wastes space and encourages inconsistency

Example:

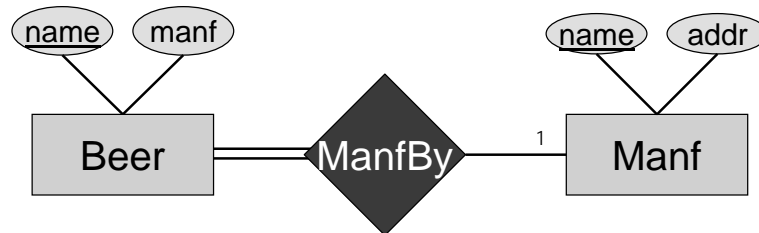


*Which is good design,  
and which is bad?*

*Why?*

42

## And What About This?



43

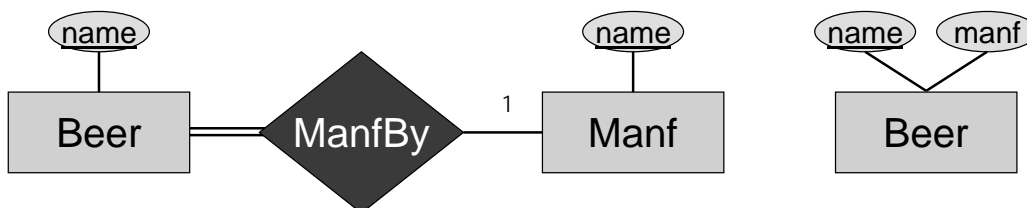
## Entities Vs. Attributes

Sometime it is not clear

- which concepts are worthy of being entities, and
- which are handled more simply as attributes

Example:

Which are the pros and cons of each of the two designs below?



44

## Entity Vs. Attribute: Rules of Thumb

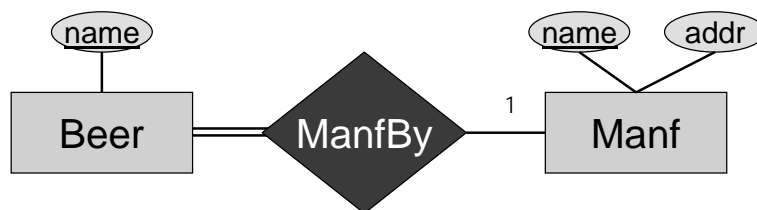
Only make an entity if either:

1. It is more than a name of something; *i.e.*, it has non-key attributes or relationships with a number of different entities, or
2. It is the “many” in a many-one relationship

45

## Entity Vs. Attribute: Example

The following design illustrates both points:



- *Manfs* deserves to be an entity because we record *addr*, a non-key attribute
- *Beers* deserves to be an entity because it is at the “many” end
- If not, we would have to make “set of beers” an attribute of *Manfs*

46

## Hints for ER Modelling

- Identify entity types by searching for nouns and noun phrases
- Assume all entities are strong and check for weak ones on a later pass
- You need an identifier for each strong entity
- Assume all relationships have optional participation and check for mandatory (total) ones on a later pass
- Expect to keep changing your mind about whether things are entities, relationships or attributes
- Keep the level of detail relevant and consistent  
(for example leave out attributes at first)
- Approach diagram through different views ...  
... and merge them

47

## Use the Schema to Enforce Constraints

- The conceptual *schema* should enforce as many constraints as possible
- Don't rely on future data to follow assumptions

Example:

If the university wants to associate only one instructor with a course,

- don't allow sets of instructors and
- don't count on departments to enter only one instructor per course

48



## Exercise: A Record Company Database

- A record company wishes to use a computer database to help with its operations regarding its performers, recordings and song catalogue.
- Songs have a unique song number, a non-unique title and a composition date. A song can be written by a number of composers; the composer's full name is required. Songs are recorded by recording artists (bands or solo performers). A song is recorded as a track of a CD. A CD has many songs on it, called tracks. CDs have a unique record catalogue number, a title and must have a producer (the full name of the producer is required). Each track must have the recording date and the track number of the CD.
- A song can appear on many (or no) CDs, and be recorded by many different recording artists. The same recording artist might re-record the same song on different CDs. A CD must have only 1 recording artist appearing on it. CDs can be released a number of times, and each time the release date and associated number of sales is required.

49

## Superclasses and Subclasses: The Problem

Suppose we want to model that:

- Students can be either undergraduates or graduates, and every student belongs to one of these groups
- Among the university employees, there are academic, administrative, and technical staff  
(and there may be others)
- Only undergraduate students have tutors, which are academics

*How can we express this in ER diagrams?*

*How do we translate such diagrams into relations?*

50

## Superclasses and Subclasses: Specialisation and Generalisation

### Subclasses and Superclasses

- A subclass entity type is a specialized type of a superclass entity type
- A subclass entity type represents a subset or subgrouping of the superclass entity type's instances

Example: Undergraduates and postgraduates are subclasses of student

### Attribute Inheritance

- Subclasses inherit properties (attributes) of their superclasses

51

## Defining Superclasses and Subclasses

- Specialisation
  - The process of defining a set of more specialised entity types of an entity type
- Generalization
  - The process of defining a generalised entity type from a set of entity types

Two ways to define subclasses:

- *Predicate/Condition* defined classes
  - Entities that are members of a subclass are determined by a *condition on an attribute* value. All member instances of the subclass must satisfy the predicate.  
Example: first years and second year students are subclasses of undergraduates, defined by their year attribute.
- *User* defined classes
  - No condition for determining subclass membership

52

# Constraints on Specialisation and Generalization

## Disjointness

- *Overlap*
  - the same entity instance may be a member of *more than one* subclass of the specialisation
- *Disjoint*
  - the same entity instance may be a member of *only one* subclass of the specialisation

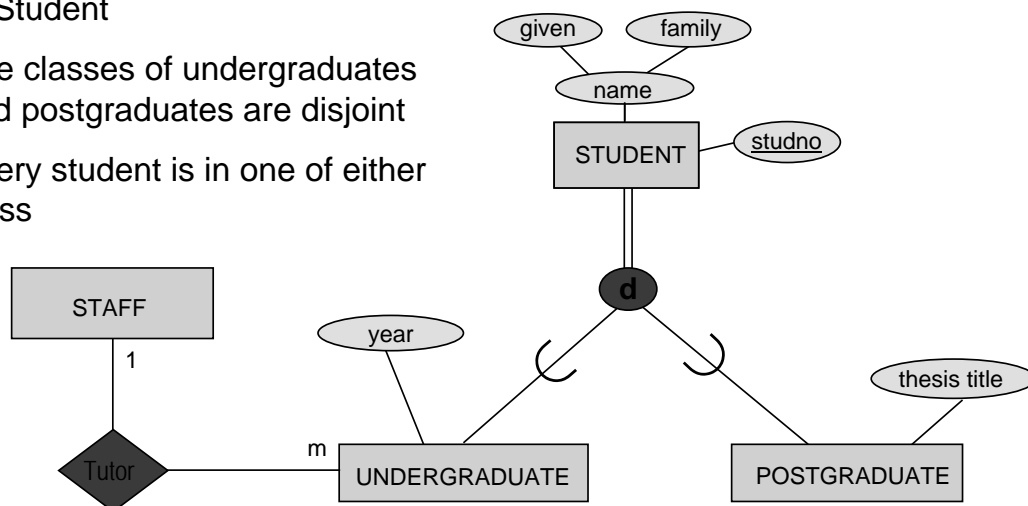
## Completeness

- *Total*
  - every entity instance in the superclass *must be* a member of some subclass in the specialisation
- *Partial*
  - an entity instance in the superclass need not be a member of any subclass in the specialisation

53

# Students are Undergraduates or Postgraduates

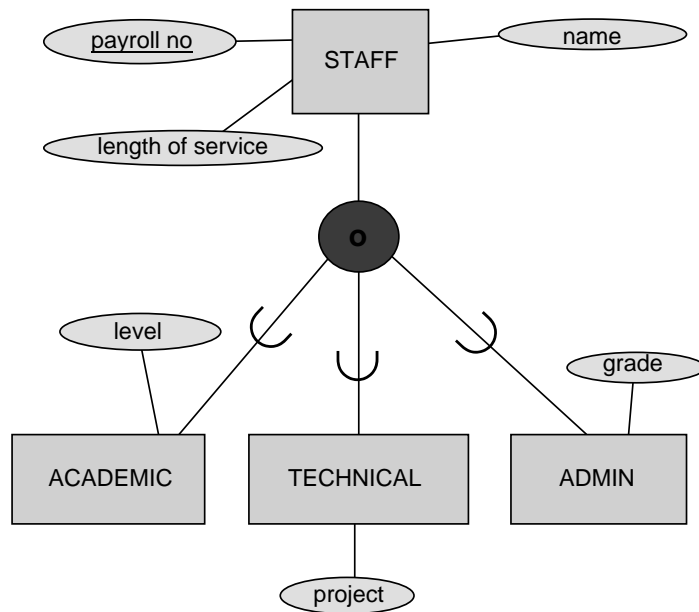
- Undergraduates and Postgraduates are subclasses of Student
- The classes of undergraduates and postgraduates are disjoint
- Every student is in one of either class



54

## Subclasses of Staff

- Academic, technical, and admin are three subclasses of staff
- The three classes may overlap



55

## Subclasses in the University Scenario

- Every person has a name and an address. A person is uniquely identified by their name.
- At a university, there are two groups of persons, employees and students. Every employee receives a salary, while every student pays a fee.
- Among the employees, there is research and teaching staff. An employee can belong to both groups.
- Among the teaching staff, there are lecturers and tutors. A tutor works for several courses.

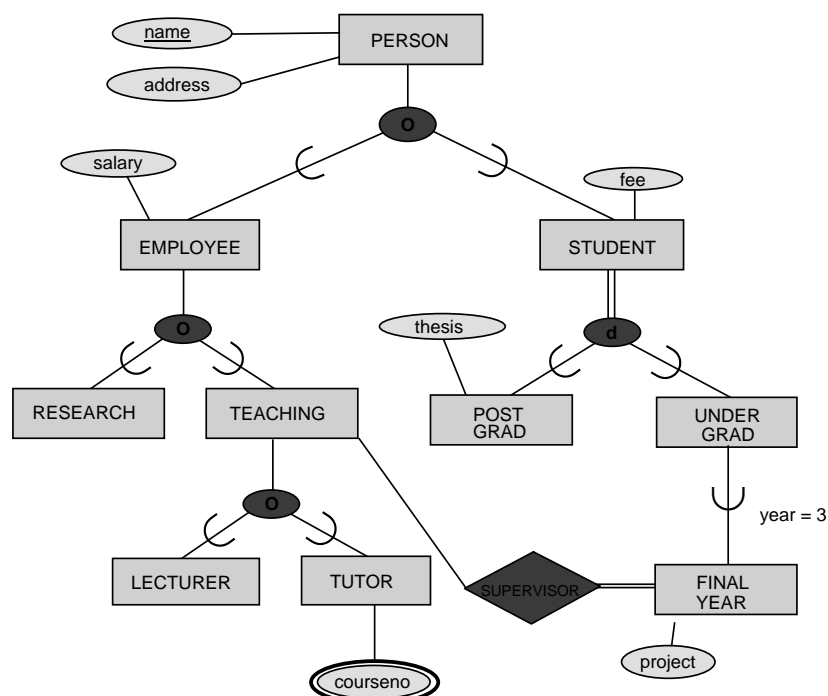
56

## Subclasses in the University Scenario

- Every student is either a postgraduate student or an undergraduate student.
- A postgraduate student has a thesis title, on which he/she is working.
- An undergraduate student is in the final year, if he/she is in his/her 3rd year.
- Every final year student is working on a project.
- Every final year student is supervised by a member of the teaching staff.

57

## Subclasses in the University Scenario



58

# References

In preparing these slides I have used several sources.  
The main ones are the following:

## Books:

- A First Course in Database Systems, by J. Ullman and J. Widom
- Fundamentals of Database Systems, by R. Elmasri and S. Navathe

Slides from Database courses held by the following people:

- Enrico Franconi (Free University of Bozen-Bolzano)
- Carol Goble and Ian Horrocks (University of Manchester)