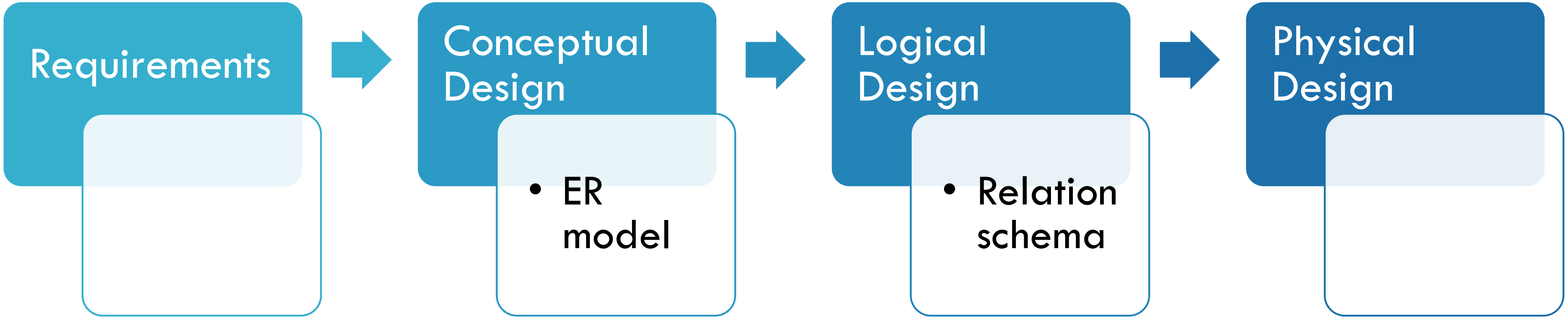
Chapter 7
Entity-Relationship Model

Based on Database System Concepts, 6th Ed. by Silberschatz, Korth and Sudarshan

Chapter 7: Entity-Relationship Model

- Design Process
- Modeling
  - Constraints
- E-R Diagram
- Reduction to Relation Schemas
- Extended E-R Features
- Design Issues
- ER notation

Design Process



Requirements

Conceptual Design

- ER model

Logical Design

- Relation schema

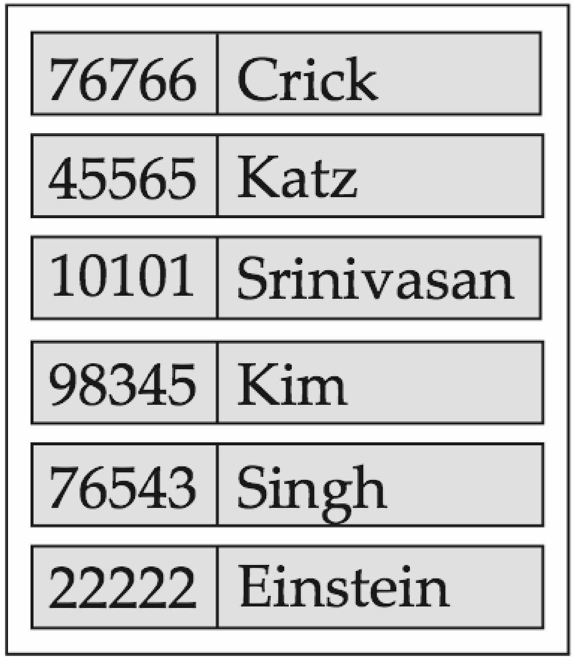
Physical Design

Design Goals : avoid <u>Redundancy</u> and <u>Incompleteness</u>

Modeling

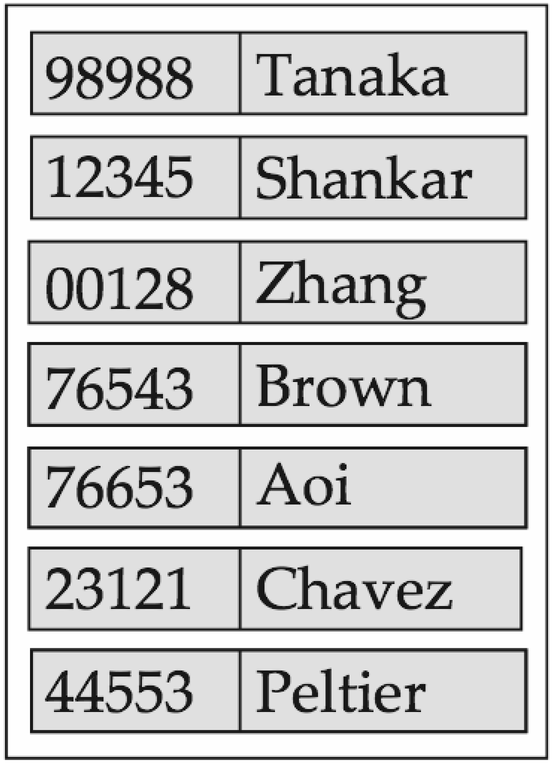
- A database can be modeled as:
  - a collection of entities
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of **attributes**.
  - Example: people have names and birthdays

Entity Sets *instructor* and *student*



instructor_ID	instructor_name
76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor



student-ID	student_name
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Relationship Sets

- A **relationship** is an association among several entities

Example:

44553 (Peltier)	advisor	22222 (Einstein)
Student entity	Relationship	instructor entity

- A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

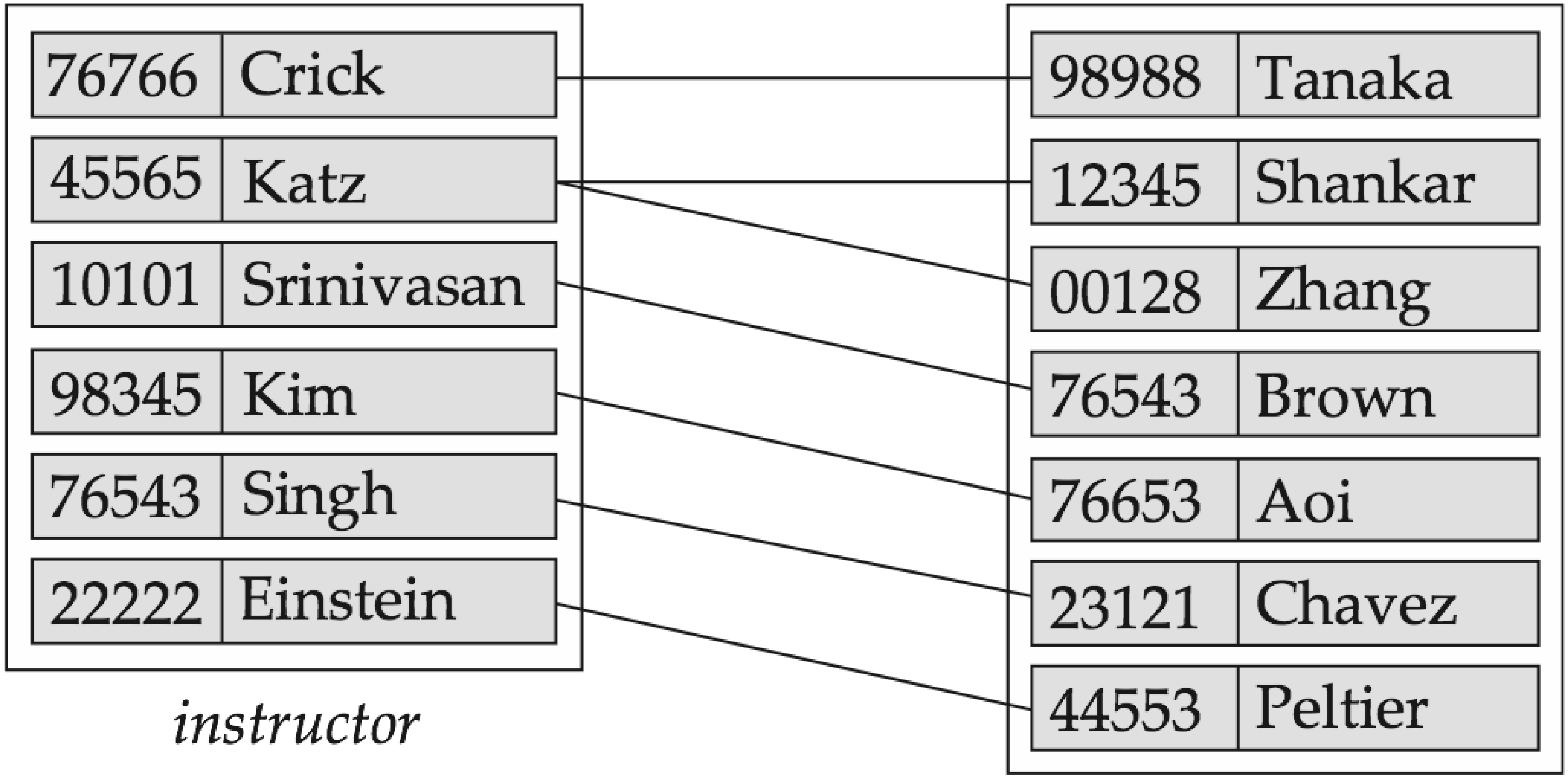
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in S_1, e_2 \in S_2, \ldots, e_n \in S_n\}$$

where $(e_1, e_2, \ldots, e_n)$ is a relationship

- Example:

$$(44553,22222) \in \mathit{advisor}$$

Relationship Set *advisor*



76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

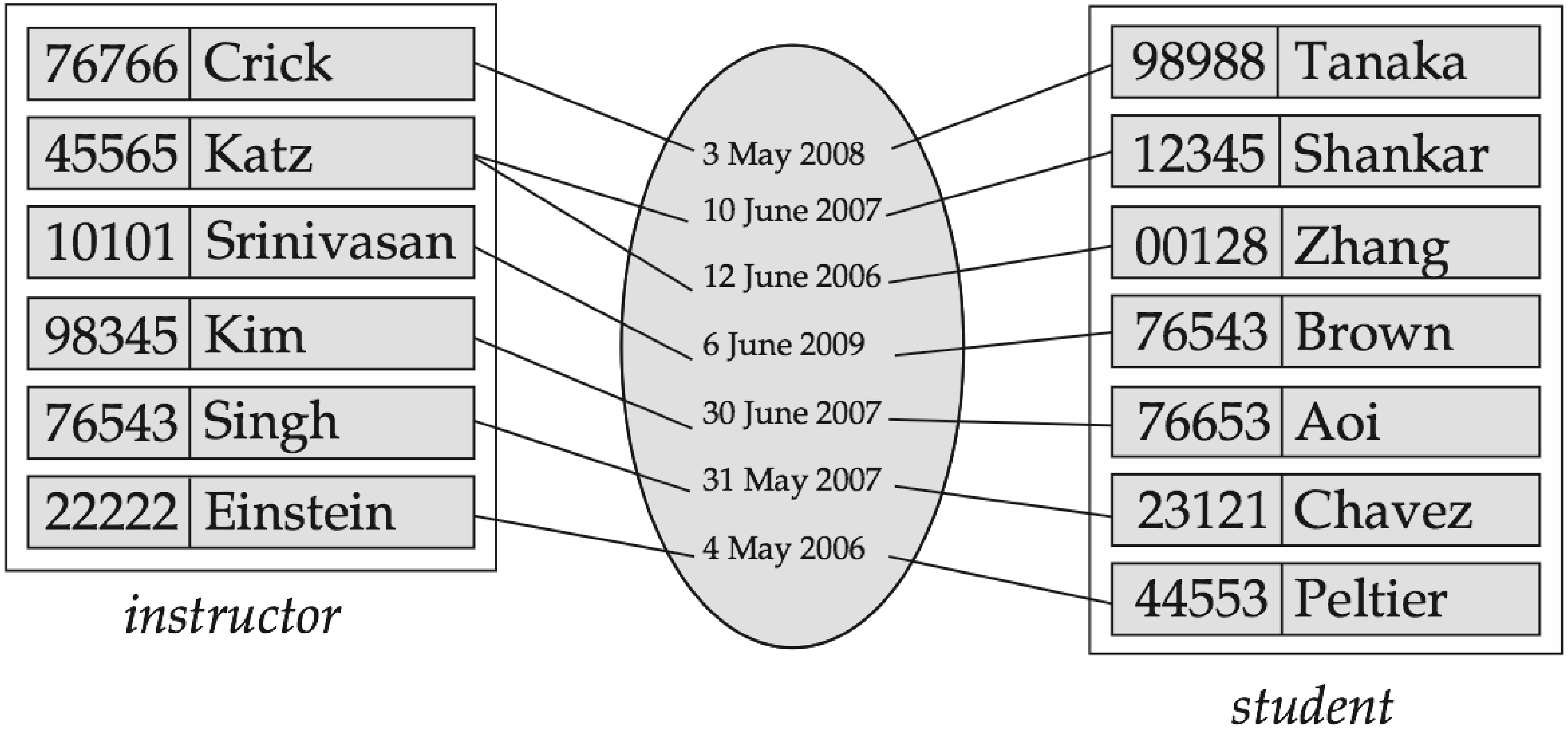
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Relationship Sets (Cont.)

- An attribute can also be associated with a relationship set

  Example: the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor



76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

3 May 2008
10 June 2007
12 June 2006
6 June 2009
30 June 2007
31 May 2007
4 May 2006

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

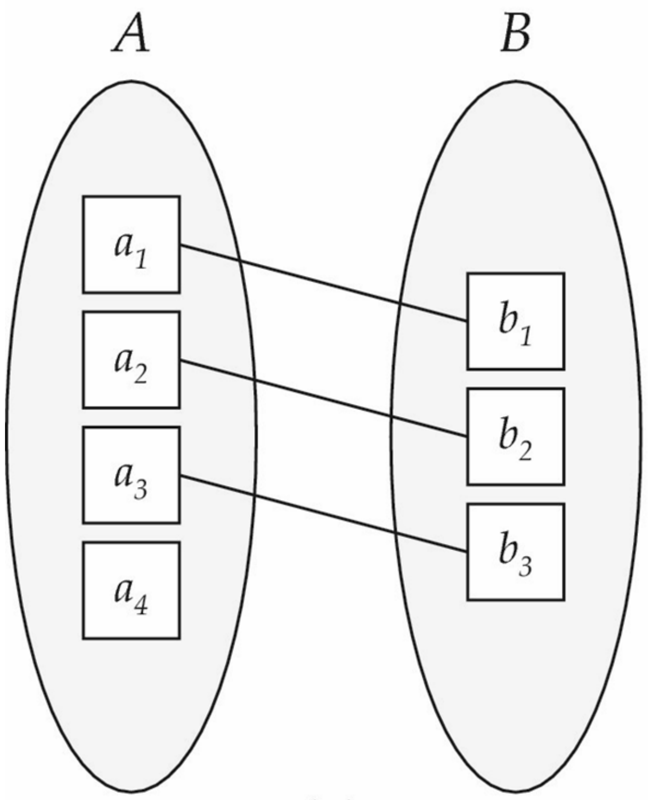
Degree of a Relationship Set

- **binary relationship**
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary
  - *Students* work on research *projects* under the guidance of an *instructor*.
  - Ternary relationship: *proj_guide* is a relationship between *instructor, student*, and *project*

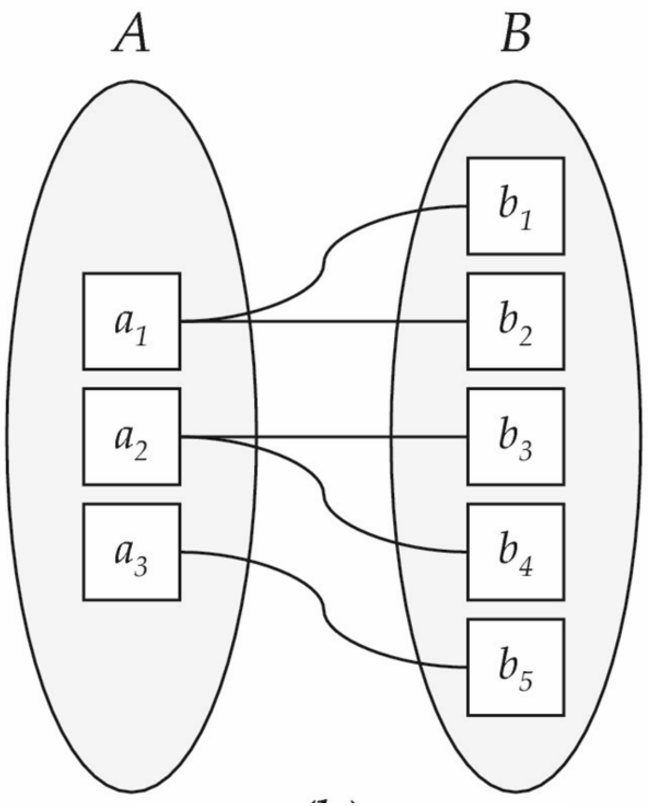
Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities



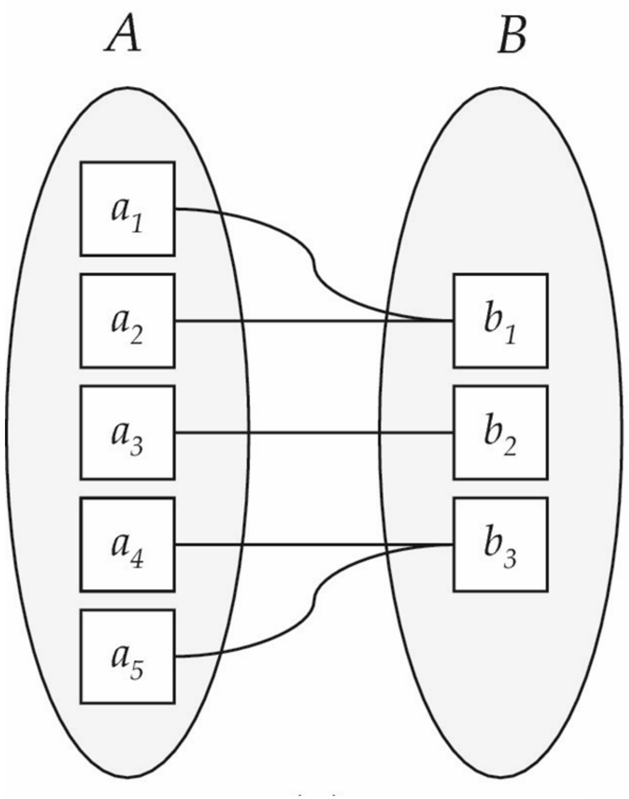
One to one



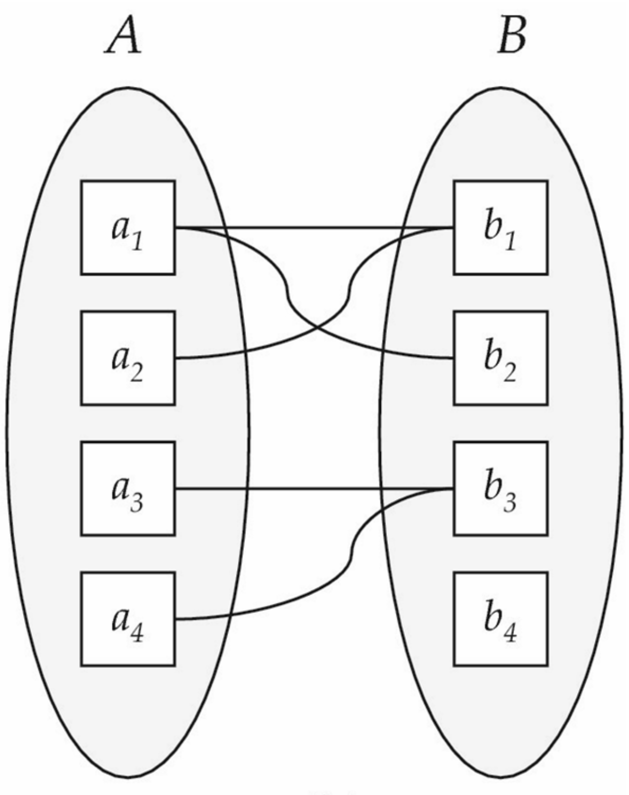
One to many

Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Mapping Cardinalities



Many to one



Many to many

Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  - Example:

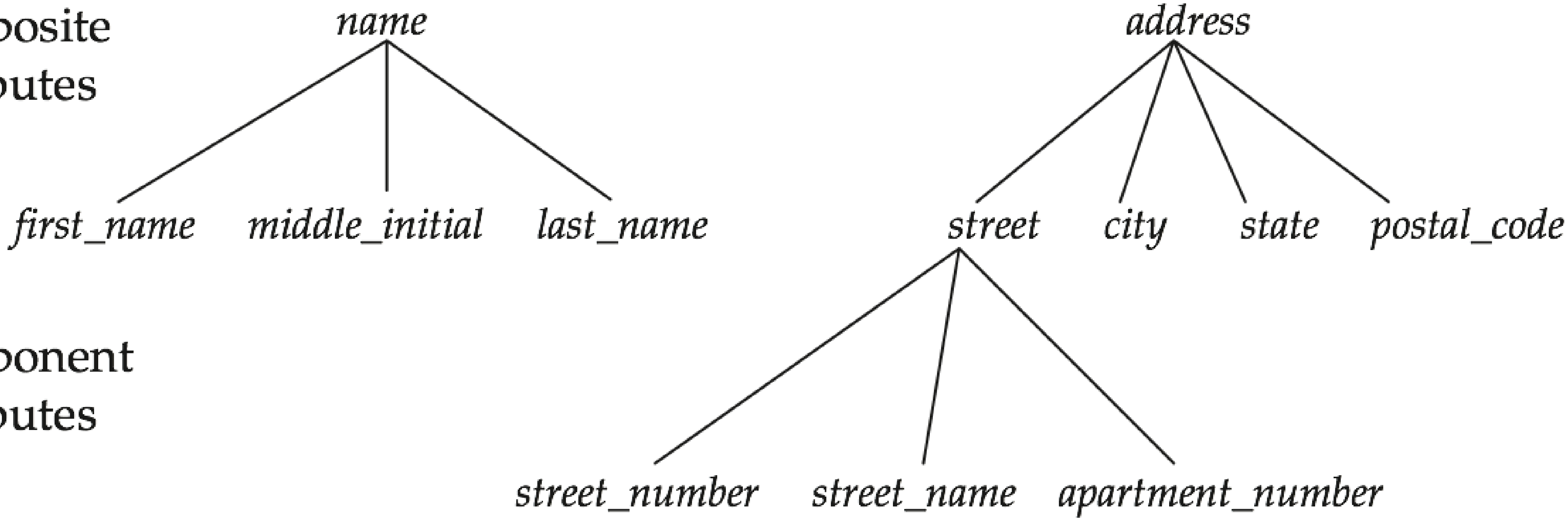
    instructor = (*ID, name, street, city, salary*)

    course= (*course_id, title, credits*)
- **Domain** – the set of permitted values for each attribute
- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes - Can be computed from other attributes
    - Example: age, given date_of_birth

Composite Attributes

composite
attributes

component
attributes



Redundant Attributes

- Suppose we have entity sets:
  - *instructor*, with attributes: *ID*, *name*, *dept_name, salary*
  - *department,* with attributes: *dept_name, building, budget*
- We model the fact that each instructor has an associated department using a relationship set *inst_dept*
- The attribute *dept_name* appears in both entity sets. Since *inst_dept* is an explicit relationship that relates instructors to departments
  - We should remove dept_name from instructor.
  - BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see later.

Keys

- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A **candidate key** of an entity set is a minimal super key
  - ID is candidate key of instructor
  - course_id is candidate key of course
- Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key**.

Example: Keys

- Consider the entity set of students in a university.
  - student has attributes NIC, index, name and GPA.
- **super keys** of student (attribute/s that can be used to uniquely identify a student)
  - (NIC), (index), (NIC,index), (NIC,name), (NIC,GPA)…… (NIC, index, name, GPA)
- **candidate keys** of student (minimal super keys)
  - (NIC), (index)
- **primary key** – based on the context (index) would be the better choice.

Keys for relationship sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - *(s_id, i_id)* is the super key of advisor
  - *NOTE: this means <u>a pair of entity sets can have at most one relationship in a particular relationship set.</u>*
    - Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot assume a relationship for each meeting. We can use a multivalued attribute though.
- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys
- Need to consider semantics of relationship set in selecting the primary key in case of more than one candidate key.

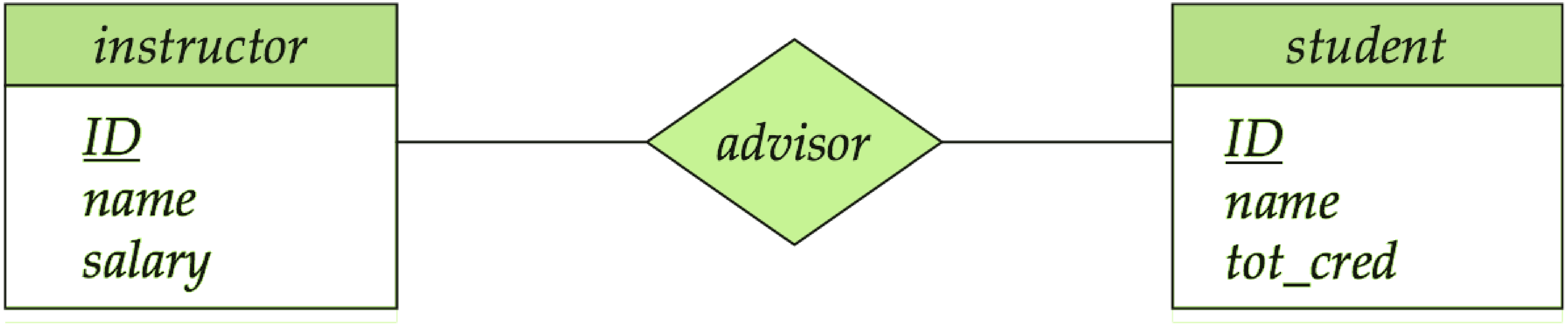
Example: Keys for relationship sets

- Consider many-to-many relationship set *takes* between *students* and *courses*
  - *(student_id, course_id)* is the super key
  - This also becomes a candidate key and the primary key
- Consider one-to-many relationship set *is_in* between *folder* and *file*
  - *(folder_id, file_id)* and *(file_id)* is the super key. (since one file is only in folder we can uniquely identify the folder for a given file)
  - (file_id) is the candidate key and the primary key.
- Follow the same logic for one-to-one and many-to-one relationship sets

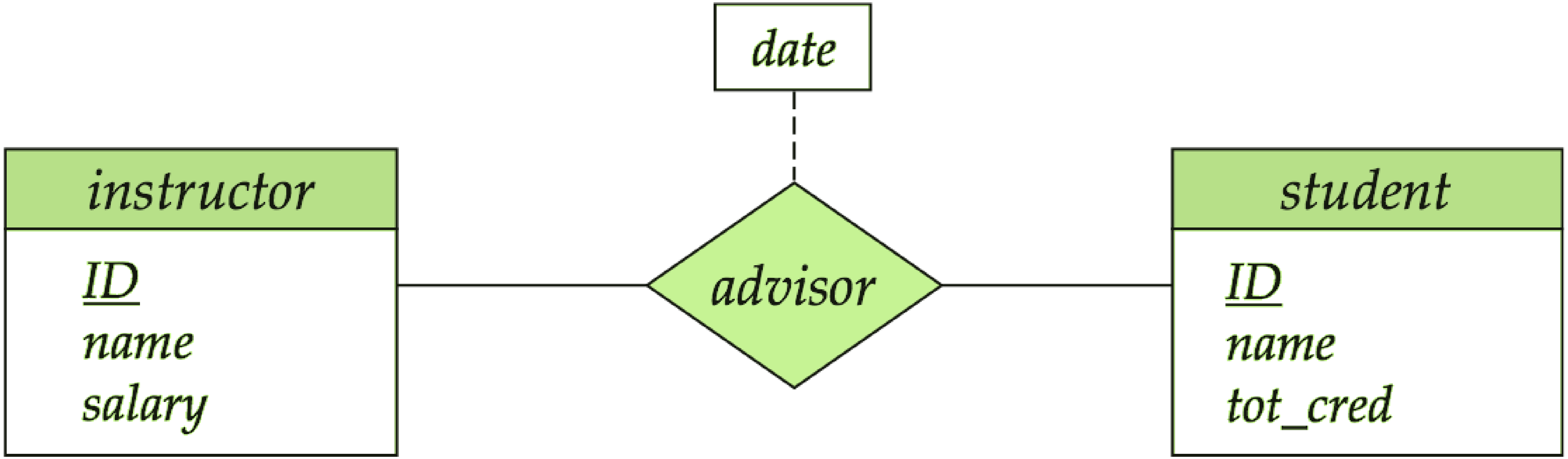
Entity Relationship Diagram

E-R Diagrams

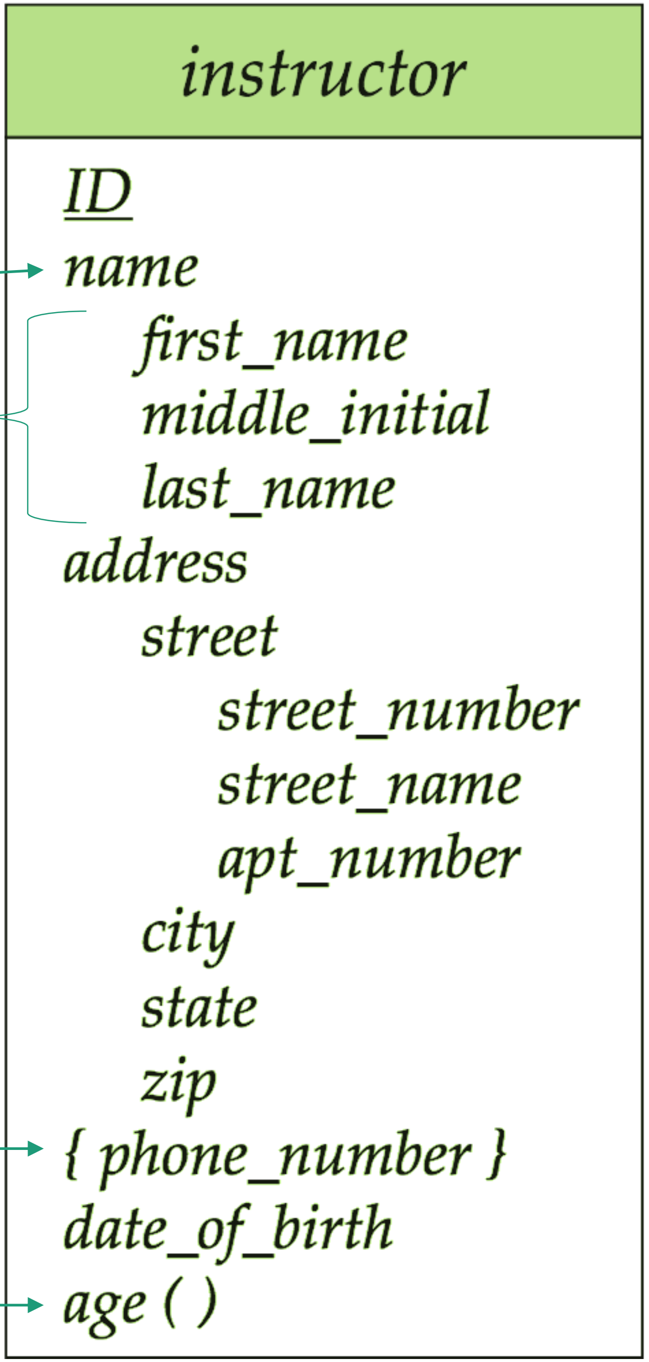
- Rectangles represent entity sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes
- Diamonds represent relationship sets.



Relationship Sets with Attributes



Complex Attributes



instructor

- <u>*ID*</u>
- *name*
  - *first_name*
  - *middle_initial*
  - *last_name*
- *address*
  - *street*
    - *street_number*
    - *street_name*
    - *apt_number*
  - *city*
  - *state*
  - *zip*
- *{ phone_number }*
- *date_of_birth*
- *age ( )*

Composite Attribute

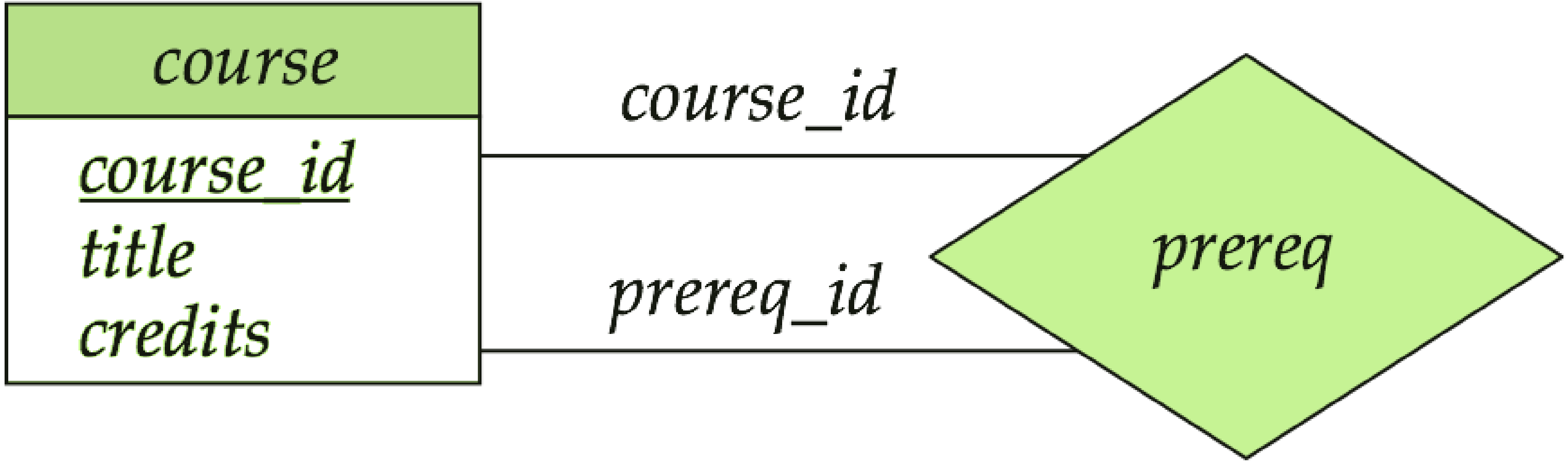
Component Attributes

Multivalued Attribute

Derived Attribute

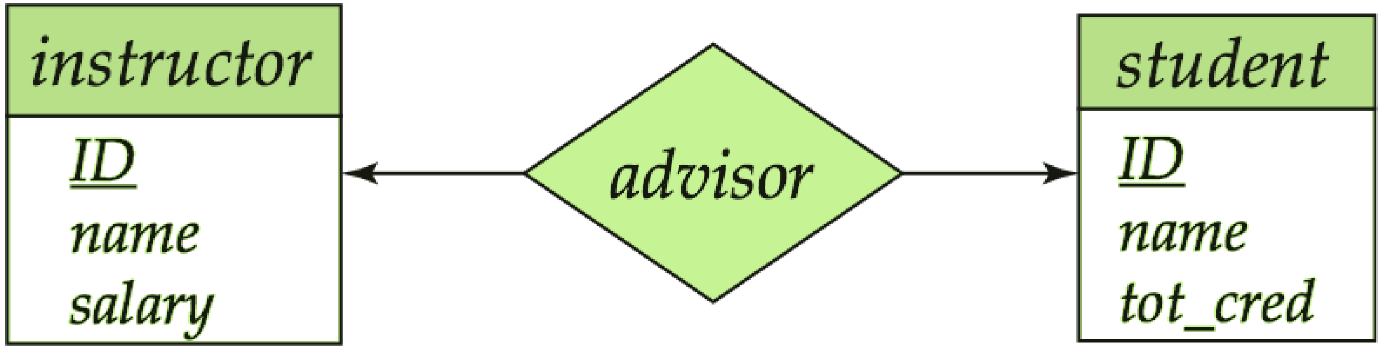
Roles

- Entity sets of a relationship need not be distinct
- Each occurrence of an entity set plays a "role" in the relationship
- The labels "*course_id*" and "*prereq_id*" are called **roles**.



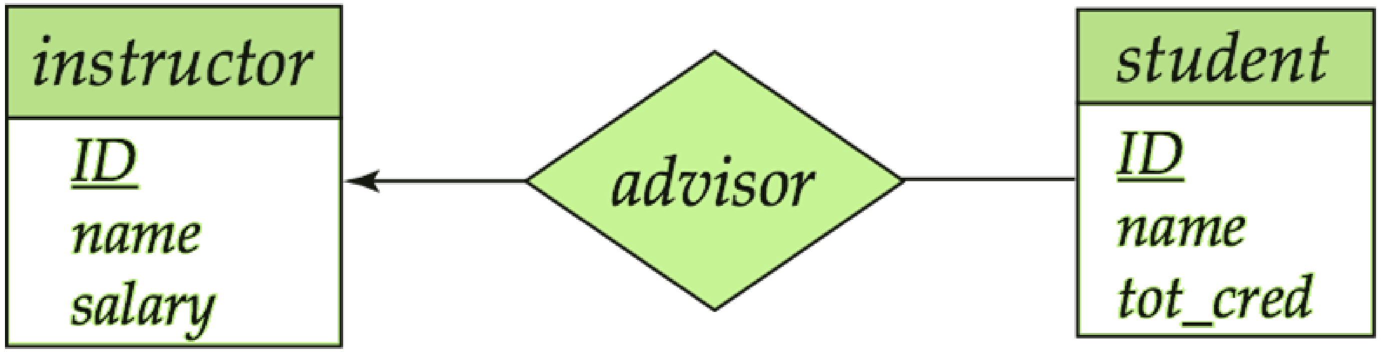
Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.
- **One-to-one relationship** between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*



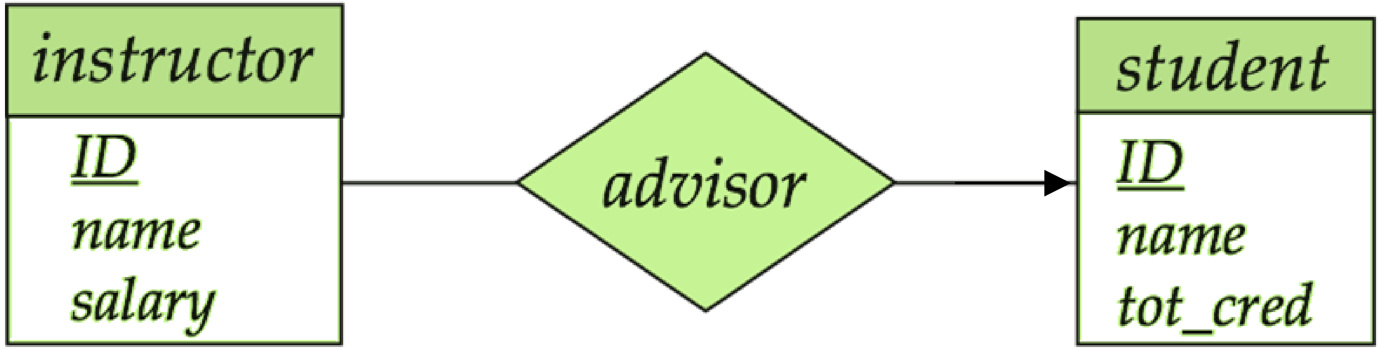
Cardinality Constraints

- **One-to-many relationship** between an instructor and a student
  - an instructor is associated with several (including 0) students via advisor
  - a student is associated with at most one instructor via advisor



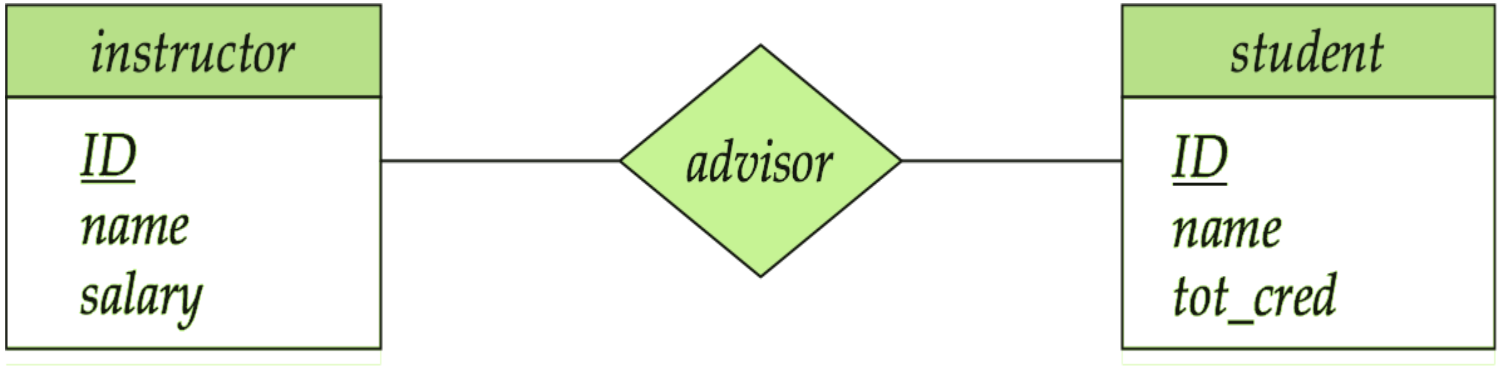
Cardinality Constraints

- **Many-to-one relationship** between an instructor and a student
  - an instructor is associated with at most one student via advisor,
  - a student is associated with several (including 0) instructors via advisor



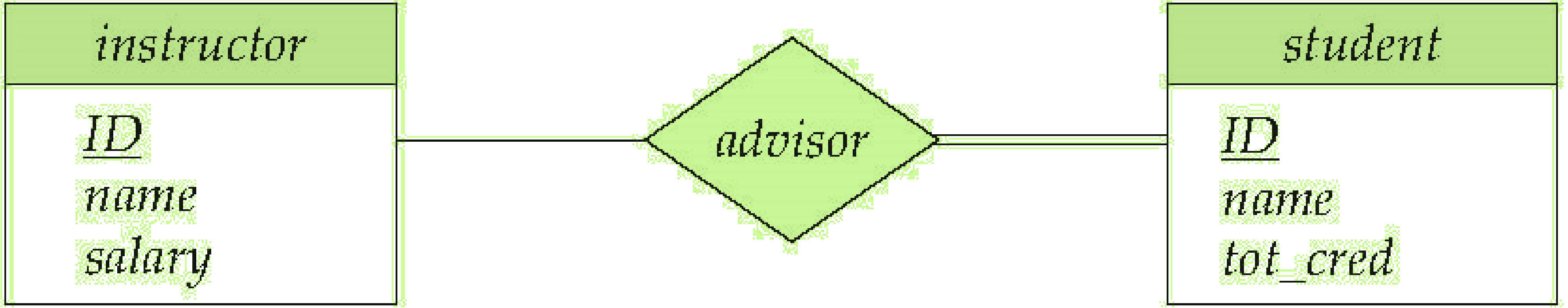
Cardinality Constraints

- **Many-to-many relationship** between an instructor and a student
  - An instructor is associated with several (possibly 0) students via advisor
  - A student is associated with several (possibly 0) instructors via advisor



Total and Partial Participation

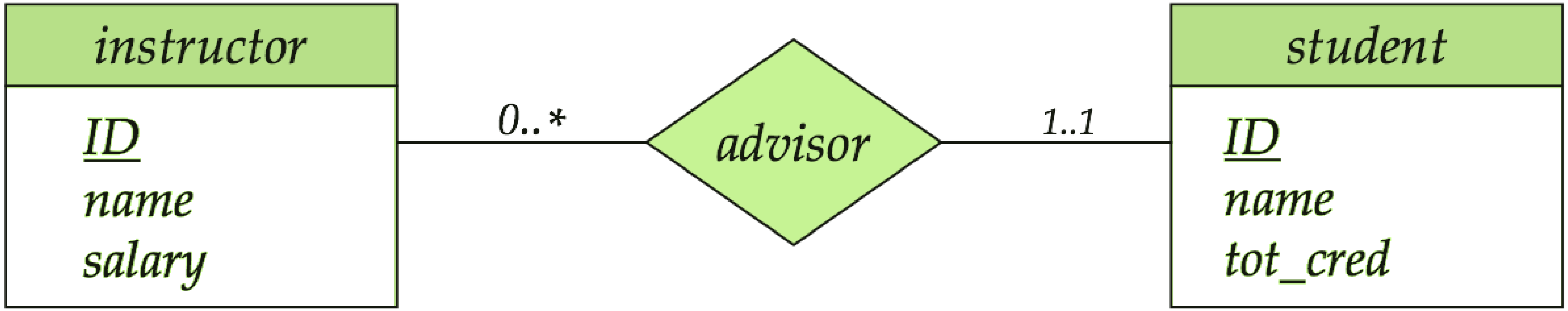
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set



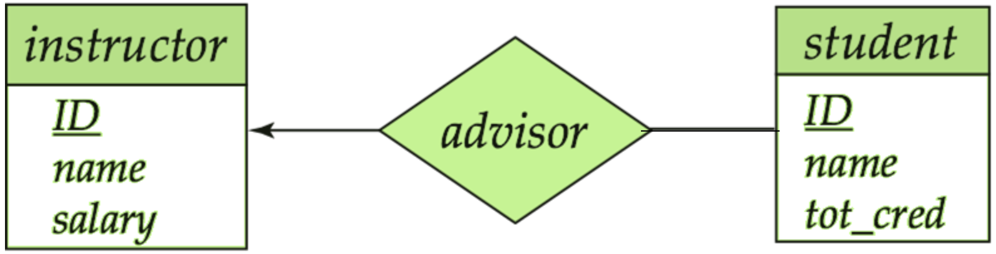
  - participation of *student* in *advisor* relation is total
  - every *student* must have an associated instructor
- Partial participation: some entities may not participate in any relationship in the relationship set
  - participation of *instructor* in *advisor* is partial

Alternative Notation for Cardinality Limits

- A line may have an associated minimum and maximum cardinality, shown in the form *l..h*, where l is the minimum and h the maximum cardinality



- Instructor can advise 0 or more students. A student must have exactly1 advisor.



Weak Entity Sets

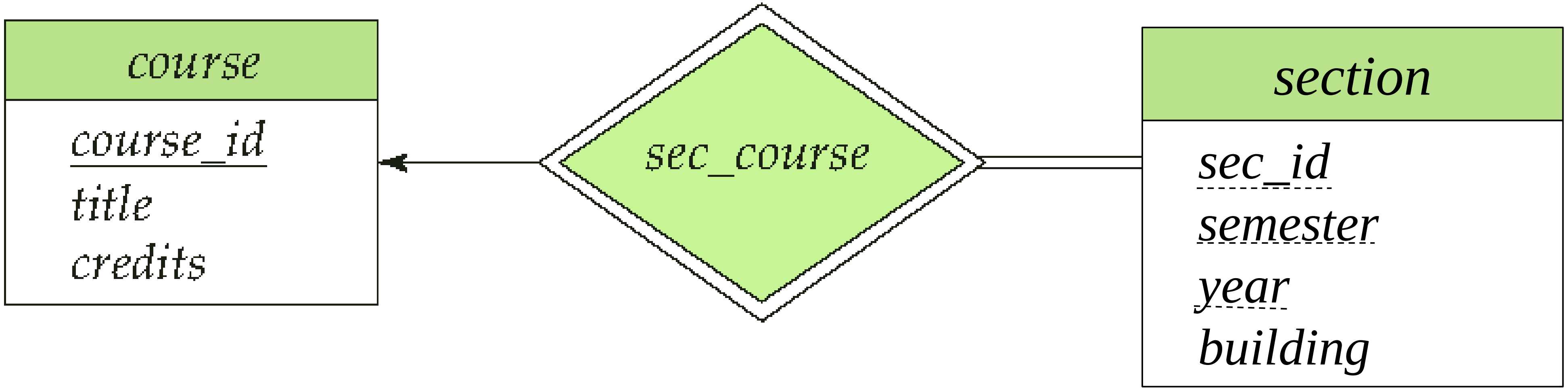
- Consider the following entities.
  - course = {course_id, tittle, credits}
  - section = {course_id, sec_id, semester, year, building}
- Clearly, section entities are related to course entities. Suppose we create a relationship set *sec_course* between entity sets *section* and *course*.
- Note that the information in *sec_course* is redundant, since *section* already has an attribute *course_id*, which identifies the course with which the section is related.
  - Can we discard the relationship *sec_course*?
  - Can we remove attribute *course_id* from *section* entity?

Weak Entity Sets

- We use **identifying relationships** and **weak entity sets** to solve this problem.
- Weak entity set is an entity set without a primary key.
  - section = {sec_id, semester, year, building}
- The existence of a weak entity set depends on an **identifying entity set**.
  - *course* is the identifying entity set for *section*.
- The relationship between the two entity sets is called an **identifying relationship**.
  - Total participation from weak entity set
  - Relationship is one-to-many from identifying entity set to weak entity set.
- The set of attributes that allows distinguishing among weak entities is called the **discriminator** or **partial key**.
- Primary key of weak entity set is formed by the primary key of the identifying entity set and the discriminator of the weak entity set.

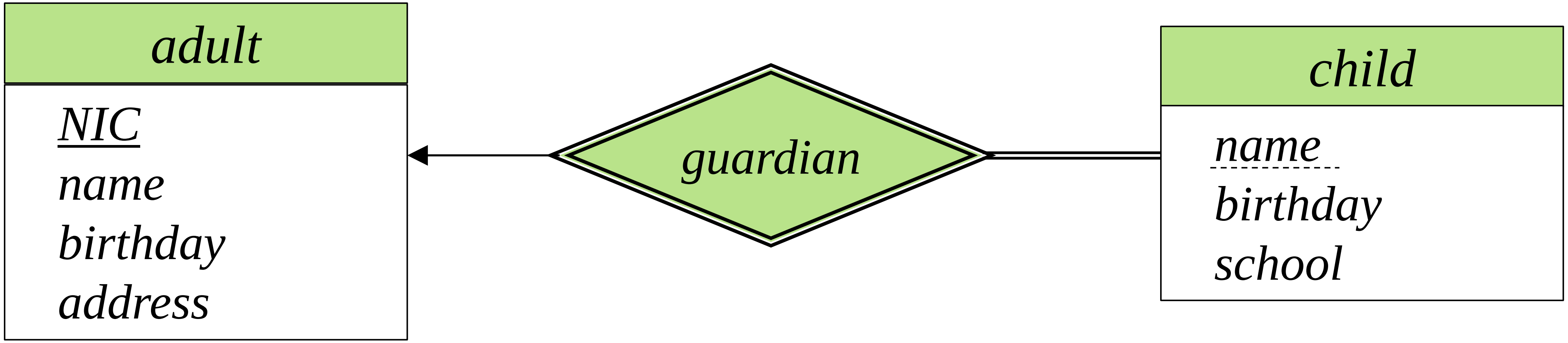
Weak Entity Sets

- **Identifying relationship** is denoted by a double diamond.
- The **discriminator** of a weak entity set is underlined with a dashed line.
- Primary key for section – (course_id, sec_id, semester, year)



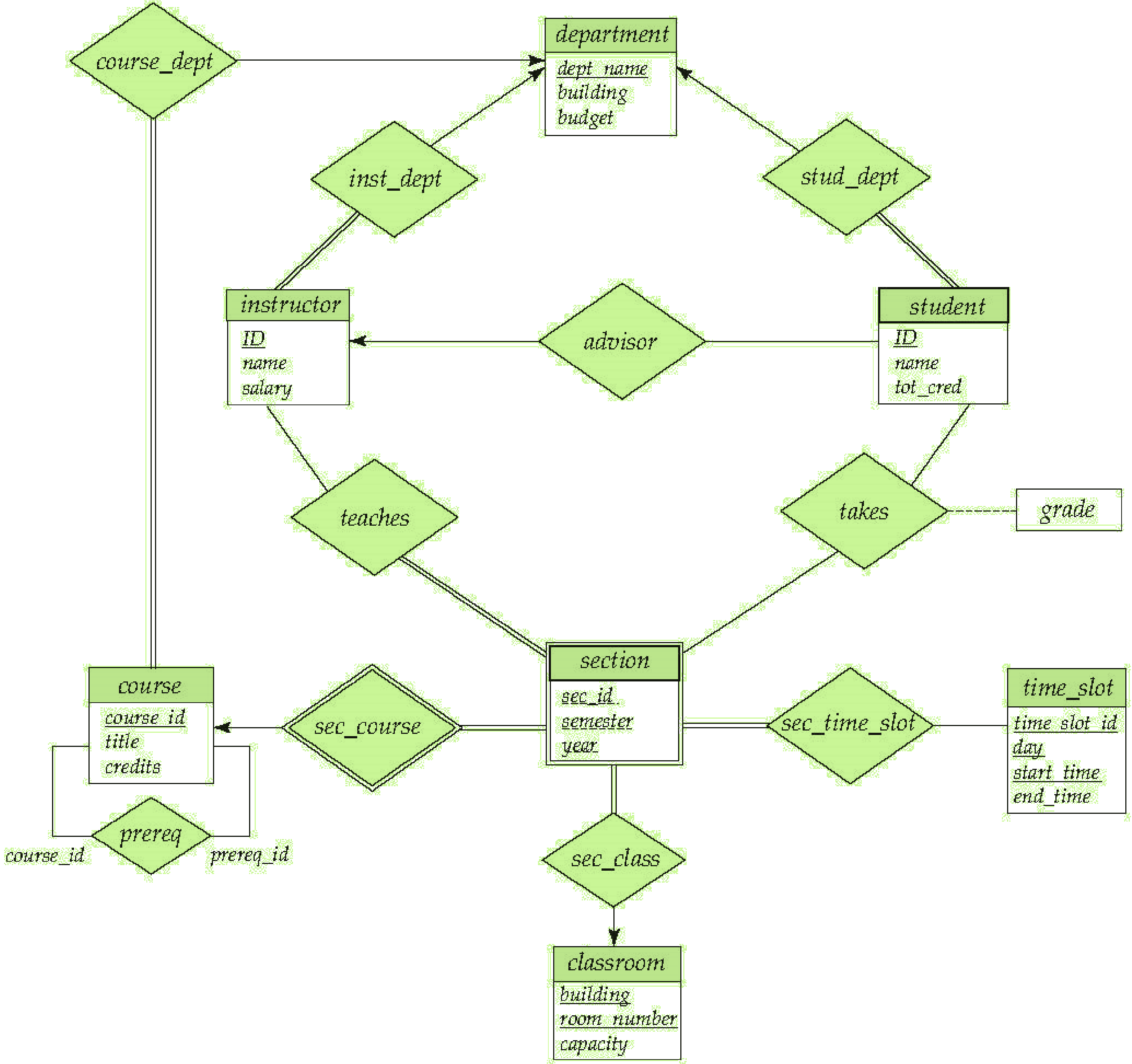
Example: Weak Entity Sets

- Consider how banks store details about children (for saving accounts). Since a child cannot be uniquely identified (no NIC) we associate a child with an adult who is his guardian.
- Assume that one guardian would not name all his children with the same name. Then *name* becomes a partial key for the weak entity set *child* and *adult* is the associated identifying entity set.
- Primary key for *child* is (NIC, name)



Example

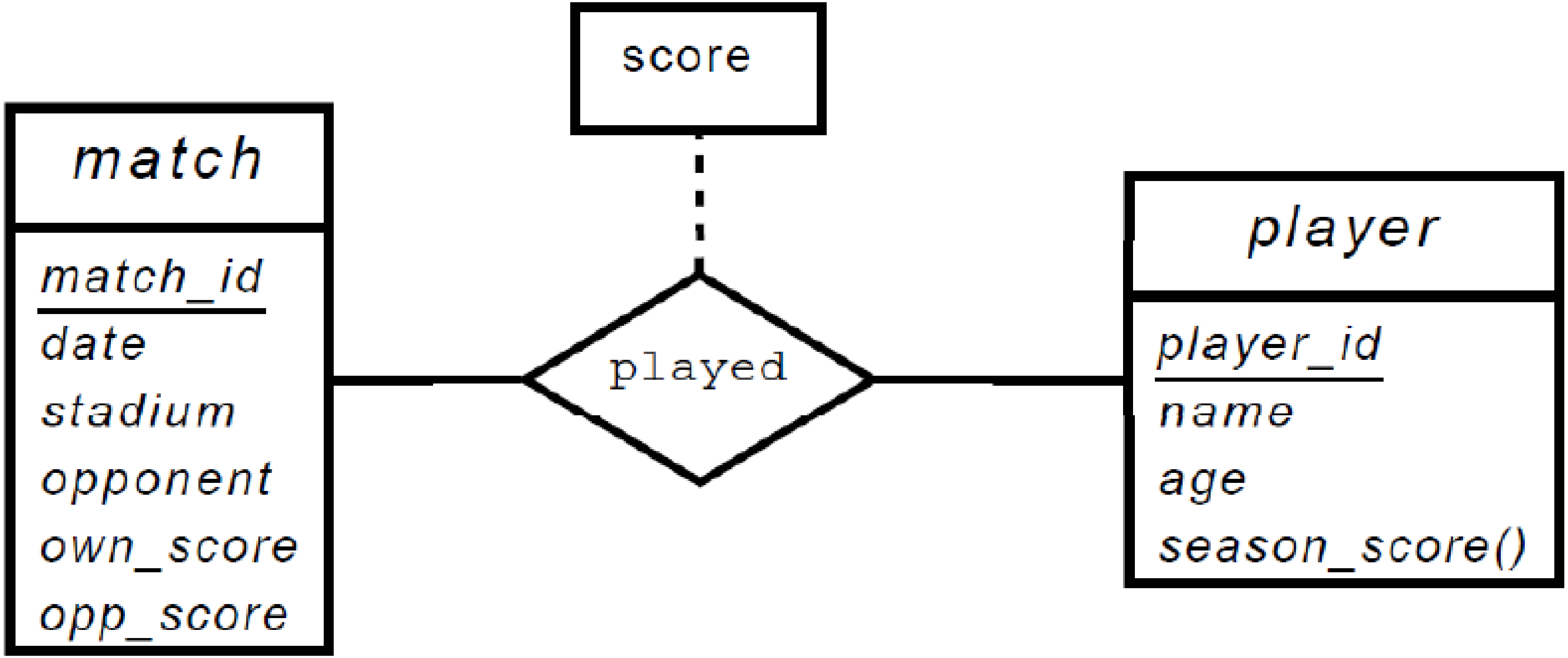
ER diagram for a University Enterprise



Exercise 1

Design an ER diagram for keeping track of the exploits of your favourite SLPL team. You should store the matches played, the scores in each match, the players in each match, individual player statistics (batting only) for each match. Summary statistics should be modeled derived attributes

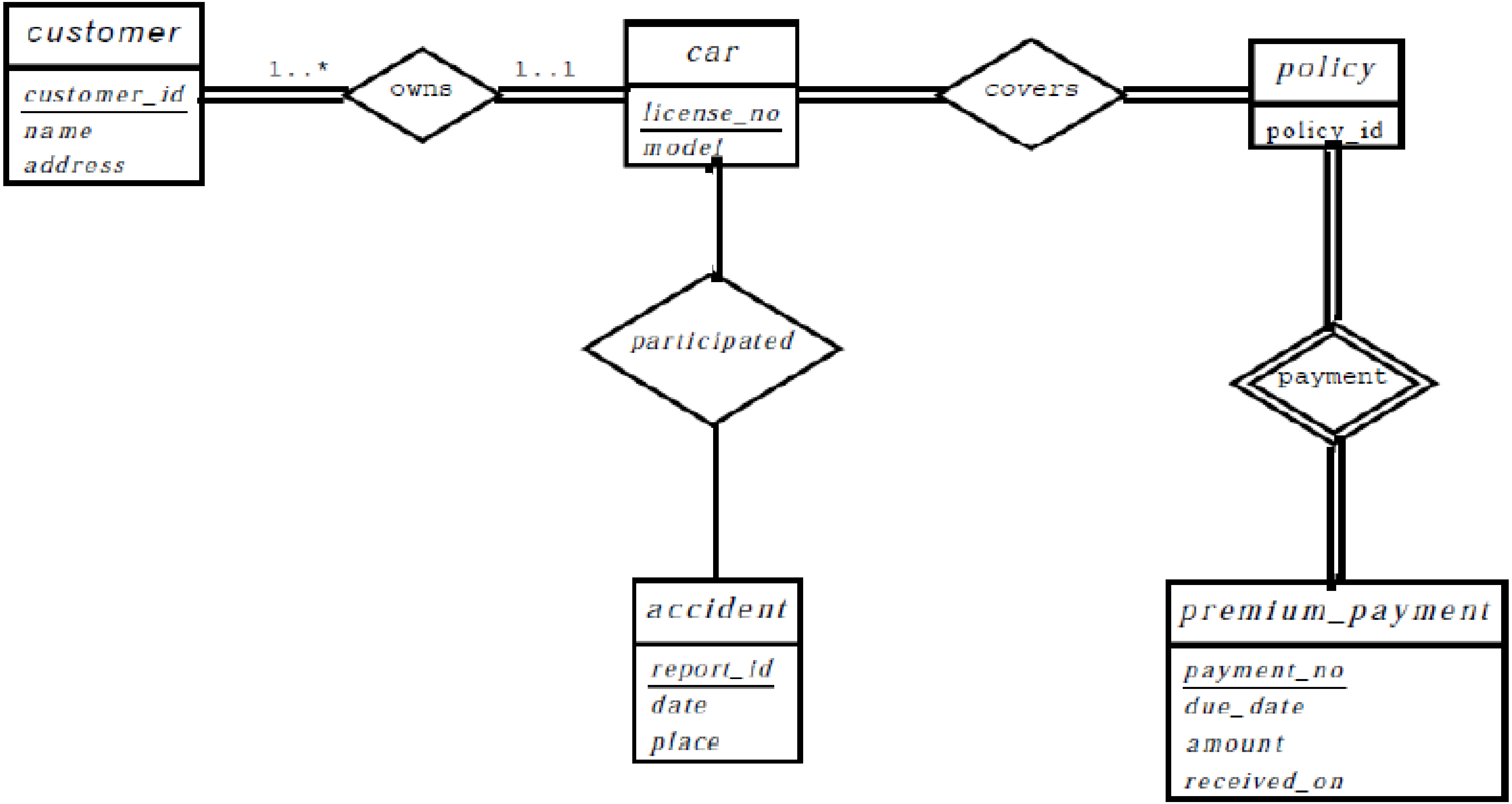
Sample answer



Exercise 2

Construct and ER diagram for a car Insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payment associated with it. Each payment is for a particular period of time and has an associated due date, and the date when the payment was received.

Sample answer



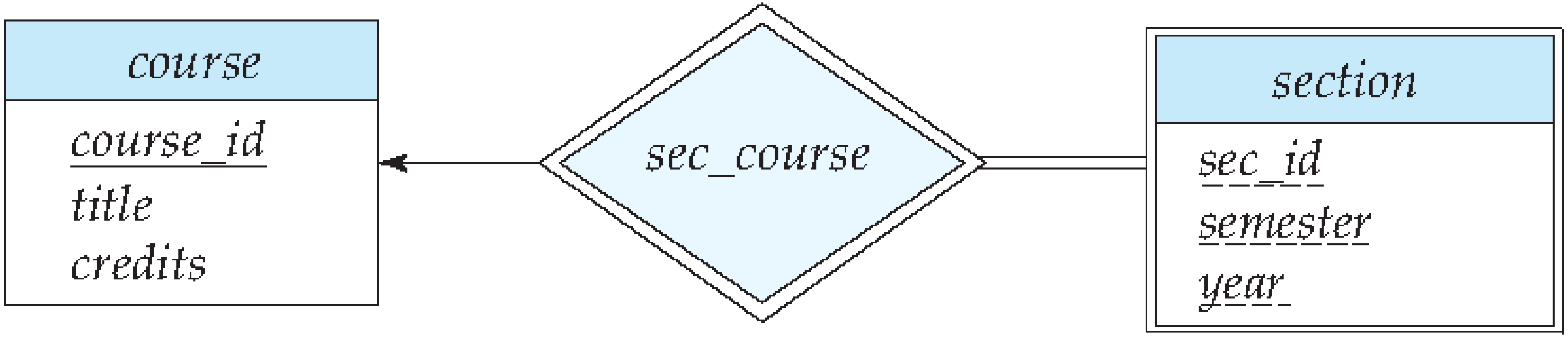
Reduction to Relational Schema

Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

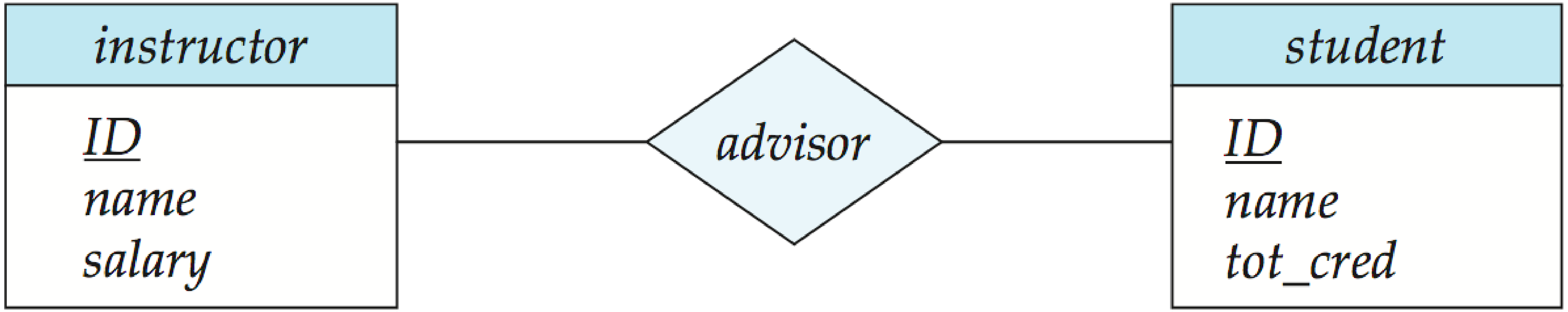
Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes
  - *course(course_id, title, credits)*
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
  - *section (course_id, sec_id, sem, year)*



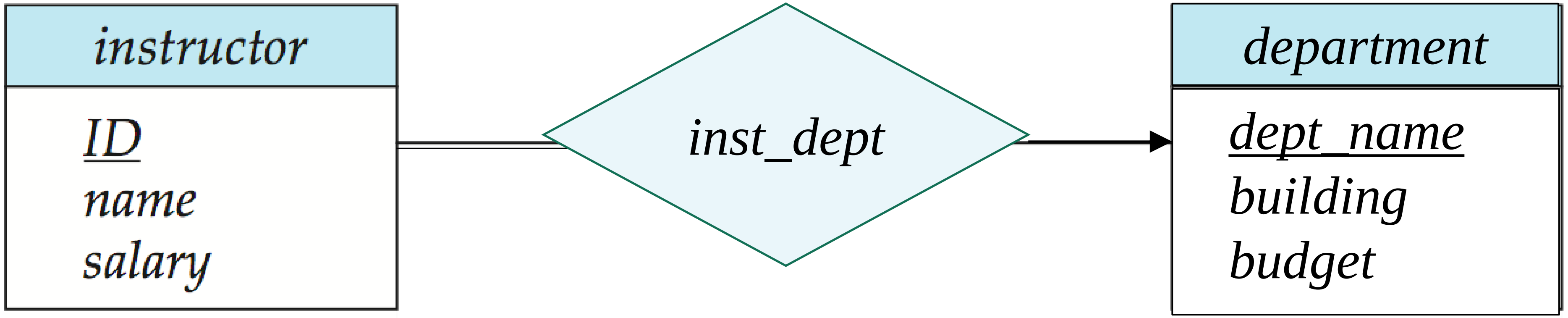
Representing Relationship Sets

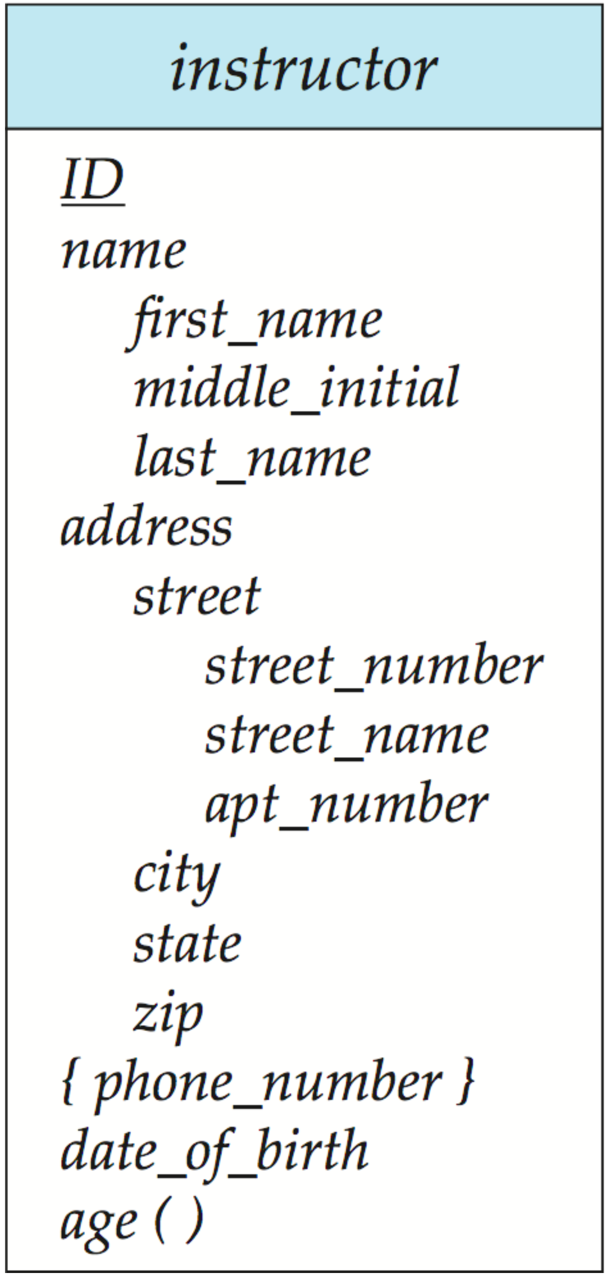
- A **many-to-many relationship** set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
  - *student = (s_id, name, tot_cred)*
  - *instructor = (i_id, name, salary)*
  - *advisor* = (*s_id, i_id*)



Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
  - *department*= (*dept_name*, *building*, *budget*)
  - *instructor* = (*i_id*, *name*, *salary*, *dept_name*)





instructor

- *ID*
- *name*
  - *first_name*
  - *middle_initial*
  - *last_name*
- *address*
  - *street*
    - *street_number*
    - *street_name*
    - *apt_number*
  - *city*
  - *state*
  - *zip*
- *{ phone_number }*
- *date_of_birth*
- *age ( )*

Composite Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* and *name_last_name*
  - Prefix omitted if there is no ambiguity (*name_first_name* could be *first_name*)
- Ignoring multivalued attributes, extended instructor schema is
  - *instructor(ID, first_name, middle_initial, last_name,*
    street_number, street_name, apt_number, city,
    state, zip_code, date_of_birth)

Multivalued Attributes

- A multivalued attribute *M* of an entity *E* is represented by a separate schema *EM*
- Schema *EM* has attributes corresponding to the primary key of *E* and an attribute corresponding to multivalued attribute *M*
- Example: Multivalued attribute *phone_number* of *instructor* is represented by a schema:

  inst_phone= (*ID*, *phone_number*)
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema *EM*
  - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:

    (22222, 456-7890) and (22222, 123-4567)

Extended ER Features

Specialization

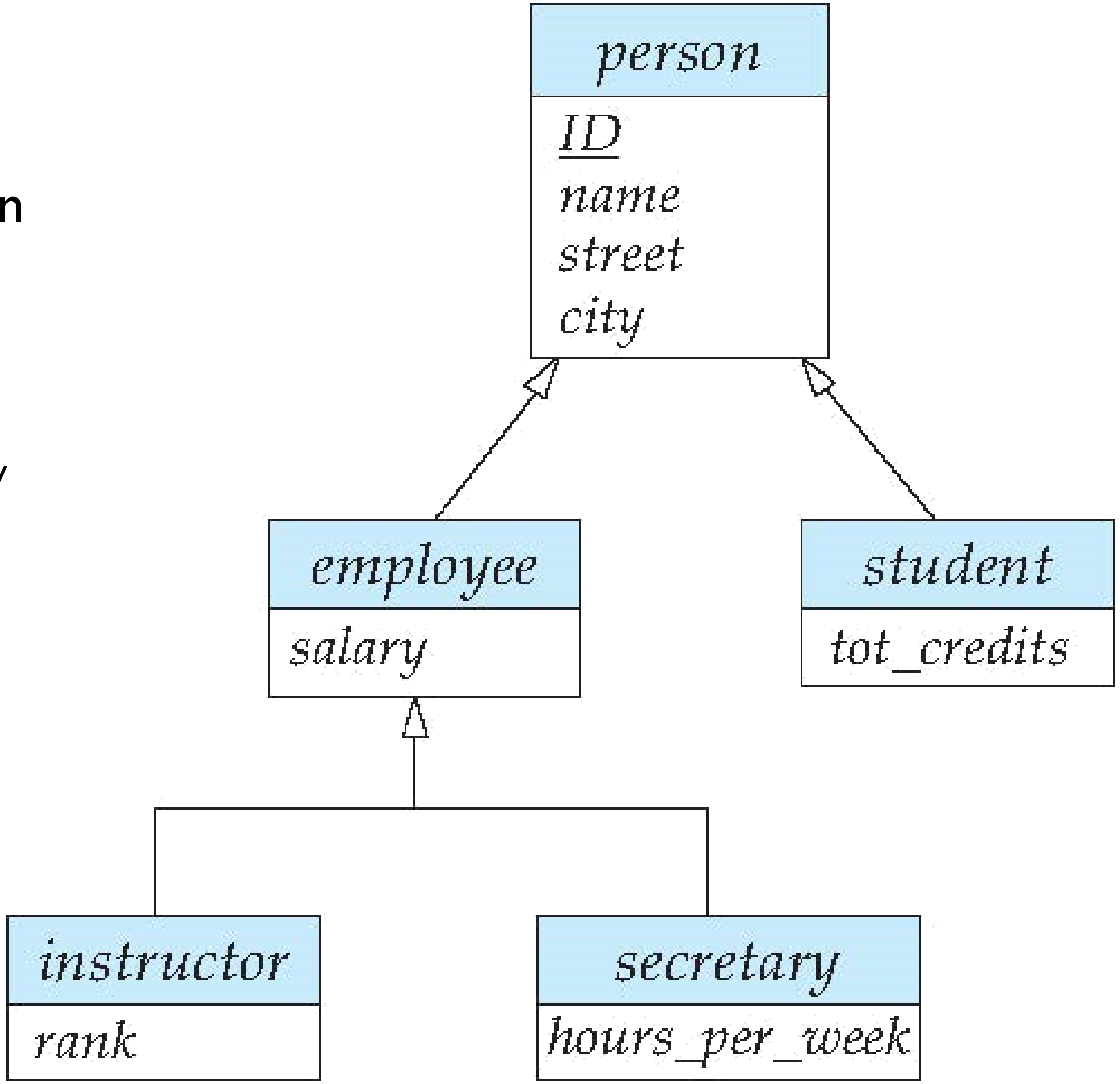
- Top-down design process; we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* 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.

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.

Example

- Can have multiple specializations based on different features.
  - *Permanent* vs. *temporary* in addition to *instructor* vs. *secretary*
  - A particular employee can *permanent/ temporary* and a *instructor/secretary.*
- **Overlapping** – *employee* and *student*
- **Disjoint** – *instructor* and *secretary*



Design Constraints on Specialization

- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - E.g.: all customers over 65 years are members of senior citizen entity set
  - user-defined – DB user assigns members
- 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
  - Overlapping - entity can belong to more than one lower level entity set

Design Constraints on a Specialization

- 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
    - E.g. : Student → grad/ undergrad
  - **partial**: an entity need not belong to one of the lower-level entity sets
    - E.g. : Employee - Employee teams if assigned after 3 months

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

Method 2:

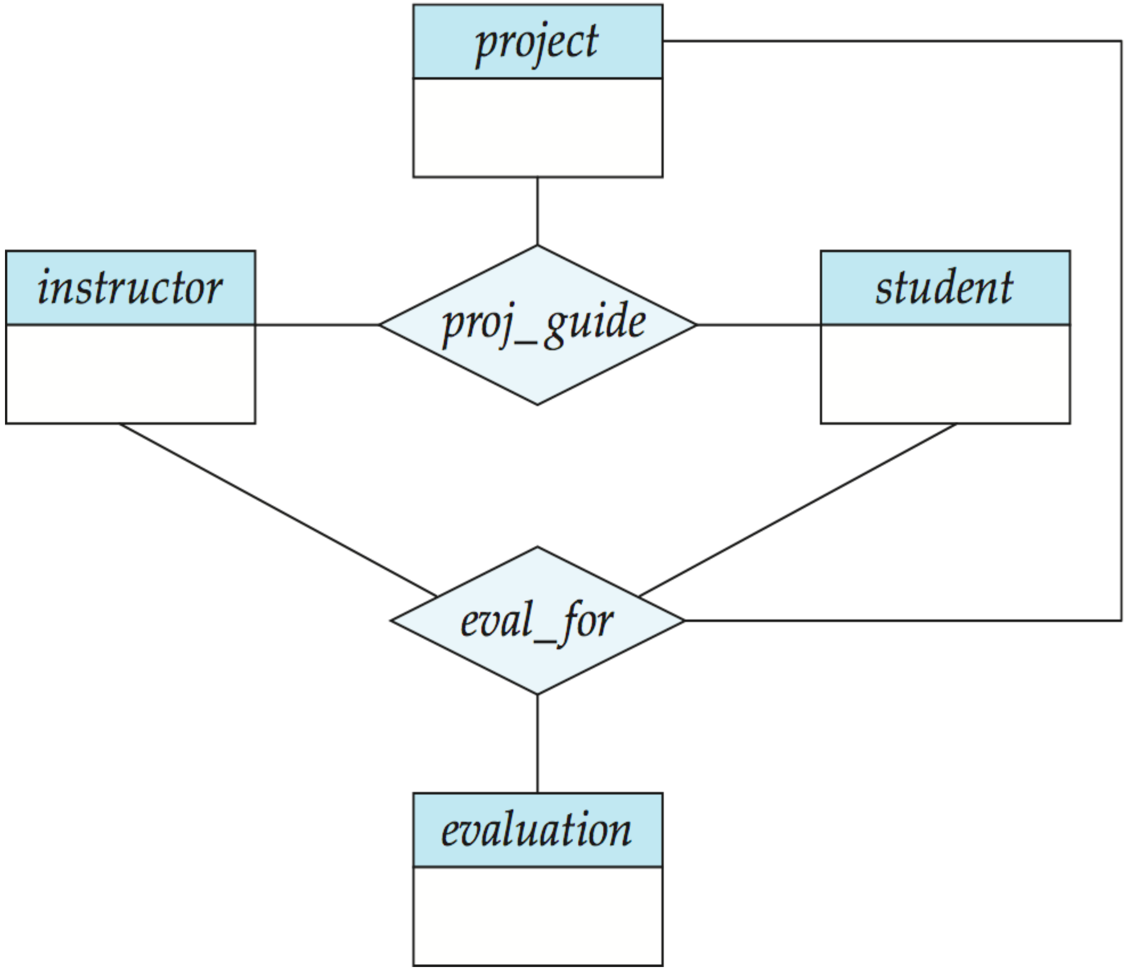
- Form a schema for each entity set with all local and inherited attributes

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
  - But explicit schema may still be needed for foreign key constraints
- Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

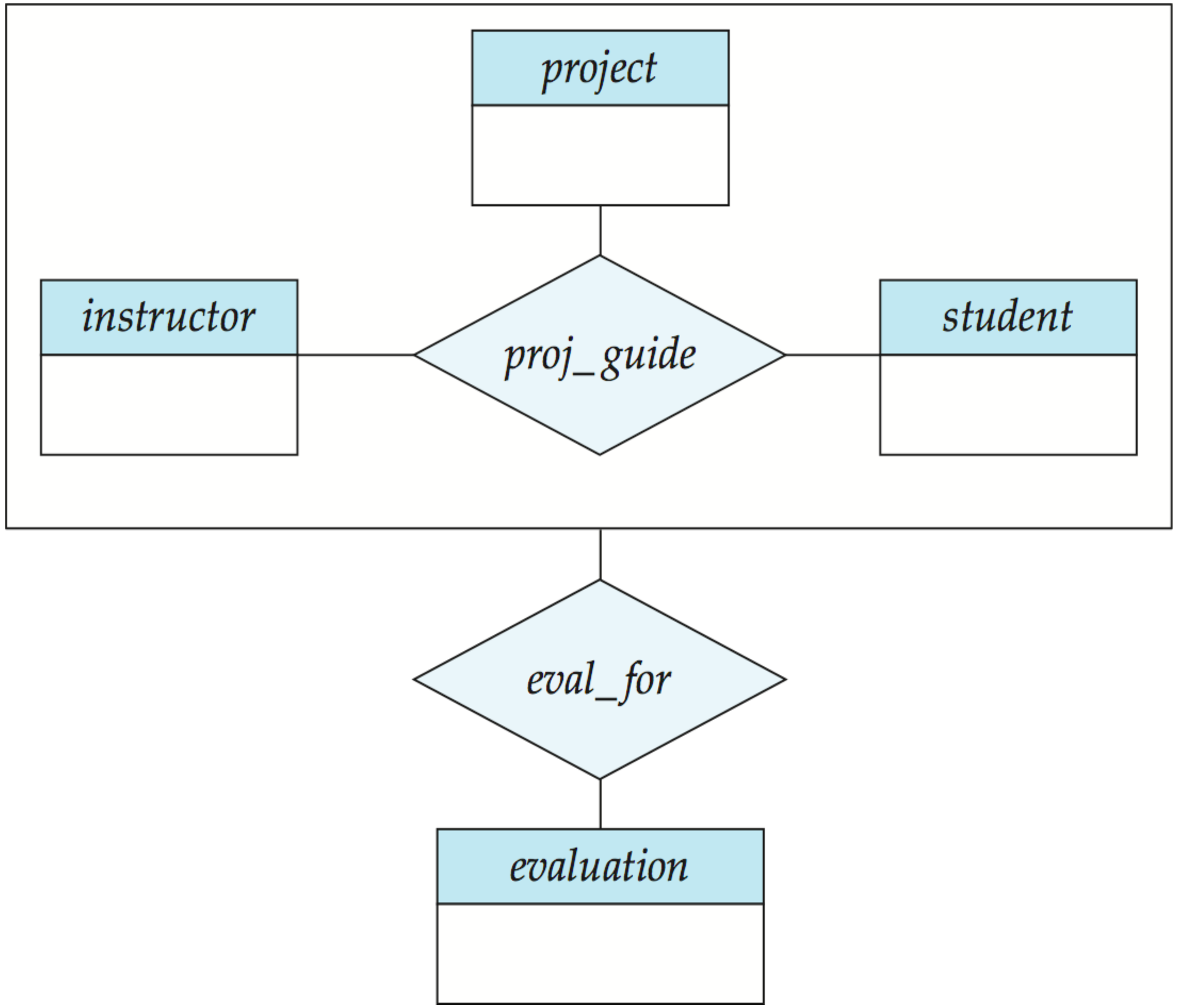
Aggregation

- Suppose a *student* can have at most one *instructor* as guide on a *project*, represented by a ternary relationship *proj_guide*.
- Suppose we want to record evaluations of a student by a guide on a project
- Relationship sets *eval_for* and *proj_guide* represent overlapping information
  - Every *eval_for* relationship corresponds to a *proj_guide* relationship
  - Some *proj_guide* relationships may not correspond to any *eval_for* elationships
  - So we can't discard the *proj_guide* relationship



Aggregation

- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity
- Using Aggregation
  - 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

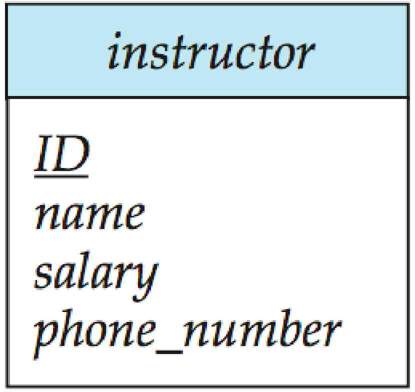
- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- In our example:
  - The schema *eval_for* is *(s_ID, project_id, i_ID, evaluation_id)*
  - Depending participation in the aggregated relationship schema *proj_guide* can be redundant.

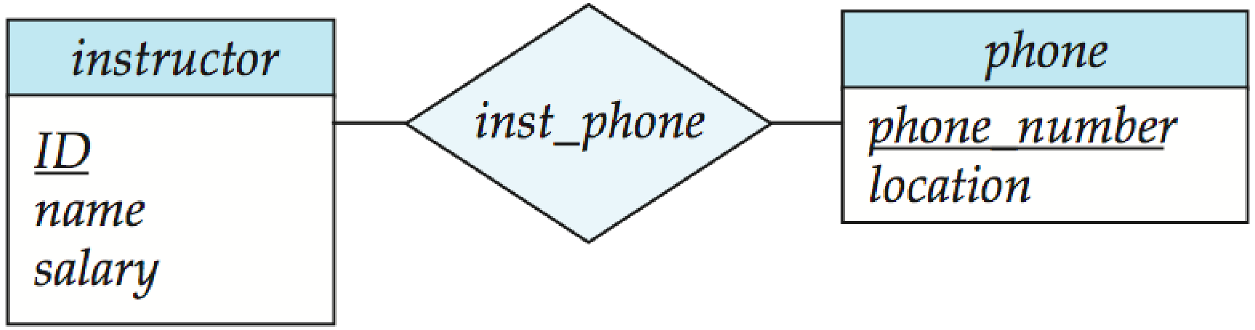
E-R Design Decisions

- 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.

Design Issues

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





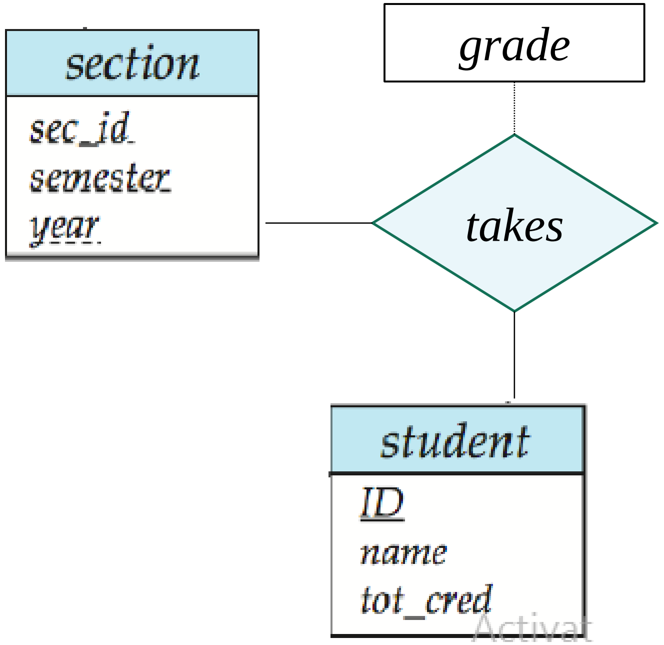
- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

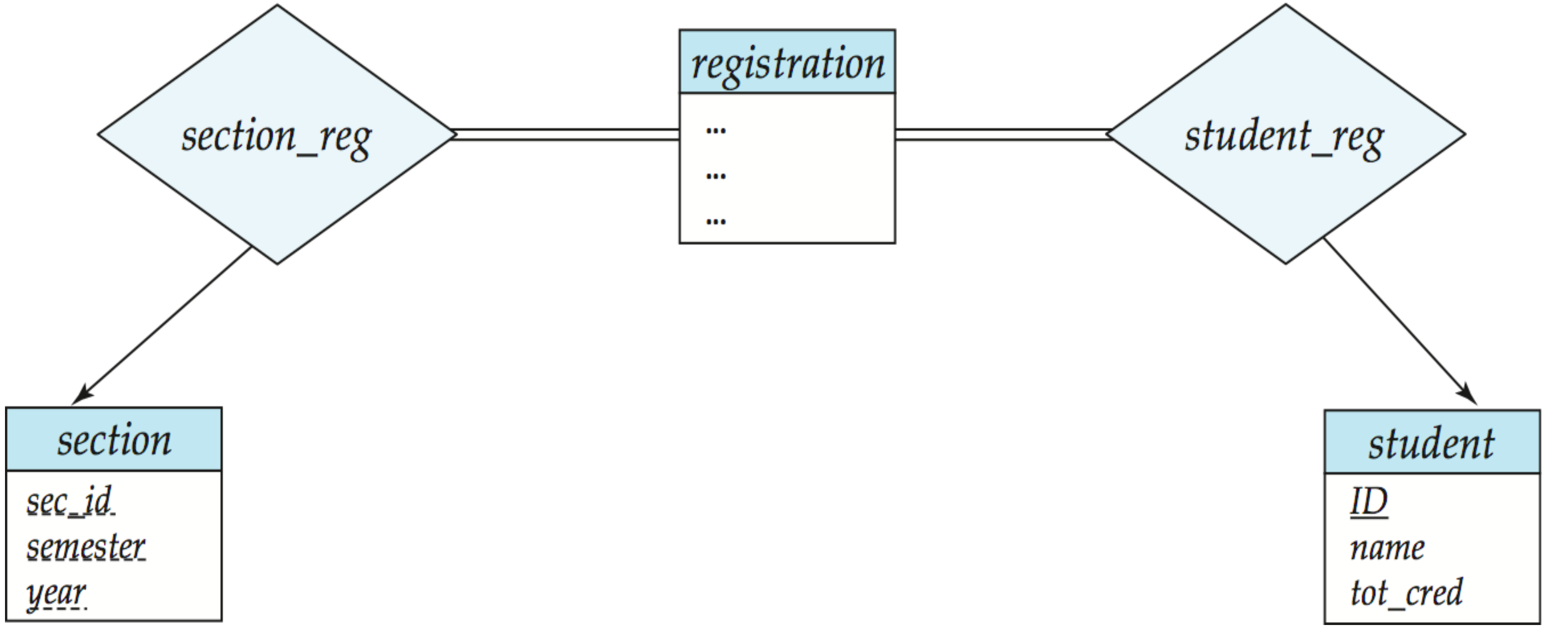
Design Issues

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

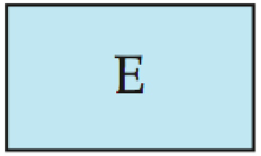
  Possible guideline is to designate a relationship set to describe an action that occurs between entities

  Decide based on placement of relationship attributes

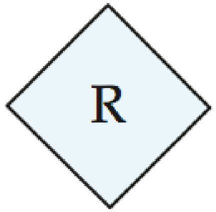




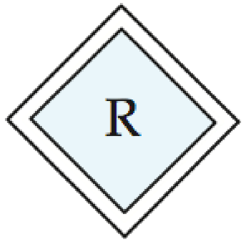
Symbols Used in E-R Notation



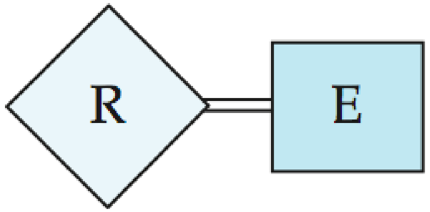
entity set



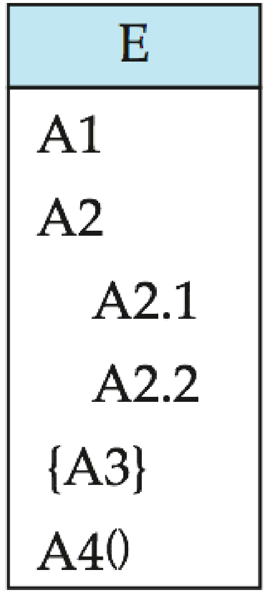
relationship set



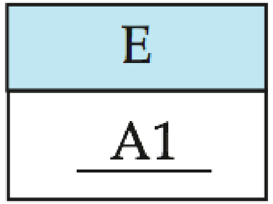
identifying relationship set for weak entity set



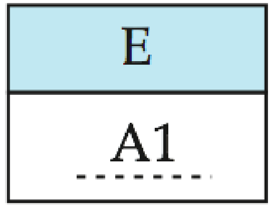
total participation of entity set in relationship



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

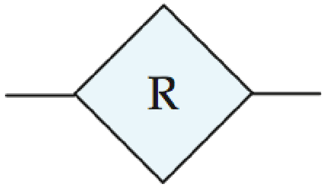


primary key

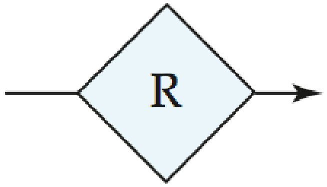


discriminating attribute of weak entity set

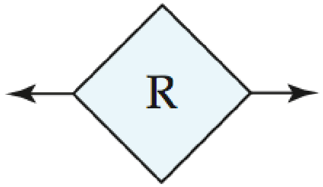
Symbols Used in E-R Notation



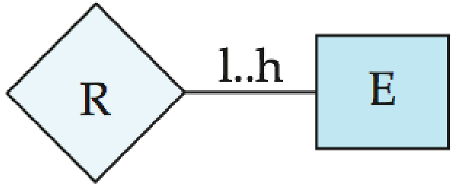
many-to-many relationship



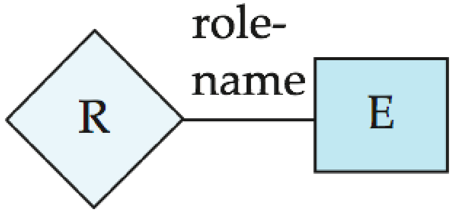
many-to-one relationship



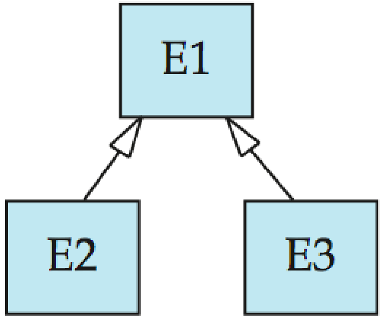
one-to-one relationship



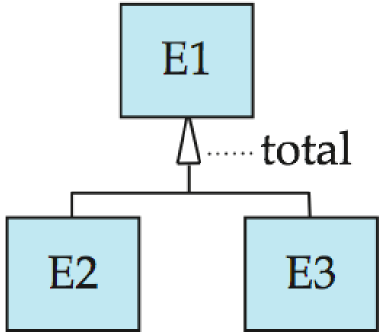
cardinality limits



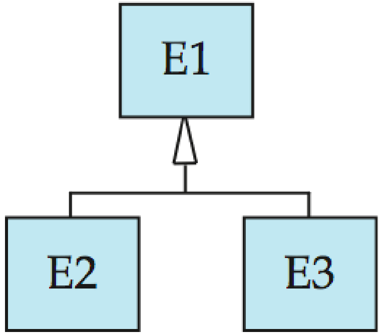
role indicator



ISA: generalization or specialization



total (disjoint) generalization

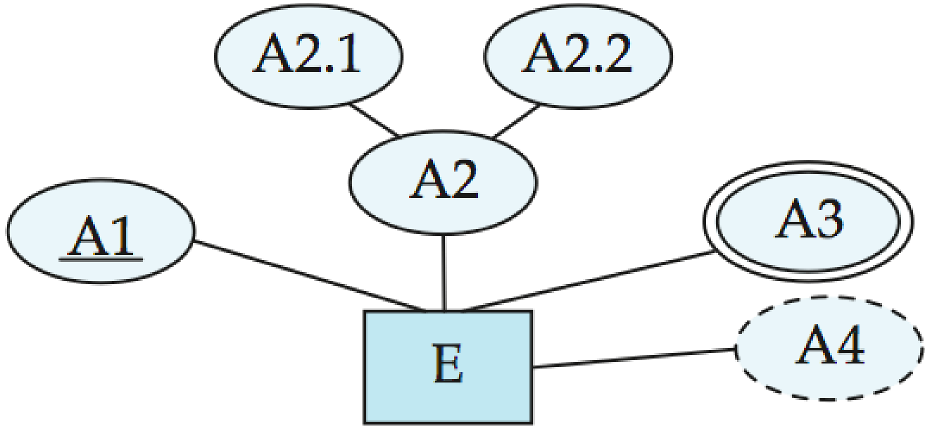


disjoint generalization

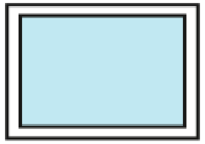
Alternative E-R Notation

- Chen, IDE1FX, …

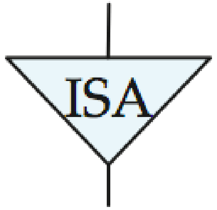
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1



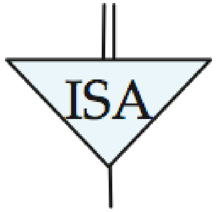
weak entity set



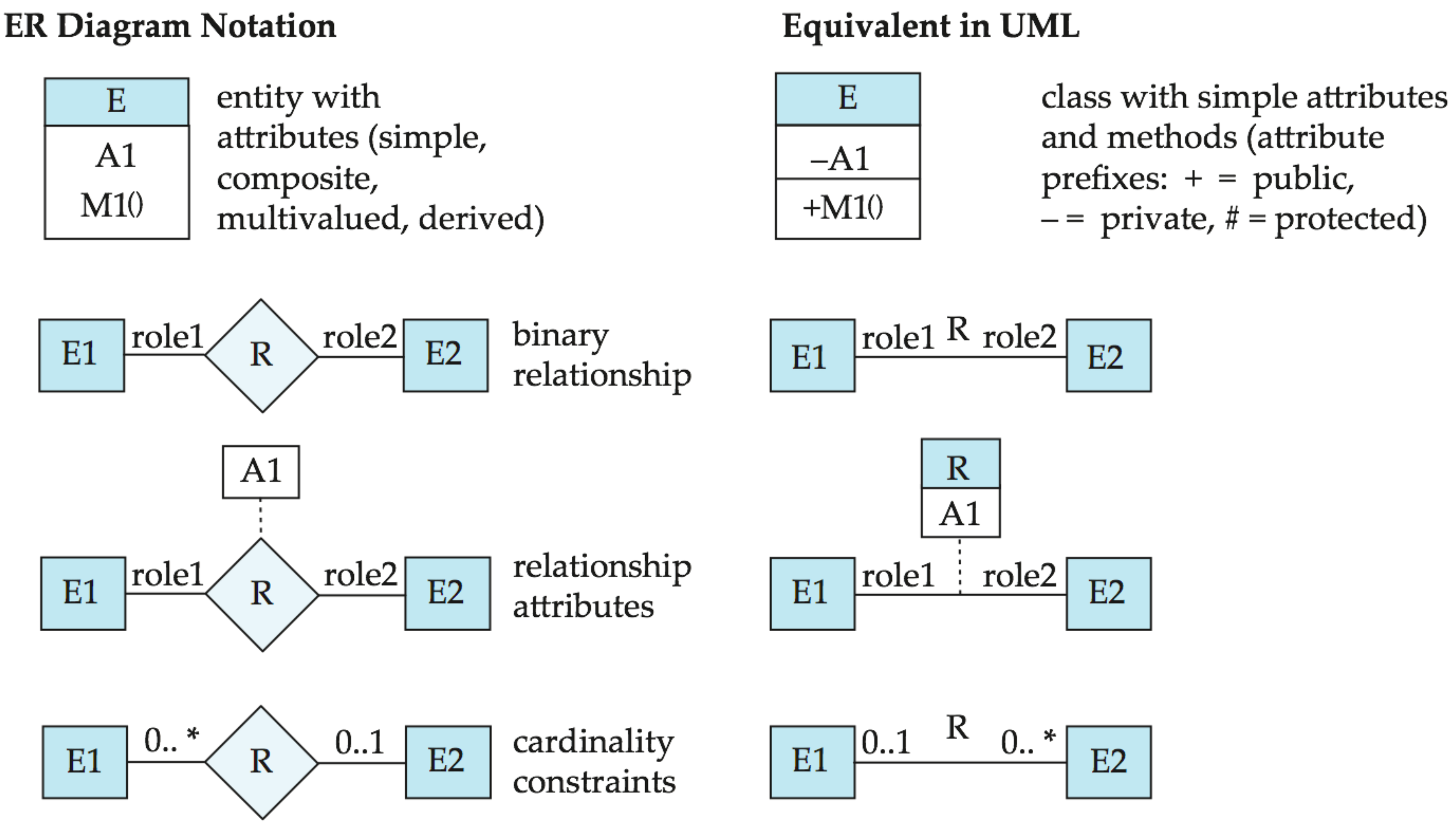
generalization



total
generalization



ER vs. UML Class Diagram



ER Diagram Notation

Equivalent in UML

entity with attributes (simple, composite, multivalued, derived)

class with simple attributes and methods (attribute prefixes: + = public, – = private, # = protected)

binary relationship

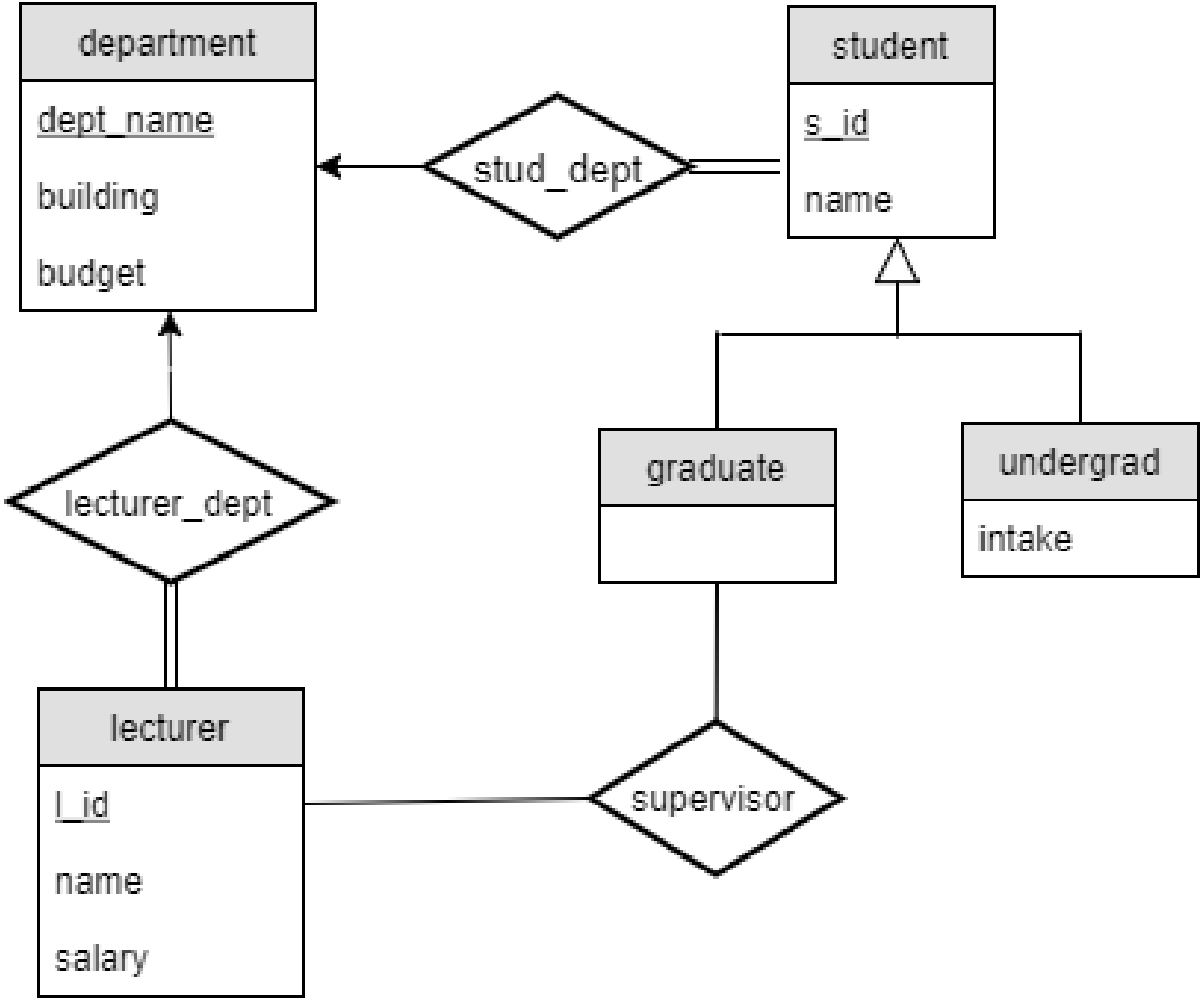
relationship attributes

cardinality constraints

*Note reversal of position in cardinality constraint depiction

Example

- Convert the following ER diagram into relational schema.
- Note that:
- All students have a department
- Only graduates can have supervisors
- Undergraduates have a specific intake
- Specialization is partial and disjoint
- supervisor is a many-to-many relationship



Example (Cont.)

department (dept_name, building, budget)

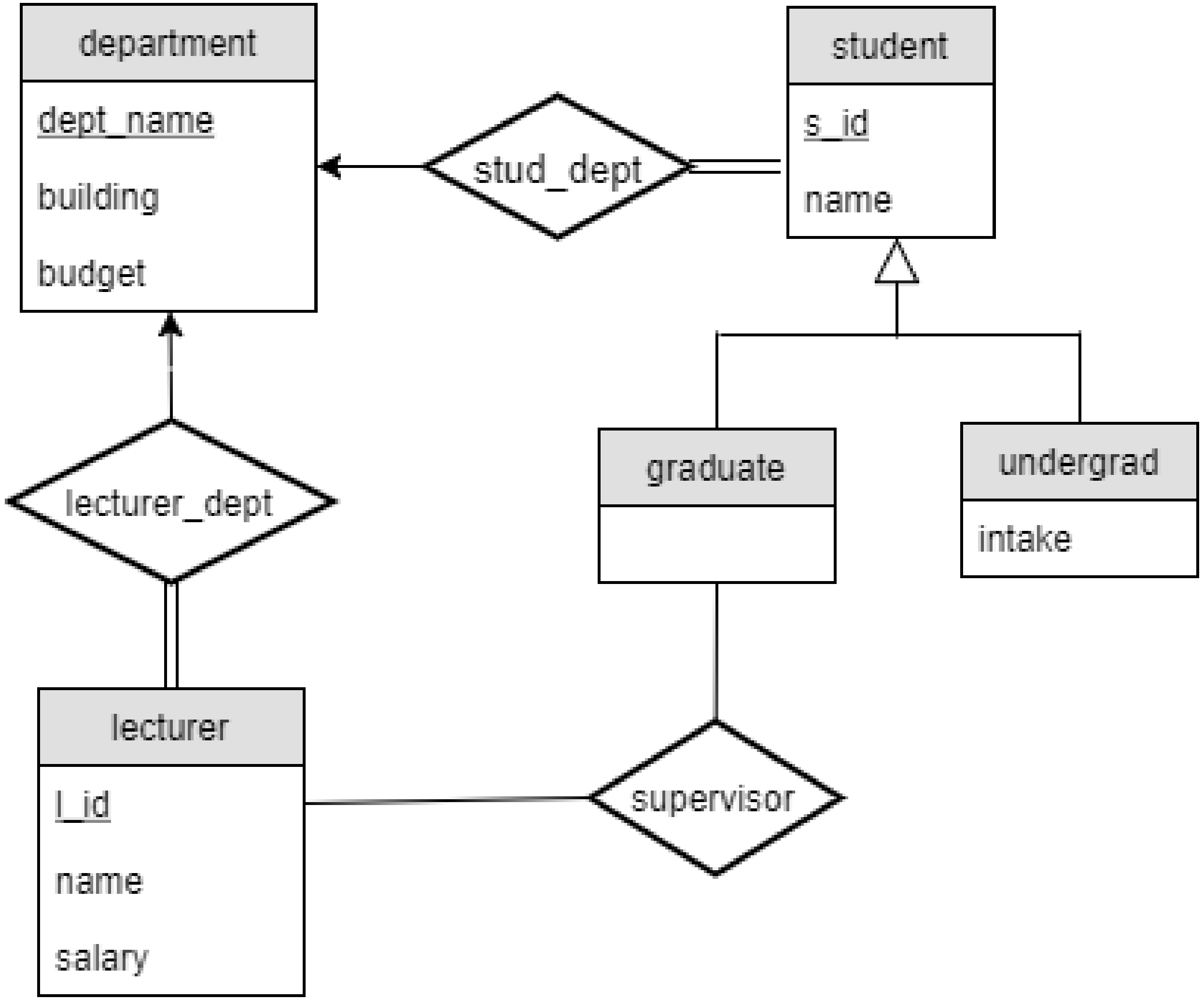
lecturer (l_id, name, salary, dept_name)

student (s_id, name, dept_name)

undergrad (s_id, intake)

graduate (s_id)

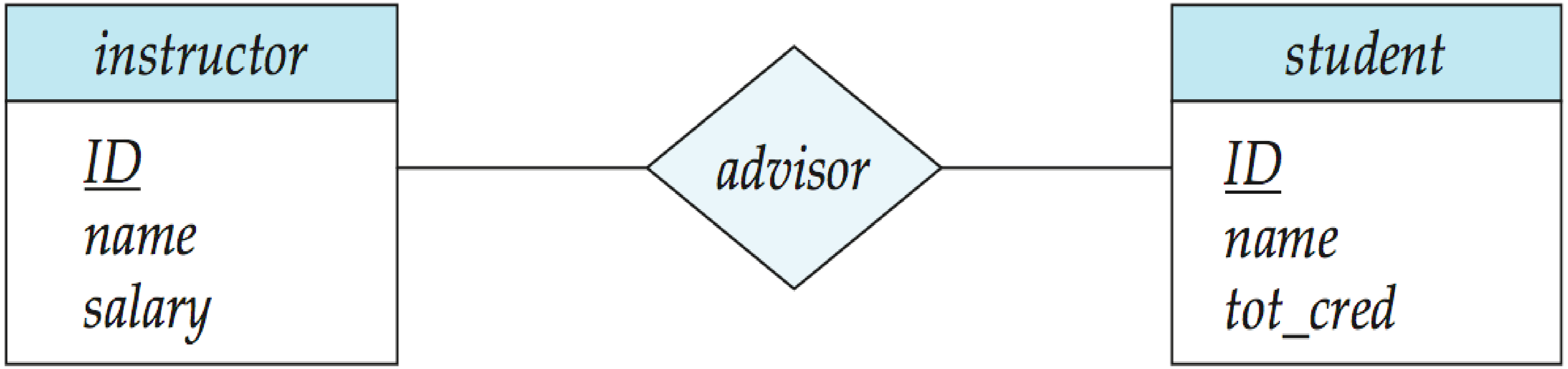
supervisor(l_id, s_id)



*We have used Method 1 to represent specialization

Exercise 3

- Modify the following ER diagram so as to enable it to keep track of time. The data model should be able to keep track of time periods when a student had two different/same instructors as advisors. It should be able to keep track of time periods when different salary revisions were valid, and changes to total credits.
- Convert the above modified ER diagram to a set of relations.



- Solution: represent each required attributes as a multi-valued composite attribute that contains attribute *value, valid_start_date* and *valid_end_date* as components.

H/W

- Start ER Diagram for the Group project!

Thank you!

Questions?