# PART 2

DATABASE DESIGN

# Extended ER Features

features of EER :-

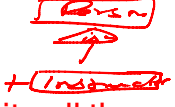
1) Specialization & generalization

2) Aggregation

# Extended E-R Features: Specialization

- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- The process of designating subgroupings within an entity set is called **specialization**.

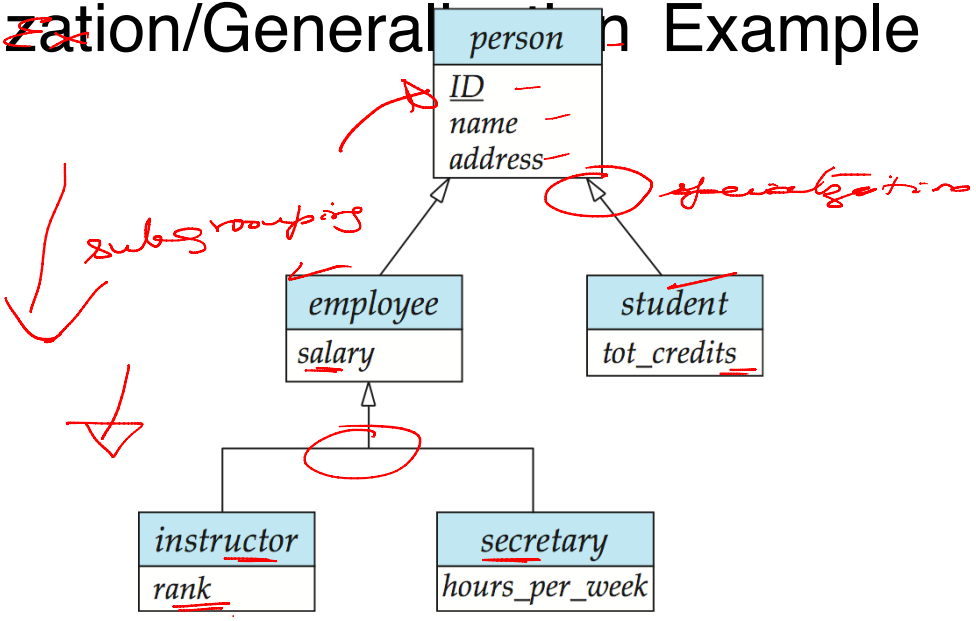
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (E.g., *instructor* "is a" *person*).

- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

- In terms of an E-R diagram, specialization is depicted by a hollow arrow-head pointing from the specialized entity to the other entity (see Figure 7.21).
- We refer to this relationship as the ISA relationship, which stands for "is a" and represents, for example, that an instructor "is a" employee.
- For an overlapping specialization (as is the case for *student* and *employee* as specializations of *person*), two separate arrows are used.
- For a disjoint specialization (as is the case for *instructor* and *secretary* as specializations of *employee*), a single arrow is used.

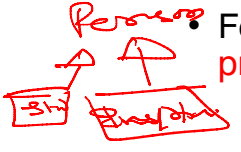
# Specialization/Generalization Example




Advantage ?

# Representing Specialization via Schemas

- Method 1:
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes



| schema | attributes |
|---|---|
| *person* | *ID, name, street, city* |
| *student* | *ID, tot_cred* |
| *employee* | *ID, salary* |

  - Drawback: getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

# Representing Specialization as Schemas (Cont.)

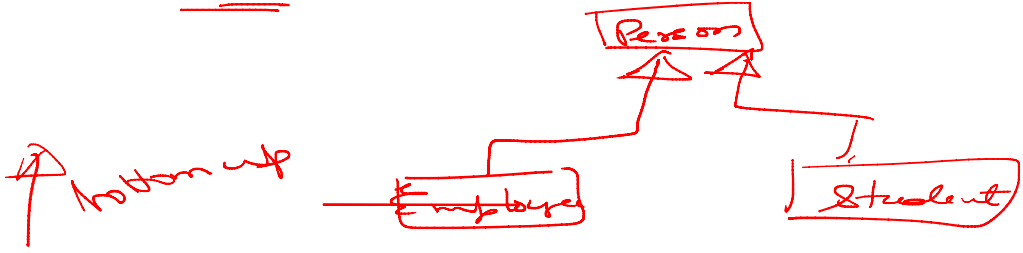
- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

| schema | attributes |
|---|---|
| *person* | *ID, name, street, city* |
| *student* | *ID, name, street, city, tot_cred* |
| *employee* | *ID, name, street, city, salary* |

  - If specialization is total, the schema for the generalized entity set (*person*) not required to store information
    - Can be defined as a "view" relation containing union of specialization relations
  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

# Extended ER Features: Generalization

- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way .
- The terms specialization and generalization are used interchangeably.



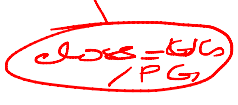
# Specialization and Generalization (Cont.)

- Can have multiple specializations of an entity set based on different features.
- E.g., *permanent_employee* vs. *temporary_employee*, in addition to *instructor* vs. *secretary*
- Each particular employee would be
  - a member of one of *permanent_employee* or *temporary_employee*,
  - and also a member of one of *instructor*, *secretary*
- The ISA relationship also referred to as **superclass - subclass** relationship

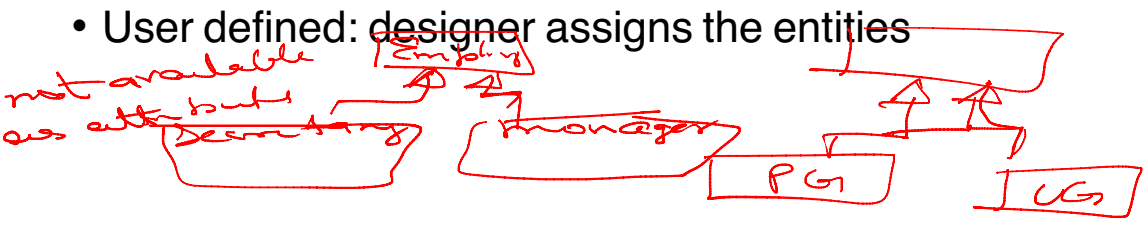
# Design Constraints on a Specialization/Generalization

Constraint 1: How to group entities?

- Conditioned defined: membership evaluated on the basis of if the entity has satisfied the condition

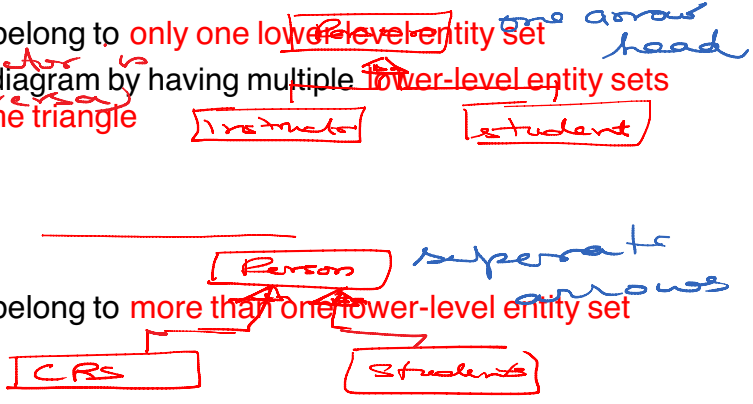




- User defined: designer assigns the entities



# Design Constraints on a Specialization/Generalization

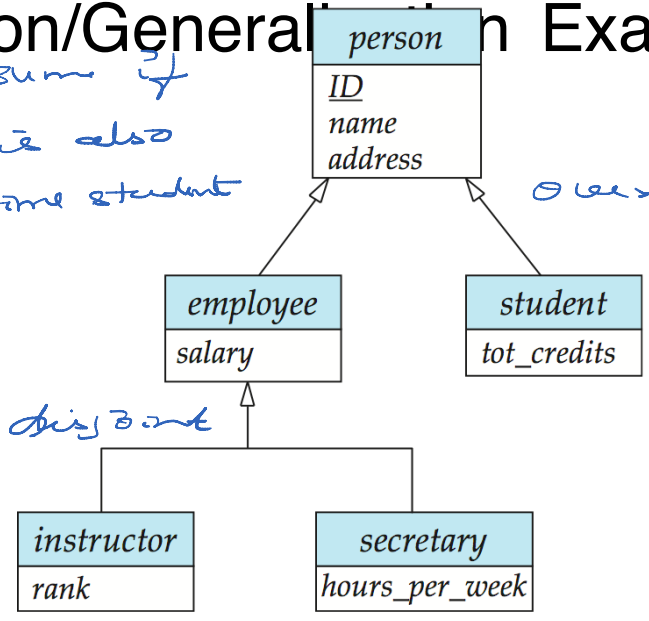
- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
  - **Disjoint**
    - an entity can belong to only one lower-level entity set
    - Noted in E-R diagram by having multiple lower-level entity sets link to the same triangle
  - **Overlapping**
    - an entity can belong to more than one lower-level entity set



# Specialization/Generalization Example

Note: Assume if employee is also a part-time student

overlapping



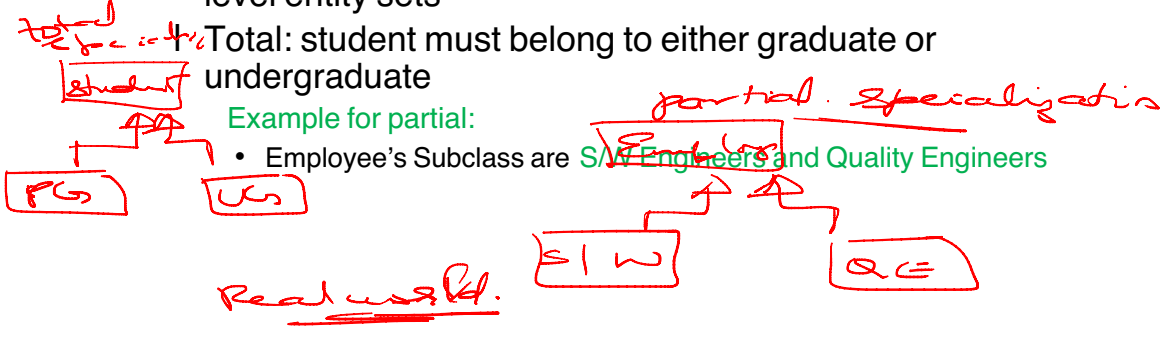

Advantage ?

# Design Constraints on a Specialization/Generalization (Cont.)

- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total**: an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong to one of the lower-level entity sets
    - Total: student must belong to either graduate or undergraduate

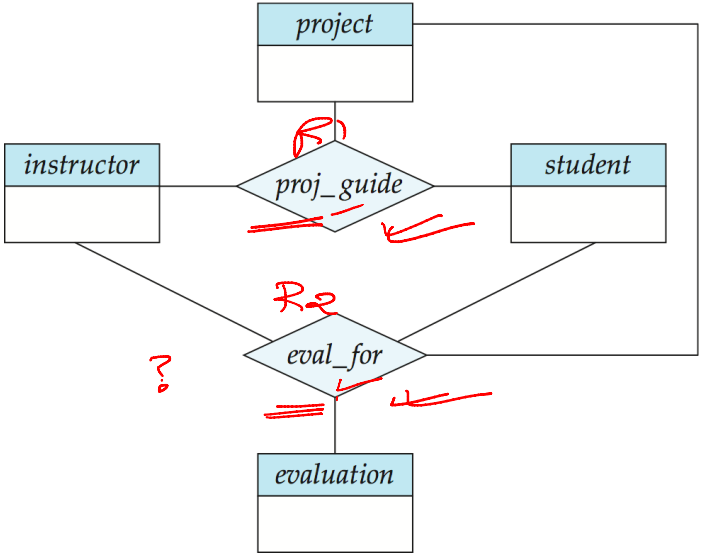
Example for partial:

- Employee's Subclass are S/W Engineers and Quality Engineers



# Aggregation

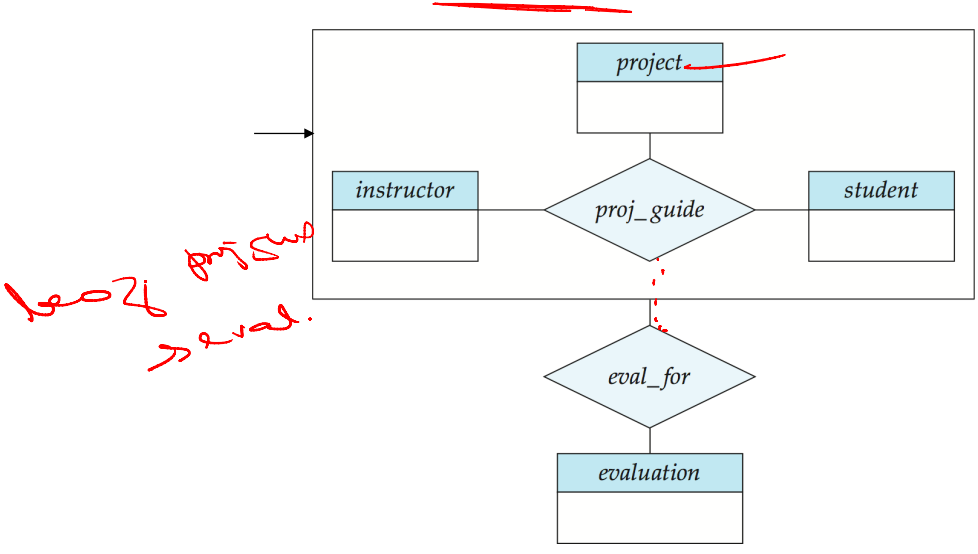
- Consider the ternary relationship *proj_guide*, which we saw earlier
- Suppose we want to record evaluations of a student by a guide on a project



# Aggregation (cont.)

- Without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation
  - Allows relationships between relationships

Abstract entity



- Aggregation is an abstraction through which relationships are treated as higher-level entities. Thus the relationship between entities and is treated as if it were an entity.

Represented by bounded rectangle around the entities and its associated relation

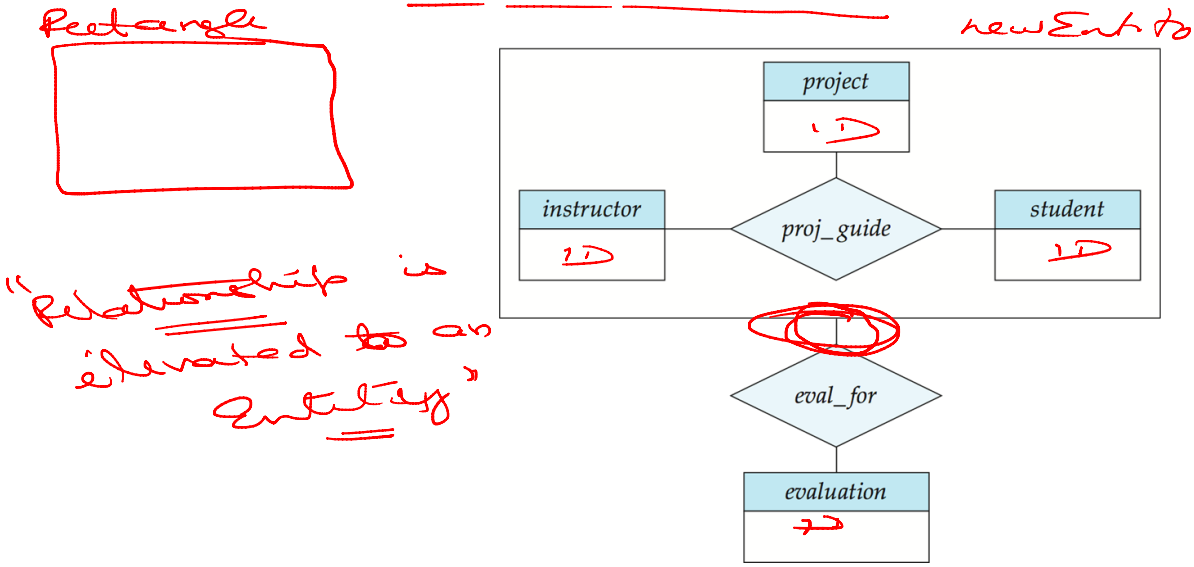
# Aggregation (Cont.)

- Relationship sets *eval_for* and *proj_guide* represent overlapping information
  - Every *eval_for* relationship corresponds to a *proj_guide* relationship
  - However, some *proj_guide* relationships may not correspond to any *eval_for* relationships
    - So we can't discard the *proj_guide* relationship
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

# Schemas Corresponding to Aggregation

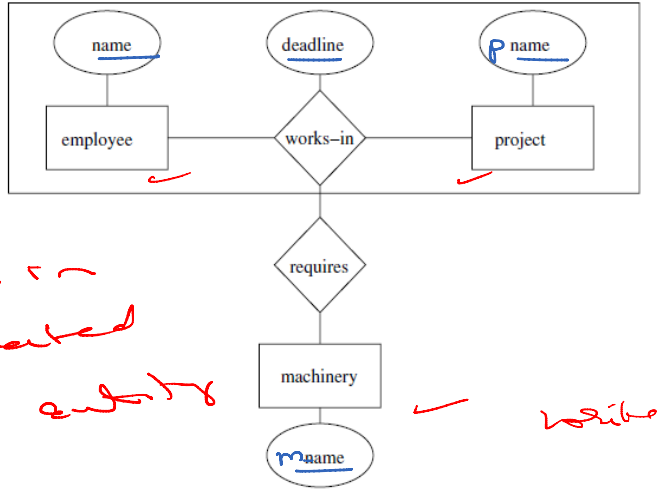
- To represent aggregation, create a schema containing
  - primary key of the aggregated relationship,
  - the primary key of the associated entity set
  - any descriptive attributes

*eval_for* (*s_ID, project_id, i_ID, evaluation_id*)



projectus

- Employee … a particular project us…



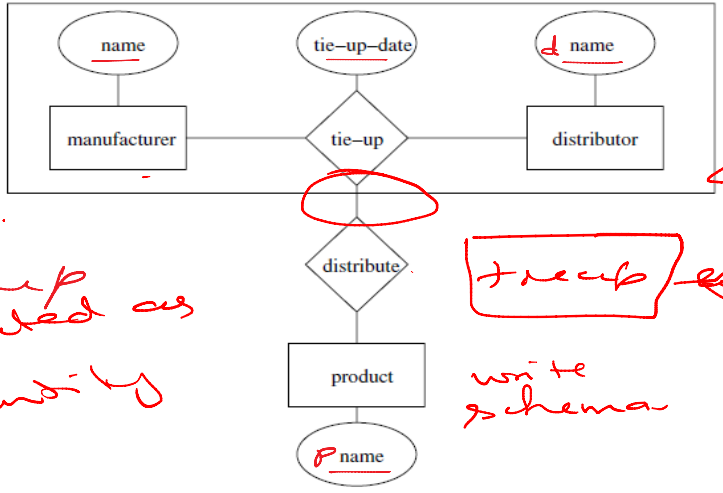

binary

works-in
elevated
as entity

table schema.

requires( name, deadline, pname, m_name) –

- Manufacturers have tie-ups with distributors to distribute products. Each tie-up is for a specific product, and there are to be distributed




Tie up derived as entity

Tie up + distributors

write schema

distribute (name, tieupdate, d name, P name)