Entity-Relationship Model Extended Features

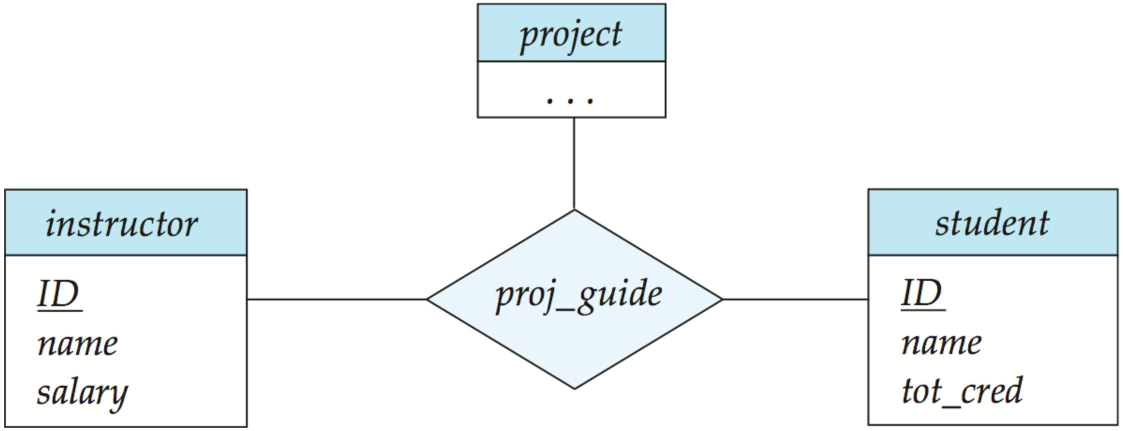
Database System Concepts, 6th Ed.

Outline

- Non-binary Relationships
- Extended E-R Features
- Design of the Bank Database
- Reduction to Relation Schemas
- Database Design
- UML

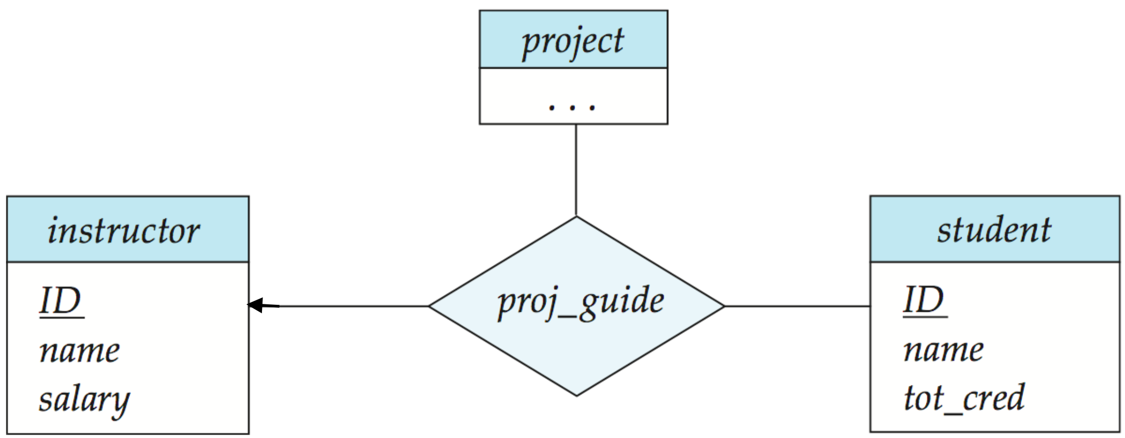
Non-binary Relationship Sets

- Most relationship sets are binary
- There are occasions when it is more convenient to represent relationships as non-binary.
- E-R Diagram with a Ternary Relationship



Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- For example, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project



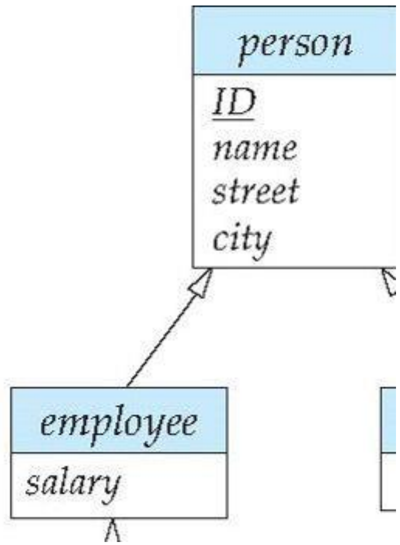
- More than one arrow is ambiguous.

IS-A: Specialization

- Consider a Person entity set

- Now suppose we want to model an Employee entity set and a Student entity set. However,
- We want to say that an “Employee is-a Person” and a “Student is-a Person”.
- That is, in addition to Person attributes, an Employee entity has an extra *salary* field.
- We depict this as follows.

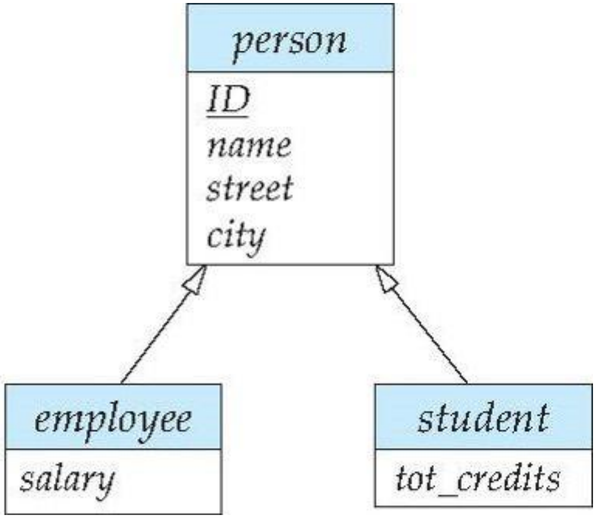
IS-A: representation



- This depicts that *employee* entity set *is a specialization of the person entity set.*
- An employee entity has all the attributes of person and in *addition has a salary attribute.*
- This is similar to **superclass-subclass** hierarchy in object-oriented langauges.
- Note the arrow from the specialized entity set to the general entity set.

IS-A: specialization

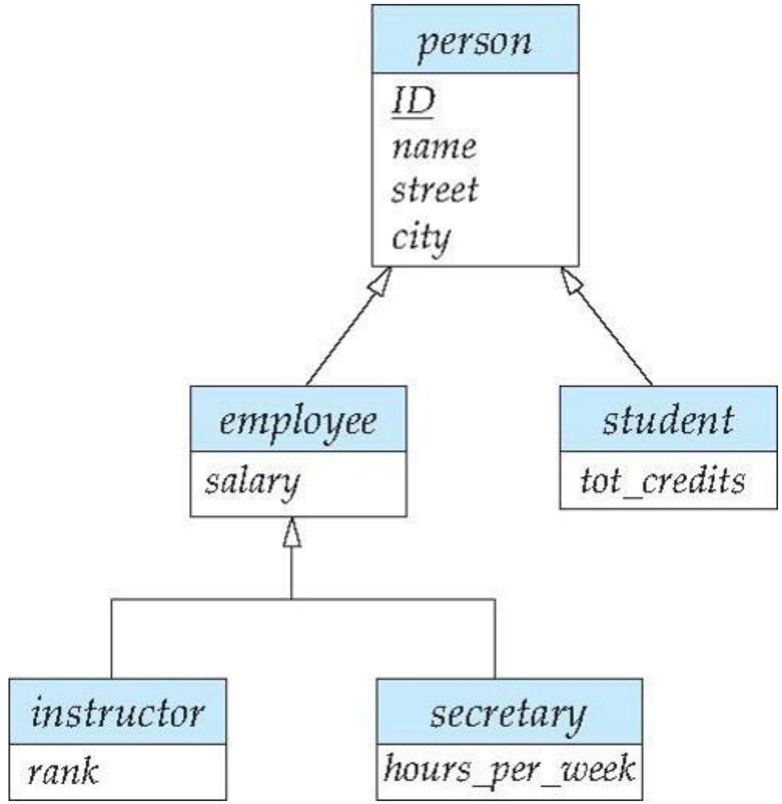
- Suppose we add another specialization called student.
- Student has an extra attribute *tot_credits*.
- Student is-a person. But a student may be or may not be an employee.
- Represented in the E-R diagram as:



- Arrow indicates that employee “IS-A” person and student “IS-A” person.

Further specialization

- Consider E-R diagram:



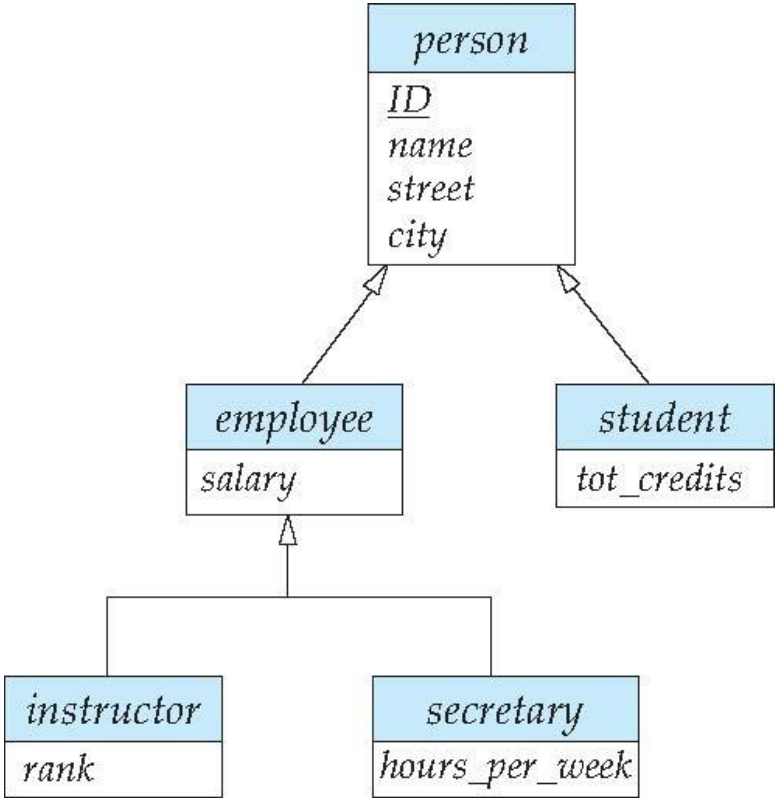
(overlapping specialization)

(disjoint specialization)

- *Instructor* “IS-A” employee, *secretary* “IS-A” *employee*,
- *Instructor* and *secretary* are **disjoint** entities.

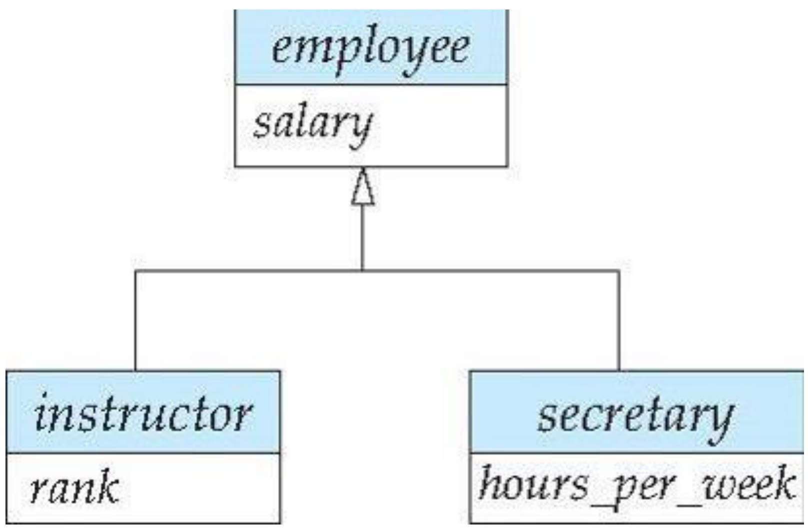
Specialization Example

- **Overlapping** – *employee* and *student*
- **Disjoint** – *instructor* and *secretary*
- Total and partial



Generalization

- Specialization is useful in top-down design approach.
- Generalization is the opposite of specialization.
- It merges together the common attributes of closely related entity sets into a higher-level entity set.
- A bottom-up design strategy.



Representing Specialization via Relation 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 (employee) 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

  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

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.

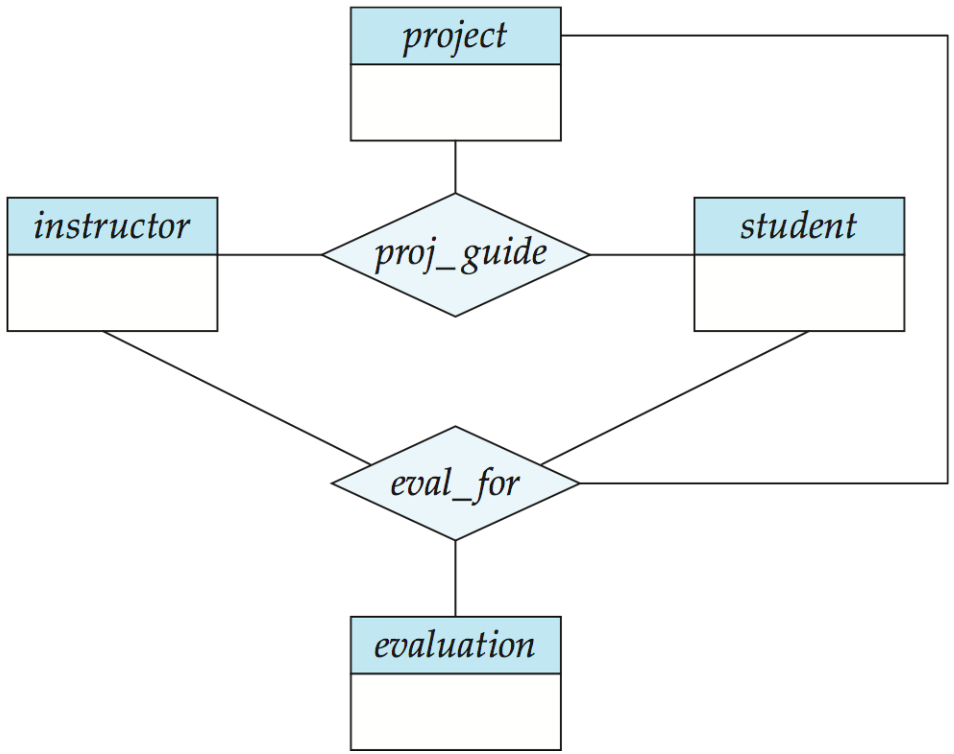
Design Constraints on a Specialization/Generalization

- **Completeness constraint for specialization/generalization:**
  - **total**: higher entity **must** belong to one of the lower-level entity sets.
  - E.g., *an employee* must be either an *instructor* or a *secretary*.
  - **partial**: higher level entity need not belong to one of the lower-level entity sets.
  - E.g., there can be *employees* that are neither *instructor* nor *secretary*.
- Partial generalization is the default.

- Specify total generalization in ER diagram by adding the keyword **total** in the diagram.

Aggregating a relationship:

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

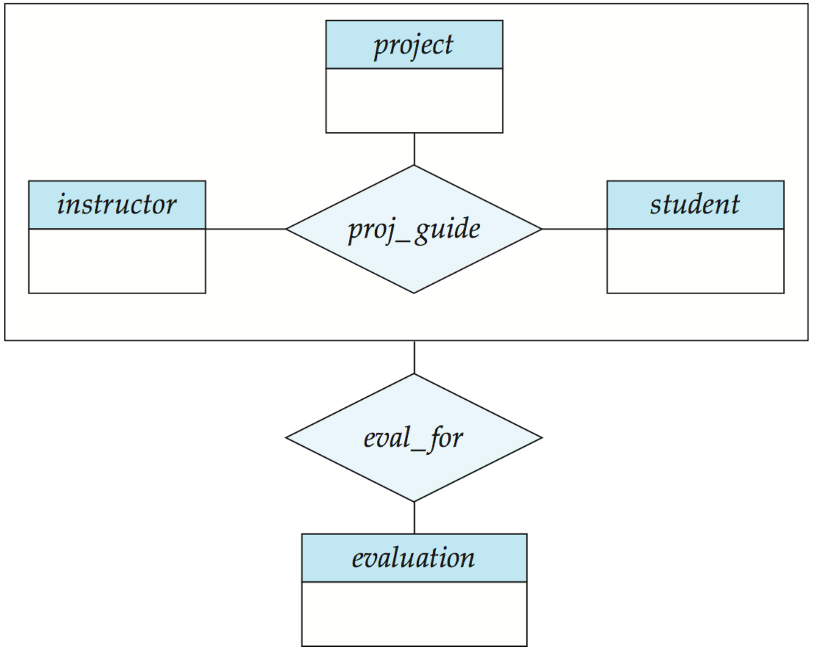


Aggregation (Cont.)

- 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

Aggregation (Cont.)

- Eliminate this redundancy via *aggregation* 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



Representing Aggregation via Schemas

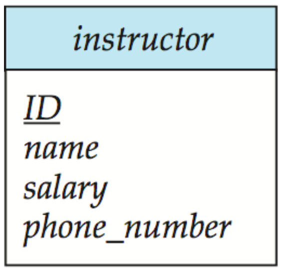
- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- E.g.
  - The schema *eval_for* is:

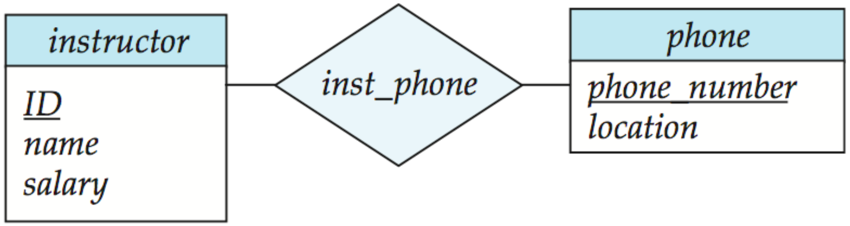
    eval_for (*s_ID, project_id, i_ID, evaluation_id*)
  - The schema *proj_guide* is redundant.

Design Issues

Entities vs. Attributes

- Use of entity sets vs. attributes



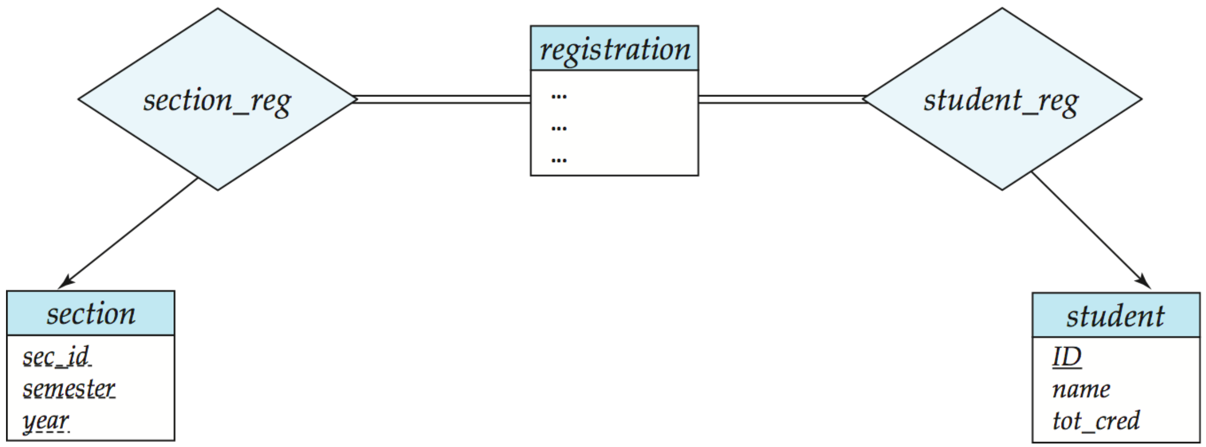


- Use of phone as an entity allows extra information about phone numbers (whether, office, residence, mobile etc.) and multiple phone numbers.

Entities vs. Relationship sets

- **Use of entity sets vs. relationship sets**

  Possible guideline is to designate a relationship set to describe an action that occurs between entities



- **Placement of relationship attributes**

  For example, attribute date as attribute of advisor or as attribute of student

Binary Vs. Non-Binary Relationships

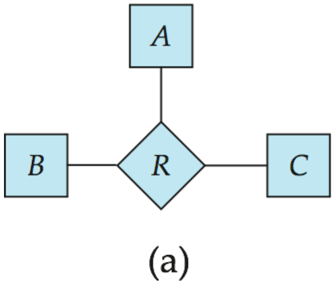
- It is possible to replace any non-binary (*n*-ary, for *n* > 2) relationship set by a number of distinct binary relationship sets.
- However: an *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*

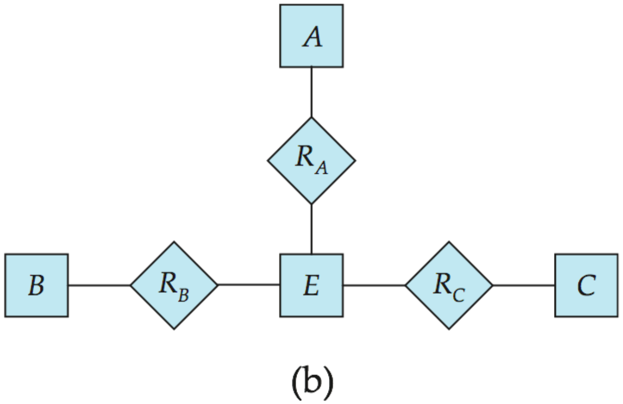
Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:

    1. R_A, relating *E* and *A* 2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*
  - Create an identifying attribute for *E and* add any attributes of *R* to *E*
  - For each relationship (a_i , b_i , c_i) in *R,* create

    1. a new entity e_i in the entity set *E* 2. add (e_i , a_i) to R_A
    3. add (e_i , b_i) to R_B 4. add (e_i , c_i) to R_C





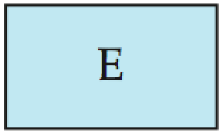
Converting Non-Binary Relationships (Cont.)

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of *R*
    - Exercise: *add constraints to the relationships* R_A, R_B *and* R_C to ensure that a newly created entity corresponds to exactly one entity in each of entity sets *A, B* and *C*
  - We can avoid creating an identifying attribute by making E a weak entity set (described shortly) identified by the three relationship sets

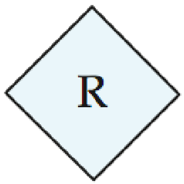
E-R Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

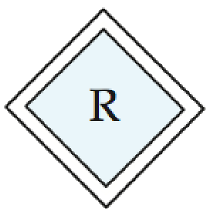
Summary of Symbols Used in E-R Notation



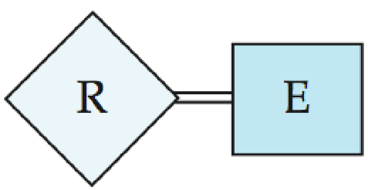
entity set



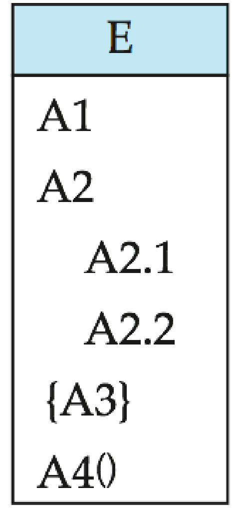
relationship set



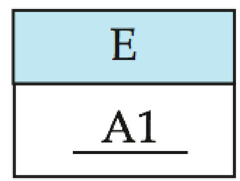
identifying
relationship set
for weak entity set



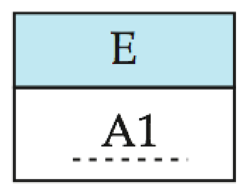
total participation
of entity set in
relationship



attributes:
simple (A1),
composite (A2) and
multivalued (A3)
derived (A4)

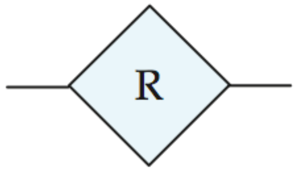


primary key

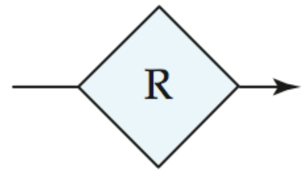


discriminating
attribute of
weak entity set

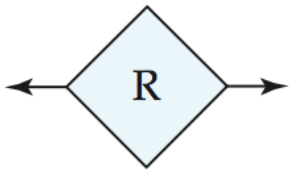
Symbols Used in E-R Notation (Cont.)



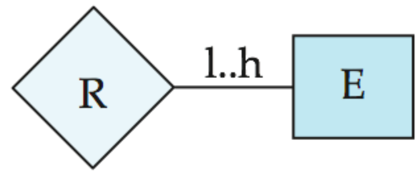
many-to-many relationship



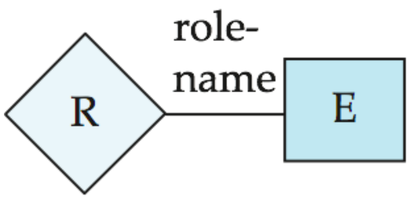
many-to-one relationship



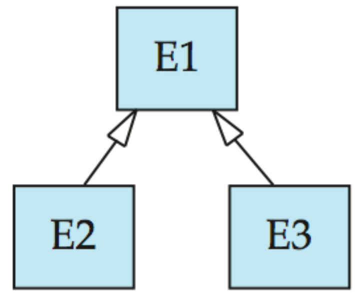
one-to-one relationship



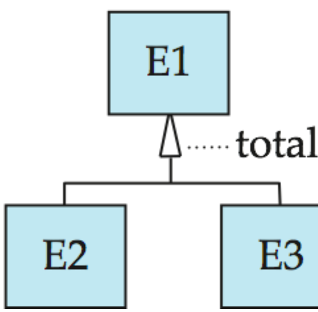
cardinality limits



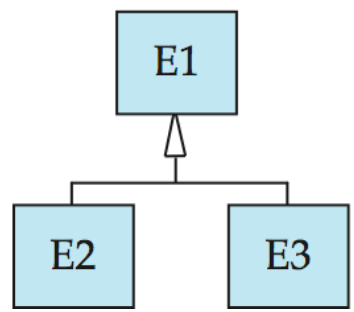
role indicator



ISA: generalization or specialization



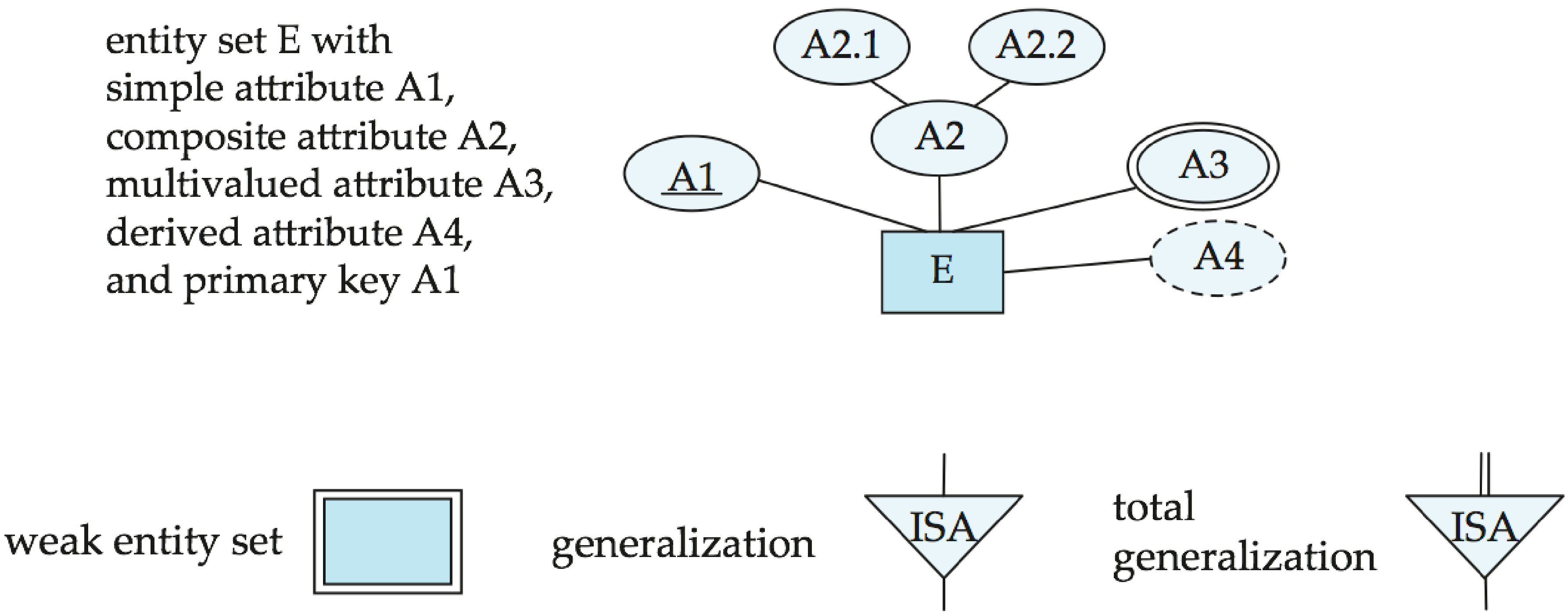
total (disjoint) generalization



disjoint generalization

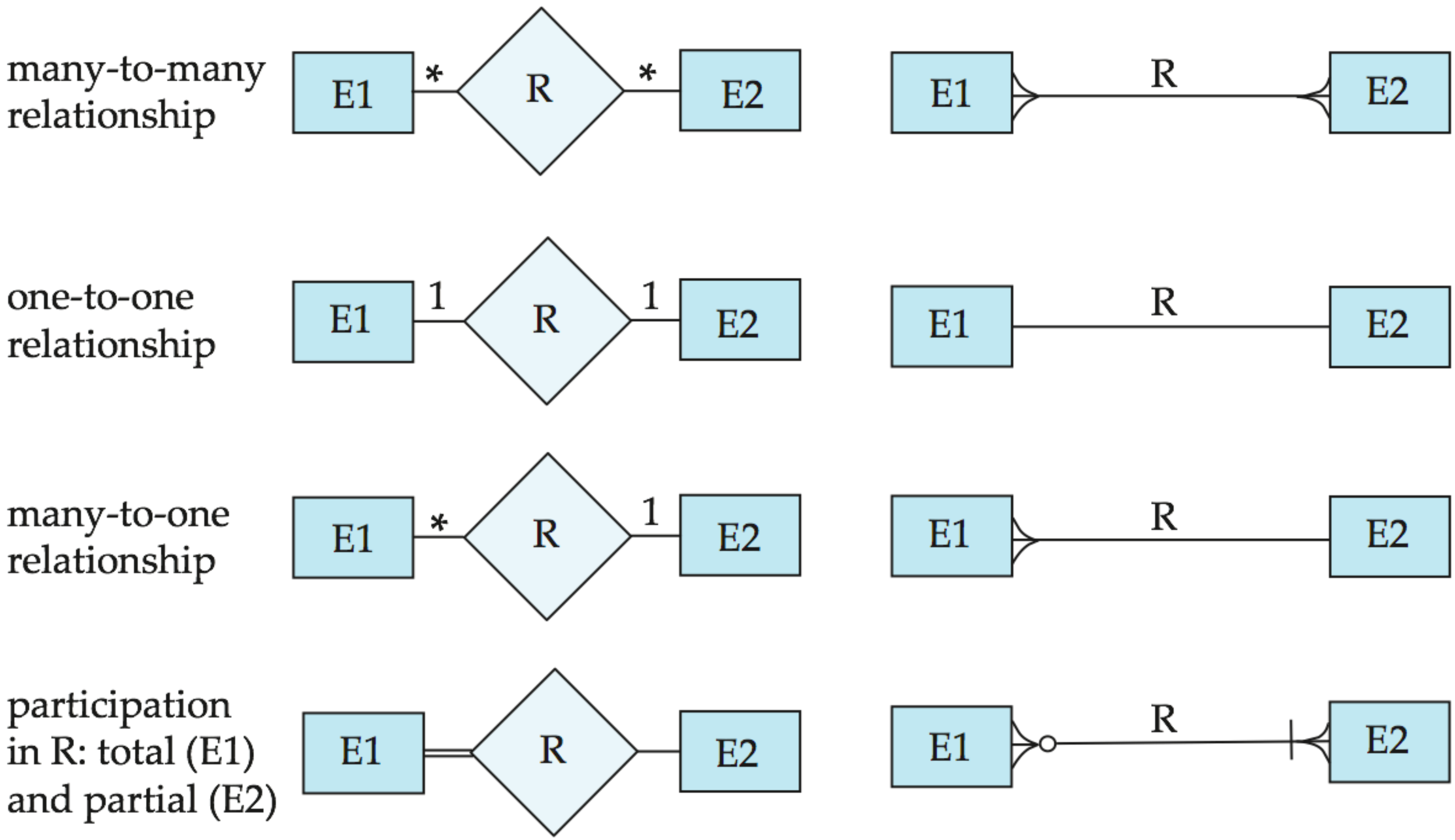
Alternative ER Notations

- Chen, IDE1FX, …



Alternative ER Notations

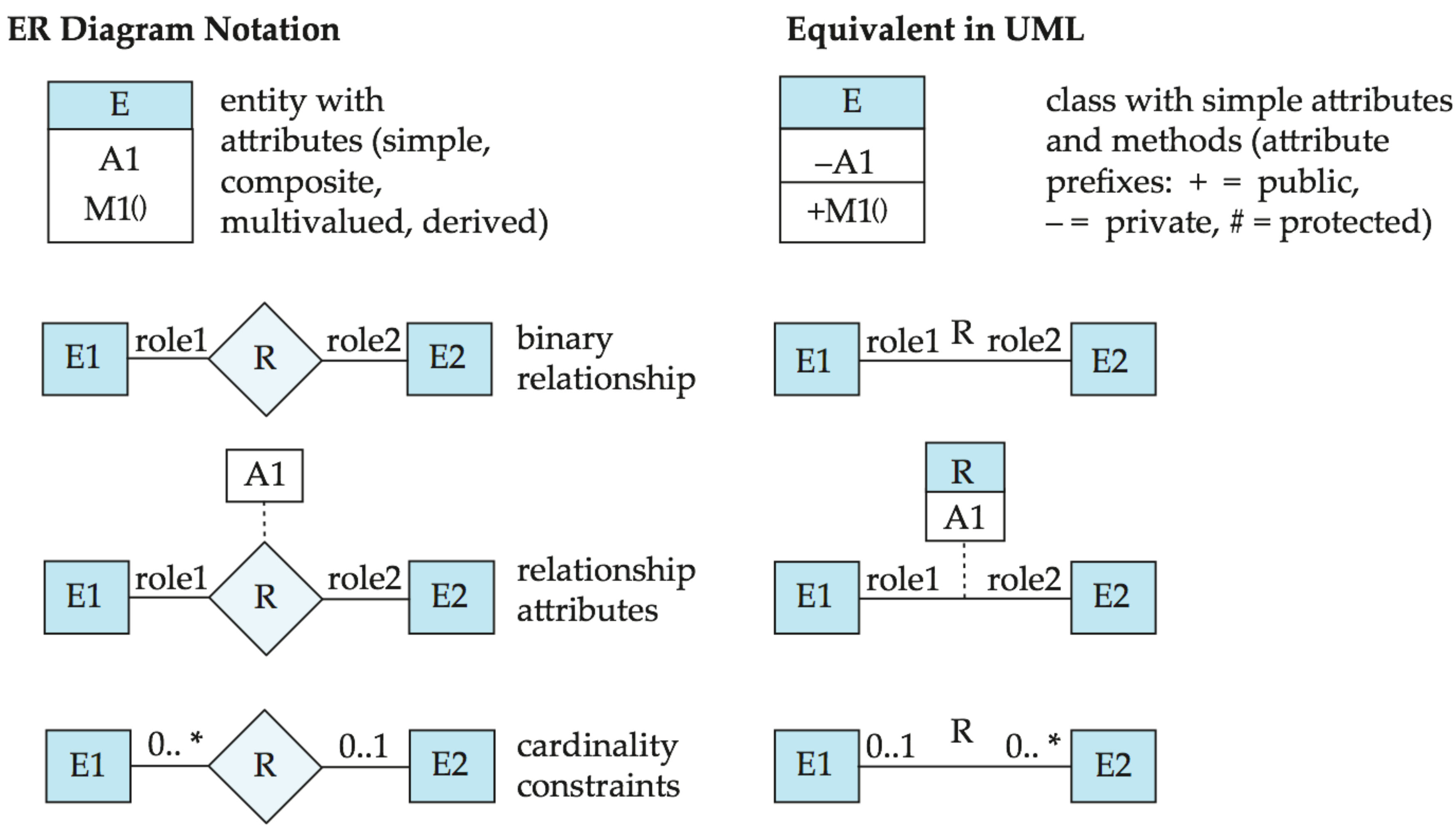
Chen | **IDE1FX (Crows feet notation)**



UML

- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.

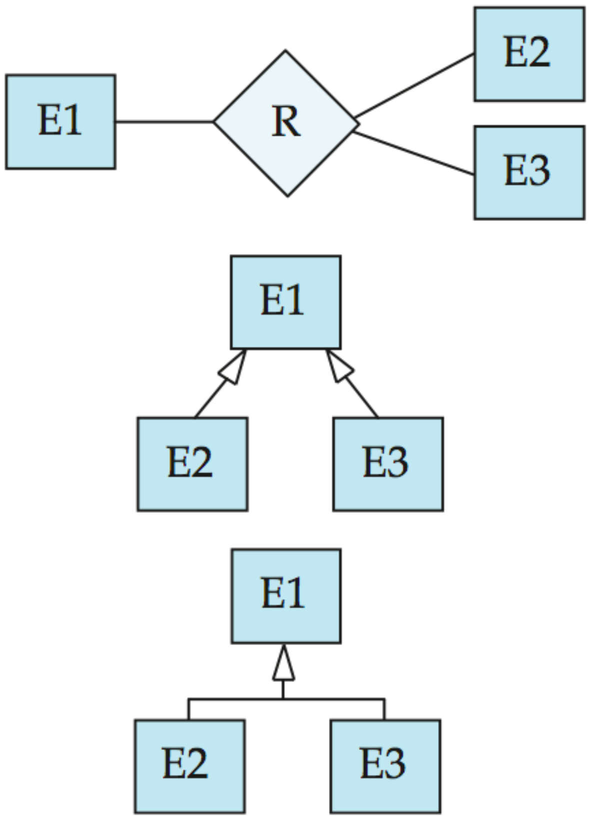
ER vs. UML Class Diagrams



*Note reversal of position in cardinality constraint depiction

ER vs. UML Class Diagrams

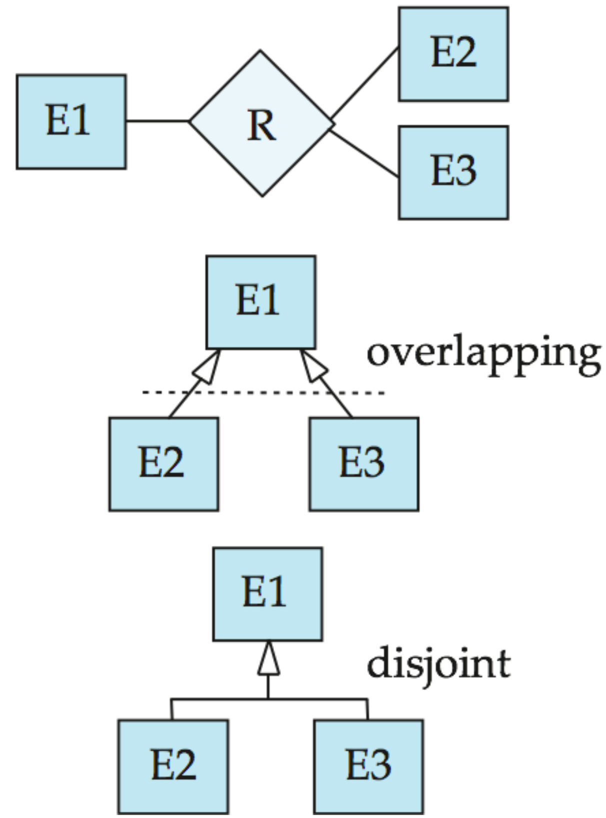
ER Diagram Notation | **Equivalent in UML**



n-ary relationships

overlapping generalization

disjoint generalization



*Generalization can use merged or separate arrows independent of disjoint/overlapping

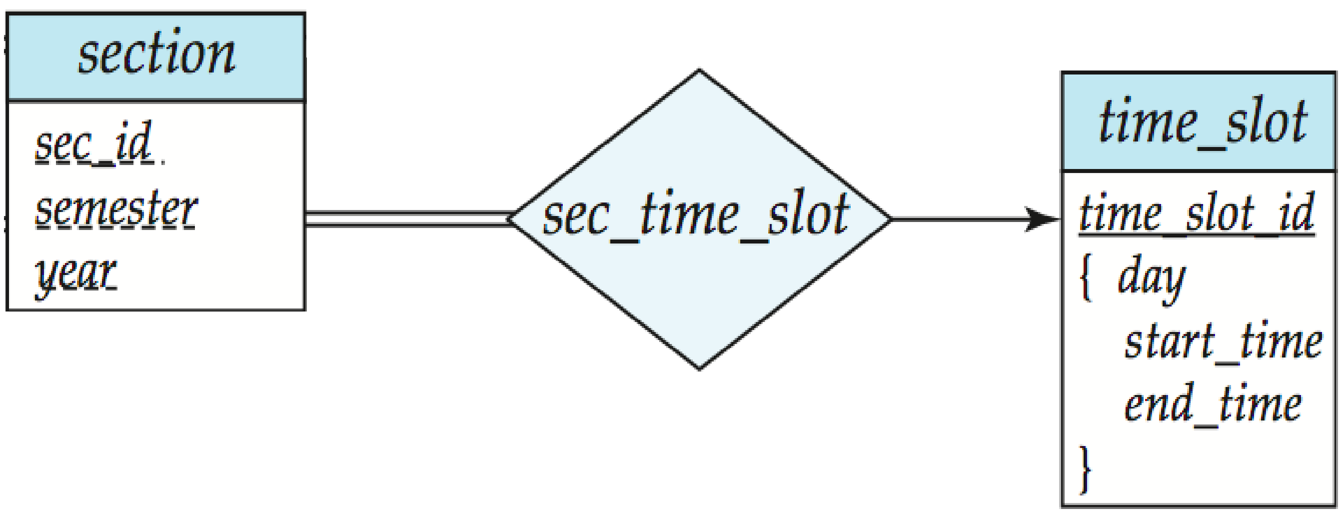
UML Class Diagrams (Cont.)

- Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.
- The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.
- The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.

End of Module 7

Multivalued Attributes (Cont.)

- Special case:entity *time_slot* has only one attribute other than the primary-key attribute, and that attribute is multivalued
  - Optimization: Don't create the relation corresponding to the entity, just create the one corresponding to the multivalued attribute
  - *time_slot*(*time_slot_id, day, start_time*, *end_time*)
  - Caveat: *time_slot* attribute of *section* (from *sec_time_slot*) cannot be a foreign key due to this optimization



Representing Aggregation via Schemas (Cont.)

- For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema

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

- Schema *proj_guide* is redundant provided we are willing to store null values for attribute *manager_name* in relation on schema *manages*