RAWDATA
Section 1

Entity-Relationship Model

Henrik Bulskov & Troels Andreasen

Chapter 7: Entity-Relationship Model

- ❑ Modeling
- ❑ Basic concepts
- ❑ E-R Diagram
- ❑ Design Issues
- ❑ Extended E-R Features
- ❑ Reduction to Relation Schemas
- ❑ Compare to UML

Modeling

- A *database* can be modeled as:
  - a collection of entities,
  - a collection of relationships among entities.

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have **attributes**
  - Example: people have *names* and *addresses*
- 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

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 entities

  Example:

44553 (Peltier)	*advisor*-	22222 (Einstein)
student entity	relationship	*instructor* entity

- A **relationship set** is a mathematical relation among two or more entities, each taken from entity sets

  – $\boldsymbol{\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}}$ $\qquad n \geq 2$

  where (e_1, e_2, …, e_n) is a relationship

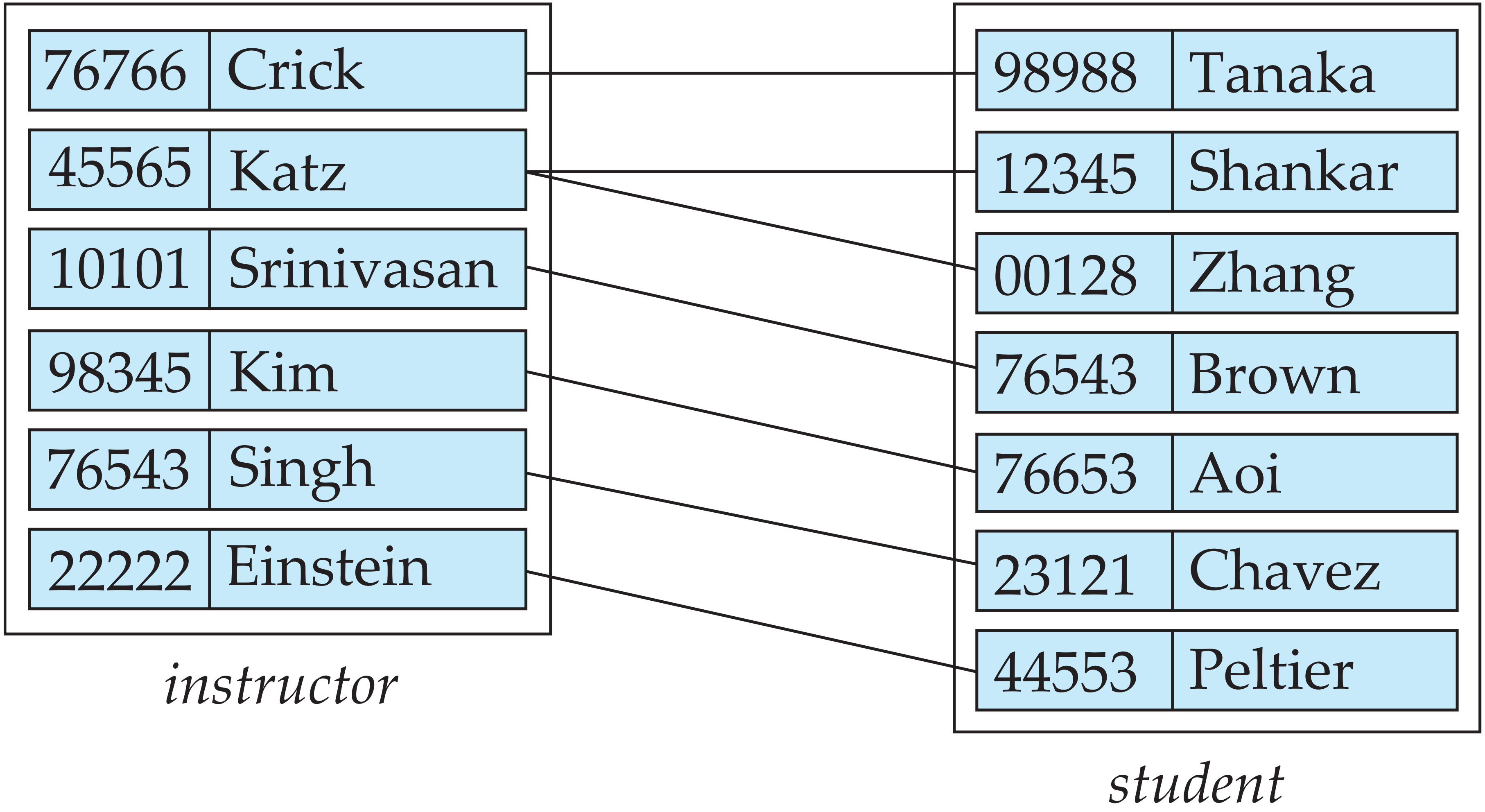
- Example:

  – *advisor* $\subseteq$ $\boldsymbol{\{(s, i) \mid s \in student, i \in instructor\}}$ $\qquad n = 2$

  – elements can be specified by id's such as (45565,12345) $\in$ *advisor* and the relationship set thus could be

    - *advisor* = {(76766,98988), (45565,12345), (45464,00128), …}

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

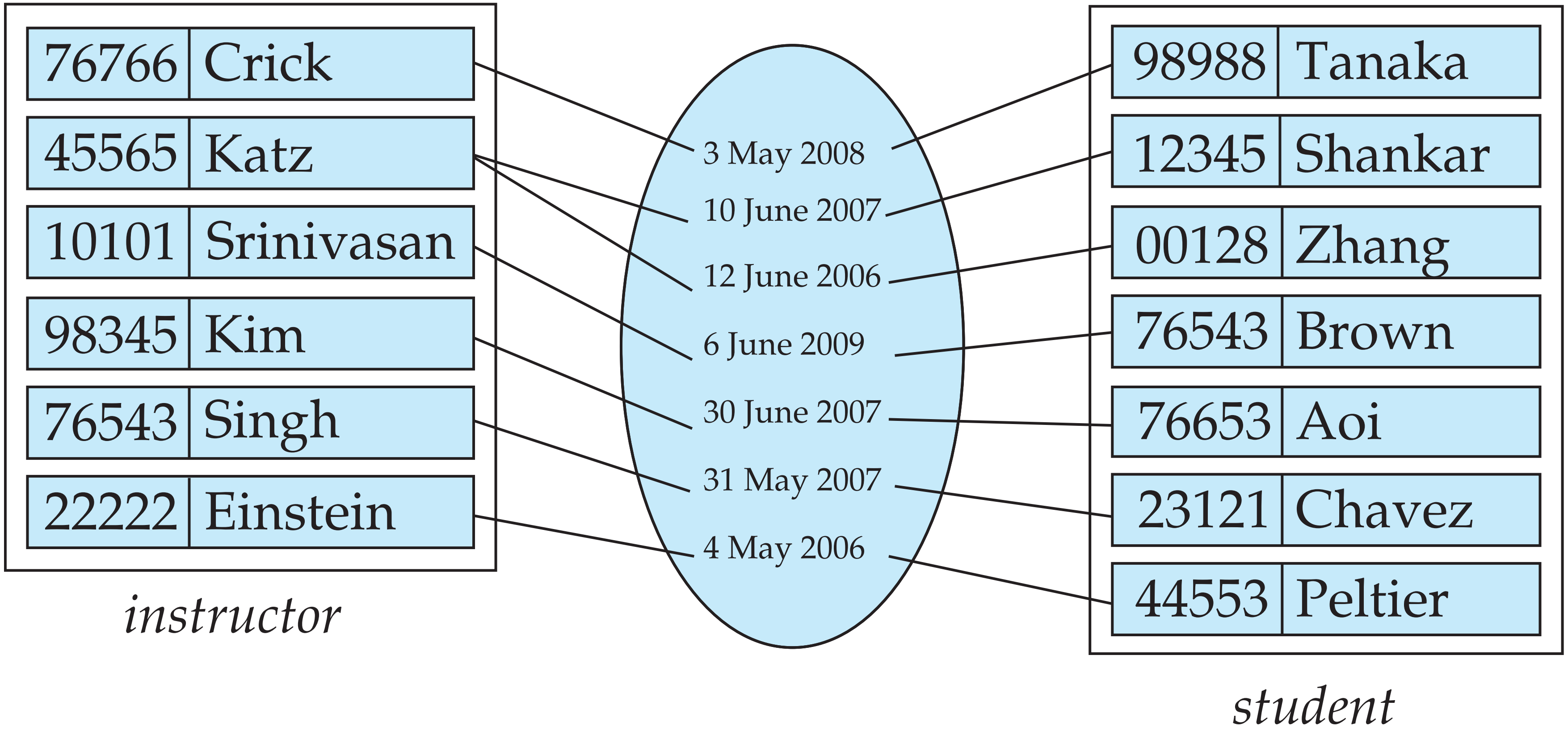
student

- thus:

  advisor = {(76766,98988), (45565,12345), (45565,00128), …}

Relationship Sets (Cont.)

- An **attribute** can also be a property of a relationship set.
- For instance, 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 (degree two).
  - most relationship sets are binary.

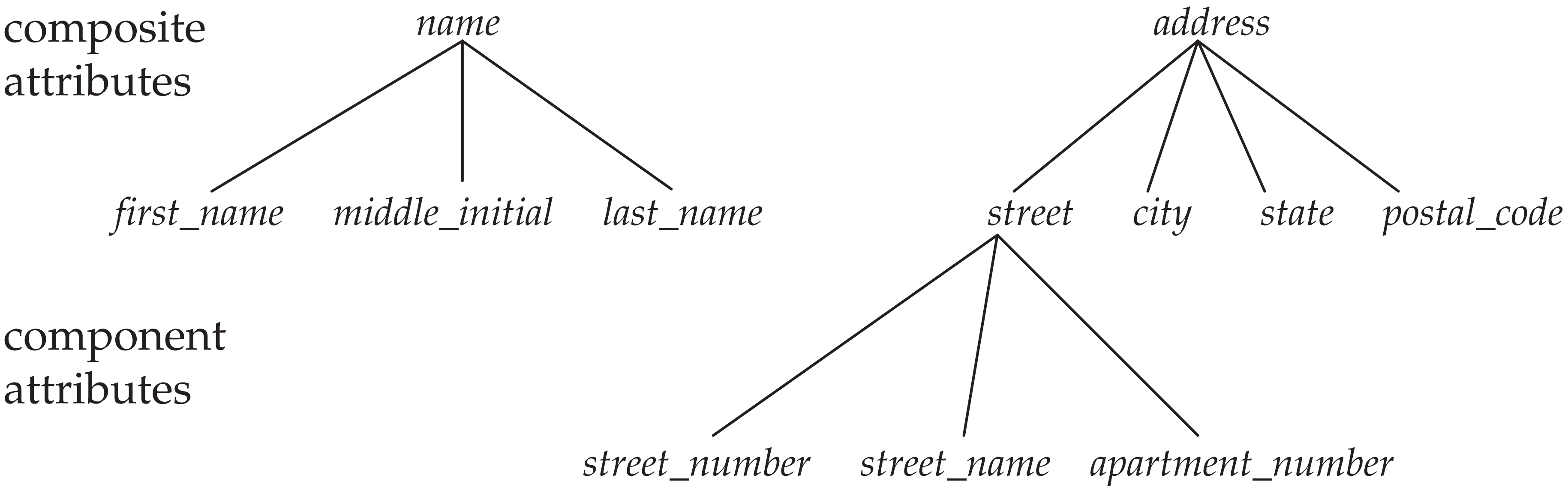
- **n-ary relationship**
  - Relationships between more than two entity sets are rare.
  - Example 3-ary: ***students*** work on research ***projects*** under the guidance of an ***instructor***.
    - ○ Can be modelled using a ternary relationship ***proj_guide*** between *instructor, student,* and *project*

Attributes

- ❑ An entity is represented by a set of attributes, that is, a set of 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

name

first_name *middle_initial* *last_name*

address

street *city* *state* *postal_code*

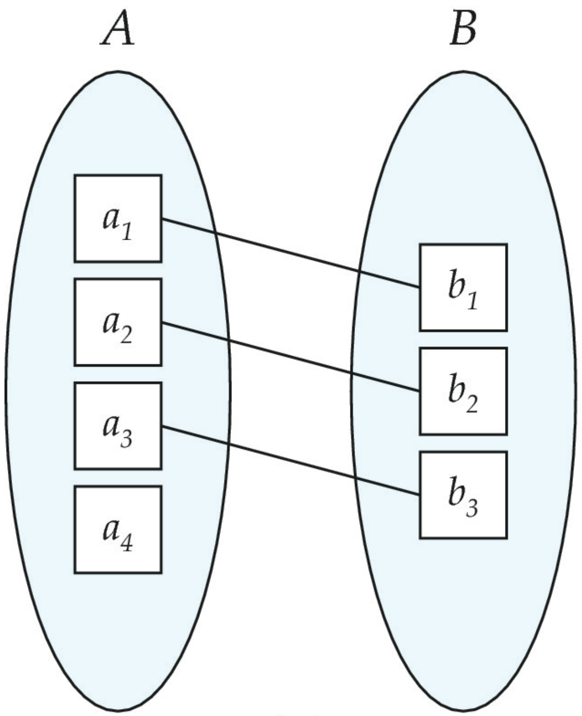
street_number *street_name* *apartment_number*

Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.

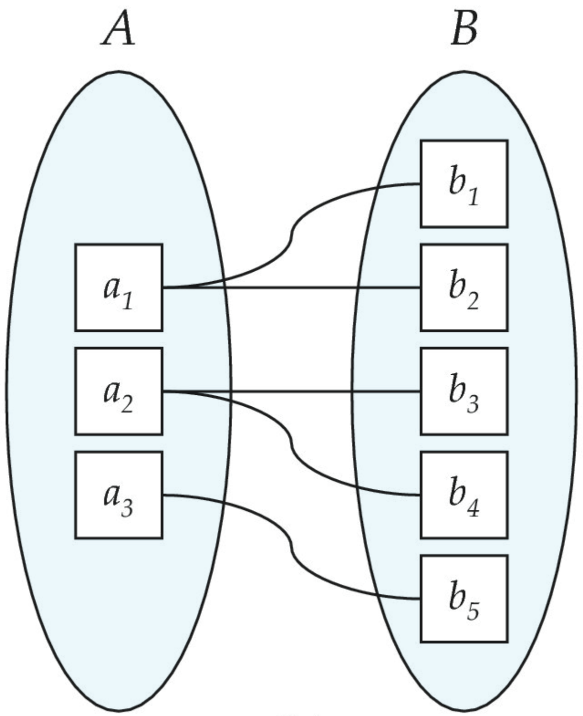
- For a binary relationship set the cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Cardinalities



(a)

One to one

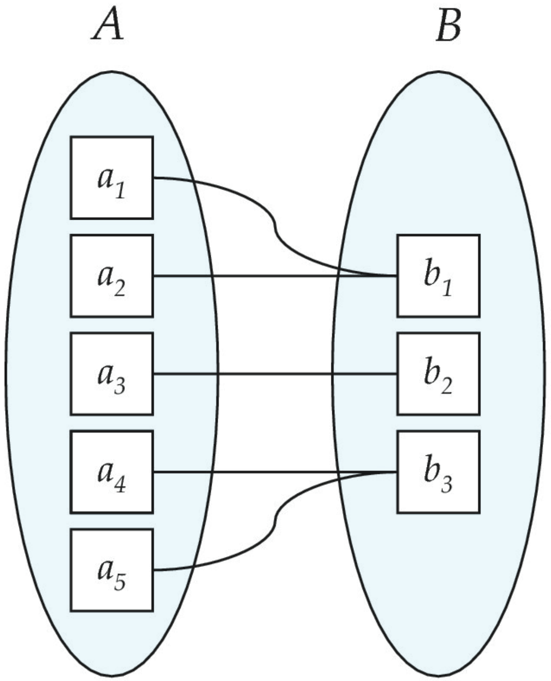


(b)

One to many

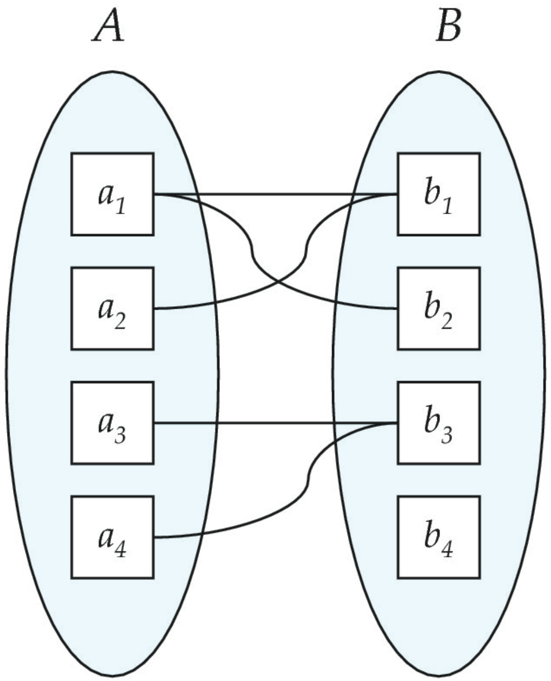
Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Cardinalities



(a)

Many to one

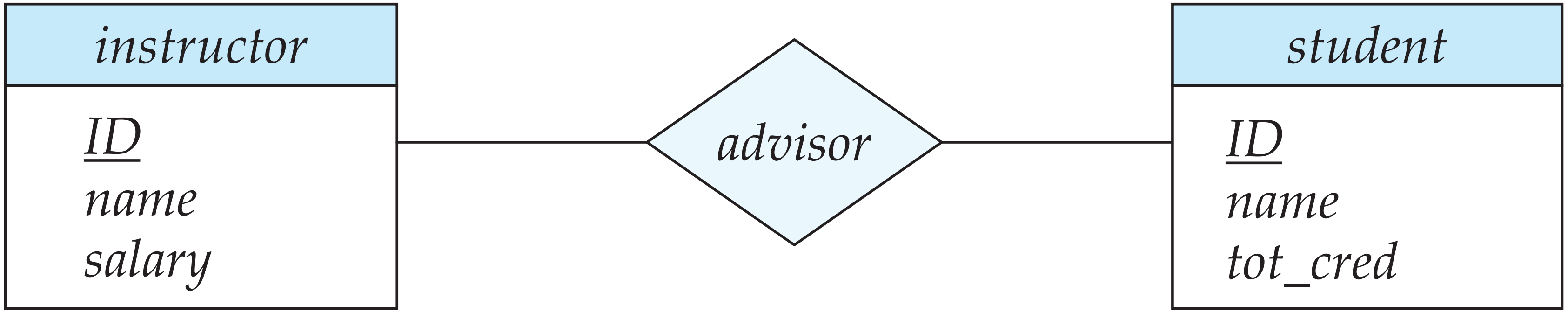


(b)

Many to many

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

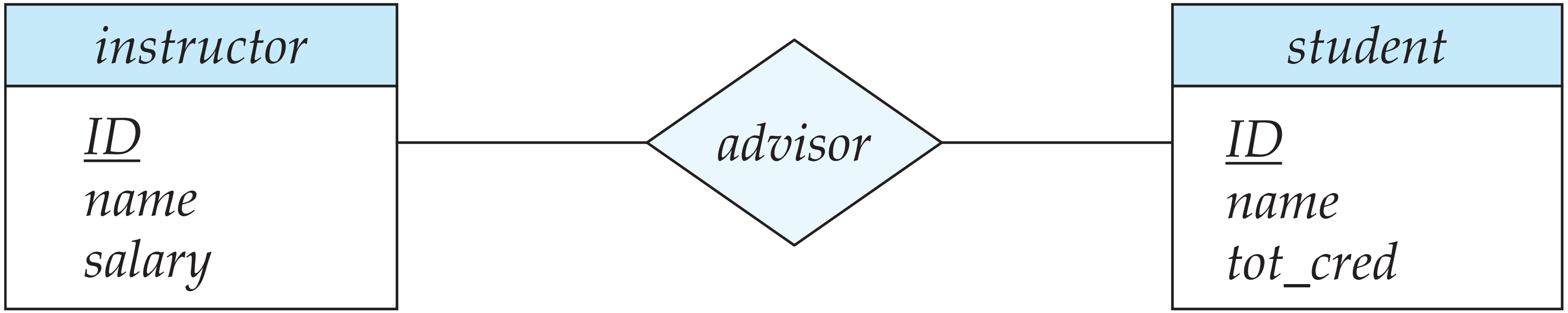
E-R Diagrams



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

Keys for Entity Sets

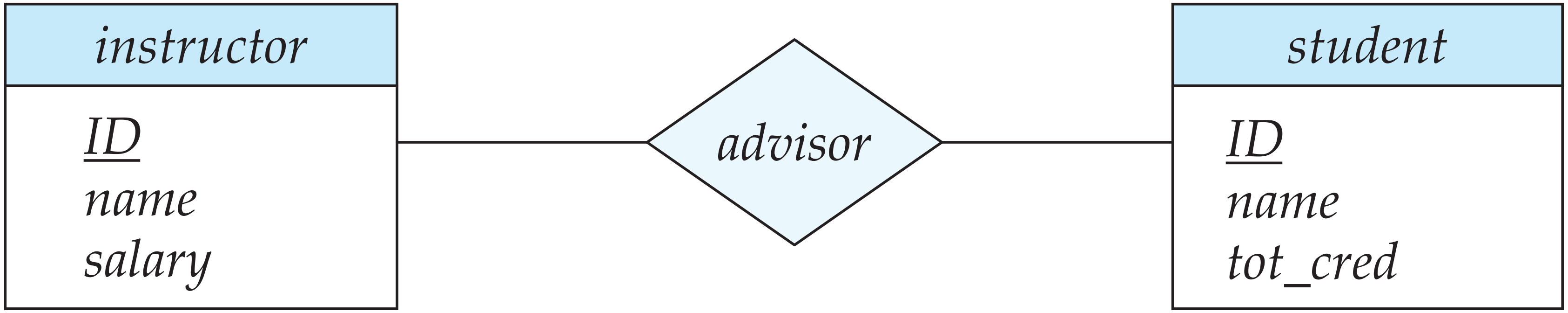
- A **super key** of an entity set is a set of one or more attributes whose values **uniquely determine each entity**.
- A **candidate key** of an entity set is **a minimal super key**
  - *ID* is candidate key of *instructor*
  - *course_id* is candidate key of *course*
- Although several candidate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



Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (*student_id, instructor_id*) is the super key of *advisor*

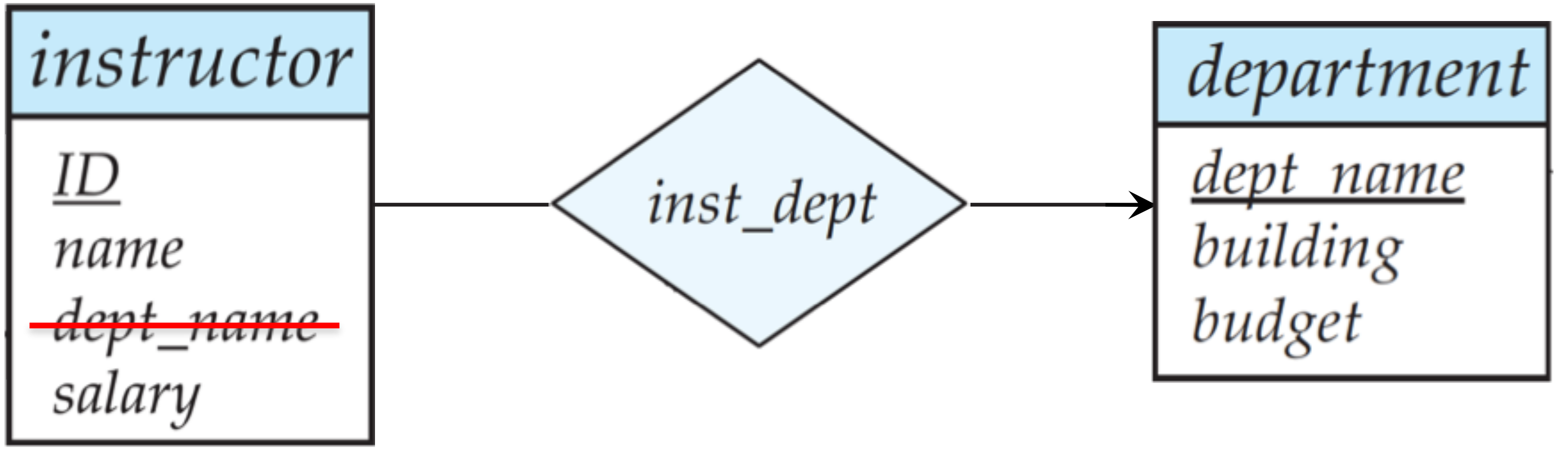
- Must consider the cardinality of the relationship set when deciding what are the candidate keys



Redundant Attributes

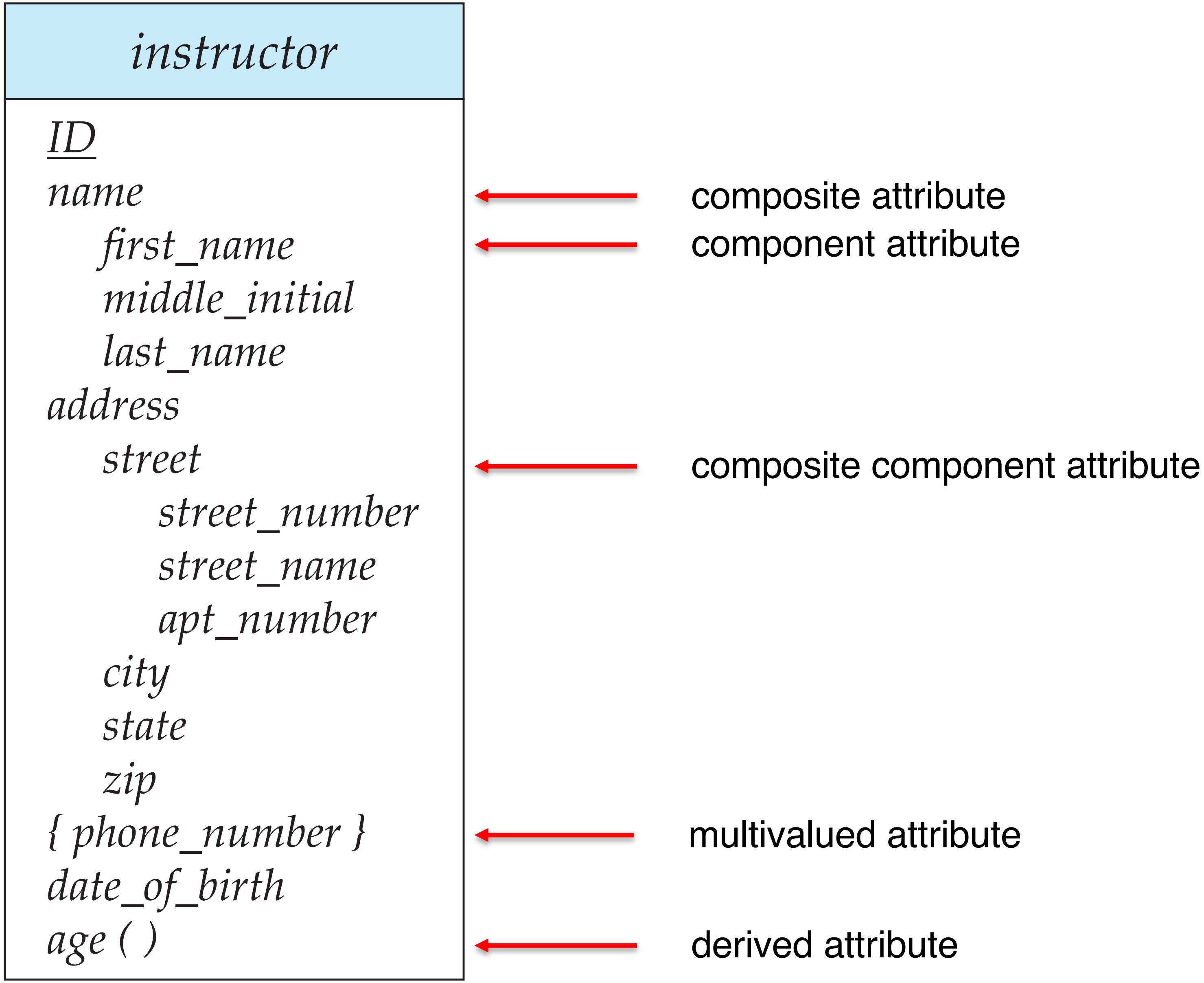
- ❑ Suppose we have entity sets
  - *instructor*, with attributes including (*ID, name, dept_name, salary)*
  - *department* (*dept_name, building, budget)*

  and a relationship
  - *inst_dept* relating *instructor* and *department*



- ❑ Attribute *dept_name* in entity *instructor* is redundant since there is an explicit relationship *inst_dept* which relates instructors to departments
  - The attribute replicates information present in the relationship, and should be removed from *instructor*
  - BUT: when converting to tables, in some cases the attribute gets reintroduced, as we will see.

Entity With Composite, Multivalued, and Derived Attributes



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 attribute

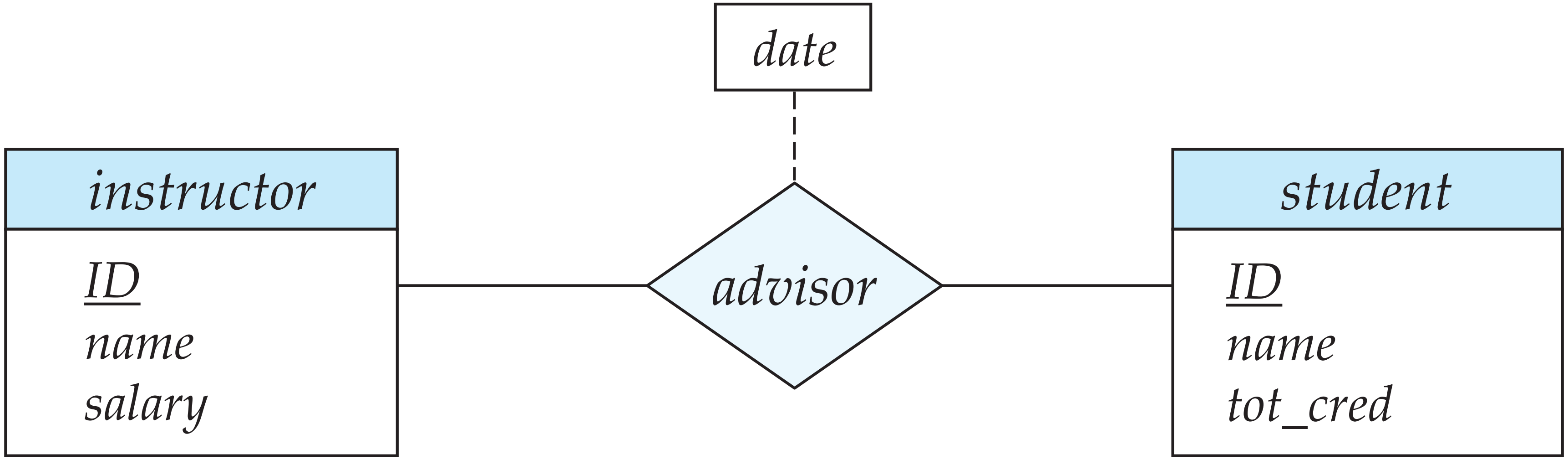
component attribute

composite component attribute

multivalued attribute

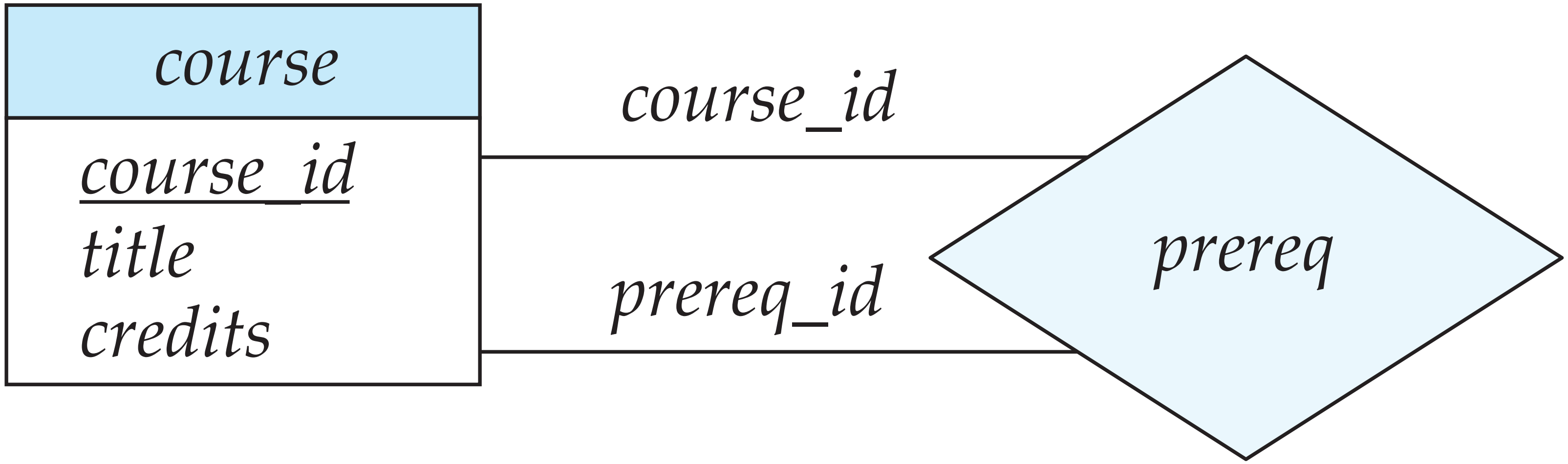
derived attribute

Relationship Sets with Attributes



Roles

- An entity set can participate more than once in a relationship
  - Each occurrence of an entity set plays a “role” in the relationship
- *course* participate twice in the relationship *prereq*
  - The labels “*course_id*” and “*prereq_id*” are called **roles**.

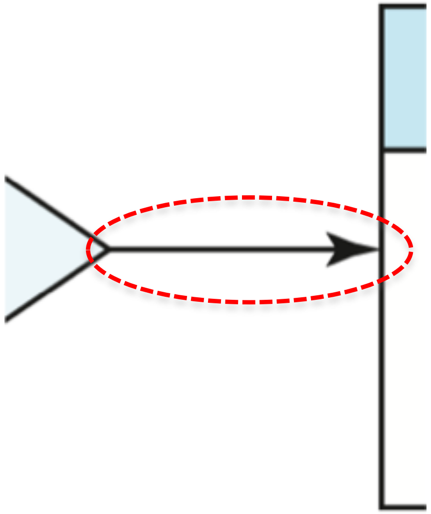


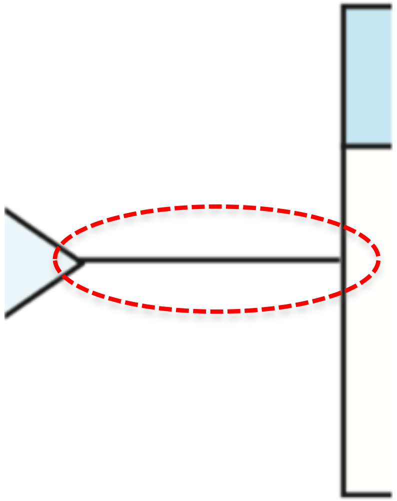
Cardinality Constraints

- We express cardinality constraints by drawing either

  - a directed line (→), signifying "one," or
  - an undirected line (—), signifying "many,"

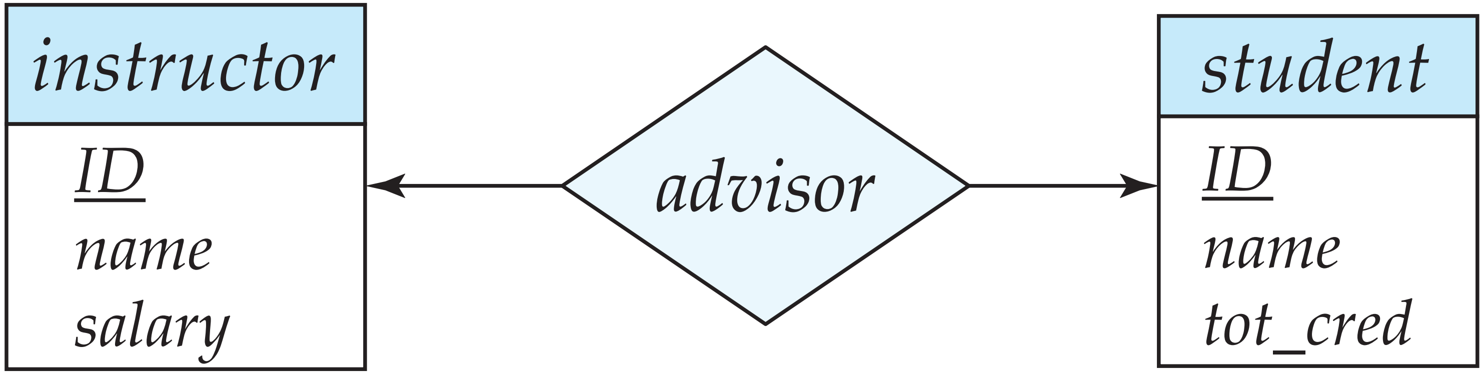
  between the relationship set and the entity set.





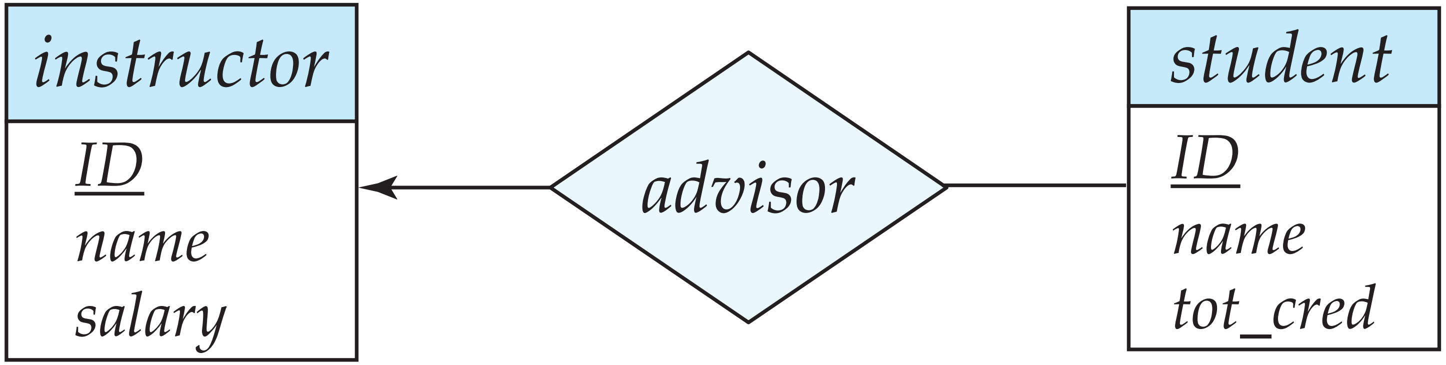
One-to-One Relationship

- one-to-one relationship between an *instructor* and a *student*
  - an instructor is associated with at most one student via *advisor*
  - and a student is associated with at most one instructor via *advisor*



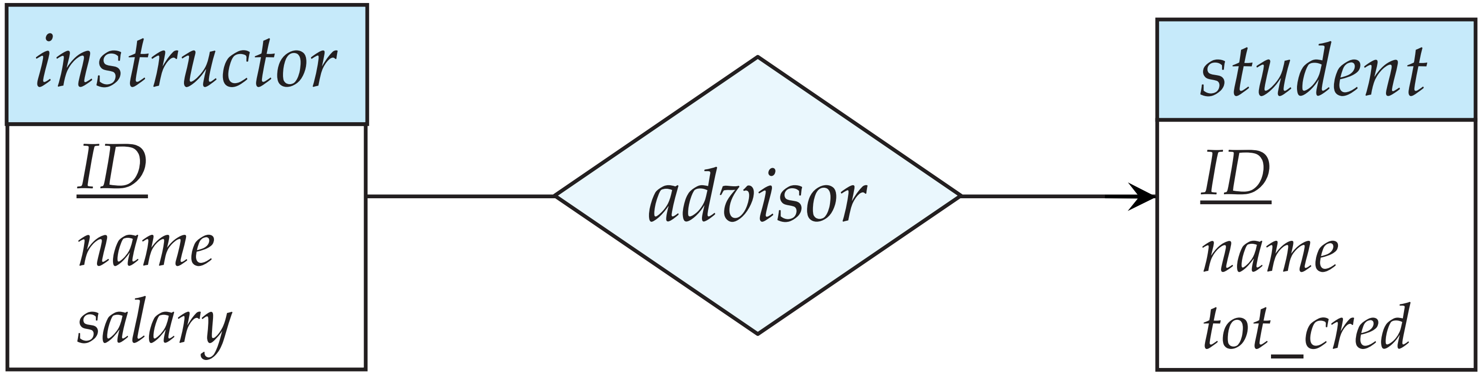
One-to-Many Relationship

- 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,



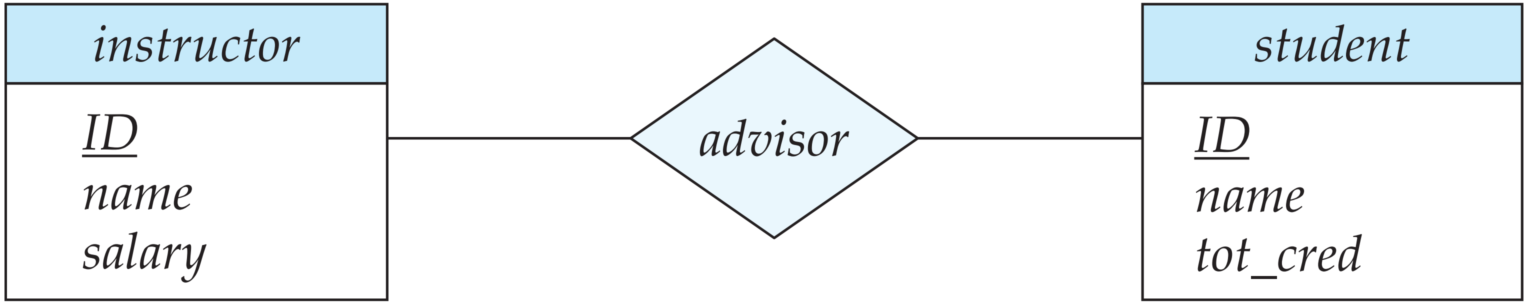
Many-to-One Relationships

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



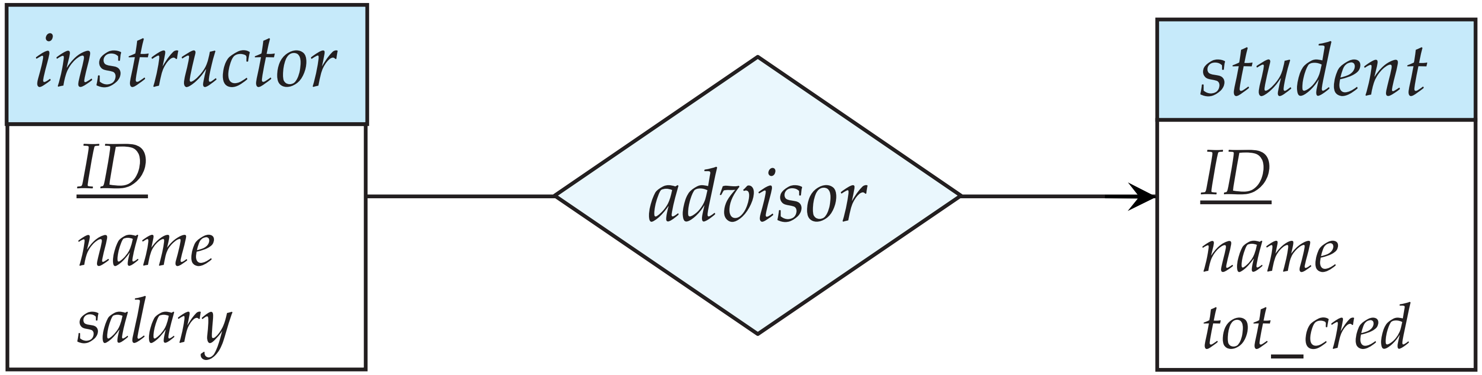
Many-to-Many Relationship

- ❑ An instructor is associated with several (possibly 0) students via *advisor*
- ❑ A student is associated with several (possibly 0) instructors via *advisor*



How to “read” a relationship

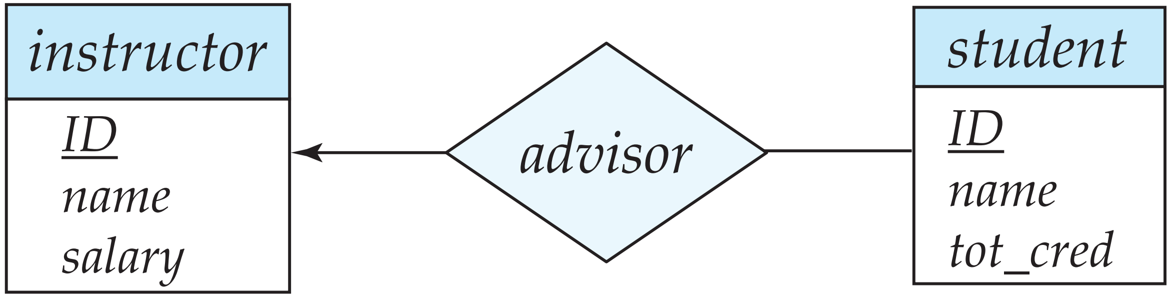
- ❑ Always read boths ways
- ❑ Always start the reading of each direction as
  - ”One of … is related to / associated with …



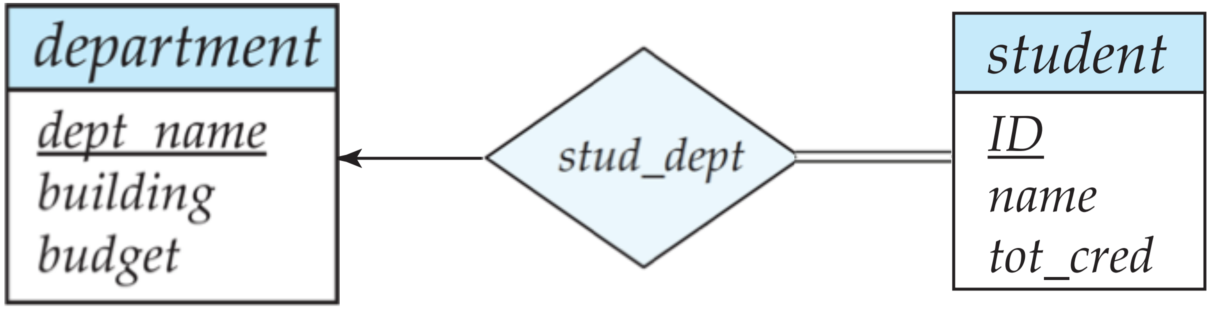
- one instructor is associated with one student,
- one student is associated with (0, one or) many instructors

Participation of an Entity Set in a Relationship Set

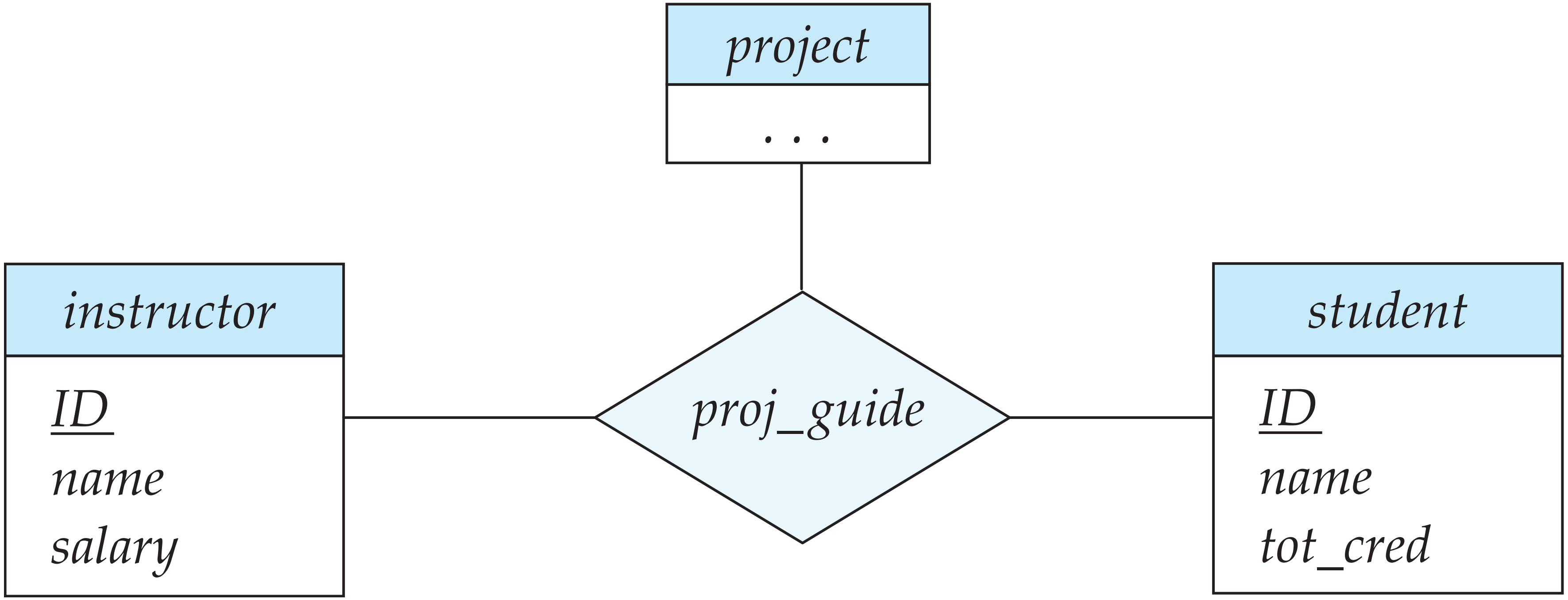
- **Partial participation**: some entities may not participate in any relationship in the relationship set
  - Example: participation of *student* in *advisor* is partial
- a student IS NOT required to have an advisor



- **Total participation**: (indicated by **double line**) every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., participation of *student* in *stud_dept* is total
- every *student* IS required to have an associated *department*



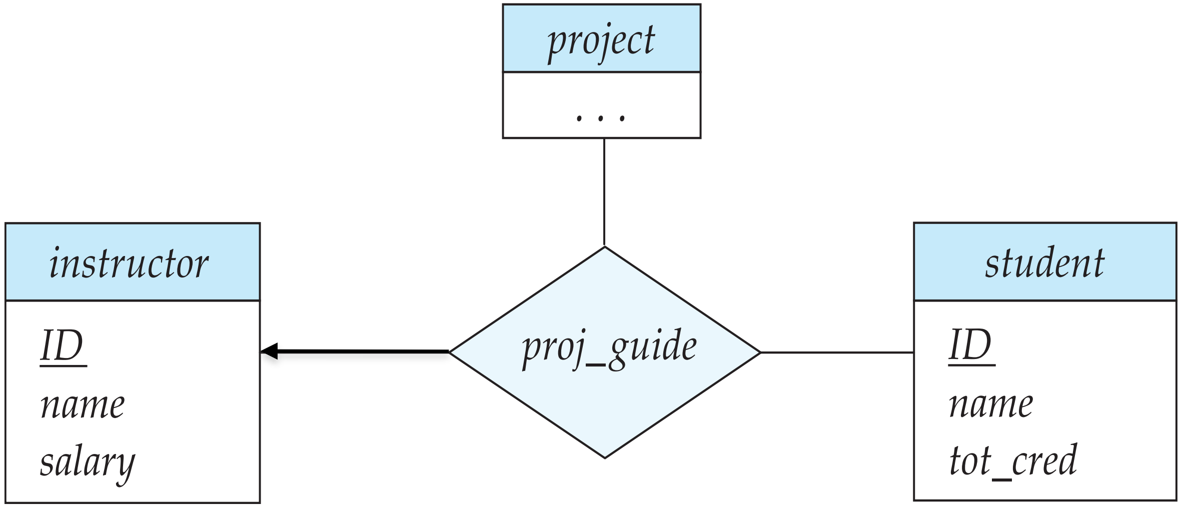
E-R Diagram with a Ternary Relationship



Cardinality Constraints on Ternary Relationship

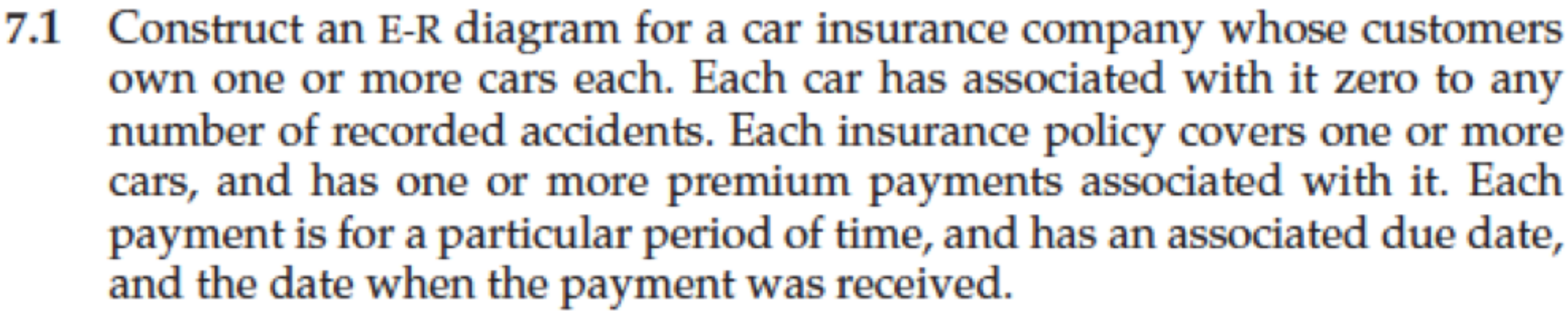
- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
  - If there is more than one arrow, the semantics become unclear

- E.g., an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project



Doing an ER design

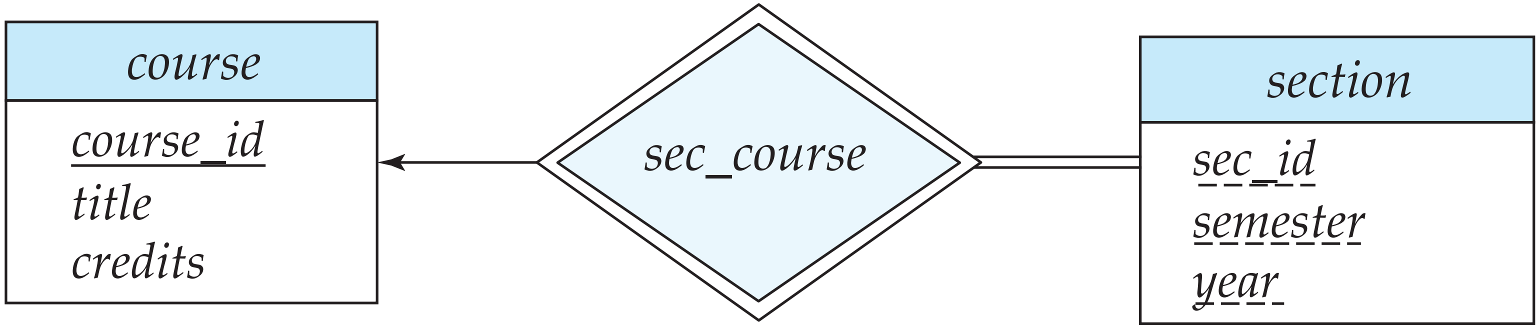
- **Design a model for this (**Exercise 7.1**)**



7.1 Construct an E-R 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 payments 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.

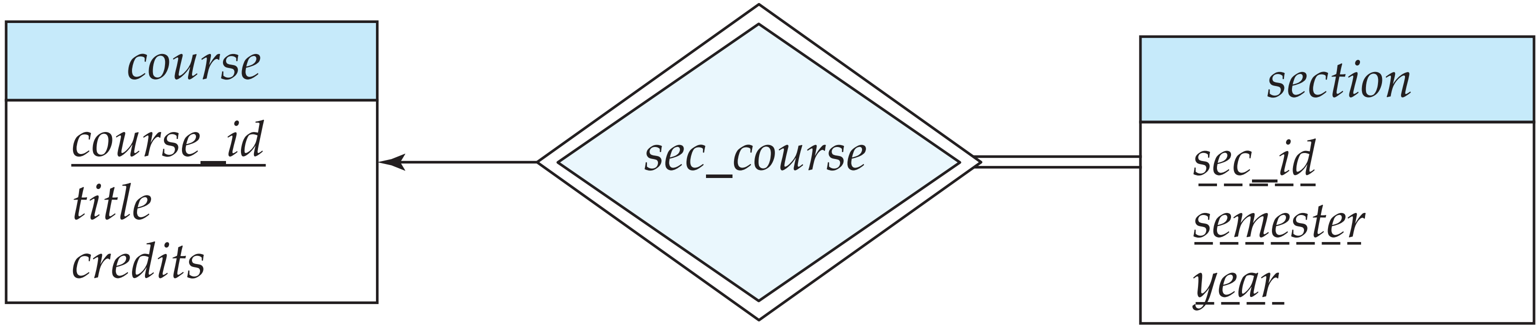
Weak Entity Sets

- An entity set that does not have a primary key is referred to as a **weak entity set**.
- The existence of a weak entity set depends on the existence of an **identifying entity set**
  - It must relate to the identifying entity set via a total, many-to-one relationship set from the weak to the identifying entity set
  - **Identifying relationship** are depicted using a double diamond



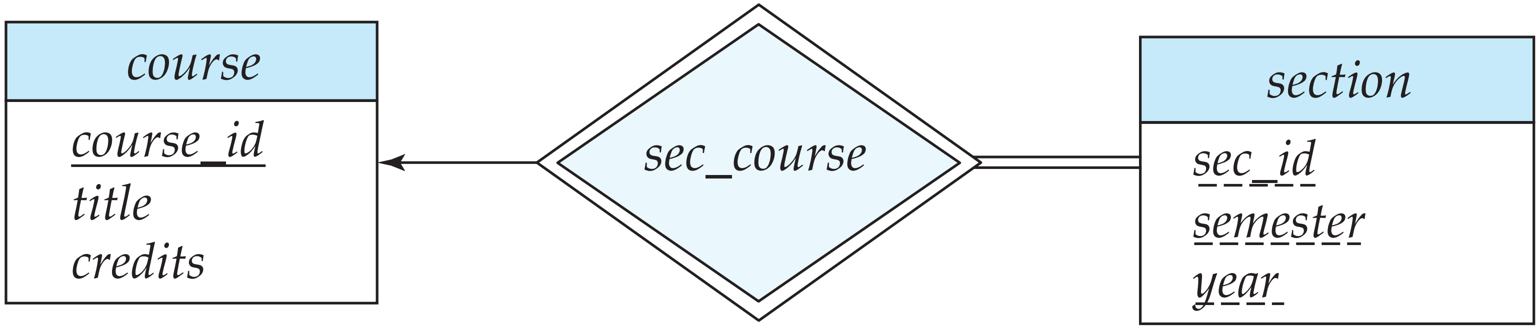
Weak Entity Sets

- The **discriminator** (*or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
  - We underline the discriminator of a weak entity set with a dashed line.

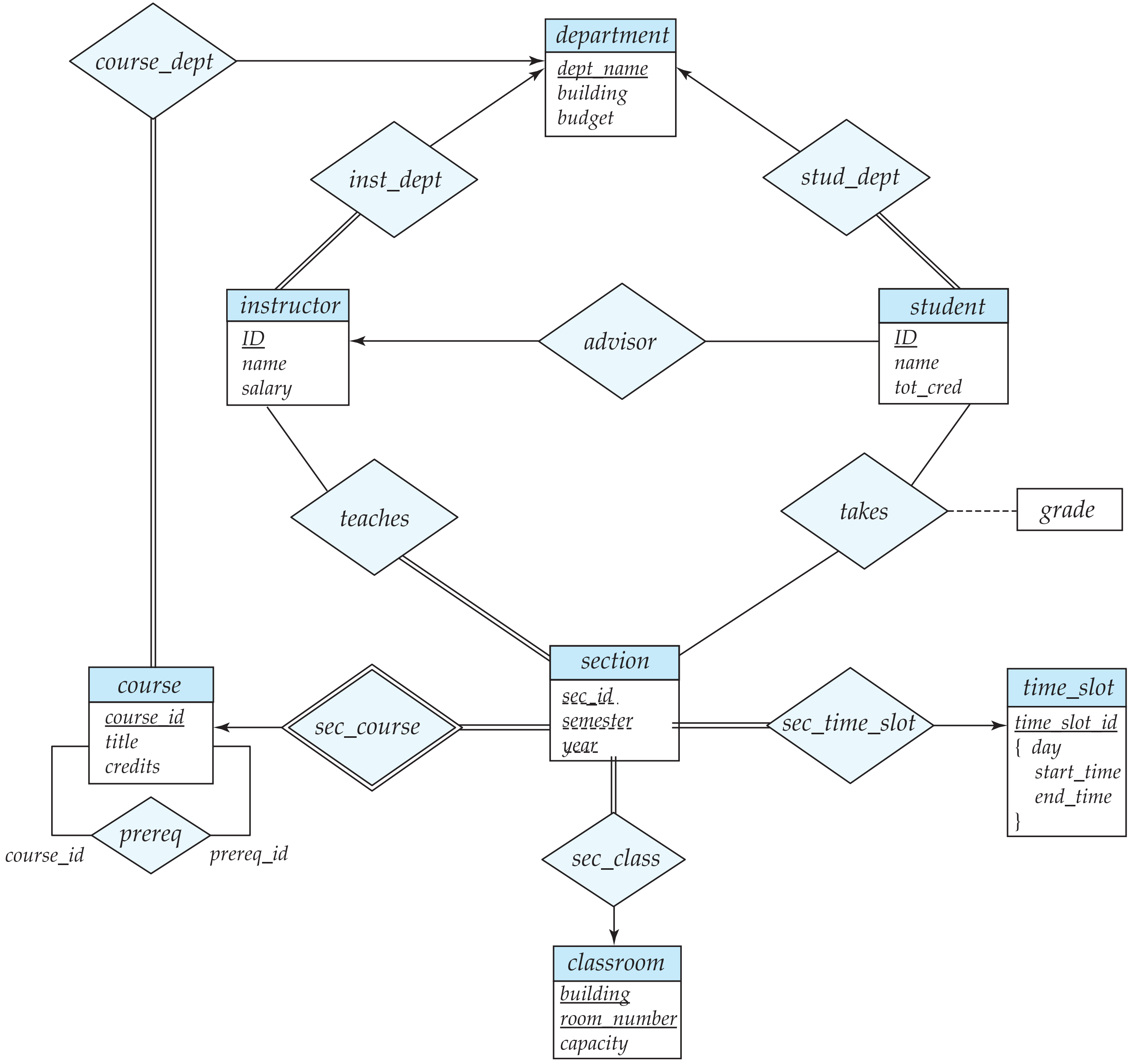


Weak Entity Sets (Cont.)

- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.
- Thus the primary key for *section* will be
  (*course_id, sec_id, semester, year*)



E-R Diagram for a University Enterprise



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.
    - One relation schema for each entity set.
    - One relation schema for each relationship set.
  - – However typically only the many-to-many relationships end up being represented as relation schemas, due to obvious simplifications

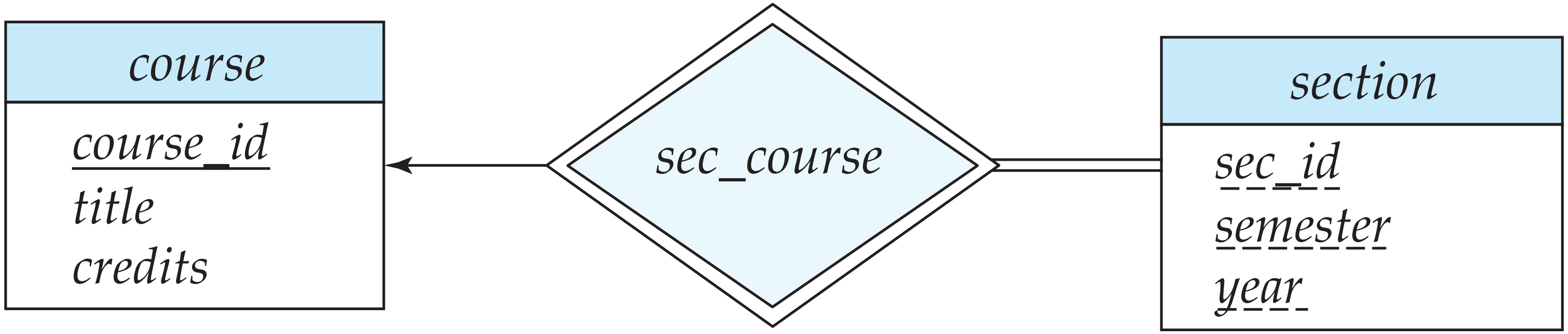
Representing Entity Sets

- A strong (non-weak) **entity set** reduces to a schema with the same attributes

  course(course_id, title, credits)

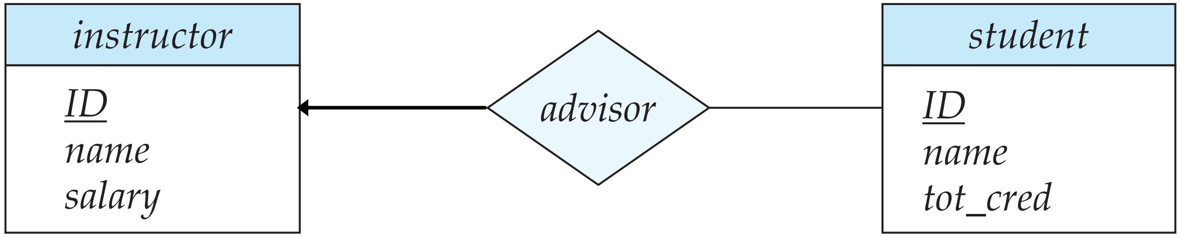
- A **weak entity set** becomes a table that includes columns for the primary key of the identifying strong entity set

  section (course_id, sec_id, semester, year)

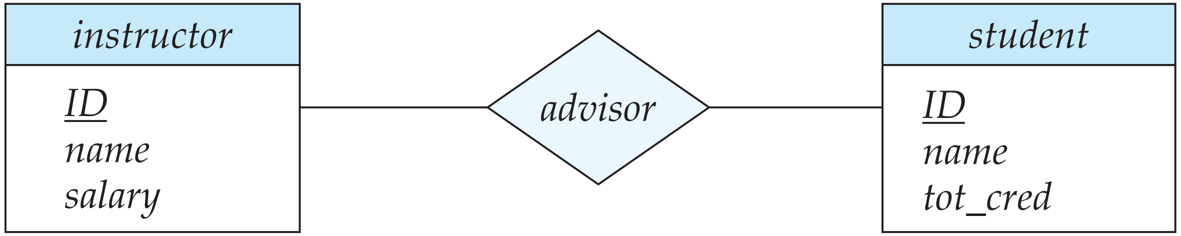


Representing Relationship Sets

- A relationship set can be represented as a schema with attributes for the primary keys of the two participating entity sets
  Primary key attributes
  - many-to-one: the primary key attributes from the many-side
  - many-to-many: the primary key attributes from both sides
- Example:



advisor = (*student_id* , *instructor_id*)

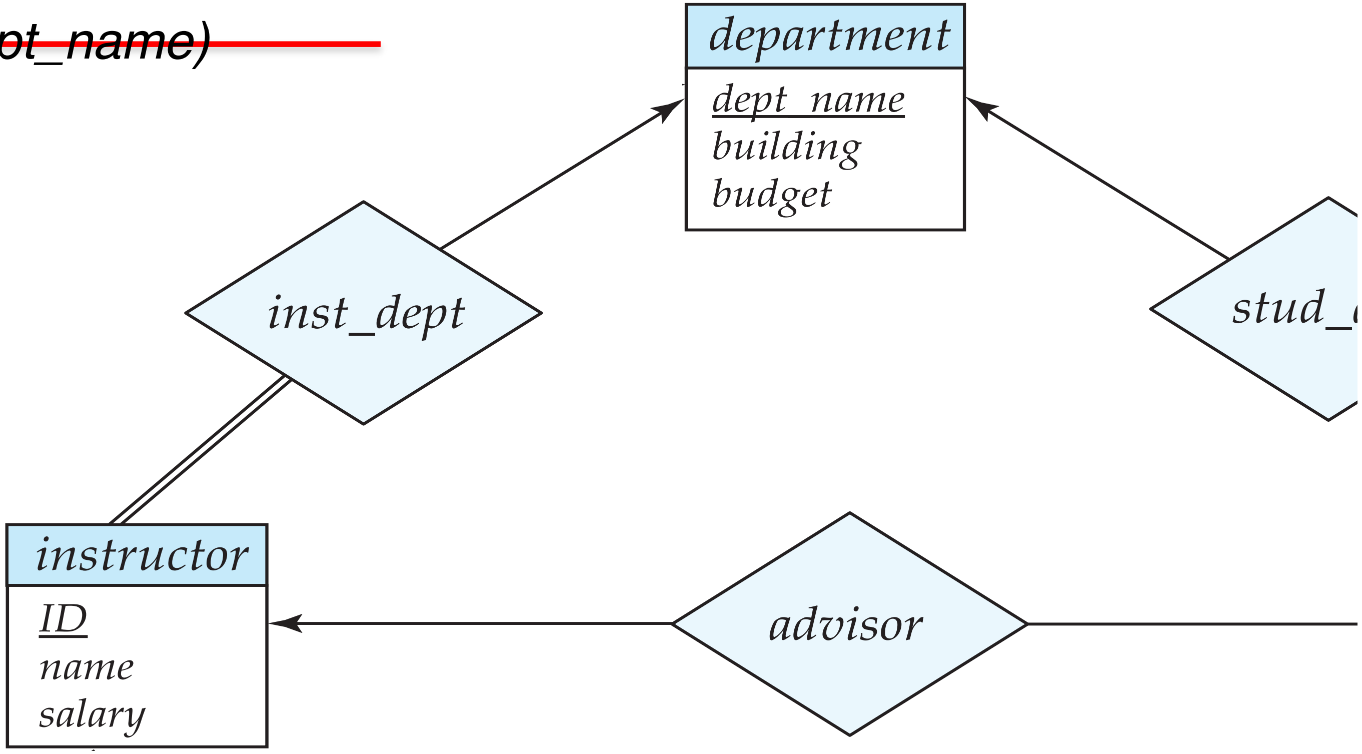


advisor = (*student_id* , *instructor_id*)

- Descriptive attributes of the relationship set may be added.

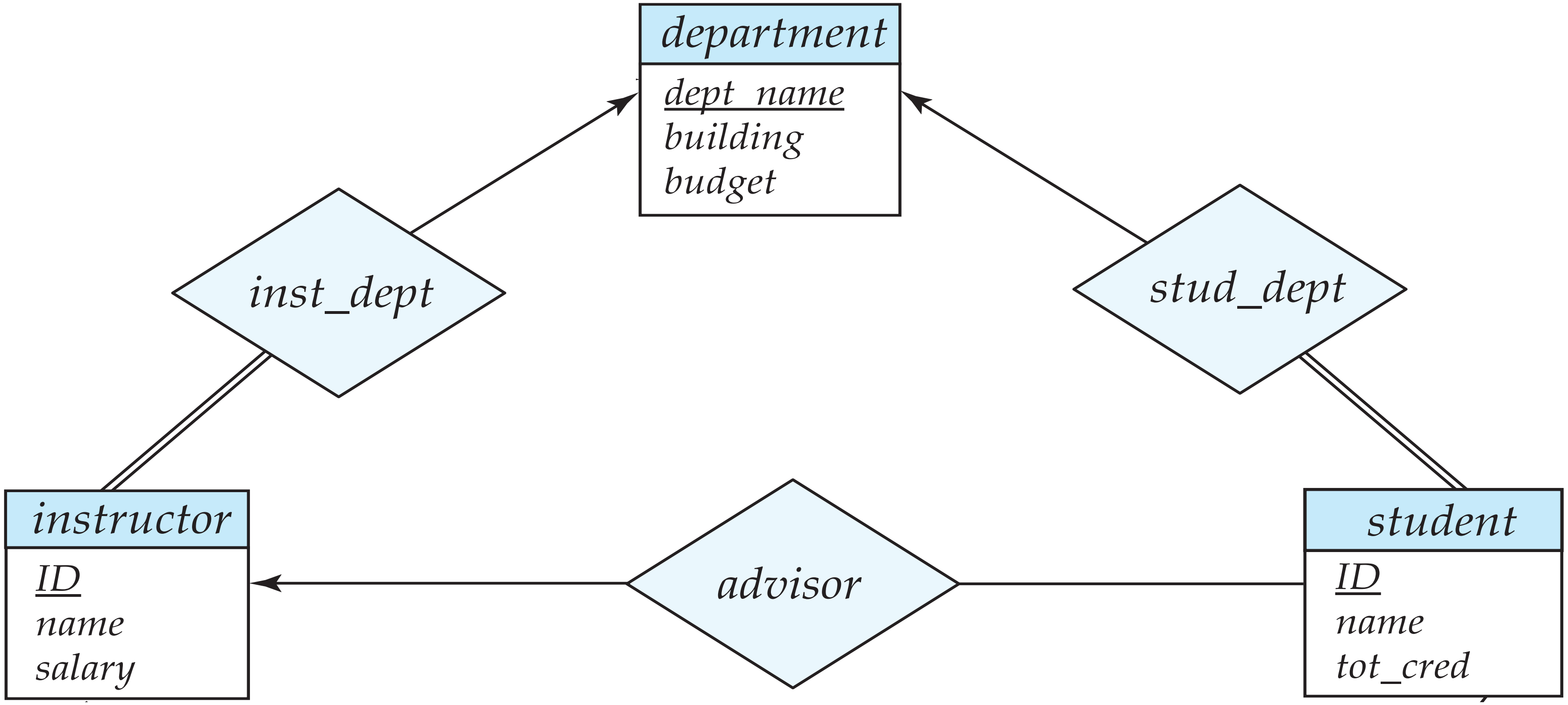
Redundancy of Schemas

- Relation schema for *instructor* and *inst_dept* will then be
  - *instructor(ID, name, salary)*
  - *inst_dept(ID, dept_name)*
- An obvious simplification would be to drop *inst_dept* and include the *dept_name* attribute in instructor
  - *instructor(ID, name, dept_name, salary)*
  - ~~*inst_dept(ID, dept_name)*~~



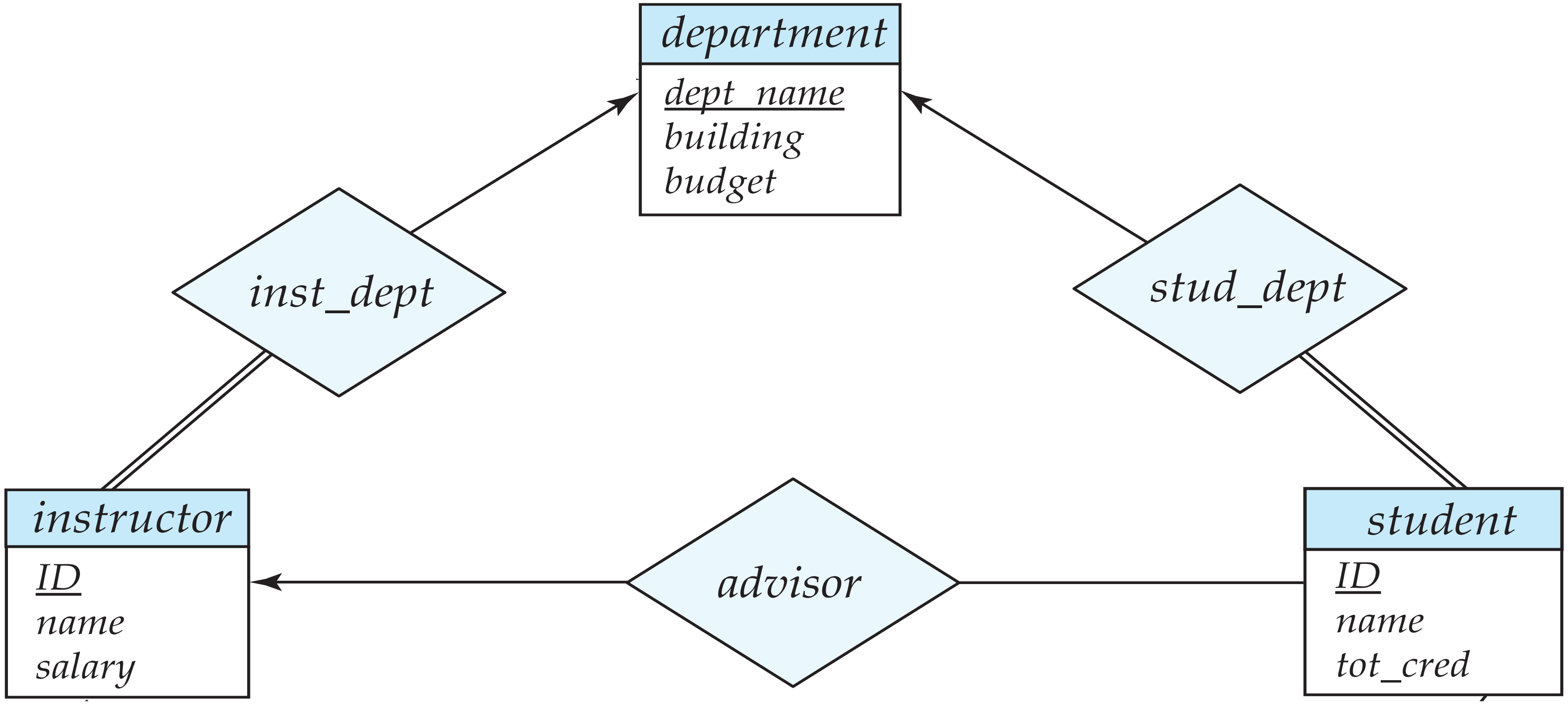
Redundancy of Schemas

- In general many-to-one and one-to-many relationship sets can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
  - *instructor(ID, name, dept_name, salary)*
  - *student(ID, name, dept_name, advisor_id, tot_cred)*



Redundancy of Schemas

- *instructor(ID, name, dept_name, salary)*
- *student(ID, name, dept_name, advisor_id, tot_cred)*



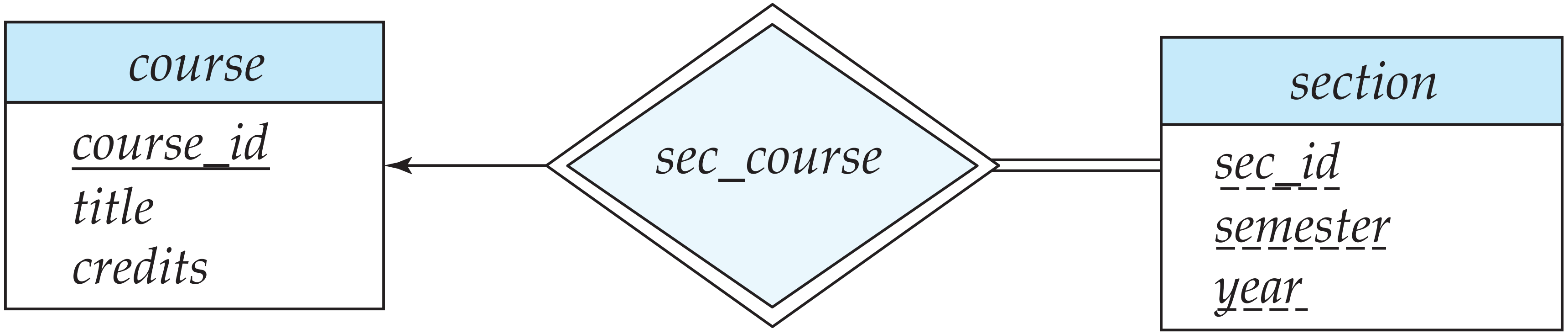
❑ If participation is *partial* on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values

❑ Thus, *advisor_id* in *student* may be null

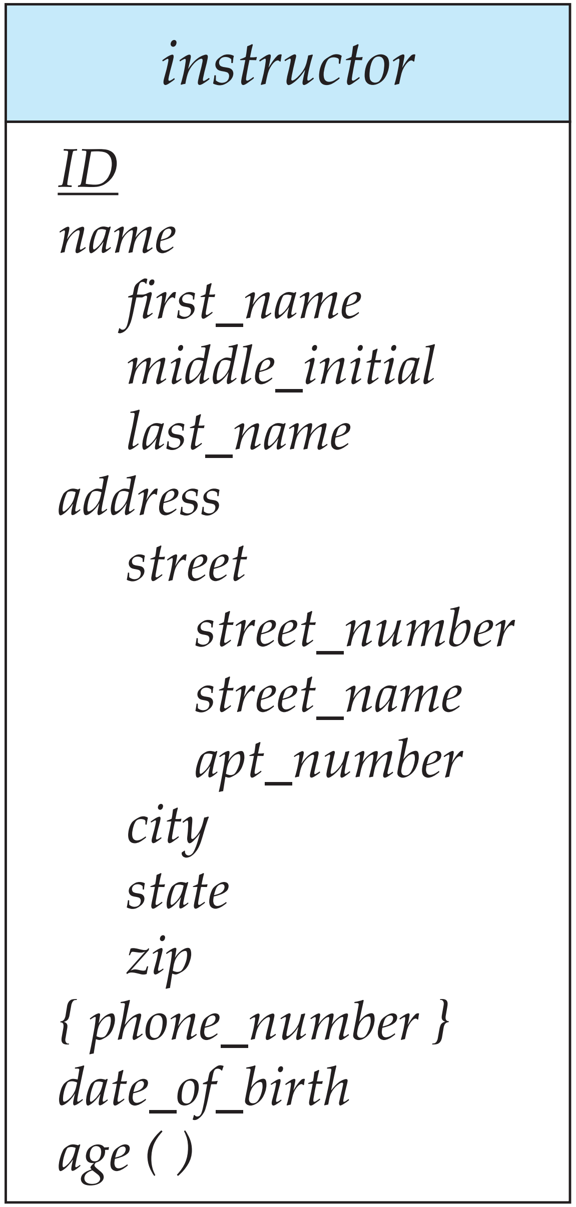
Redundancy of Schemas (Cont.)

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets

- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - Example: The *section* schema
    section(course_id, sec_id, semester, year)
    already contains the attributes that would appear in the *sec_course* schema:
    sec_course (course_id, sec_id, semester, year)
  -



Composite and Multivalued Attributes



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 are flattened out** by creating a separate attribute for each component attribute
  - – Example: 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 *first_name* and *last_name*

- ❑ Ignoring multivalued attributes, the 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)

Composite and 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 separate schema:

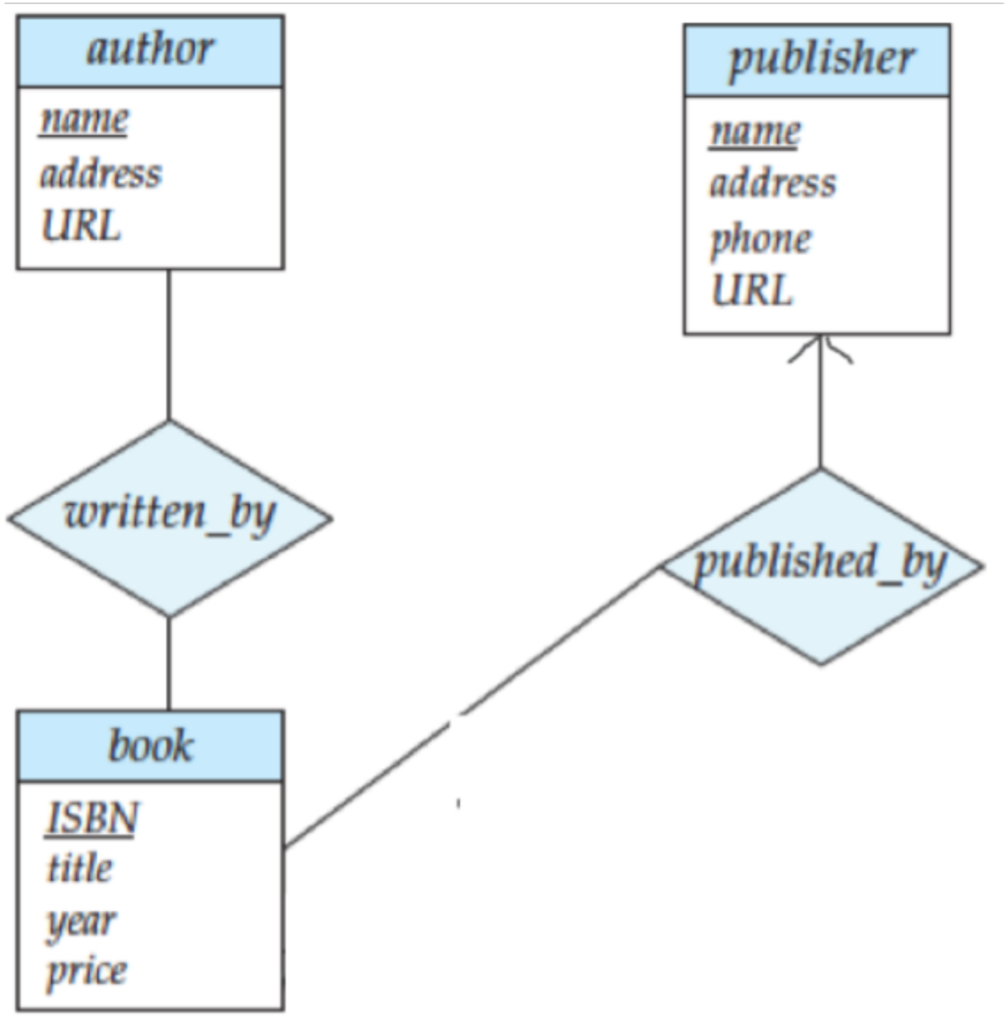
    inst_phone (*ID, phone_number*)

  - – 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)

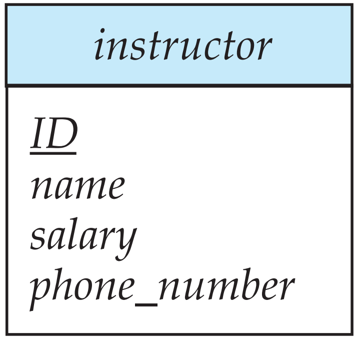
Reduction to relational schema

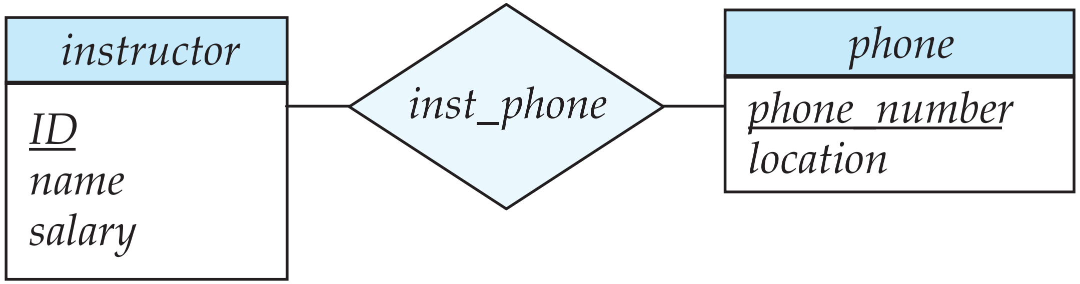
- **How to reduce this?**



Design Issues

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



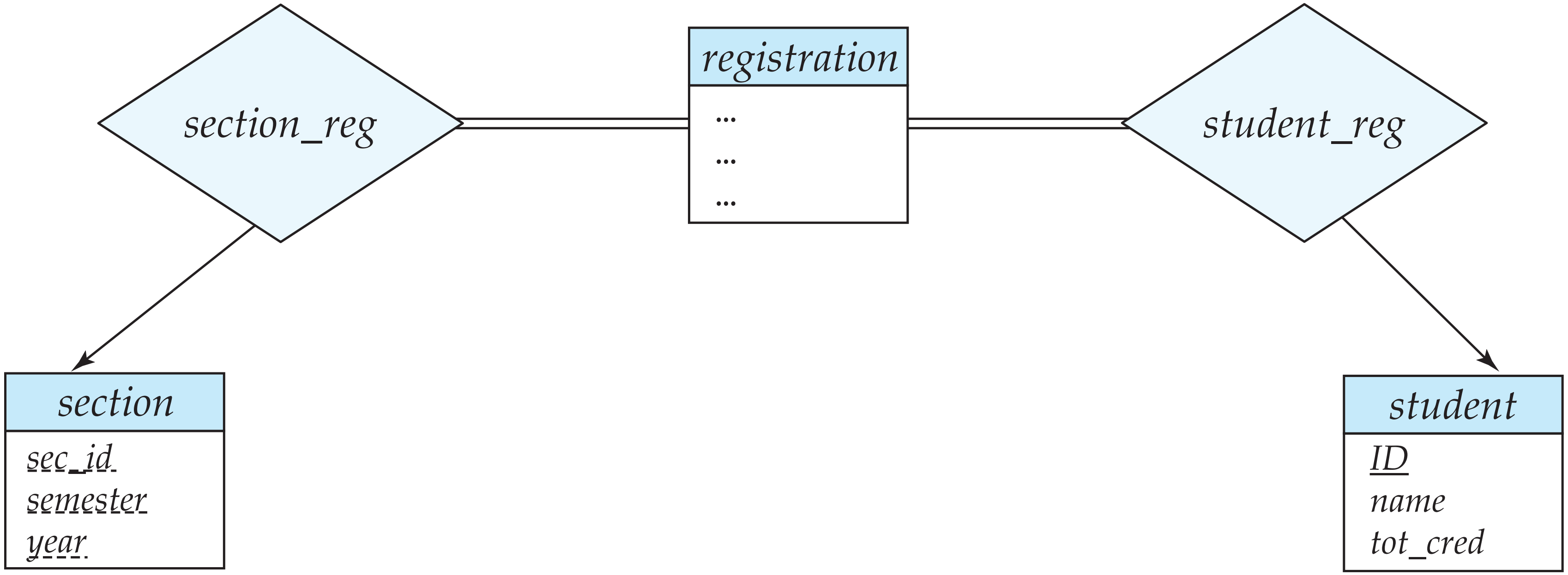


- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Design Issues

- **Use of entity sets vs. relationship sets**
  A many-to-many relationship can also be represented as an entity set

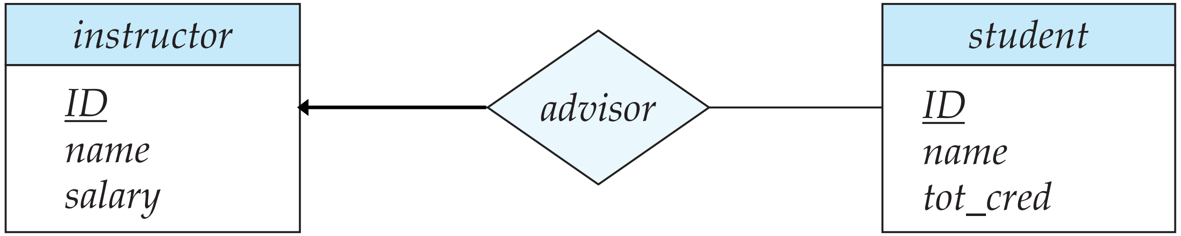
- What would be an alternative to this?



Design Issues

- **Placement of relationship attributes**

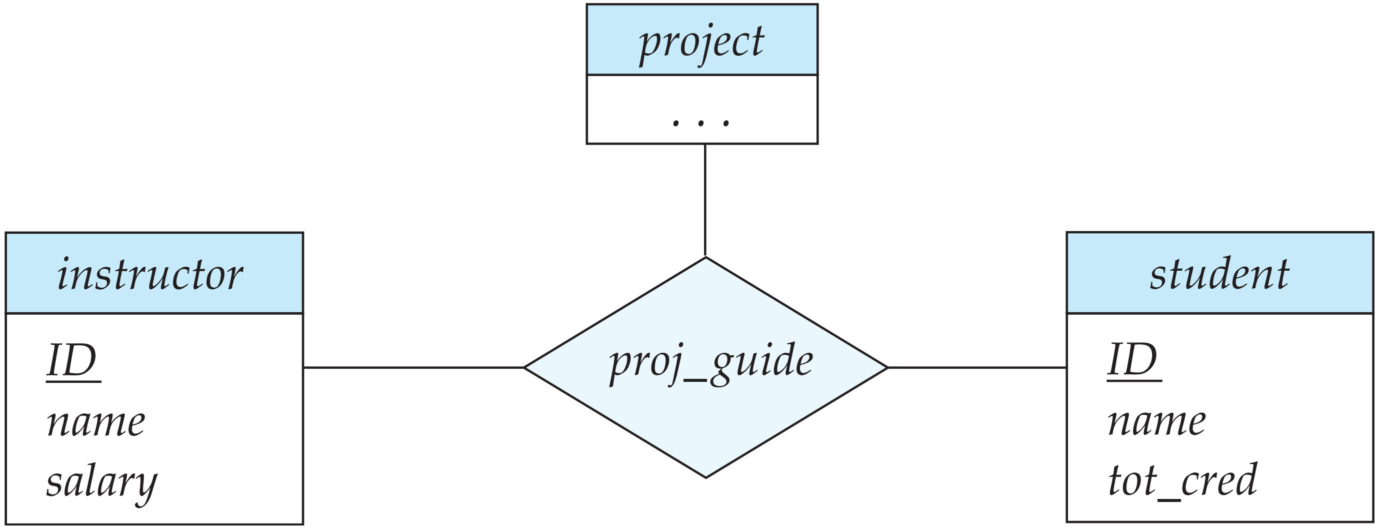
  e.g., attribute *date* as attribute of *advisor* or as attribute of *student*



Design Issues

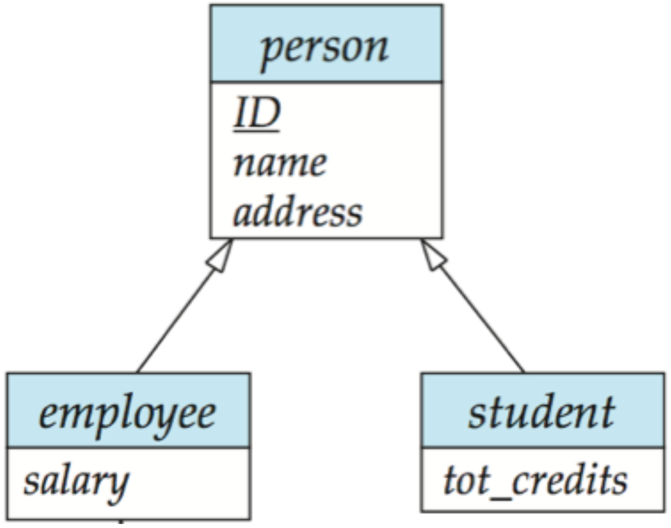
❑ Binary Vs. Non-Binary Relationships

- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g., 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*



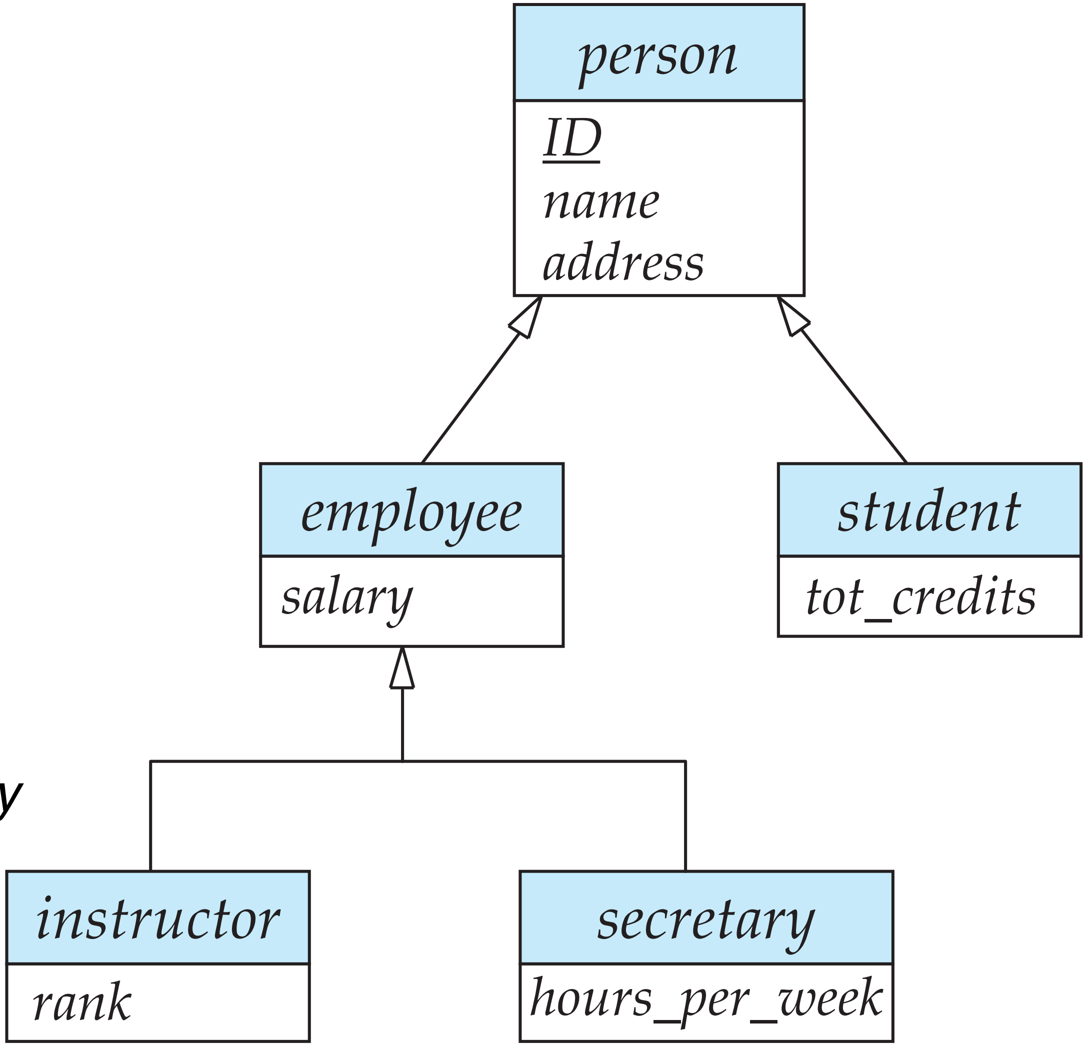
Extended E-R Features: Specialization

- Design process
  - we designate subgroupings within an entity set that are distinctive from other entities in the set.
- **Subgroupings (specializations)**
  - 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 hollow arrow-head pointing from the specialized entity to the other entity
  - also referred as an ISA relationship (E.g., *student* "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.



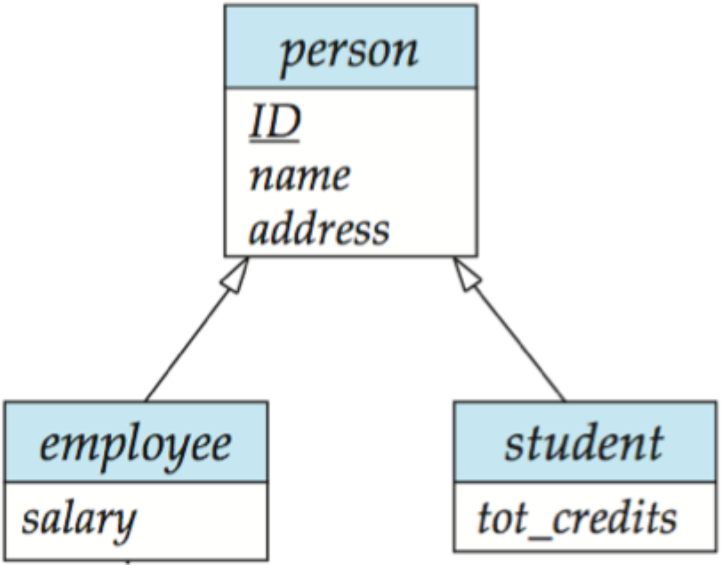
Specialization Example

- Overlapping
  - *employee* and *student*
- Disjoint
  - *instructor* and *secretary*



Reducing Specialization to Schemas

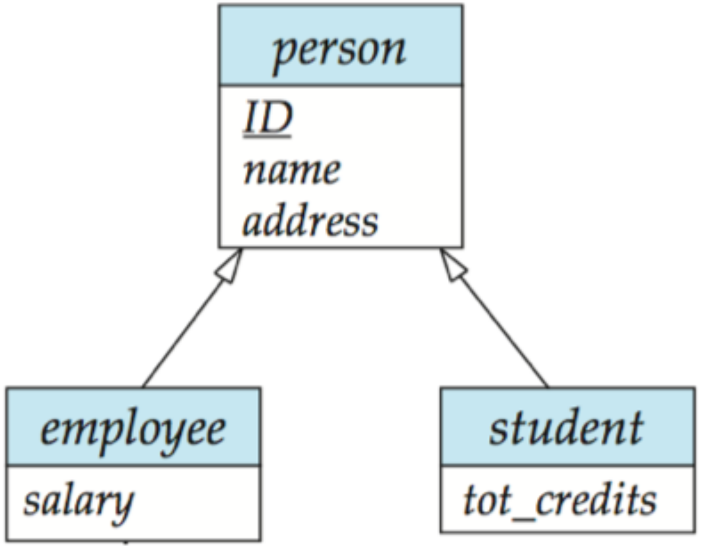
- Method 1:
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes



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

Reducing Specialization to 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, then the schema for the generalized entity set (*person*) is 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

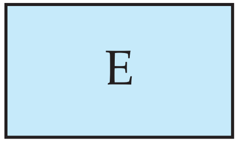
E-R Design Decisions - Summary

- ❑ 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 combination of **binary** relationships.
- ❑ The use of a **strong or weak** entity set.
- ❑ The use of **specialization** / generalization – contributes to modularity in the design.

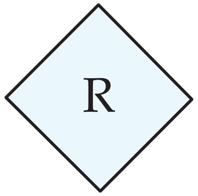
Doing another ER design

- ❑ Requirements for the database
  - – We need to register persons with unique social security number (ssn) and name.
  - – A person can either be a student, for which we store status, or a teacher for which we record salary.
  - – Further, we need departments with unique id’s (did) and name, as well as projects with project id (proid), name, and module.
  - – A student can participate in at most one project, and every project project is assigned a teacher who serves as supervisor.
  - – A student studies at one or more departments, a teacher is employed in exactly one department.

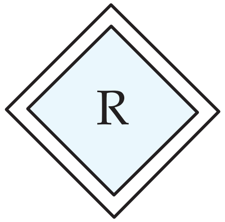
Summary of Symbols Used in E-R Notation



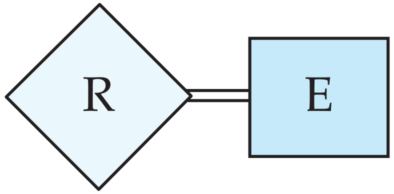
entity set



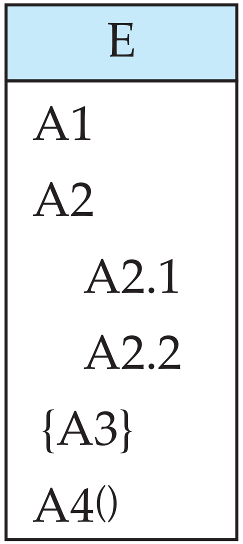
relationship set



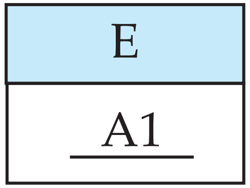
identifying
relationship set
for weak entity set



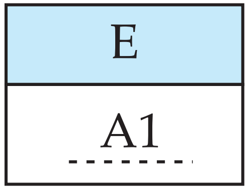
total participation
of entity set in
relationship



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

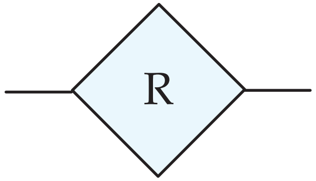


primary key

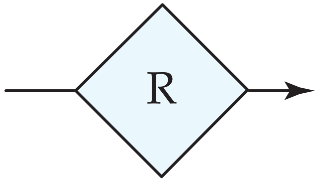


discriminating
attribute of
weak entity set

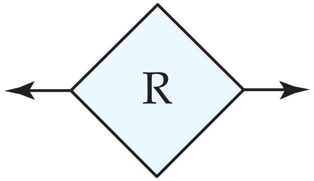
Symbols Used in E-R Notation (Cont.)



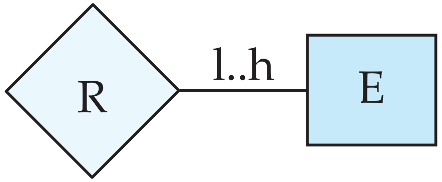
many-to-many relationship



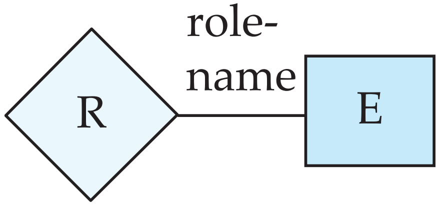
many-to-one relationship



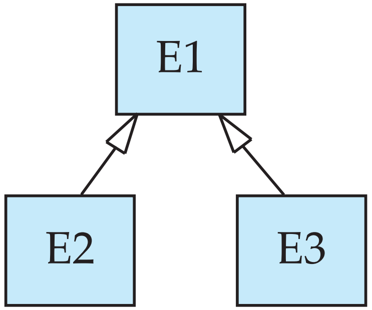
one-to-one relationship



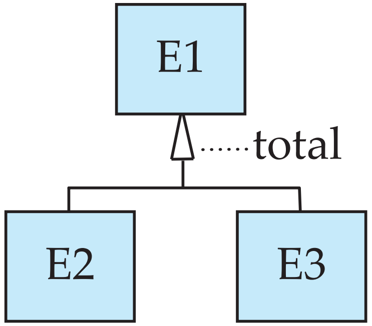
cardinality limits



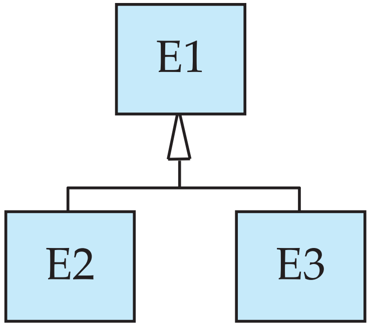
role indicator



ISA: generalization or specialization



total (disjoint) generalization



disjoint generalization

❑ Do ER modeling from scratch

❑ Don’t just reverse engineer!

Don’t just reverse engineer!

- ❑ Reverse engineering
  - – an option in many DBMS’, including MySQL
  - – given a relational schema, produce an ER-model that would translate into this relational schema
  - – build on foreign key references

- ❑ Don’t just reverse engineer! Reason no One
  - – you’ll need the skills it takes to model a database without regard to the specific tables and foreign keys
  - – don’t underestimate the power and benefits of simplifyed database drawings in development

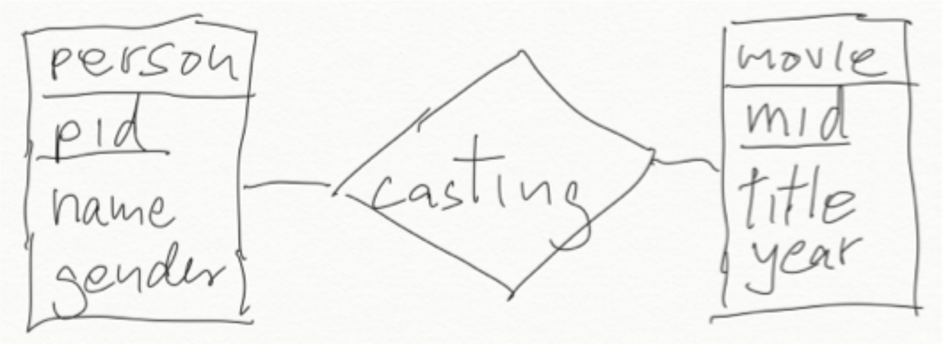
Don’t just reverse engineer!

- ❑ Reverse engineering in MySQL
  - – lead to a diagram they call EER-diagram, but this is a rather limited notation
    - the notation is also called a craw foot diagram
  - – don’t support many-to-many
  - – is very close to the final relational model

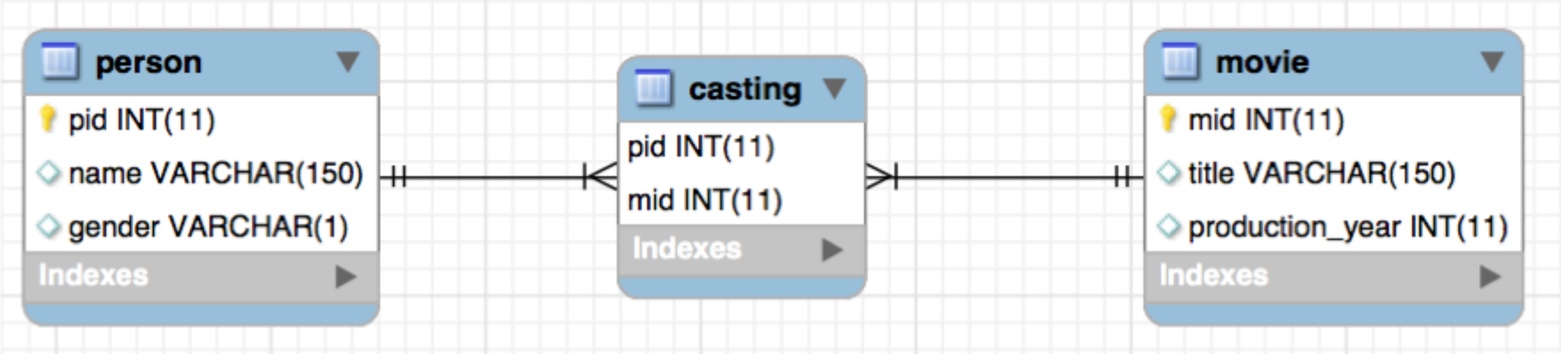
- ❑ Don’t just reverse engineer! Reason no Two
  - – the more common ER notation of the DBSC book do have more expressive power and
  - – diagrams tend to become far more readable

Compare …

- The simple movie database
  - the common ER notation of the DBSC book

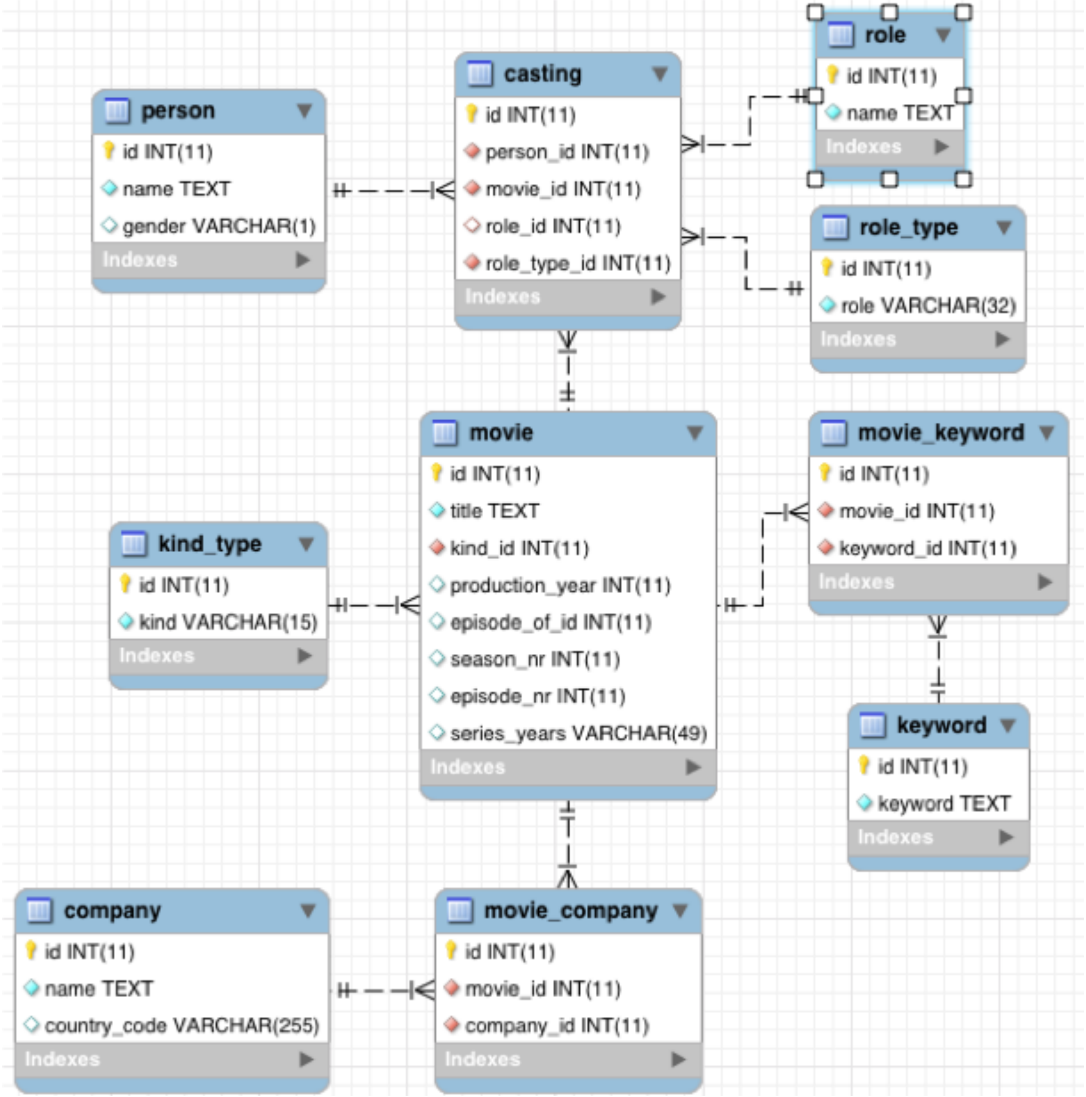


  - this is what you get when you reverse engineer a 3-table relational schema ”implementing” (or ”engineering”) this database



The database from assignment 1

- ❑ The model included in the assignment was in fact reverse engineered using the tool for that purpose included in MySQL



role
- id INT(11)
- name TEXT
- Indexes

casting
- id INT(11)
- person_id INT(11)
- movie_id INT(11)
- role_id INT(11)
- role_type_id INT(11)
- Indexes

person
- id INT(11)
- name TEXT
- gender VARCHAR(1)
- Indexes

role_type
- id INT(11)
- role VARCHAR(32)
- Indexes

movie
- id INT(11)
- title TEXT
- kind_id INT(11)
- production_year INT(11)
- episode_of_id INT(11)
- season_nr INT(11)
- episode_nr INT(11)
- series_years VARCHAR(49)
- Indexes

movie_keyword
- id INT(11)
- movie_id INT(11)
- keyword_id INT(11)
- Indexes

kind_type
- id INT(11)
- kind VARCHAR(15)
- Indexes

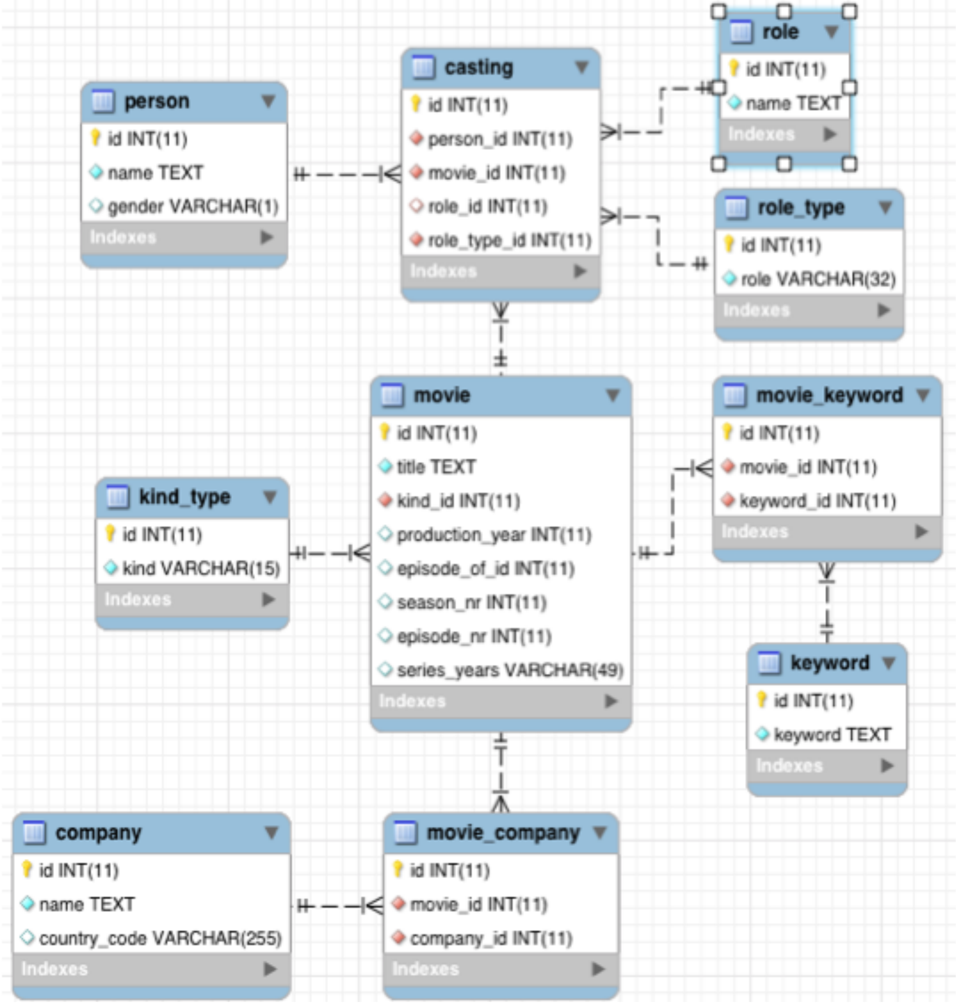
keyword
- id INT(11)
- keyword TEXT
- Indexes

company
- id INT(11)
- name TEXT
- country_code VARCHAR(255)
- Indexes

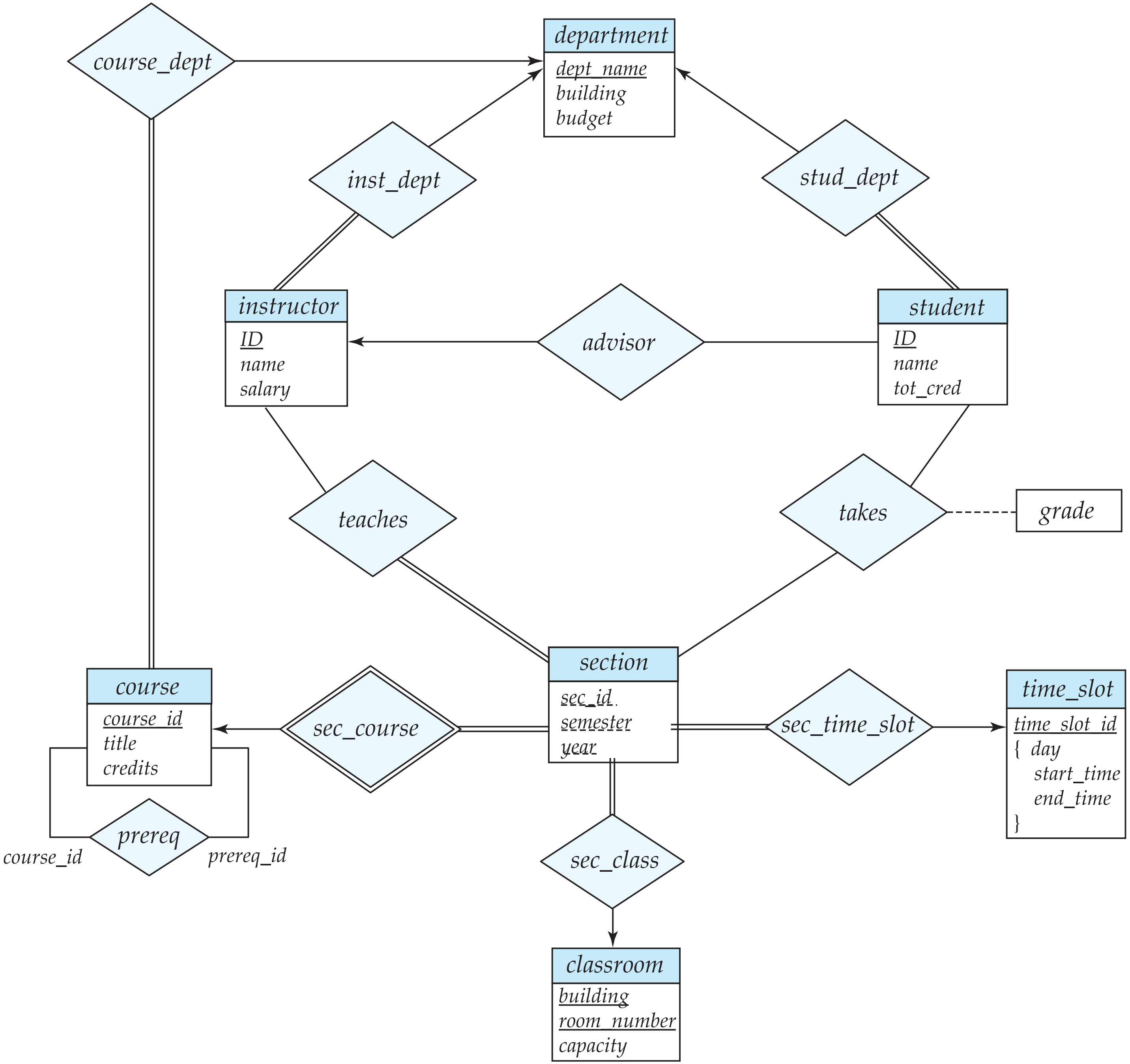
movie_company
- id INT(11)
- movie_id INT(11)
- company_id INT(11)
- Indexes

The database from assignment 1

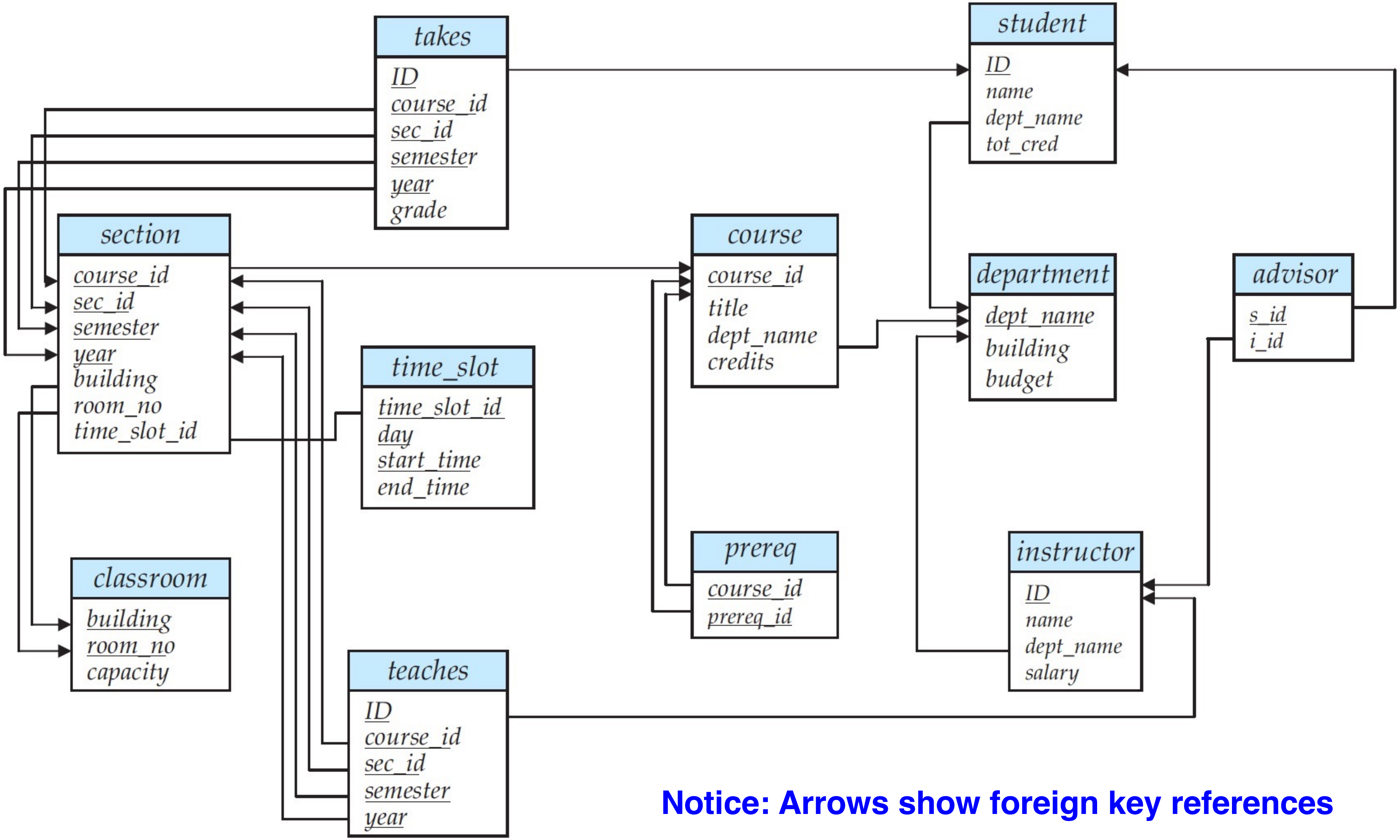
- ❑ Exercise:
  - provide a ”real” ER diagram corresponding to this craw foot diagram or, if you prefer, start with the imdb_movie relational schema
  - feel free to improve the database design
    - e.g. to drop artificial primary key attributes, when these serve no purpose



E-R Diagram for a University Enterprise



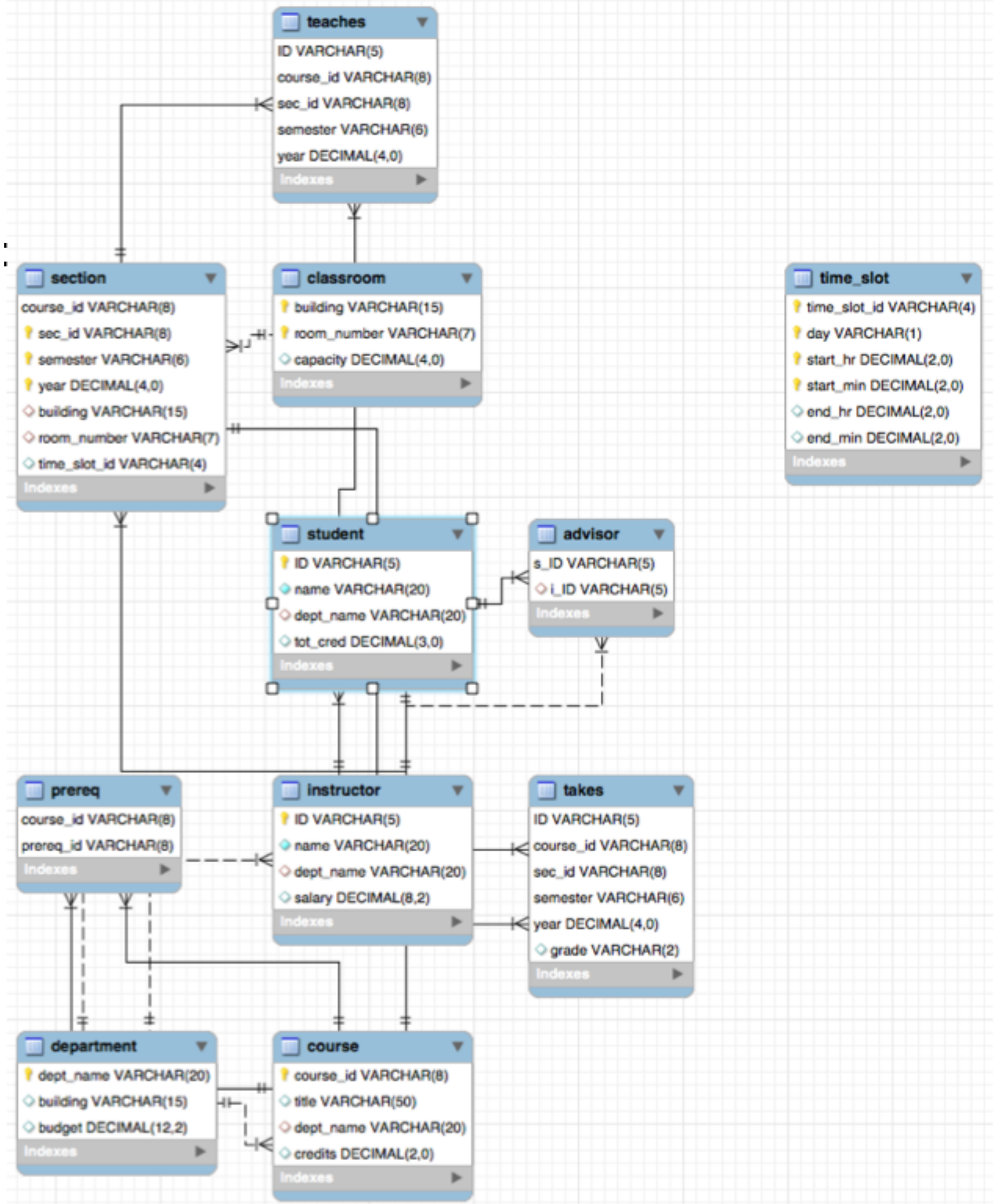
Schema Diagram for University Database



Notice: Arrows show foreign key references

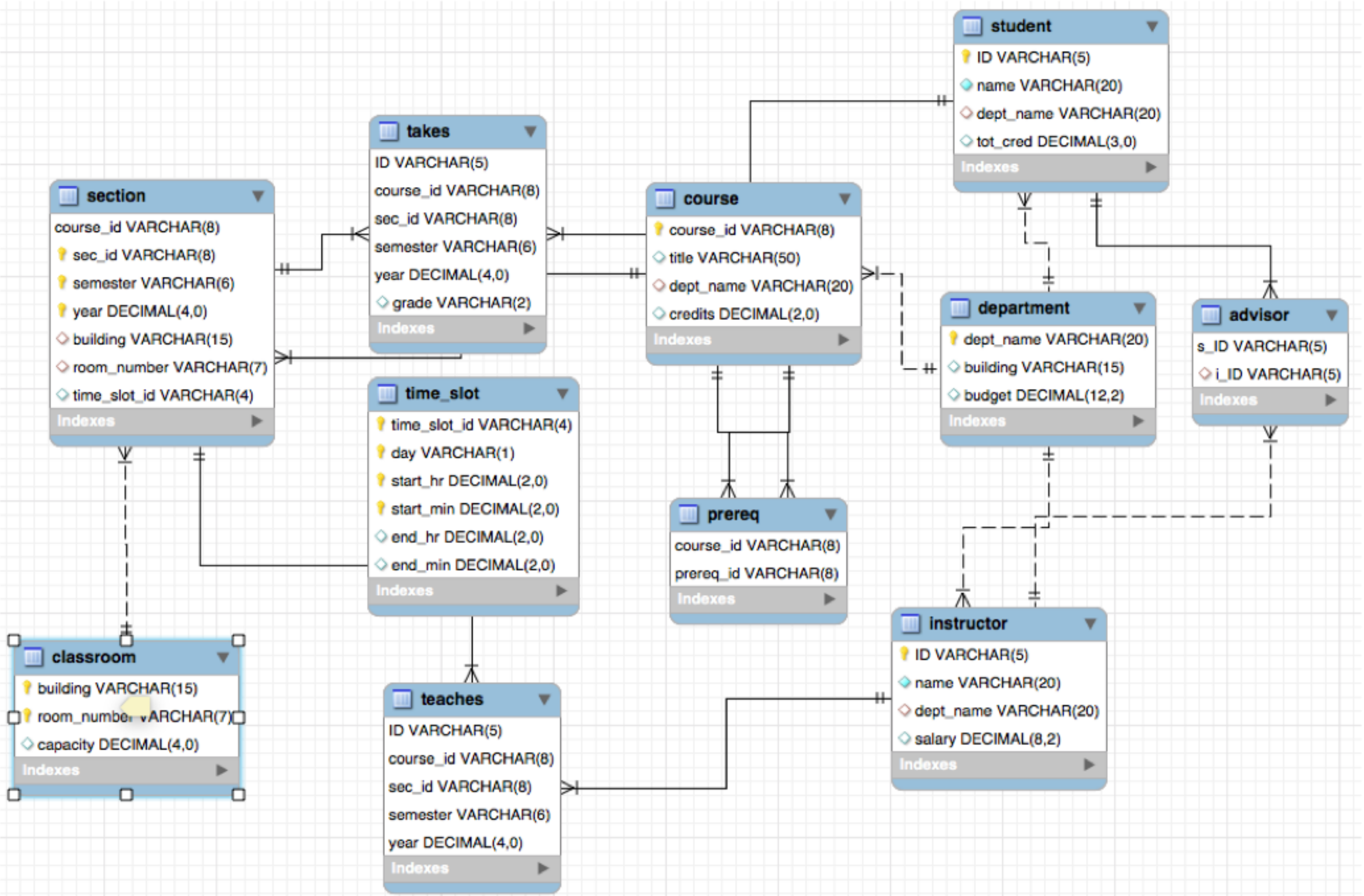
University database reverse engineered

❑ Reverse engineered in MySQL:



University database reverse engineered

- ❑ Now reordered similar to the schema diagram from the DBSC book



student
- ID VARCHAR(5)
- name VARCHAR(20)
- dept_name VARCHAR(20)
- tot_cred DECIMAL(3,0)

takes
- ID VARCHAR(5)
- course_id VARCHAR(8)
- sec_id VARCHAR(8)
- semester VARCHAR(6)
- year DECIMAL(4,0)
- grade VARCHAR(2)

section
- course_id VARCHAR(8)
- sec_id VARCHAR(8)
- semester VARCHAR(6)
- year DECIMAL(4,0)
- building VARCHAR(15)
- room_number VARCHAR(7)
- time_slot_id VARCHAR(4)

course
- course_id VARCHAR(8)
- title VARCHAR(50)
- dept_name VARCHAR(20)
- credits DECIMAL(2,0)

department
- dept_name VARCHAR(20)
- building VARCHAR(15)
- budget DECIMAL(12,2)

advisor
- s_ID VARCHAR(5)
- i_ID VARCHAR(5)

time_slot
- time_slot_id VARCHAR(4)
- day VARCHAR(1)
- start_hr DECIMAL(2,0)
- start_min DECIMAL(2,0)
- end_hr DECIMAL(2,0)
- end_min DECIMAL(2,0)

prereq
- course_id VARCHAR(8)
- prereq_id VARCHAR(8)

classroom
- building VARCHAR(15)
- room_number VARCHAR(7)
- capacity DECIMAL(4,0)

teaches
- ID VARCHAR(5)
- course_id VARCHAR(8)
- sec_id VARCHAR(8)
- semester VARCHAR(6)
- year DECIMAL(4,0)

instructor
- ID VARCHAR(5)
- name VARCHAR(20)
- dept_name VARCHAR(20)
- salary DECIMAL(8,2)