

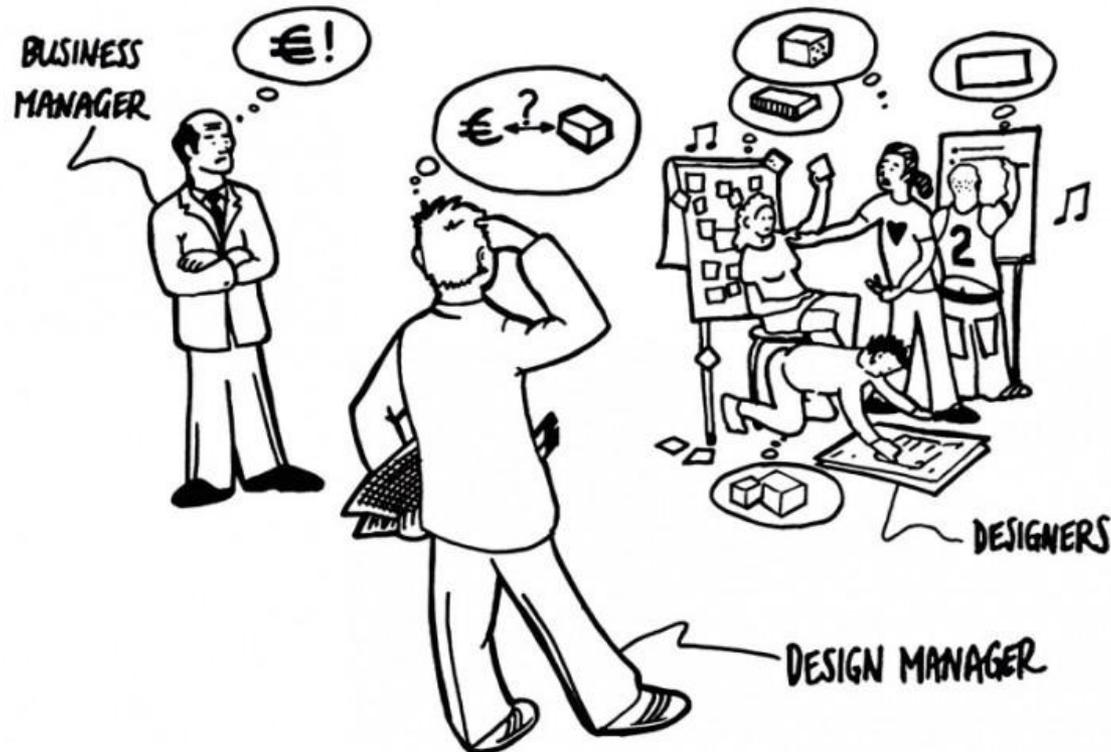
Entity Relationship Models

CS460 Databases for Data Scientists



Today

- Designing databases
 - i.e., how to get from your customer's requirements...
 - ... to a set of tables and attributes



Outline

- Design Process
- Cardinality Constraints
- ER Diagrams
- Reduction to Relation Schemas
- Design Decisions

Database Design

- Initial phase: requirements engineering
 - characterize fully the data needs of the prospective database users
 - which data needs to be stored?
 - ... and in which volumes?
 - which queries should be answered?
- Conceptual schema
 - which types of entities and relations exist?
 - what attributes do they have?

Database Design

- Final phase: from a conceptual to physical data model
 - Logical Design: find a “good” collection of relation schemas
 - Business decision – What attributes should we record in the database?
 - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
 - Physical Design – Deciding on the physical layout of the database

Database Design Approaches

- **Today**
Entity Relationship Model
 - Models an enterprise as a collection of entities and relationships
 - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
 - Described by a set of attributes
 - Relationship: an association among several entities
 - Represented diagrammatically by an entity-relationship diagram
- **Next lecture**
Normalization Theory
 - Formalize what designs are bad, and test for them

Entity Relationship Model

- Dates back to the 1970s
 - Chen, Peter Pin-Shan: *The Entity–Relationship Model – Toward A Unified View of Data.*
ACM Transactions on Database Systems. 1(1): 9–36, 1976
 - Developed to facilitate database design by allowing the specification of an enterprise schema that represents the overall logical structure of a database
- The ER data model employs three basic concepts:
 - Entity sets,
 - Relationship sets,
 - Attributes



Entity Sets

- An entity is an object that exists and is distinguishable from other objects
 - Example: Peter Chen, Mannheim, Star Wars
- An entity set is a set of entities of the same type that share the same properties
 - Example: set of all persons, cities, movies
- Each entity is represented by a *set of attributes*
 - Example:
instructor = (ID, name, street, city, salary)
course= (course_id, title, credits)
- A subset of the attributes form a *primary key* of the entity set i.e., uniquely identifying each member of the set

Entity Sets – Example

- instructor (instructor_id, instructor_name)
- student (student_id, student_name)

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

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

Example:

44553 (Peltier) advisor 22222 (Einstein)
student entity relationship set instructor entity

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

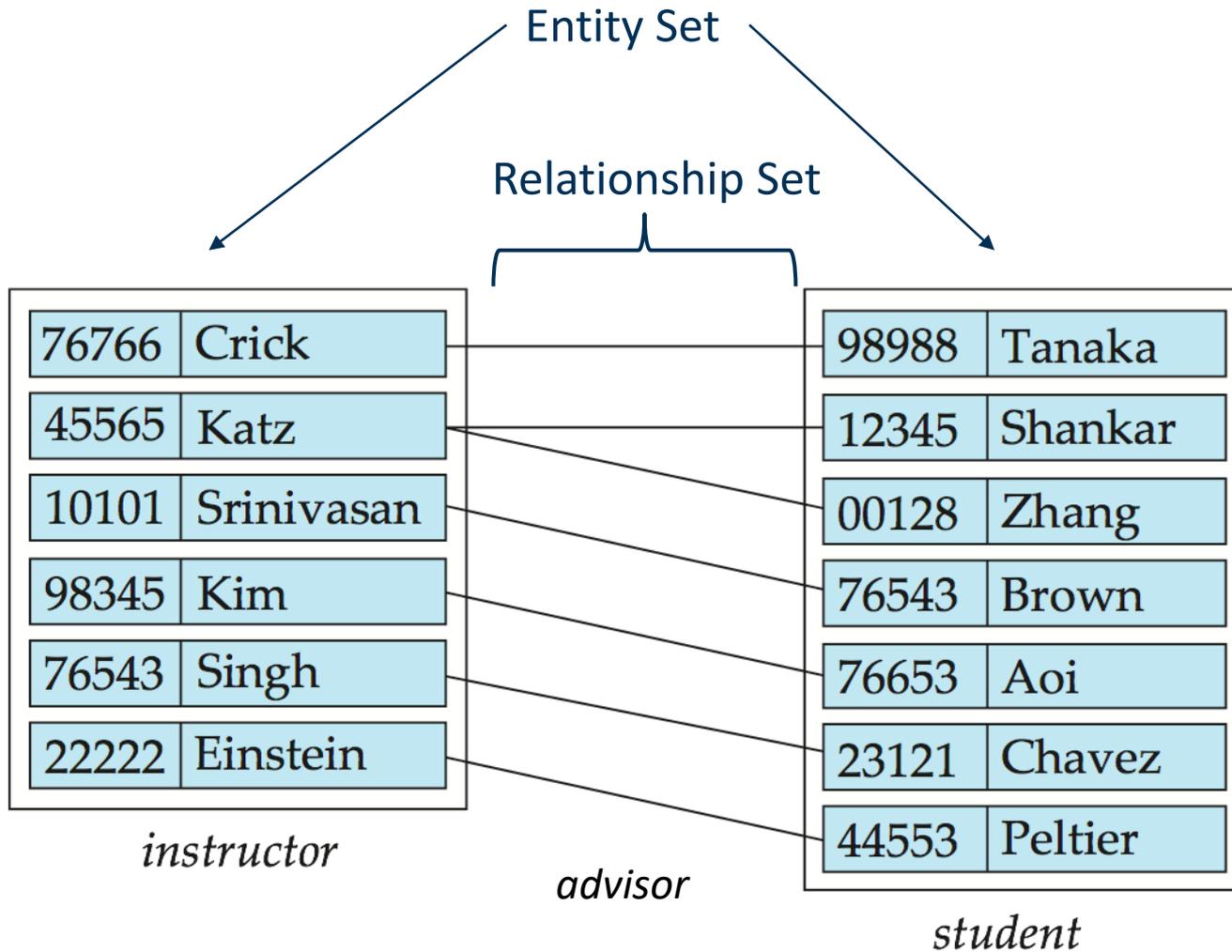
$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$

where (e_1, e_2, \dots, e_n) is a relationship

- Example:

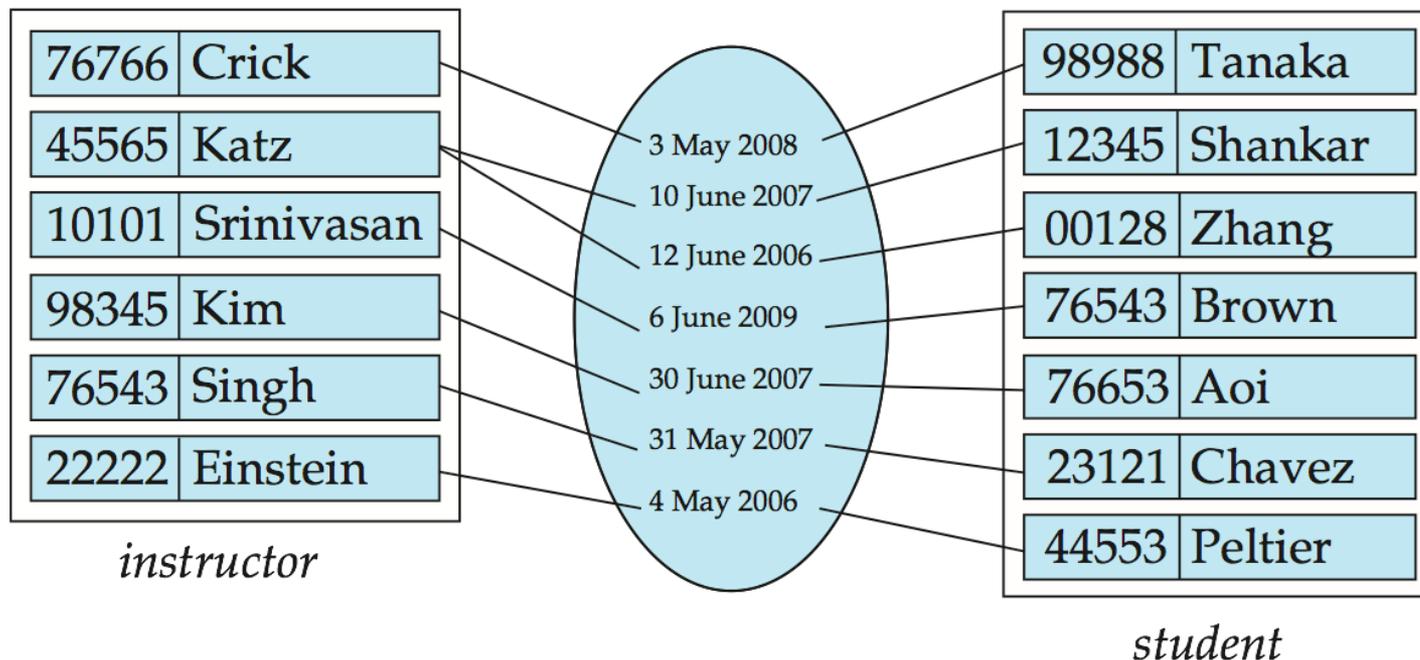
$(44553, 22222) \in \text{advisor}$

Relationship Sets



Relationship Sets

- An attribute can also be associated with a relationship set
- E.g., *advisor* relationship set:
 - date which captures the start of the supervision



Degree of a Relationship

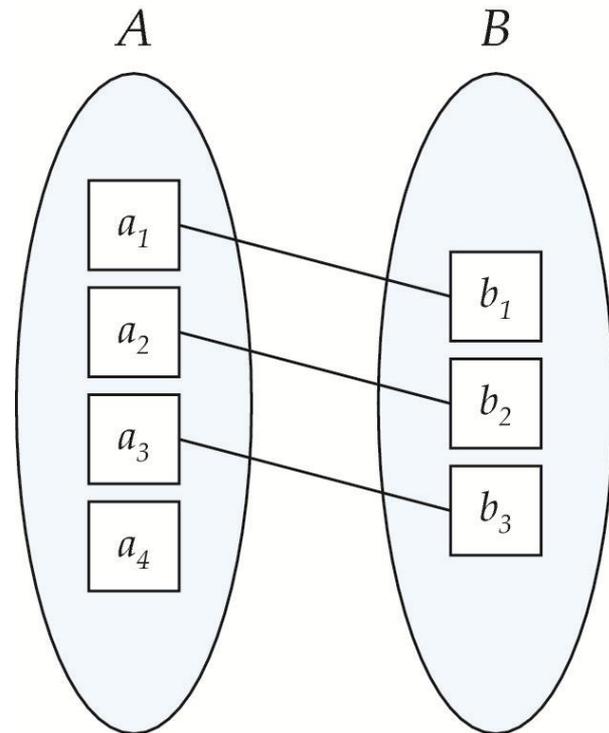
- Definition: degree of a relationship
i.e., number of entity sets that are involved in relation set
- Binary relationship (degree two)
 - Involve two entity sets
 - The by far most frequent case
- Relationships between more than two entity sets (degree >2)
 - e.g.: students work on projects under the guidance of an instructor
 - Relationship `proj_guide` is a *ternary relationship* between instructor, student, and project
 - Those are rather rare

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:
 - 1:1 (one to one)
 - 1:n (one to many)
 - n:1 (many to one)
 - n:m (many to many)

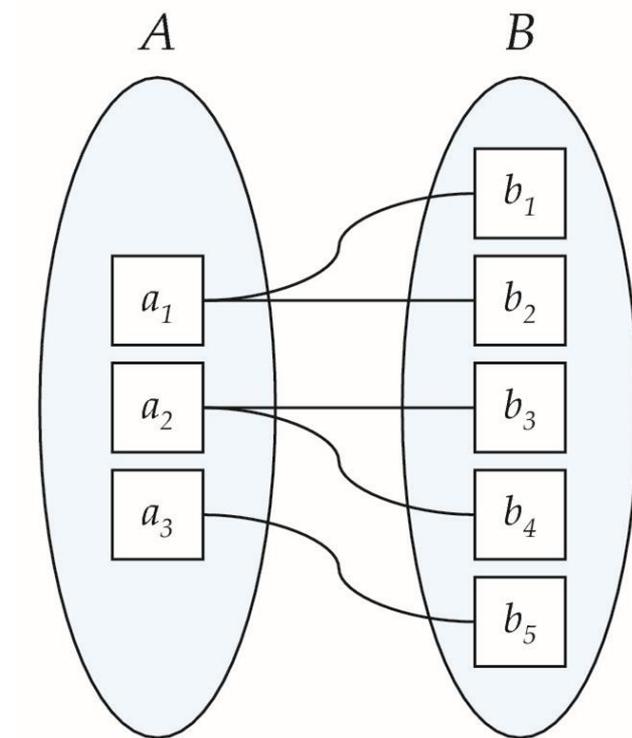
Mapping Cardinalities – One to One

- One to one (1:1)
 - Note: Some elements in A and B may not be mapped to any elements in the other set
- Examples
 - student_works_on_thesis
 - department_has_dean



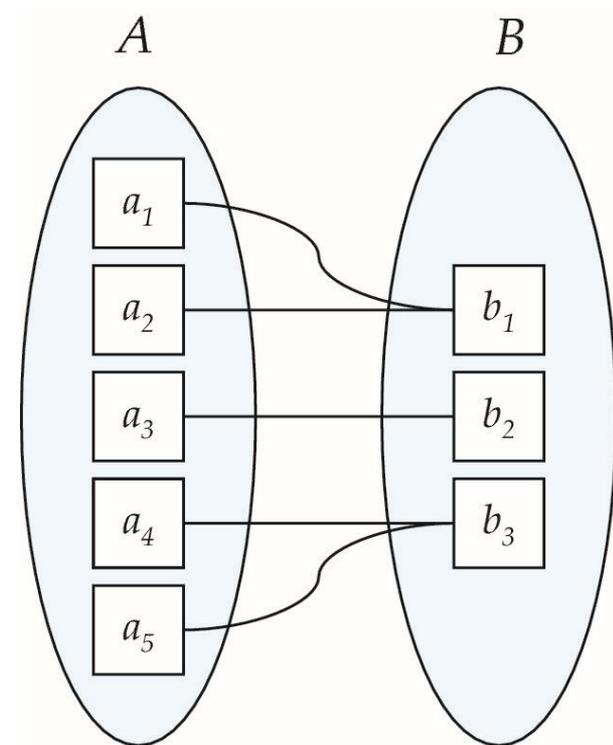
Mapping Cardinalities – One to Many

- One to many (1:n)
 - Note: Some elements in A and B may not be mapped to any elements in the other set
- Examples
 - building_has_room
 - faculty_has_member



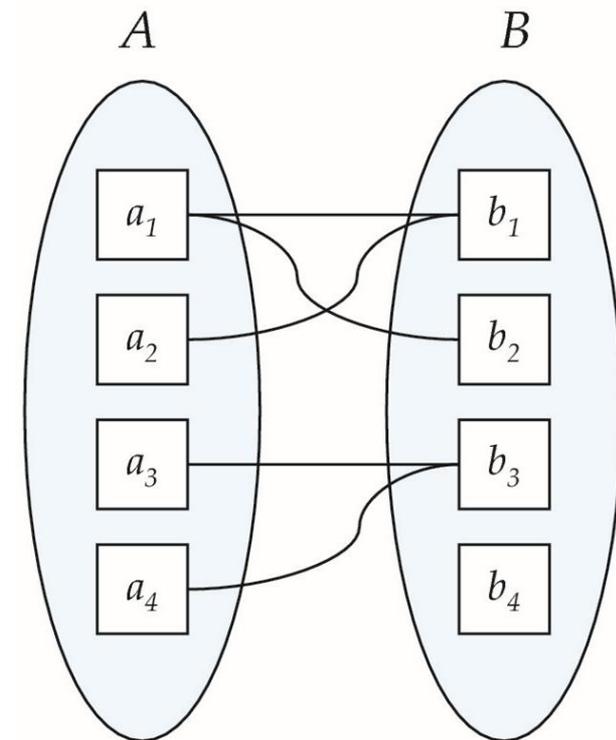
Mapping Cardinalities – Many to One

- Many to one (n:1)
 - Note: Some elements in A and B may not be mapped to any elements in the other set
- Examples
 - room_located_in_building
 - member_of_faculty
- Note:
 - the inverse of a 1:n relation is a n:1 relation
 - and vice versa



Mapping Cardinalities – Many to Many

- Many to many (n:m)
 - Note: Some elements in A and B may not be mapped to any elements in the other set
- Examples
 - student_takes_course
 - student_has_advisor



Distinguishing 1:n/n:1 Cardinalities and n:m Cardinalities

- Rule of thumb
 - Always ask for the cardinality the other way around
- “A building may have multiple rooms...”
 - “...but can a room be in multiple buildings?”
 - No → building_has_room is 1:n
- “A department can be located in multiple buildings...”
 - “...but can a building host multiple departments?”
 - Yes → department_located_in_building is n:m

Relation Sets from the Same Entity Set

- The two entity sets in a relation set may be the same
- This holds independently from the cardinality!

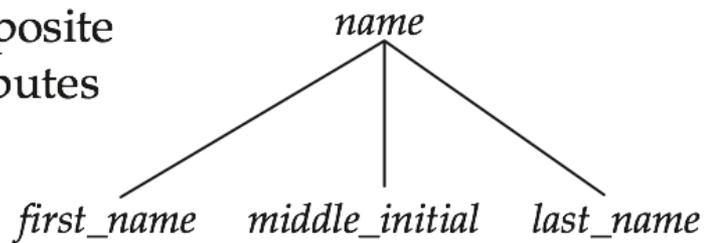
- person_married_to_person
 - 1:1
- person_is_father_of_person
 - 1:n
- person_has_father
 - n:1
- person_is_parent_of_person
 - n:m

Attribute Types & Domains

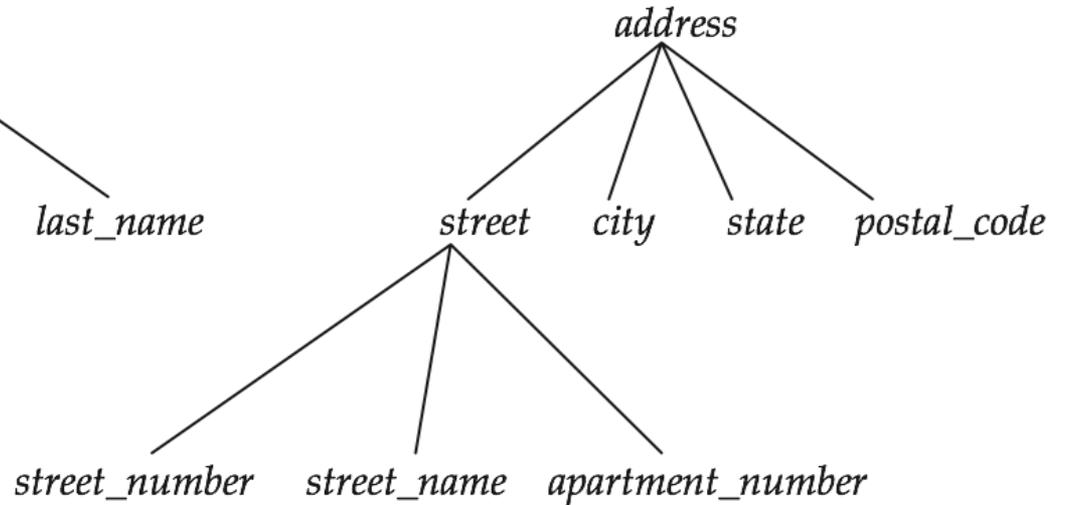
- Attribute types:
 - Simple and composite attributes
 - Single-valued and multi-valued attributes
 - Example: multi-valued attribute: phone_numbers
- Derived attributes
 - Can be computed from other attributes
 - Example: age (given date_of_birth)
- Domain – the set of permitted values for each attribute

Composite Attributes

composite
attributes



component
attributes



Weak Entity Sets

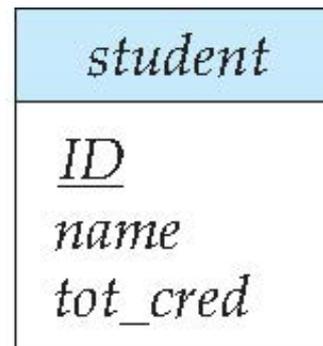
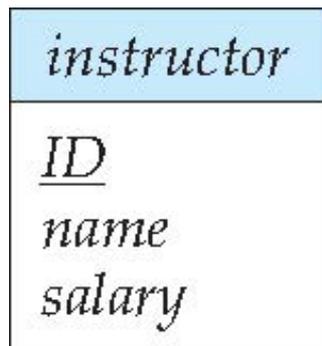
- Consider the set of buildings and rooms
 - Entity set building(building_name,address)
 - Entity set room(number,capacity)
 - Relation set room_in_building (number,building_name)
- Question:
 - What is the primary key of the the entity set *room*?

Weak Entity Sets

- Weak entity sets are entity sets that
 - Do not have a set of attributes sufficient to identify each entity uniquely
 - Require an additional relation set to identify each entity uniquely
- Those relation sets are called *identifying relation set*
- Weak entities do not have primary keys
 - A weak entity set has an *identifying entity* and a *discriminator*
 - Example: room(number, capacity)
 - *Building* is the identifying entity
 - *Number* is the discriminator
- A weak entity cannot exist without the identifying entity
 - e.g., a room cannot exist without the building

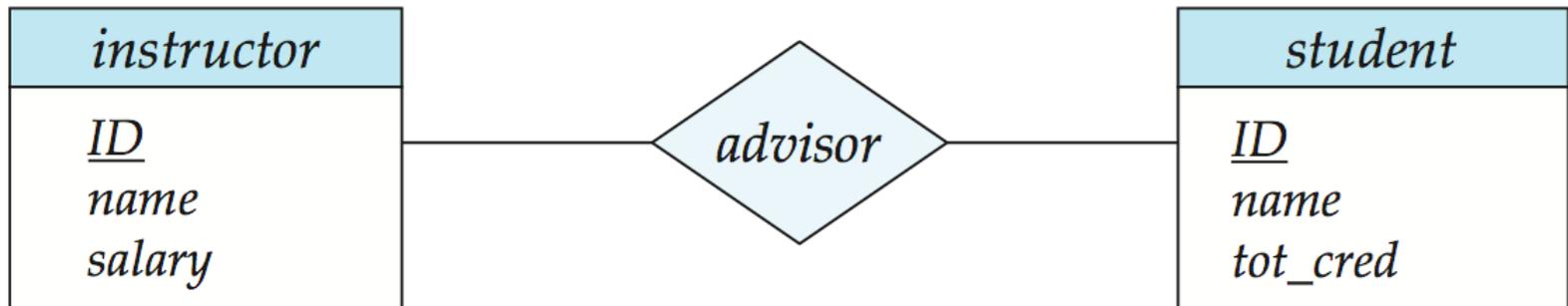
ER Diagrams

- Entity Relationship Diagrams (ER diagrams)
 - Are the graphical notation of entity relationship models
- Notation of entity sets:
 - Rectangles represent entity sets
 - Attributes listed inside entity rectangle
 - Underlining indicates primary key attributes



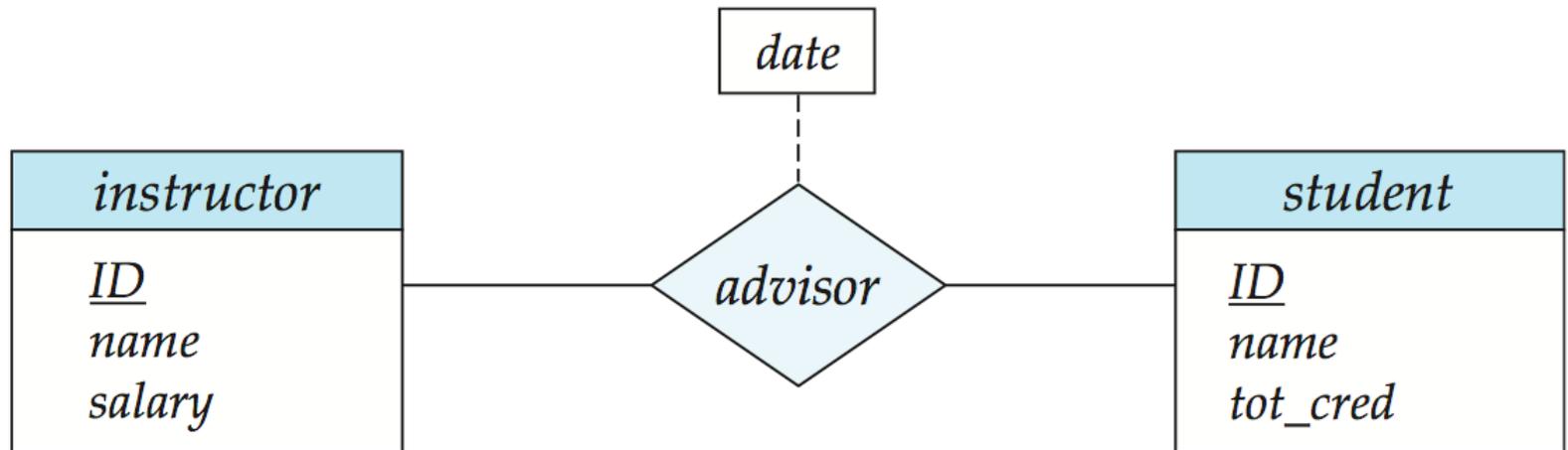
ER Diagrams

- Diamonds represent relationship sets



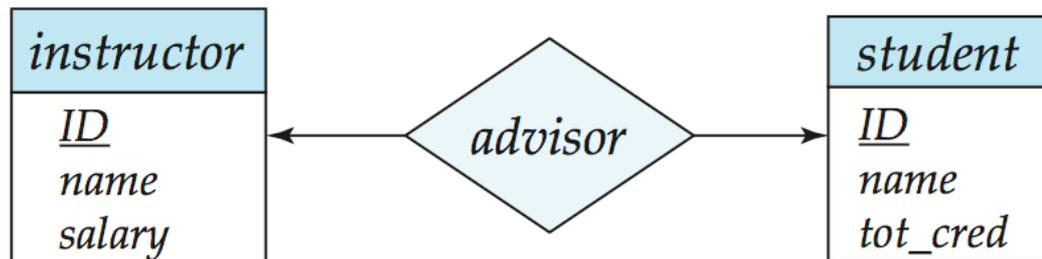
ER Diagrams

- Diamonds represent relationship sets
 - Attributes can be attached to relationship sets



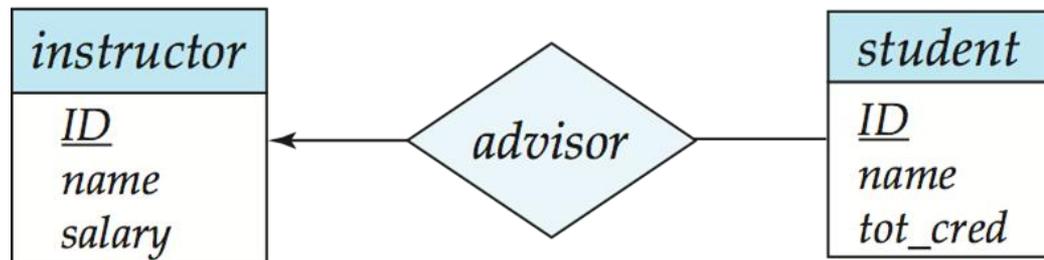
Cardinalities in ER Diagrams

- We express cardinality constraints by drawing either a directed line (\rightarrow), 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
 - An instructor is associated with at most one student via the relationship advisor



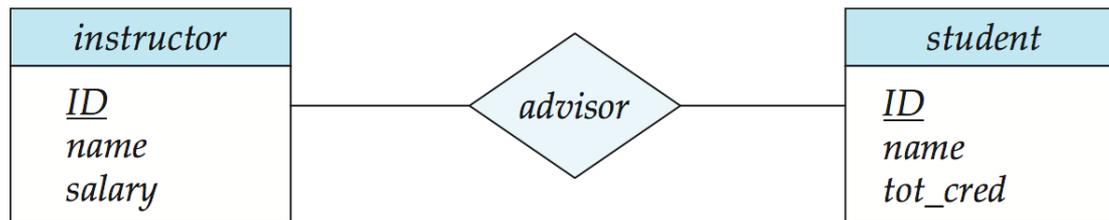
Cardinalities in ER Diagrams

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



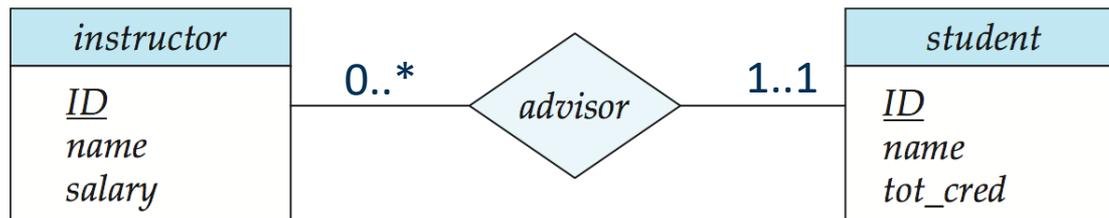
Cardinalities in ER Diagrams

- Many to many relationships
 - An instructor is associated with several (possibly 0) students via advisor
 - A student is associated with several (possibly 0) instructors via advisor



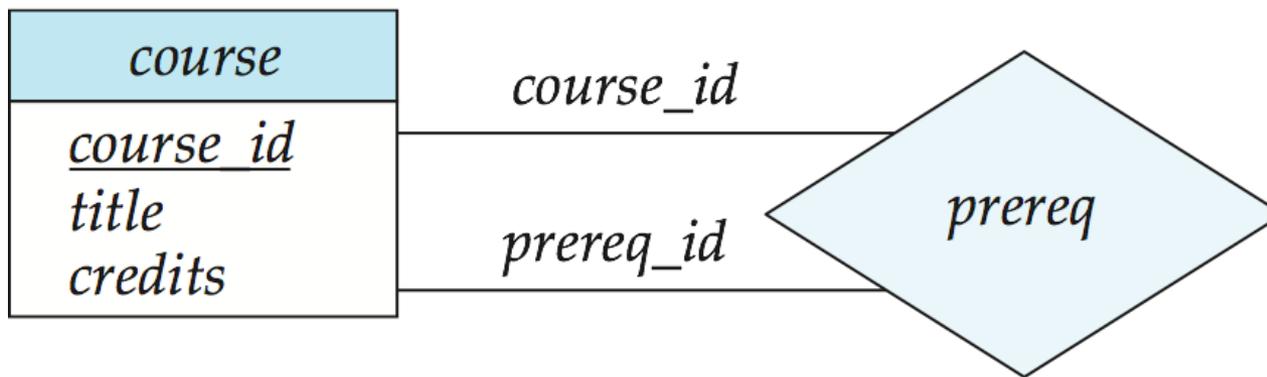
Complex Cardinality Constraints

- Notation for minimum/maximum cardinality of a relation
 - Each student has *exactly one* advisor (i.e., min=max=1)
 - Each instructor can be the advisor of multiple students, but needs not to be (i.e., min=0,max=∞)
- Notation:
 - min..max
 - * indicates no limit



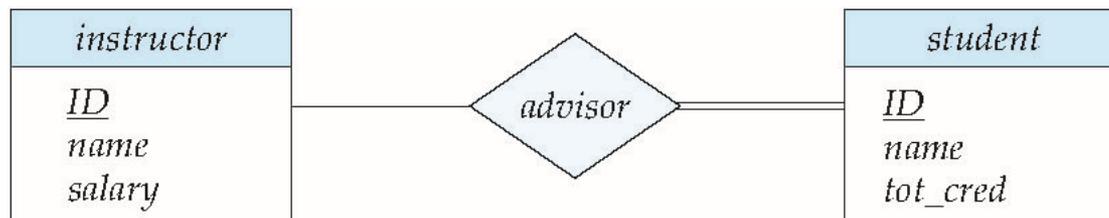
Roles

- Entity sets of a relationship need not be distinct
 - i.e., there may be a relationship set involving the same entity set twice
- Each occurrence of an entity set plays a “role” in the relationship
 - The labels “*course_id*” and “*prereq_id*” are called **roles**

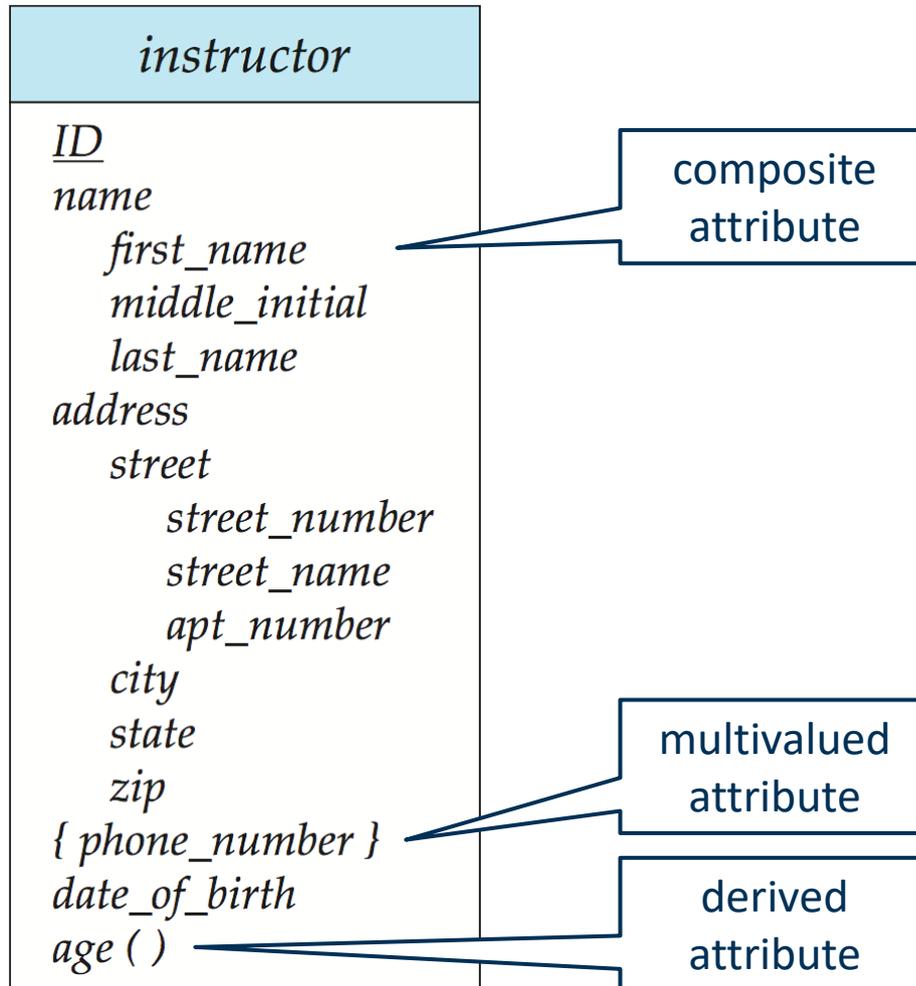


Total and Partial Participation

- Total participation (double line)
 - Every entity in the entity set participates in at least one relationship in the relationship set
 - i.e., every student *must* have an advisor
 - recap: think of *not null* constraints
- Partial participation (single line)
 - Some entities may not participate in the relationship
 - e.g., not every instructor has to supervise a student

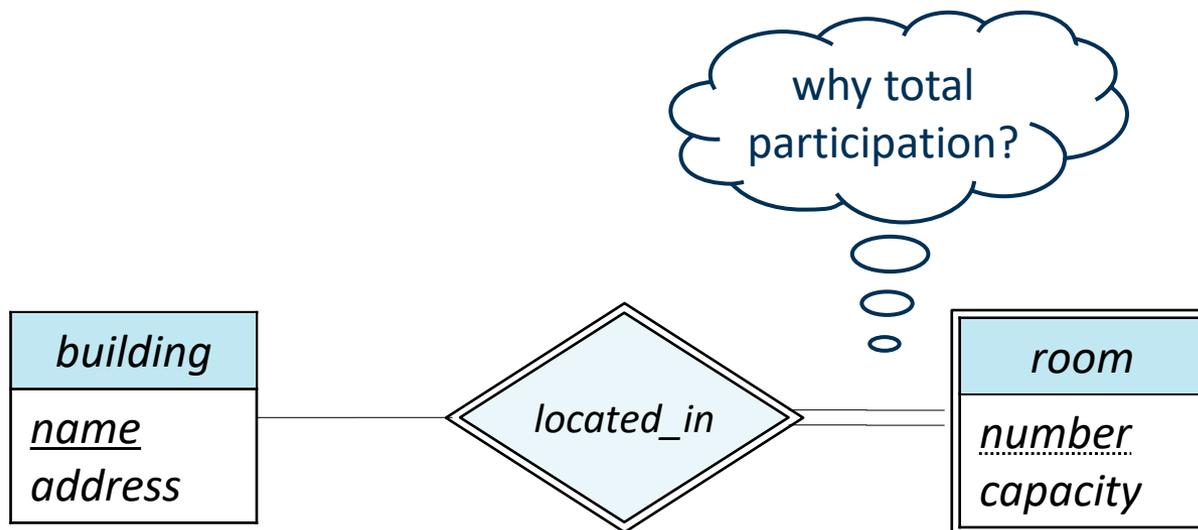


Notation of Attribute Types



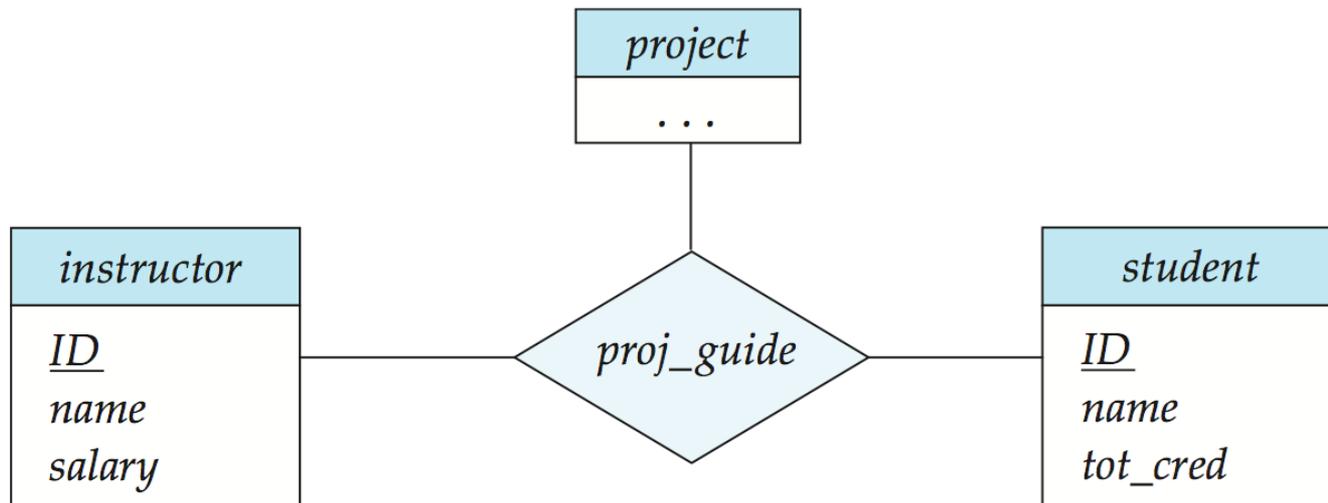
Expressing Weak Entity Sets

- A weak entity set is depicted via a double rectangle
 - The identifying relationship set is depicted by a double diamond
- The *discriminator* is underlined with a dashed line
 - Identifier for room: (number, building.name)



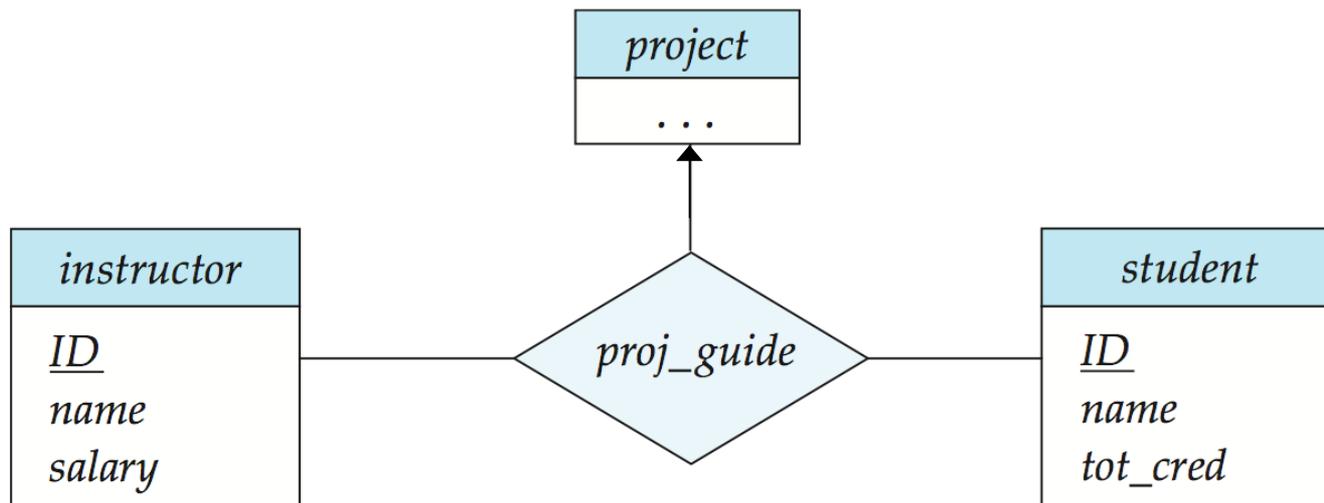
Higher Arity Relationship Sets

- Most relationship sets are binary
- Sometimes, ternary (or higher arity) relations occur
 - ER models support that
- Example:
 - Students work on projects under supervision of an instructor



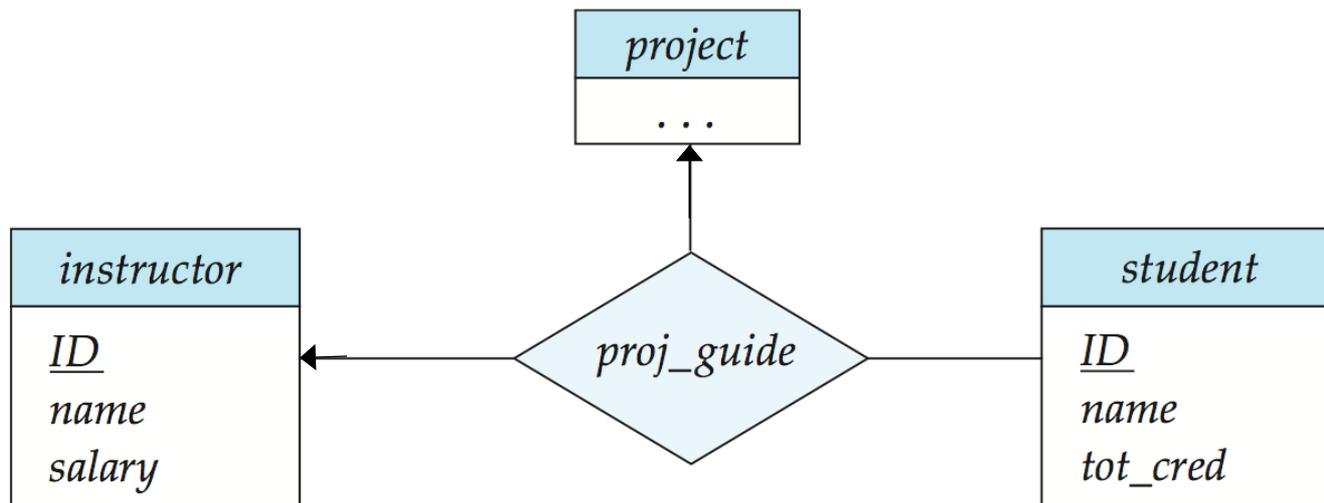
Cardinality Constraints for Ternary Relations

- Only one single arrow (i.e., cardinality restriction) is allowed for a ternary relation
 - Example: each student can work in at most one project under the supervision of some instructor(s)



Cardinality Constraints for Ternary Relations

- Multiple single arrows (i.e., cardinality restrictions) would lead to different possible interpretations
 - Each student works on at most one project under at most one instructor
 - For each project a student works on, there is at most one instructor
 - For each instructor supervising a student, there is at most one project
- Hence, we do not allow for them



Specialization

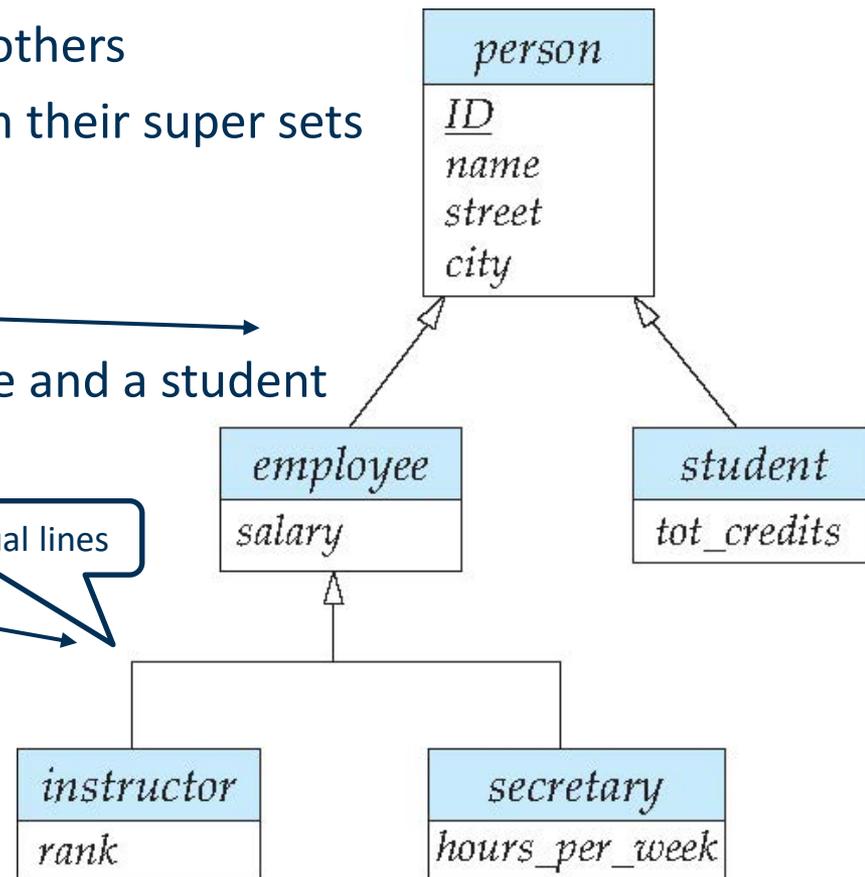
- A concept very common in (object oriented) programming
 - Entity sets are sub-/super sets of others
 - They inherit all the attributes from their super sets

- Overlapping

- A person can be *both* an employee and a student

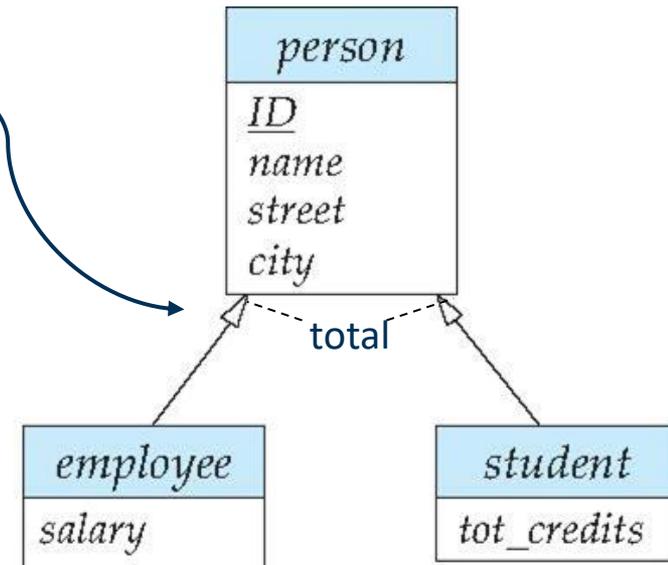
- Disjoint

- An employee can be *either* an instructor *or* a secretary
- An entity belong to no more than one lower-level entity set



Partial vs. Total Specialization

- Total specialization
 - Each higher-level entity (person) must belong to a lower-level entity set
 - There are no other persons than employees and students (in the DB)
 - Needs to be specified in the diagram
 - Analogy in OOP: abstract classes
- Partial specialization
 - Some higher-level entities may not belong to any lower-level entity set
 - the default case



Exercise

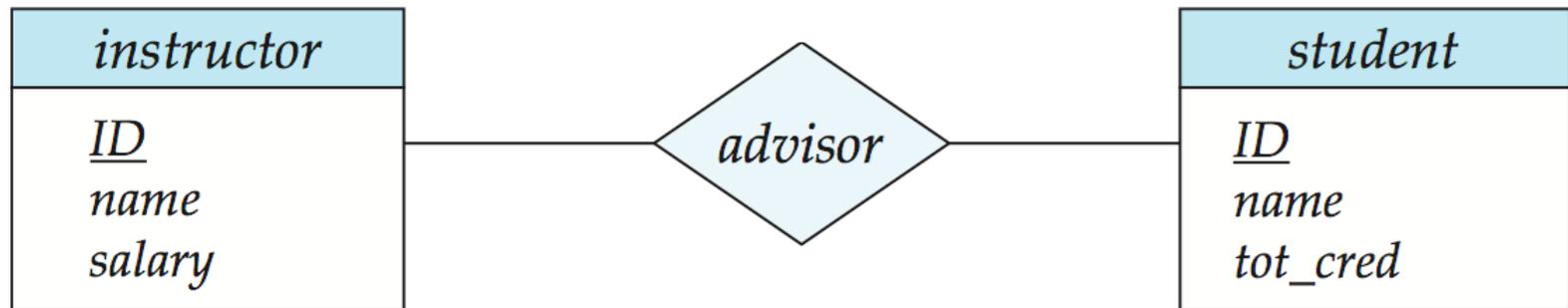
- A small library wants to keep track of books and persons which borrow these books. Design an **Entity-Relationship Diagram (ERD)** with appropriate attributes.
 - What question(s) do you have?
 - What changes if you also want to include the borrowing history?

Reduction to Relation Schemas

- How to get to from an ER model to a relational database model?
 - Recap: relational database models consists of relations
- We have
 - Entity sets and relationship sets
- Goal
 - Translate entity and relationship sets uniformly to *relation schemas*
- Mechanism:
 - For each entity set and relationship set there is a unique relation that is assigned the name of the corresponding entity set or relationship set
 - Each relation has a number of columns (generally corresponding to attributes), which have unique names

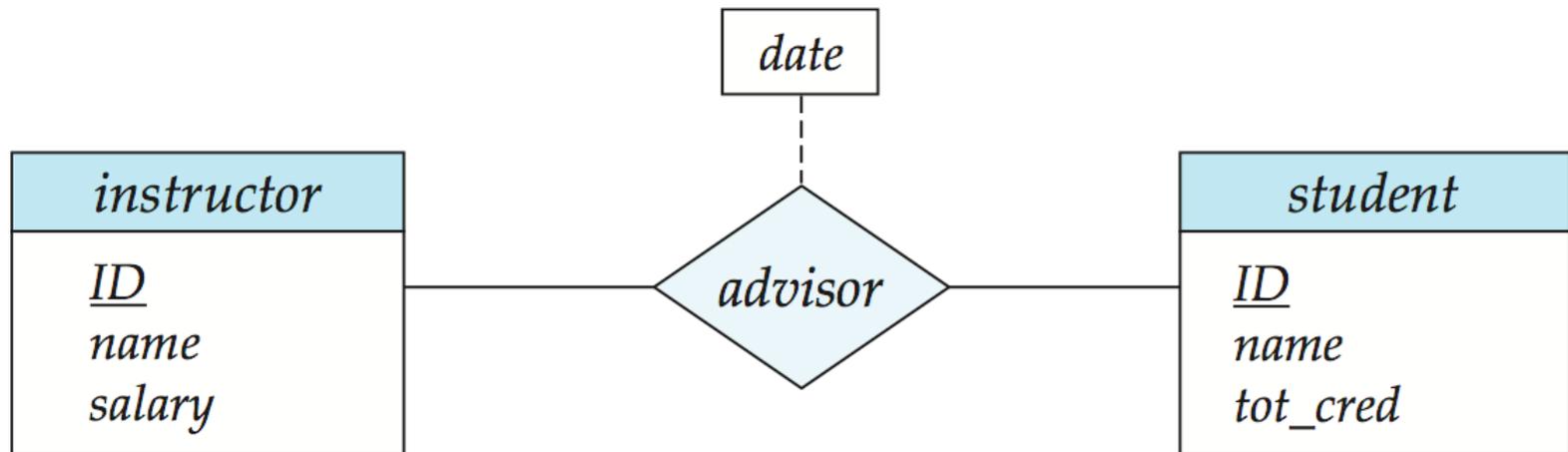
Representing Relationship Sets

- Many-to-many relationship sets
 - Represented as a relation with attributes for the primary keys of the two participating entity sets
 - They are foreign keys to the individual tables as well
- Example: schema for relationship set *advisor*
advisor = (*student ID*, *instructor ID*)



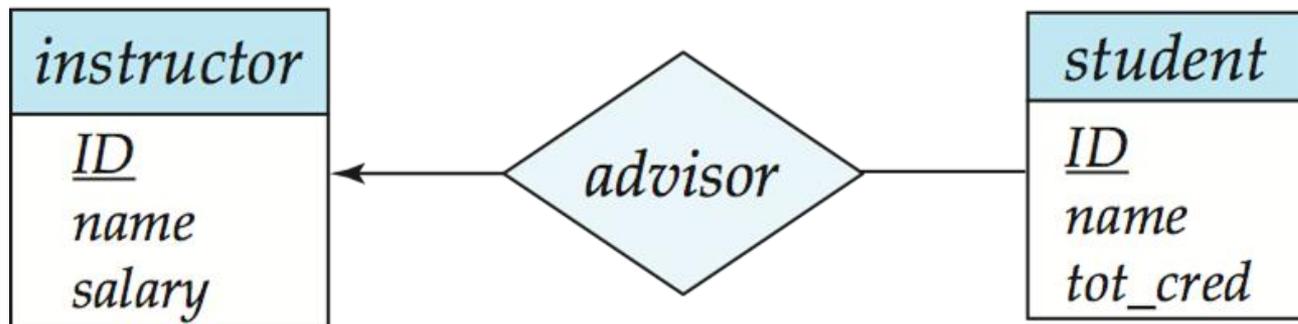
Representing Relationship Sets

- Many-to-many relationship sets
 - Additional attributes of the relationship set become attributes of the representing relation
- Example: schema for relationship set *advisor*
advisor = (*student ID*, *instructor ID*, *date*)



Representing Relationship Sets

- One-to-many relationship sets
 - The primary key of the “one” side can become a foreign key attribute on the “many” side
- student = (ID, name, tot_cred, instructor_ID)*
- In case of partial participation, this may cause **null** values



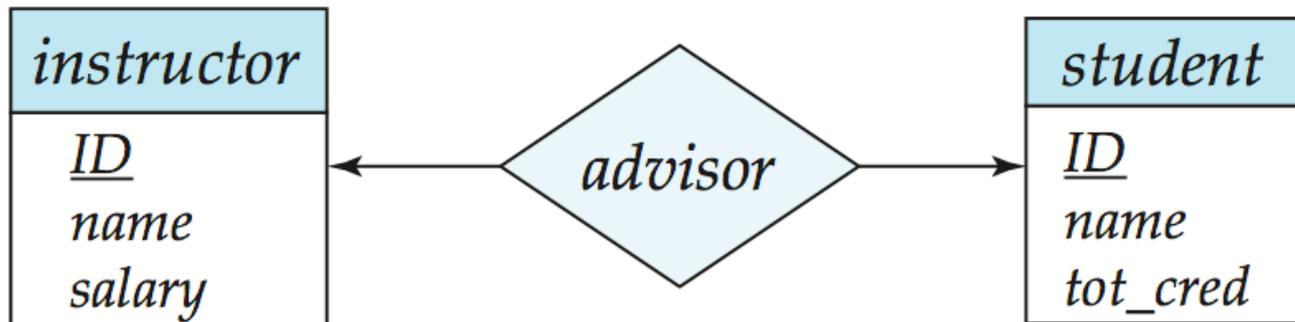
Representing Relationship Sets

- Special case for one-to-one relationship sets
 - The primary key on one side can be included on the other side

student = (ID, name, tot_cred, instructor_ID) or
instructor = (ID, name, salary, student_id)

why not on both sides?

- In case of partial participation, this may cause **null** values



Representing Attributes

- **Composite attributes** are flattened out by creating a separate attribute for each component attribute
- Add prefix of super attribute in case ambiguous names occur
 - e.g., `street_number`, `phone_number`

instructor(ID,
first_name, *middle_initial*, *last_name*,
street_number, *street_name*,
apt_number)

<i>instructor</i>
<u>ID</u>
<i>name</i>
<i>first_name</i>
<i>middle_initial</i>
<i>last_name</i>
<i>address</i>
<i>street</i>
<i>street_number</i>
<i>street_name</i>
<i>apt_number</i>

Representing Multi-valued 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* 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
- 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)

<i>instructor</i>
<u><i>ID</i></u>
<i>name</i>
<i>first_name</i>
<i>middle_initial</i>
<i>last_name</i>
<i>address</i>
<i>street</i>
<i>street_number</i>
<i>street_name</i>
<i>apt_number</i>
<i>city</i>
<i>state</i>
<i>zip</i>
{ <i>phone_number</i> }

Representing Derived Attributes

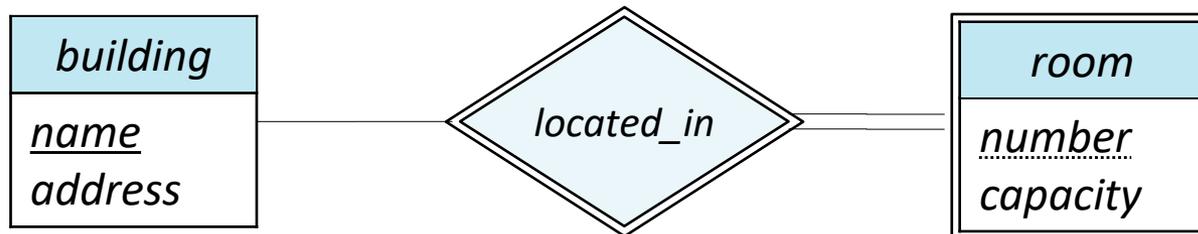
- Derived attributes can be represented as a view
 - Example: age()

```
CREATE VIEW instructor_age AS  
  SELECT ID,NOW()-date_of_birth AS age  
  FROM instructor
```

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

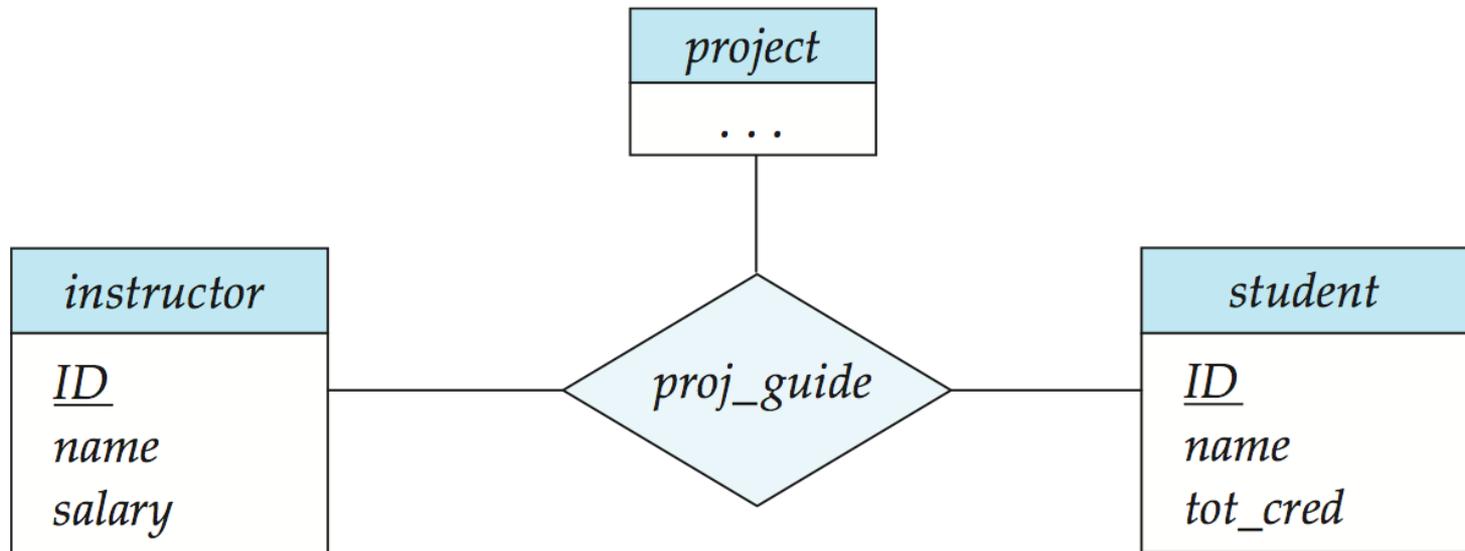
Representing Weak Entities

- A strong entity set becomes a relation with the same attributes
building(name, address)
- A weak entity set becomes a relation that includes
 - the column(s) of the primary key of the identifying strong entity set:
 - room (name, number, capacity)
- At the same time, name is a foreign key
 - which integrity constraints should we use?



Representing Higher Arity Relations

- Higher arity relationship sets are represented just like binary ones
 - i.e., as one relation with the primary keys of the related entity sets
 - `proj_guide(instructor ID, student ID, project ID)`



Representing Specialization

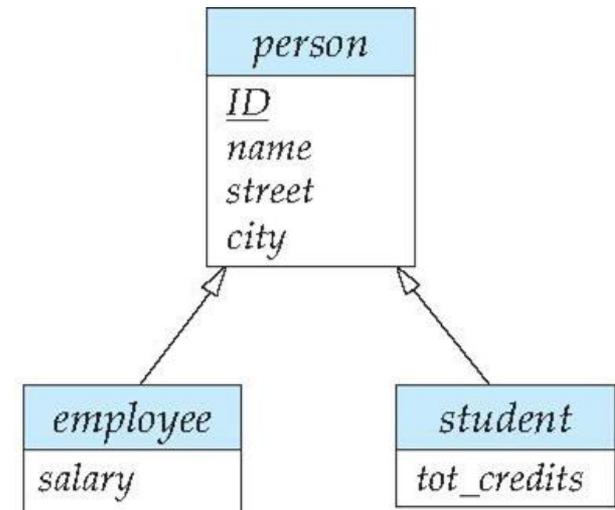
- **Method 1**

- All three relations become relations
 - primary key is shared
- Shared attributes are only represented in the higher level entity

person(ID, name, street, city)

employee(ID, salary)

student(ID, tot_credits)



- **Drawback:**

- Accessing person information for employees and students requires access to two relations

Representing Specialization

- **Method 2**

- All three relations become relations
 - primary key is shared
- Shared attributes are only represented in each entity

person(ID, name, street, city)

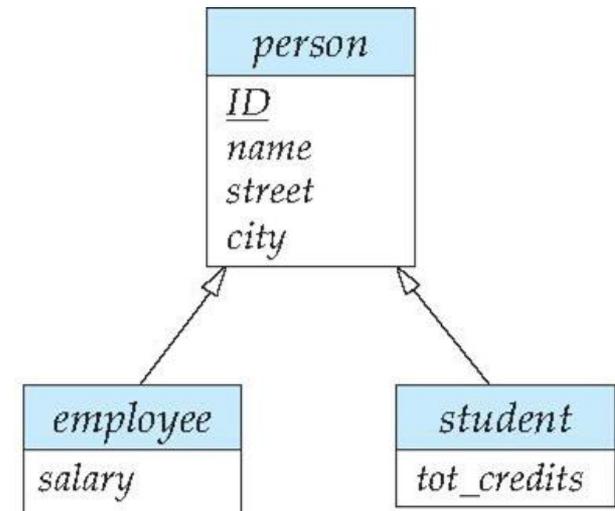
employee(ID, name, street, city, salary)

student(ID, name, street, city, tot_credits)

- Super relation can be omitted for total specialization

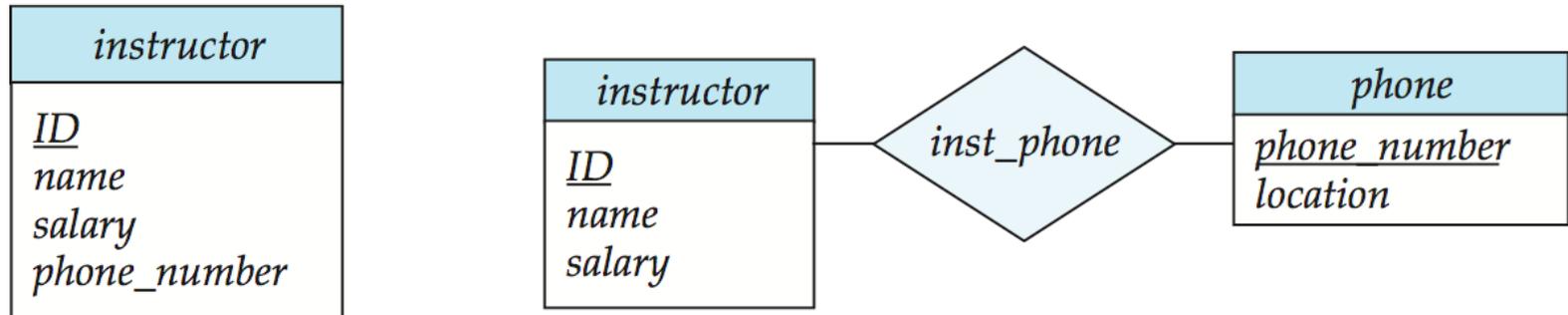
- **Drawback:**

- Redundant storage for overlapping specialization
 - i.e., for persons that are both employees and students



Design Decisions in ER Modeling

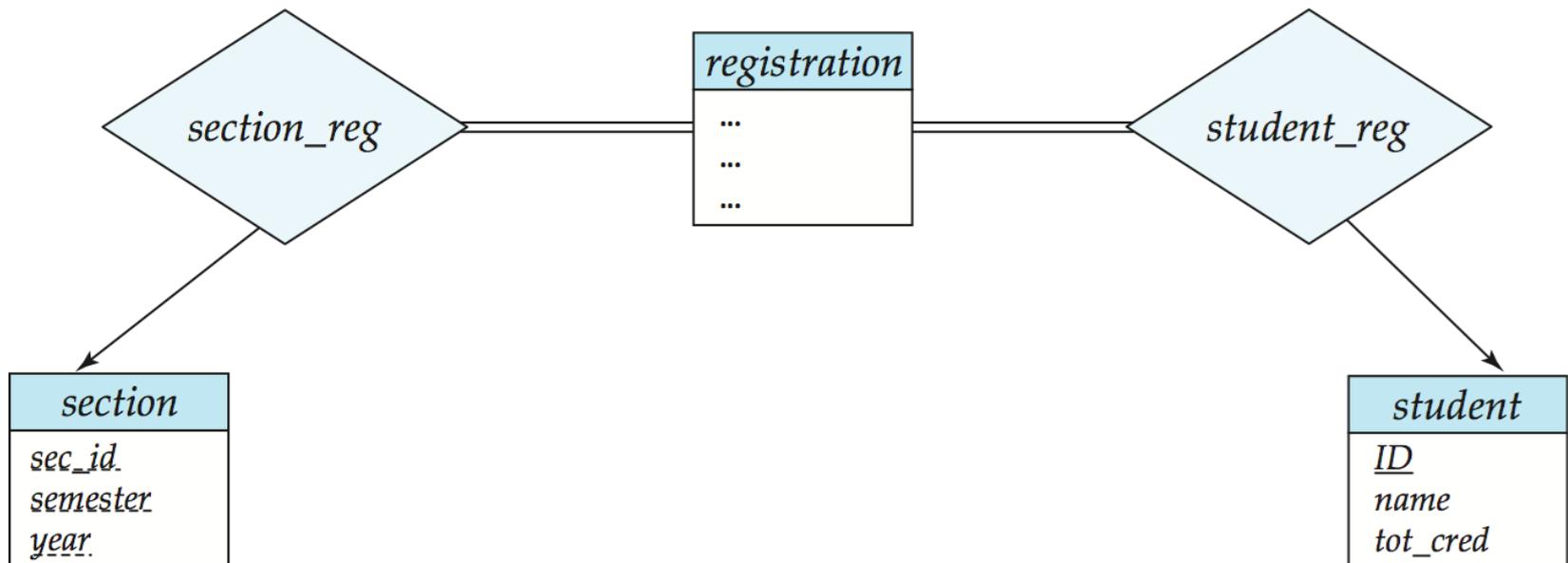
- Entity sets vs. attributes



- Entity set
 - Allows for additional information
 - Supports multi-valued attributes
 - in that case, the attribute would end as a relation in the DB anyways

Entity Sets vs. Relationship Sets

- Students register for course sections
 - This could be a simple relationship set as well
- Entity set can store additional information, e.g.
 - Date of registration

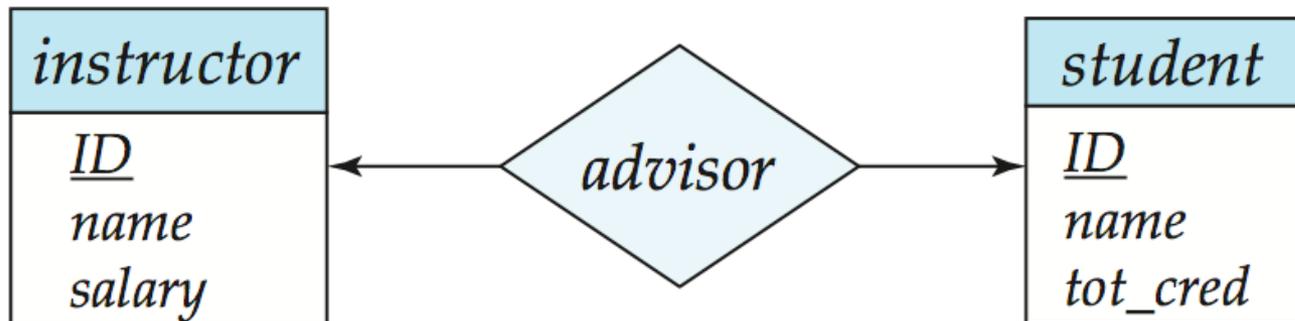


Placement of Attributes for 1:1 Relationships

- The primary key on one side can be included on the other side

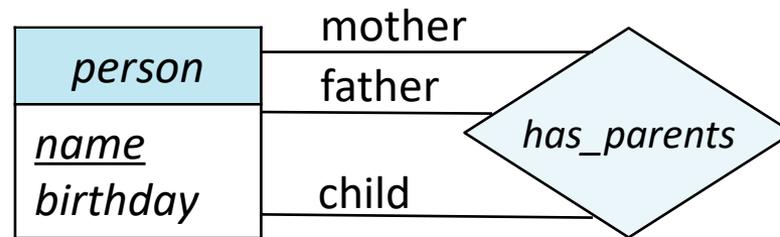
*student = (ID, name, tot_cred, instructor_ID) or
instructor = (ID, name, salary, student_id)*

- Which one?



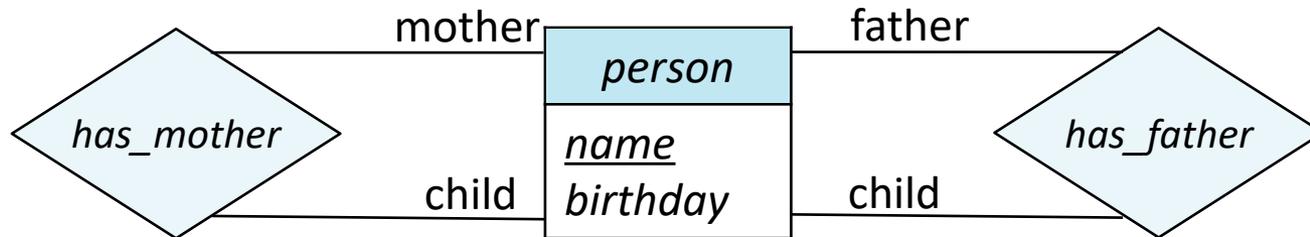
Binary vs. Non-Binary Relationships

- Sometimes, non-binary relationships can be replaced by binary ones



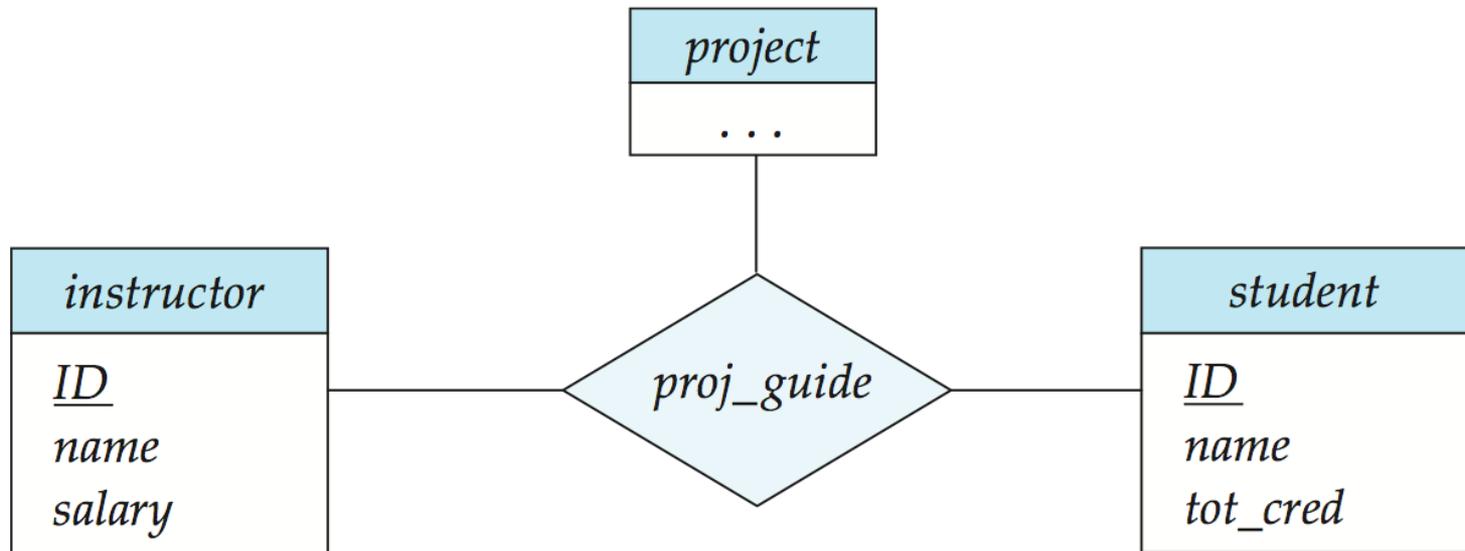
Binary vs. Non-Binary Relationships

- Sometimes, non-binary relationships can be replaced by binary ones
 - This is usually the preferred solution



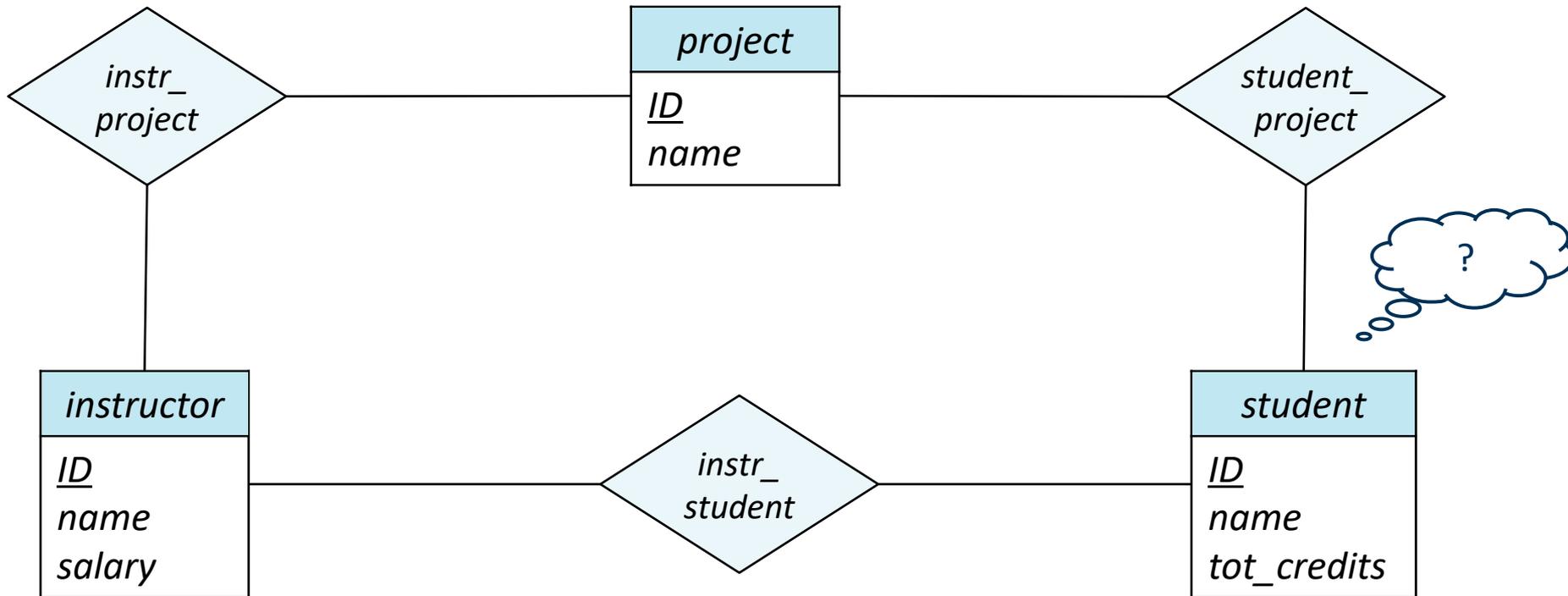
Binary vs. Non-Binary Relationships

- Sometimes, non-binary relationships can be replaced by binary ones
 - but sometimes, they are n-ary by nature



Binary vs. Non-Binary Relationships

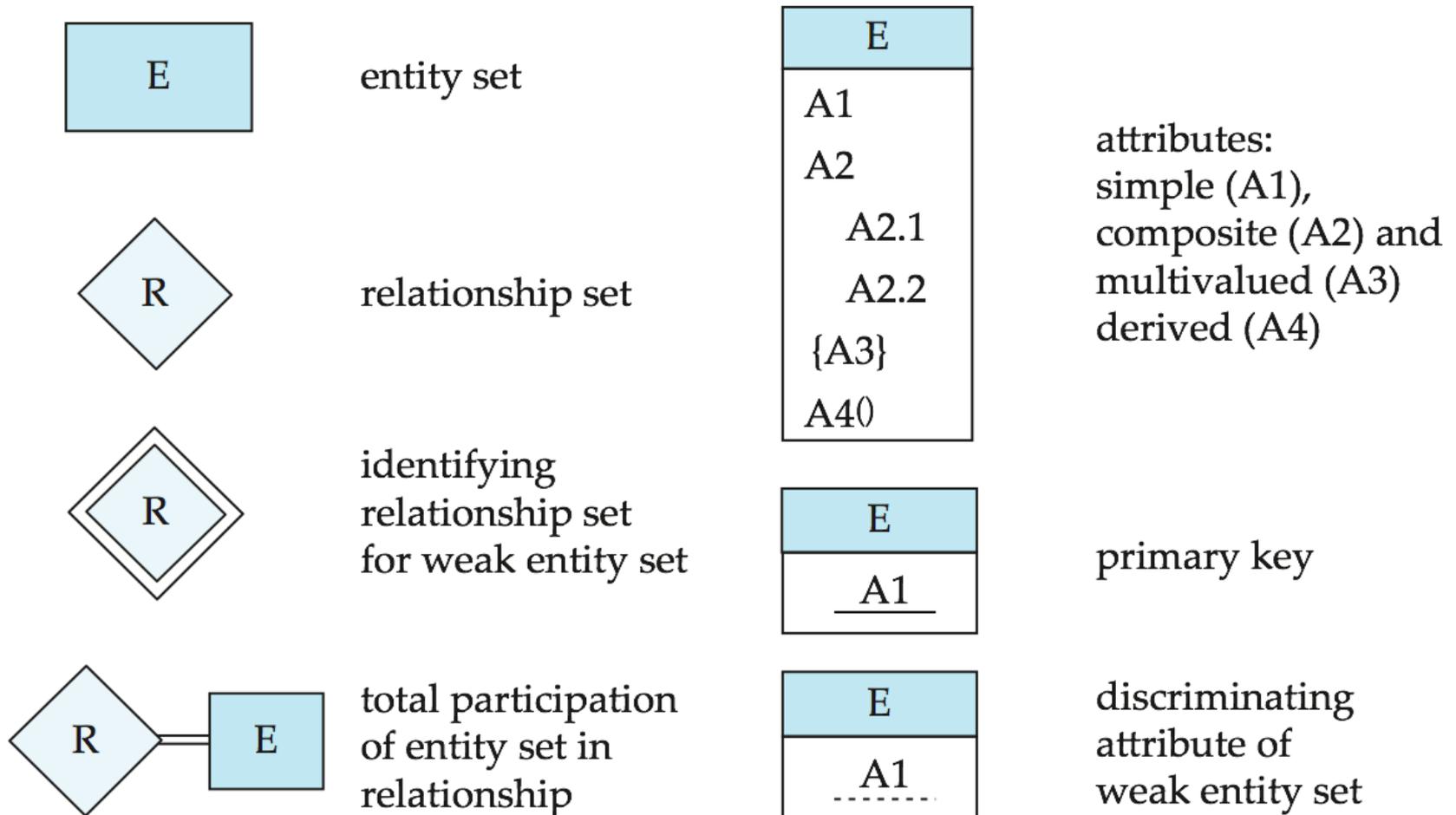
- Sometimes, non-binary relationships can be replaced by binary ones
 - but sometimes, they are n-ary by nature



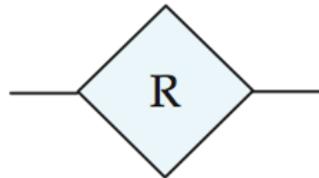
Summary

- Designing databases
 - i.e., how to get from your customer's requirements...
 - ...to a set of tables and attributes
- ER Models are an intermediate step
 - Conceptual view on the database
 - Graphical notation
 - Can be used for discussion with customers
- Translation rules for ER to RDBMS
- Design decisions
 - For ER Models (mostly business decisions)
 - For translation to RDBMS (mostly computer science decisions)

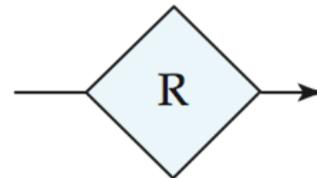
Summary of ER Notation



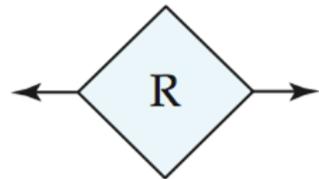
Summary of ER Notation (ctd.)



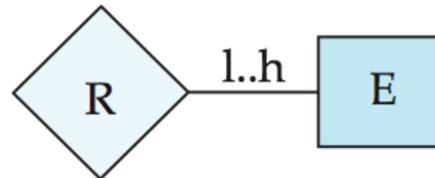
many-to-many
relationship



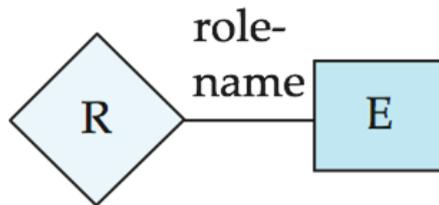
many-to-one
relationship



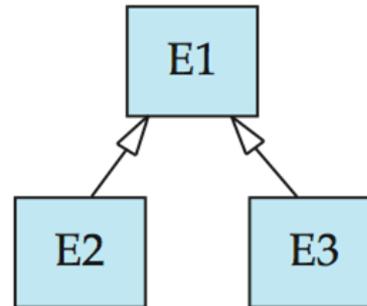
one-to-one
relationship



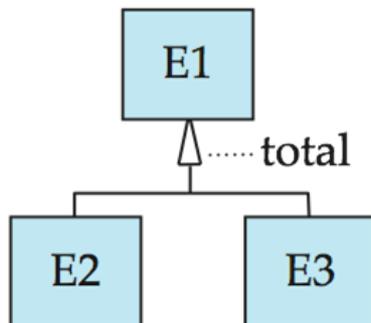
cardinality
limits



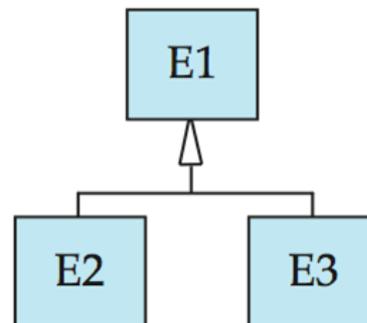
role indicator



ISA: generalization
or specialization



total (disjoint)
generalization



disjoint
generalization

Questions?



A Full Example

