CS34800
Information Systems

Entity-Relationship Model

Prof. Chris Clifton

16 September 2016

Indiana
Center for
Database
Systems
™

Announcements

- Project 1 due at 11:59pm
  - Some concerns about Part 2 (since we didn't cover some of the material until last class). Possible extension? (Frequency code DD)
    - A. Deadline for both Part 1 and Part 2 tonight 11:59pm
    - B. Part 1 due tonight 11:59pm, Part 2 tomorrow night 11:59pm
- Assignment 2 will be out later today
  - Relational Algebra

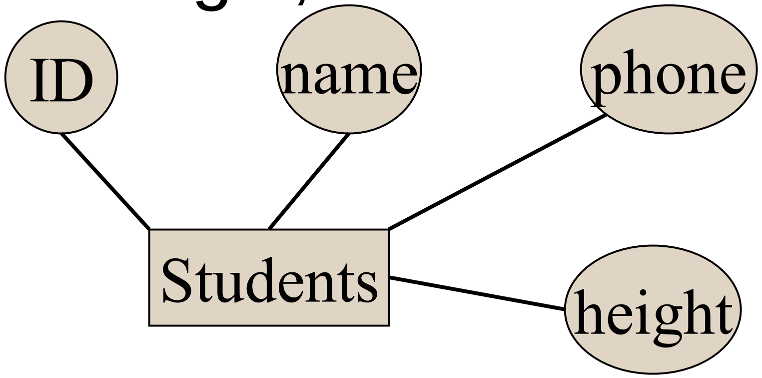
Main categories of data models

- **Logical models**: used to describe, organize and access data in DBMS; application programs refers to such models. They are independent from the physical data structures
  - examples: relational data model, hierarchical data model, object-relational data model
- **Conceptual models**: support the representation of data independently from specific DBMS. Their goal is to provide representations, which are rich in semantics, of the real word entities, their properties and relationships. These models are mainly used for the conceptual design of databases
  - The **Entity-Relationship** is the most well known model in such category

Entity/Relationship Model

Diagrams to represent designs.

- *Entity* like object, = “thing.”
- *Entity set* like class = set of “similar” entities/objects.
- *Attribute* = property of entities in an entity set, similar to fields of a struct.
- In diagrams, entity set → rectangle; attribute → oval.

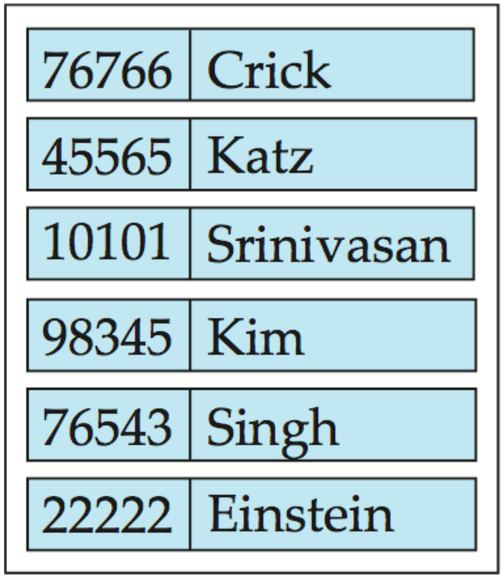


Entity Sets

- 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; i.e., descriptive properties possessed by all members of an entity set.
  - Example:

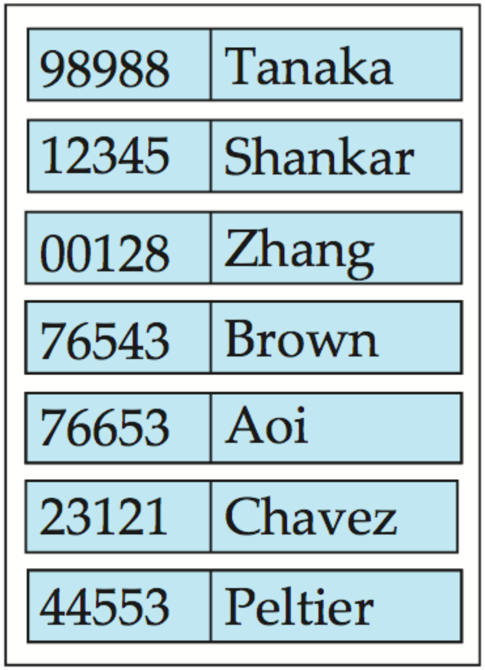
    instructor = (*ID, name, street, city, salary*)
    course= (*course_id, title, credits*)
- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifiying each member of the set.

Entity Sets -- *instructor* and *student*



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

instructor

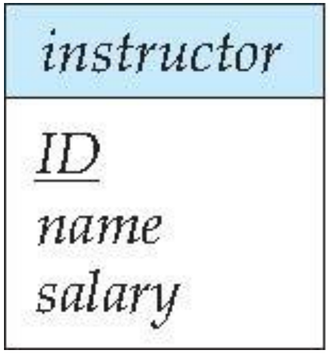


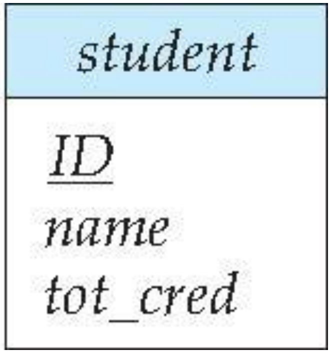
student-ID	student_name
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Entity Sets

- Entities can be represented graphically as follows:
  - Rectangles represent entity sets.
  - Attributes listed inside entity rectangle
  - Underline indicates primary key attributes





Relationships

- Connect two or more entity sets.
- Represented by diamonds.

Relationship Set

Think of the "value" of a relationship set as a table.

- One column for each of the connected entity sets.
- One row for each list of entities, one from each set, that are connected by the relationship.

Students	Courses
Sally	CS348
Sally	CS355
Joe	CS348
…	…

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

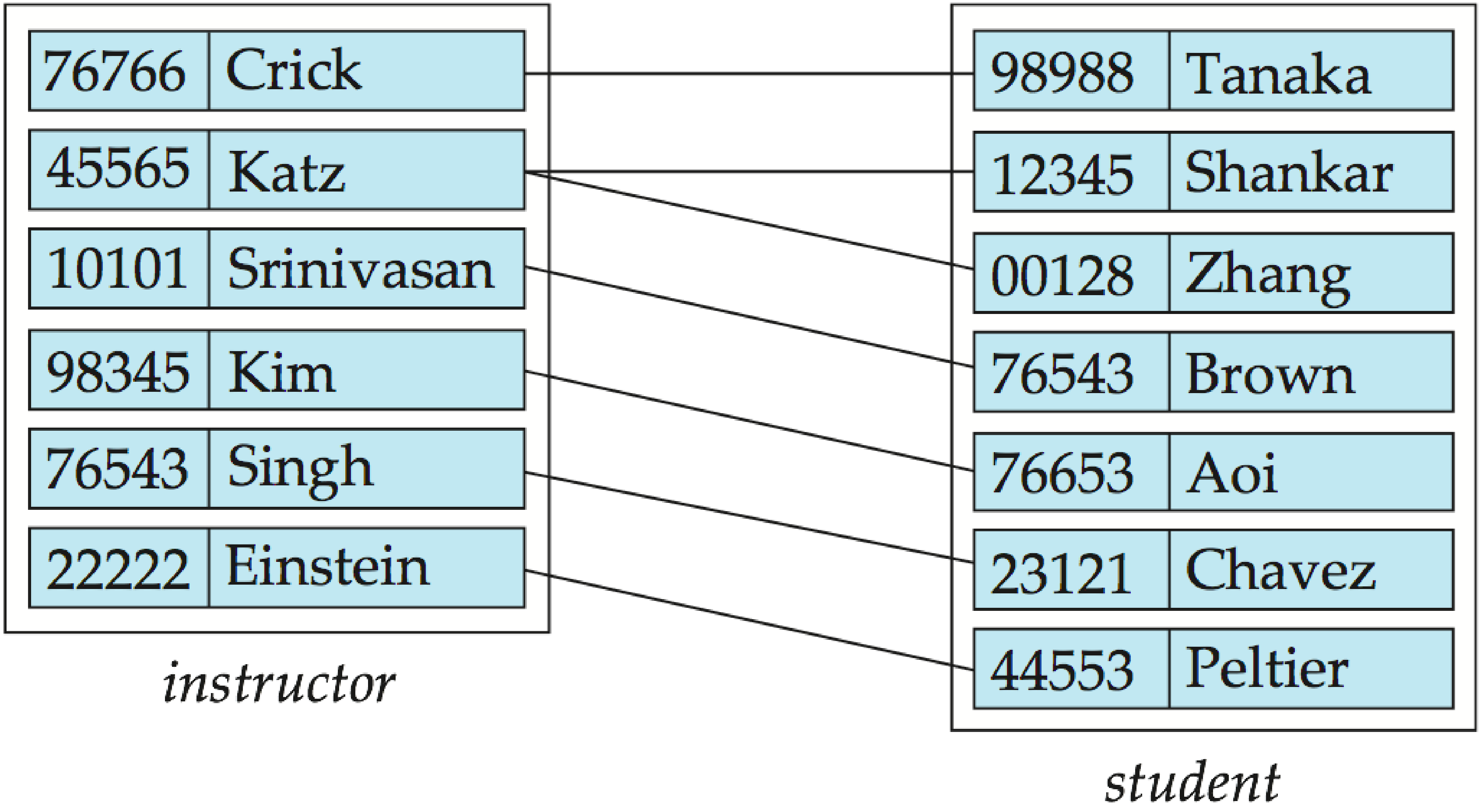
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

    $(44553,22222) \in advisor$

Relationship Set *advisor*



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

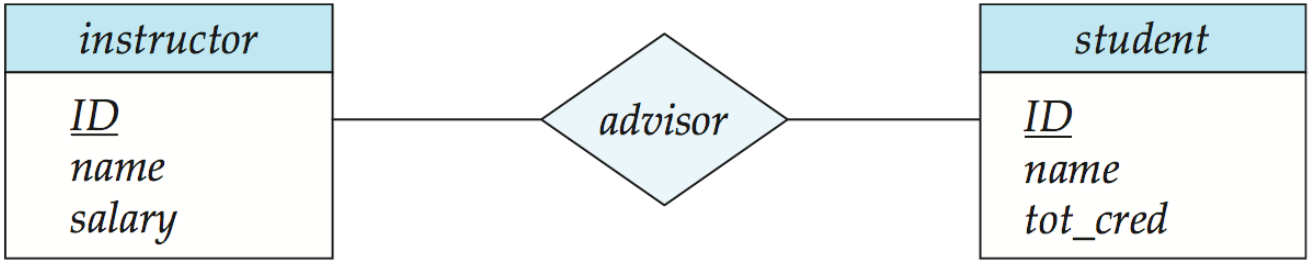
instructor

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

student

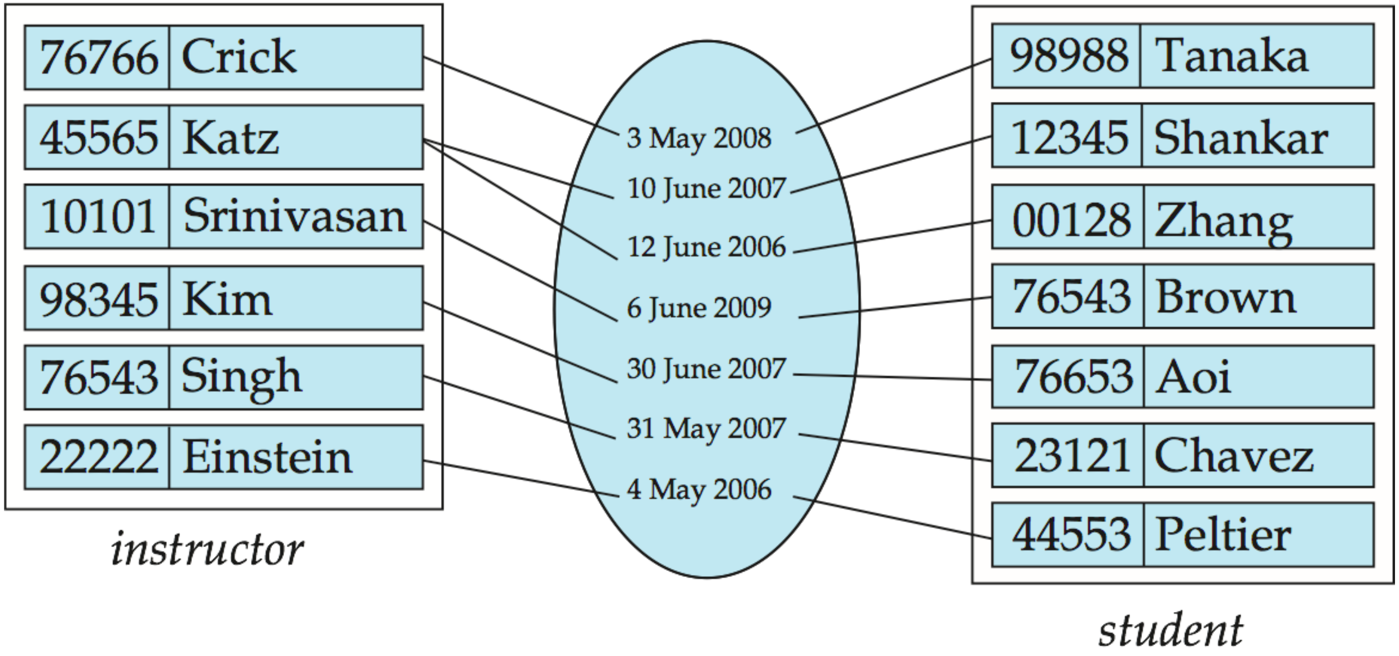
Relationship Sets

- Diamonds represent relationship sets.

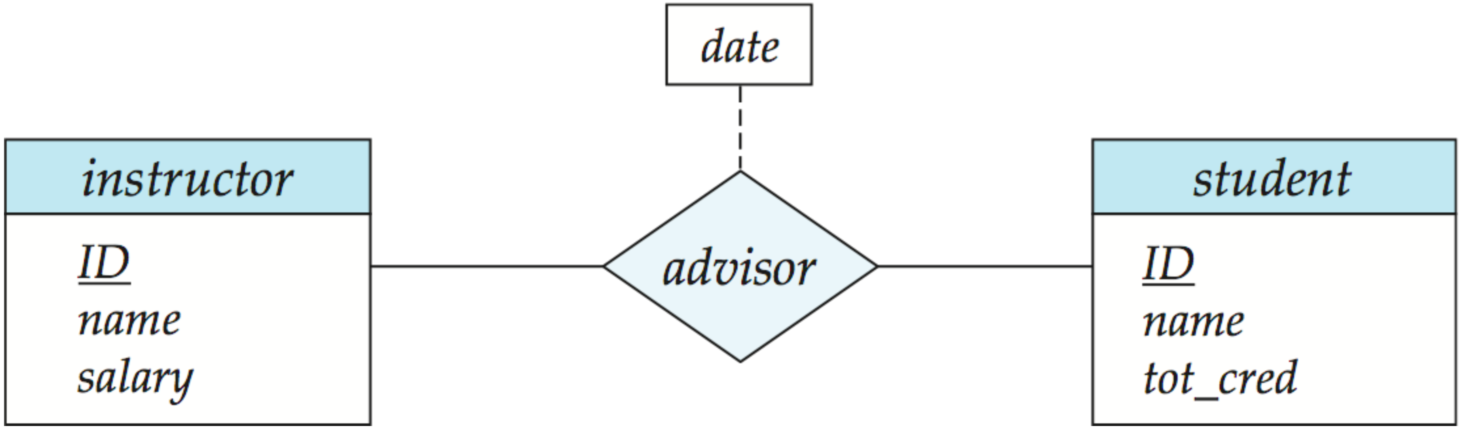


Relationship Sets (Cont.)

- An attribute can also be associated with 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



Relationship Sets with Attributes



Multiway relationships

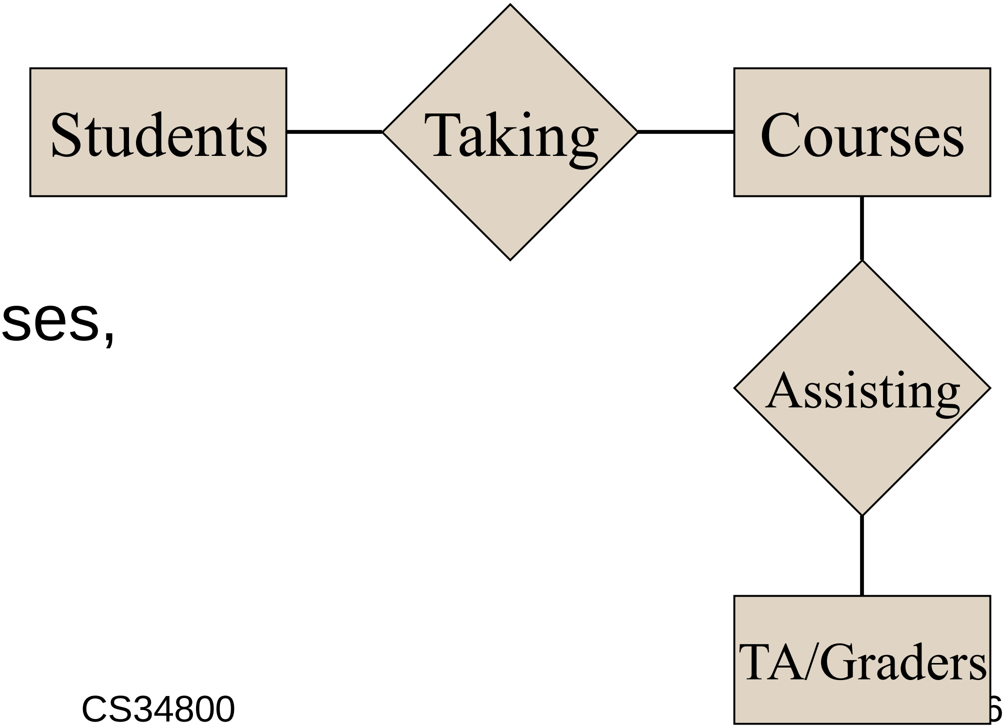
- Professor/student works
- But does this work for TAs?
  - Assignment of student to TA not captured in “course to student” relationship

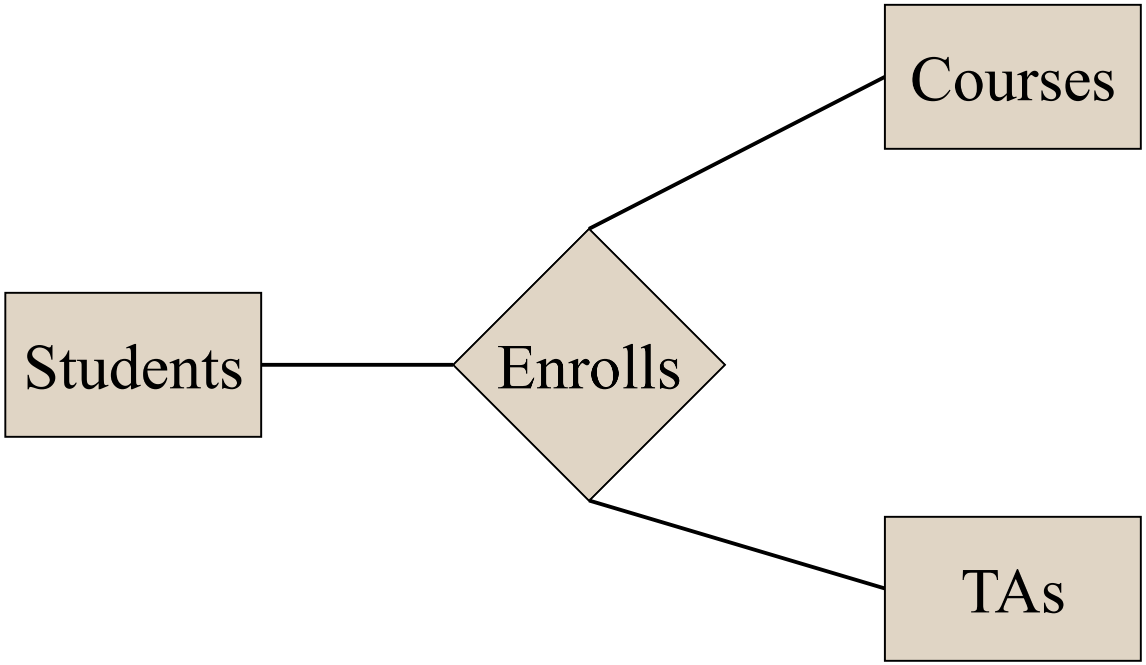
Multiway Relationships

Usually binary relationships (connecting two E.S.) suffice.

- However, there are some cases where three or more E.S. must be connected by one relationship.
- Example: relationship among students, courses, TA's (and graders).

Possibly, this E/R diagram is OK:

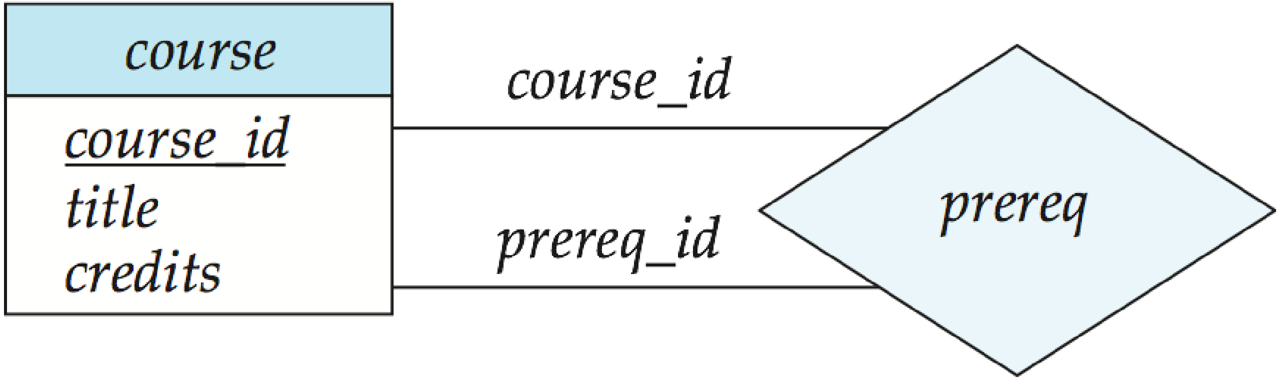




Students	Courses	TAs
Ann	CS348	Denis
Sue	CS348	Romila
Bob	CS348	Devesh
…	…	…

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**.



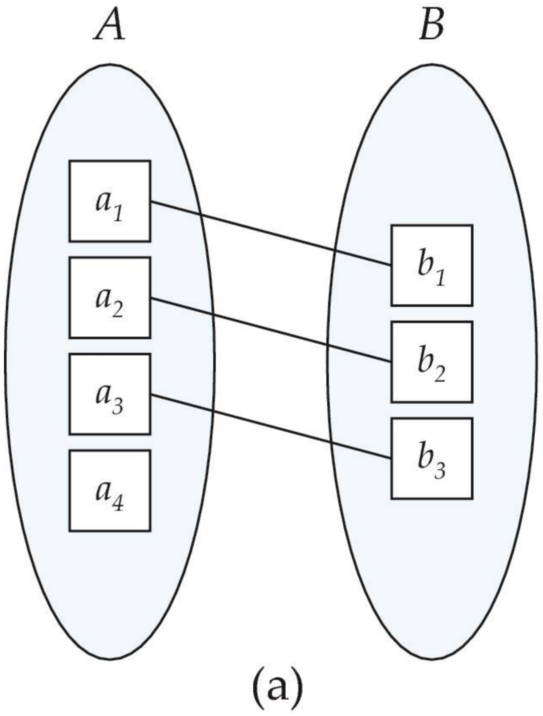
Degree of a Relationship Set

- 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. (More on this later.)
    - Example: *students* work on research *projects* under the guidance of an *instructor*.
    - relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

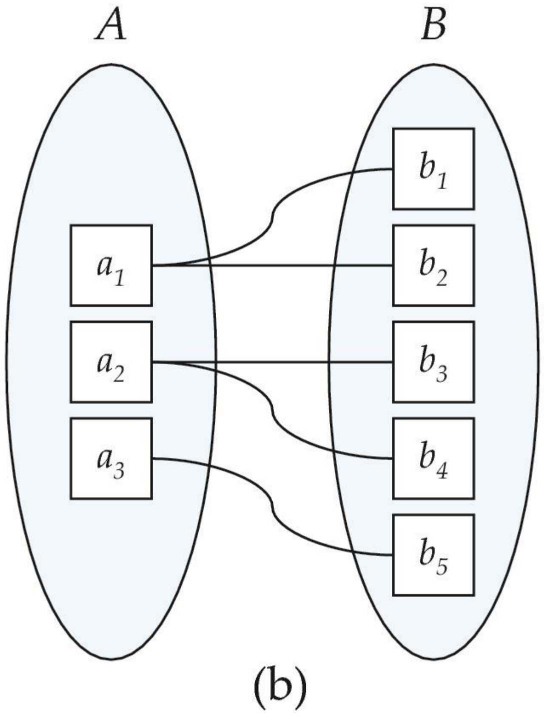
Mapping Cardinality Constraints

- 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



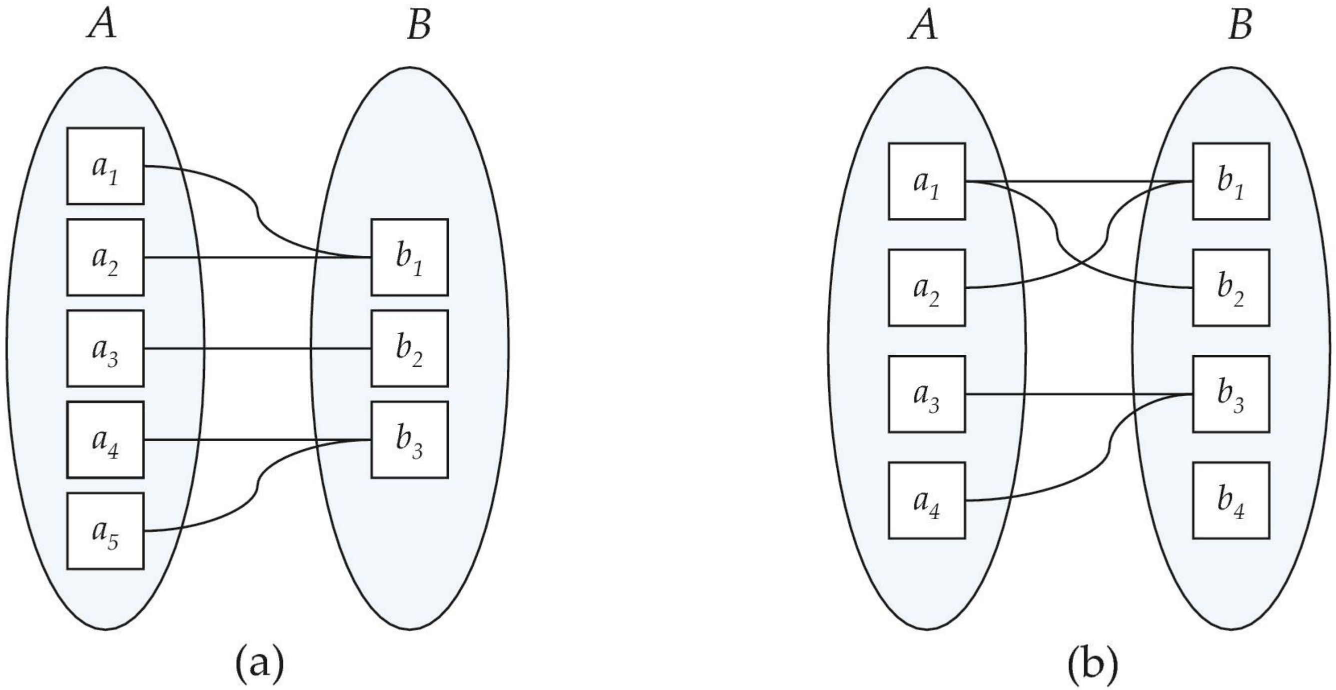
One to one



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

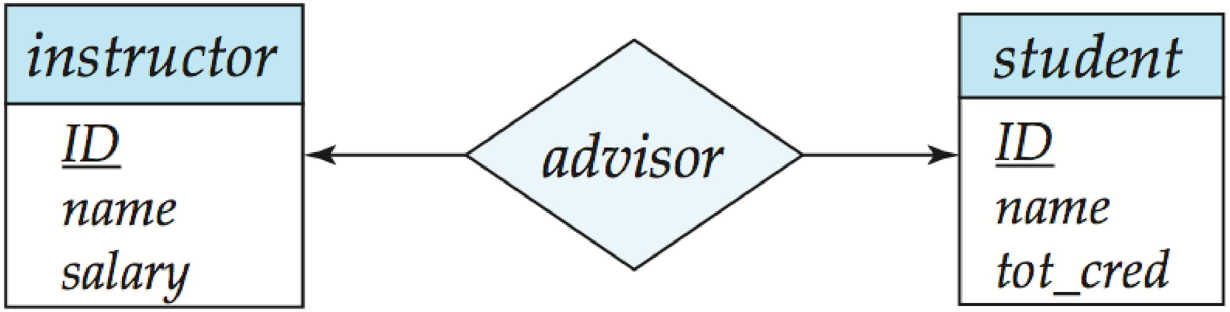
Many to many

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

Cardinality Constraints

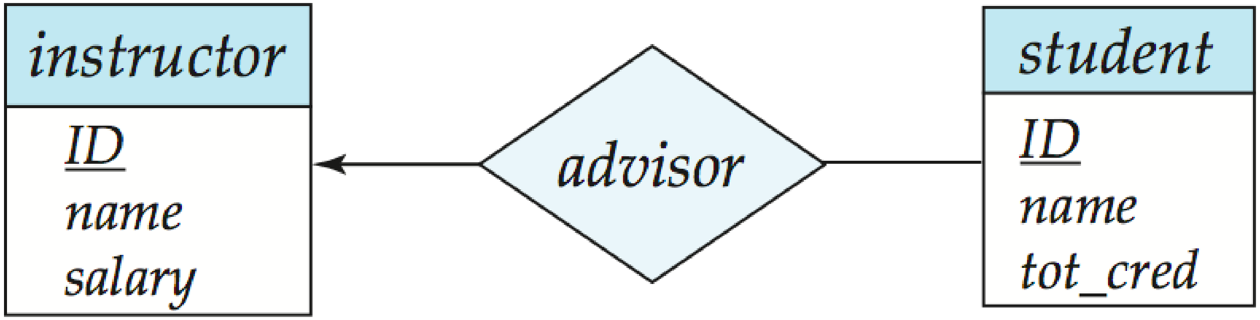
- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.

- 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*



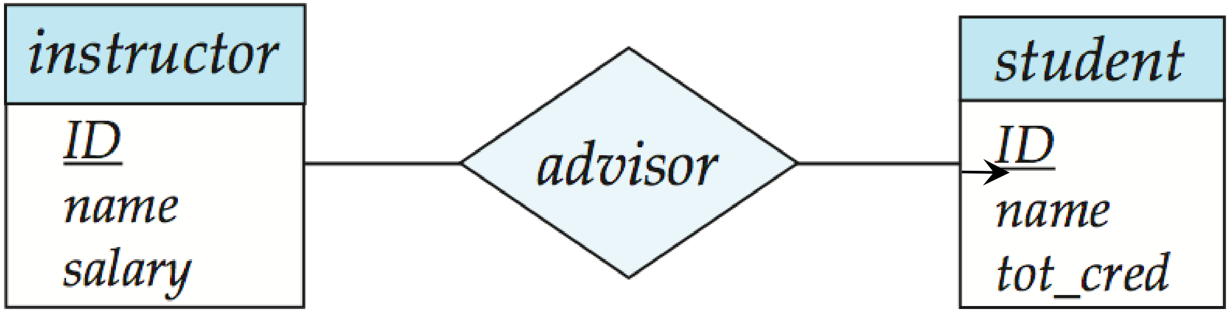
One-to-Many Relationship

- 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,



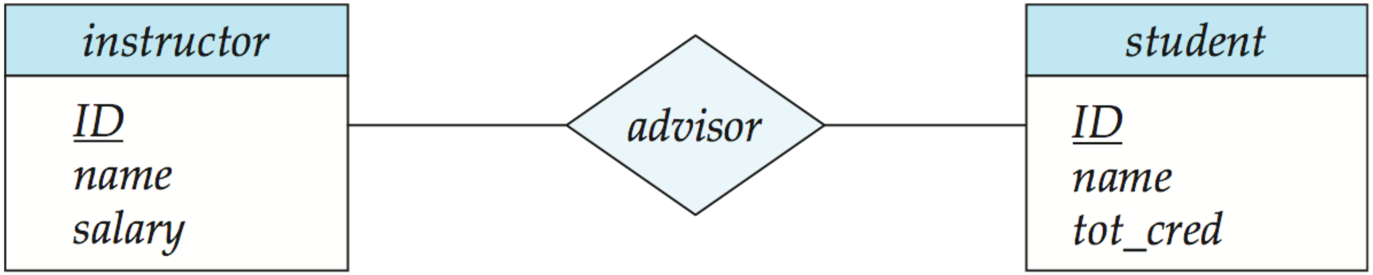
Many-to-One Relationships

- 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*



PURDUE
UNIVERSITY

CS34800
Information Systems

Entity-Relationship Model

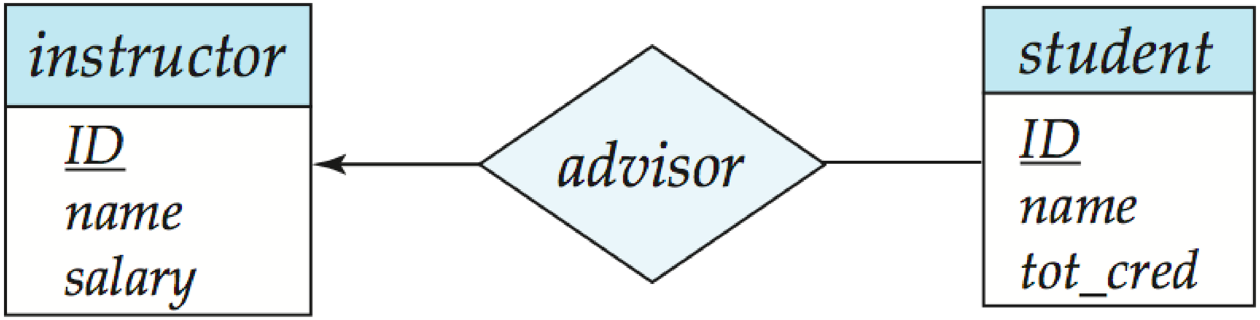
Prof. Chris Clifton

19 September 2016

Indiana
Center for
Database
Systems
™

ER Diagram: Review

- Conceptual Database Design Model
  - Entity Sets
  - Relationships



- In this diagram:
  A. *advisor* is an entity set
  B. *advisor* is a relationship
  C. each *advisor* works with at most one *instructor*
  D. each *instructor* can advise at most one *student*

Weak Entity Sets

- Consider a *section* entity, which is uniquely identified by a *course_id*, *semester, year*, and *sec_id*.
- 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.
- One option to deal with this redundancy is to get rid of the relationship *sec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

Weak Entity Sets (Cont.)

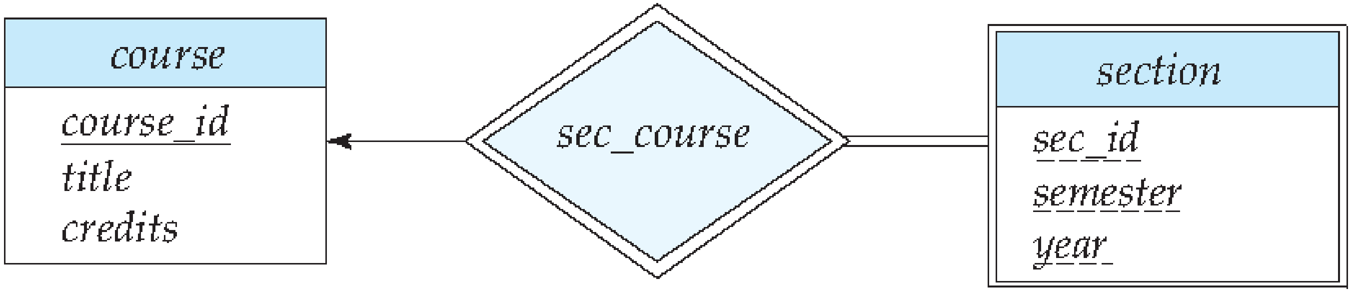
- An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes *section_id*, *year*, and *semester.* However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely; although each *section* entity is distinct, sections for different courses may share the same *section_id*, *year*, and *semester*.
- To deal with this problem, we treat the relationship *sec_course* as a special relationship that provides extra information, in this case, the *course_id*, required to identify *section* entities uniquely.
- The notion of **weak entity set** formalizes the above intuition. A weak entity set is one whose existence is dependent on another entity, called its **identifying entity**; instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called **discriminator** to uniquely identify a weak entity. An entity set that is not a weak entity set is termed a **strong entity set.**

Weak Entity Sets (Cont.)

- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set. The identifying entity set is said to **own** the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.
- Note that the relational schema we eventually create from the entity set *section* does have the attribute *course_id*, for reasons that will become clear later, even though we have dropped the attribute *course_id* from the entity set *section.*

Expressing Weak Entity Sets

- In E-R diagrams, a weak entity set is depicted via a double rectangle.
- We underline the discriminator of a weak entity set with a dashed line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond.
- Primary key for *section* – (*course_id, sec_id, semester, year*)



Don't Overuse Weak E.S.

- There is a tendency to feel that no E.S. has its entities uniquely determined without following some relationships.
- However, in practice, we almost always create unique ID's to compensate: student ID numbers, VIN's, etc.
- The only times weak E.S.'s seem necessary are when:
    a) We can't easily create such ID's; e.g., no one is going to accept a “species ID” as part of the standard nomenclature (species is a weak E.S. supported by membership in a genus).
    b) There is no global authority to create them, *e.g.*, assignments in a class.

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

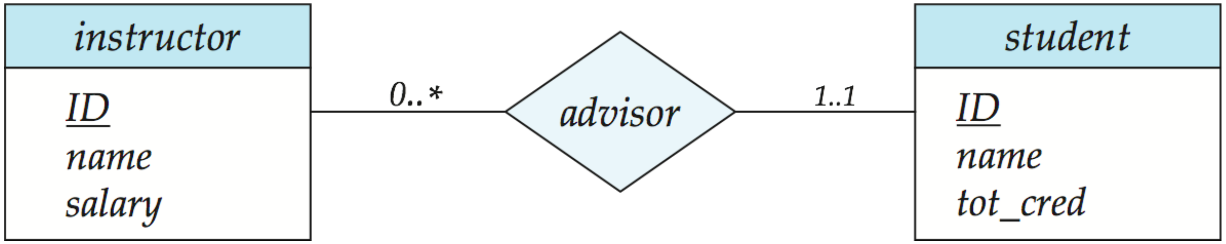
participation of *student* in *advisor* relation is total

  - every *student* must have an associated instructor

- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial

Notation for Expressing More Complex Constraints

- 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
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit.



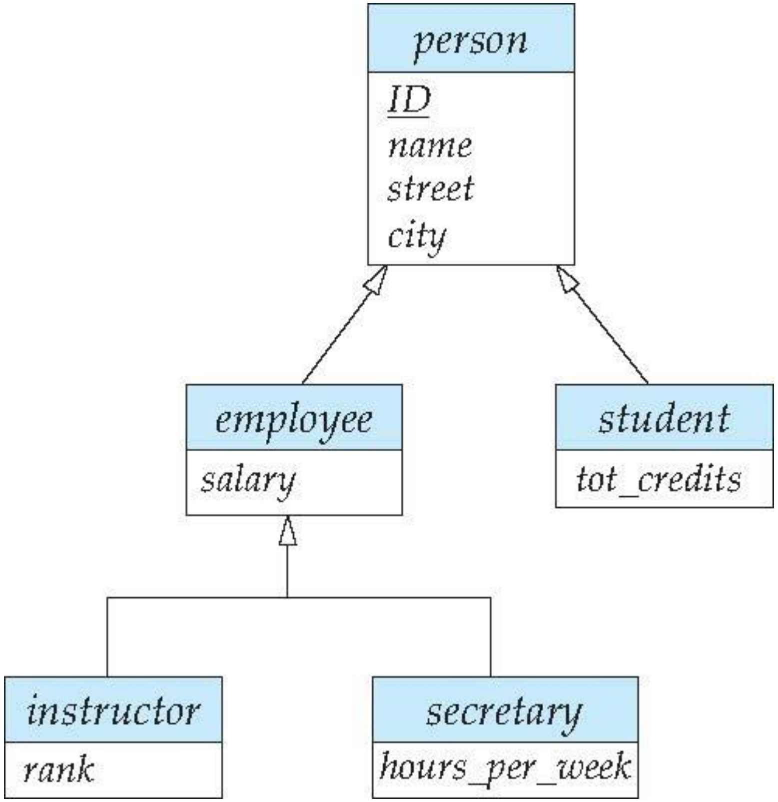
Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

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.

Specialization Example

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



Representing Specialization via Schemas

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

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

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

Representing Specialization as Schemas (Cont.)

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

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

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

Generalization

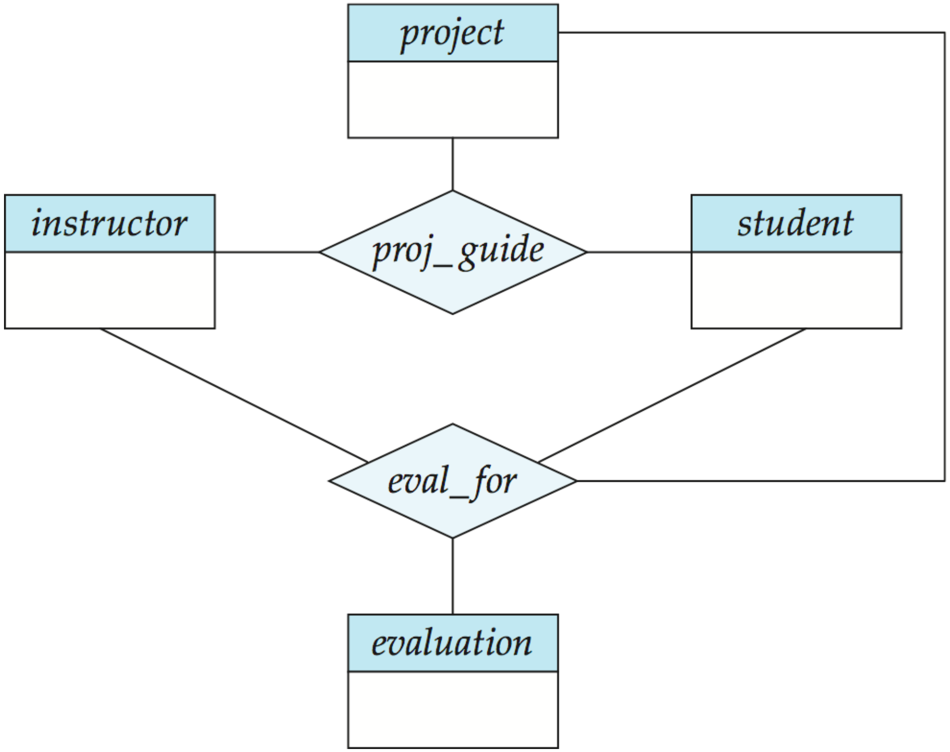
- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization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

Design Constraints on a Specialization/Generalization

- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total**: an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong to one of the lower-level entity sets
- Partial generalization is the default. We can specify total generalization in an ER diagram by adding the keyword **total** in the diagram and drawing a dashed line from the keyword to the corresponding hollow arrow-head to which it applies (for a total generalization), or to the set of hollow arrow-heads to which it applies (for an overlapping generalization).
- The *student* generalization is total: All student entities must be either graduate or undergraduate. Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets, the completeness constraint for a generalized higher-level entity set is usually total

Aggregation

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

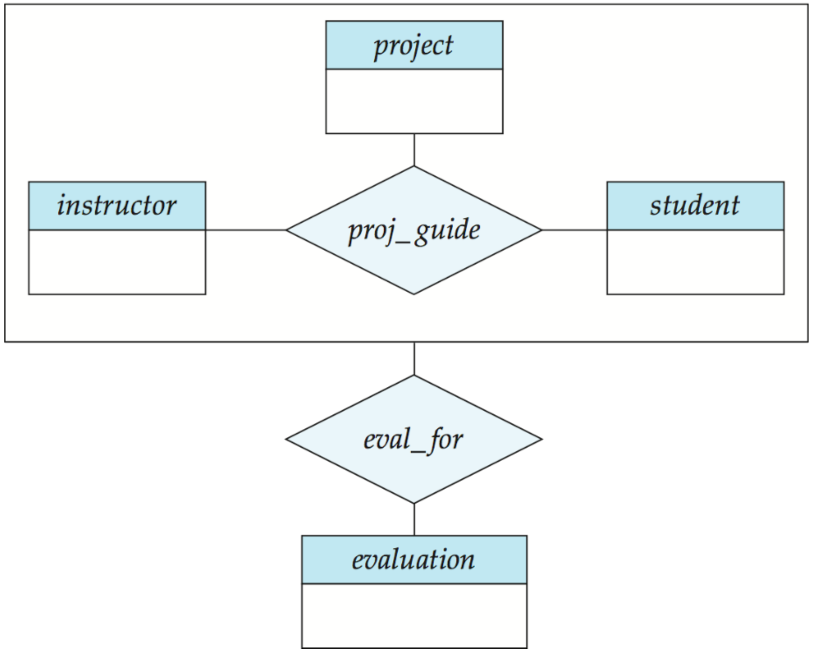


Aggregation (Cont.)

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

Aggregation (Cont.)

- Eliminate this redundancy via *aggregation* without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation



Representing Aggregation via Schemas

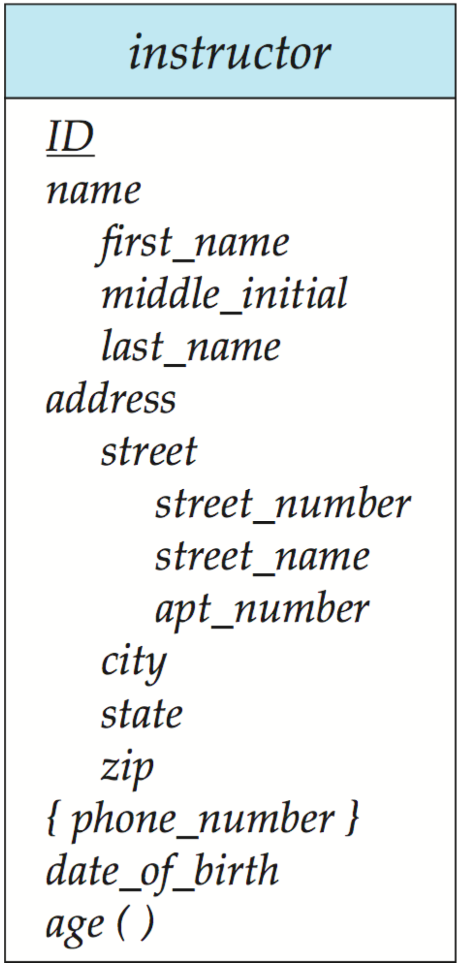
- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- In our example:
  - The schema *eval_for* is:

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

Complex Attributes

- 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
- **Domain** – the set of permitted values for each attribute

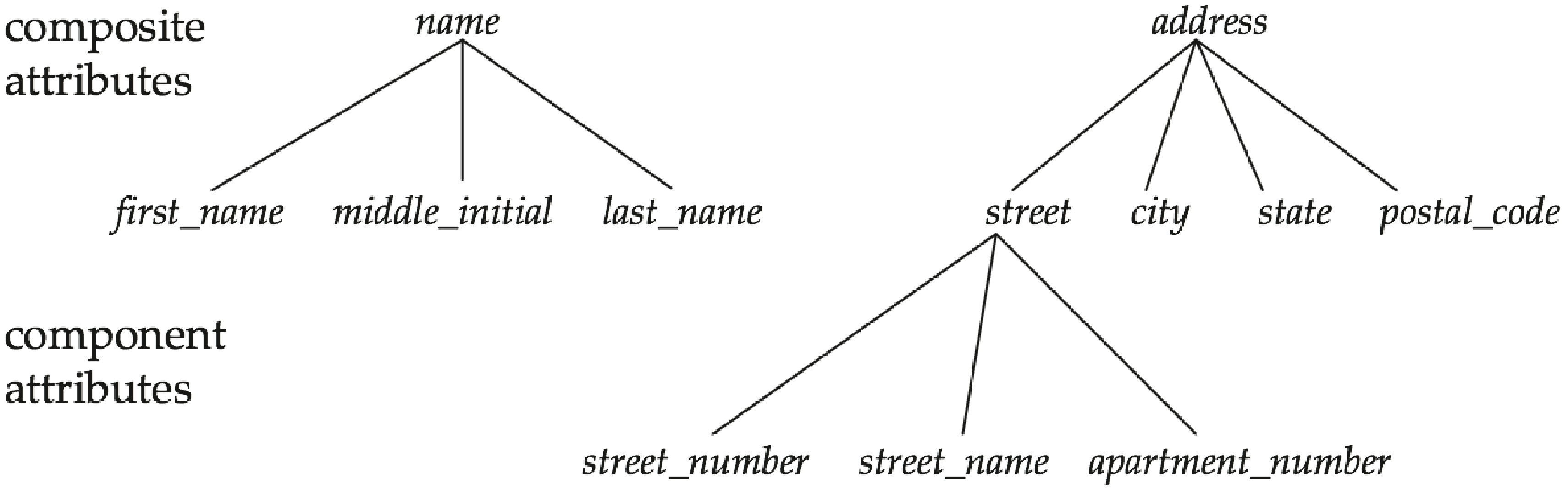
Notation to Express Entity with Complex Attributes



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

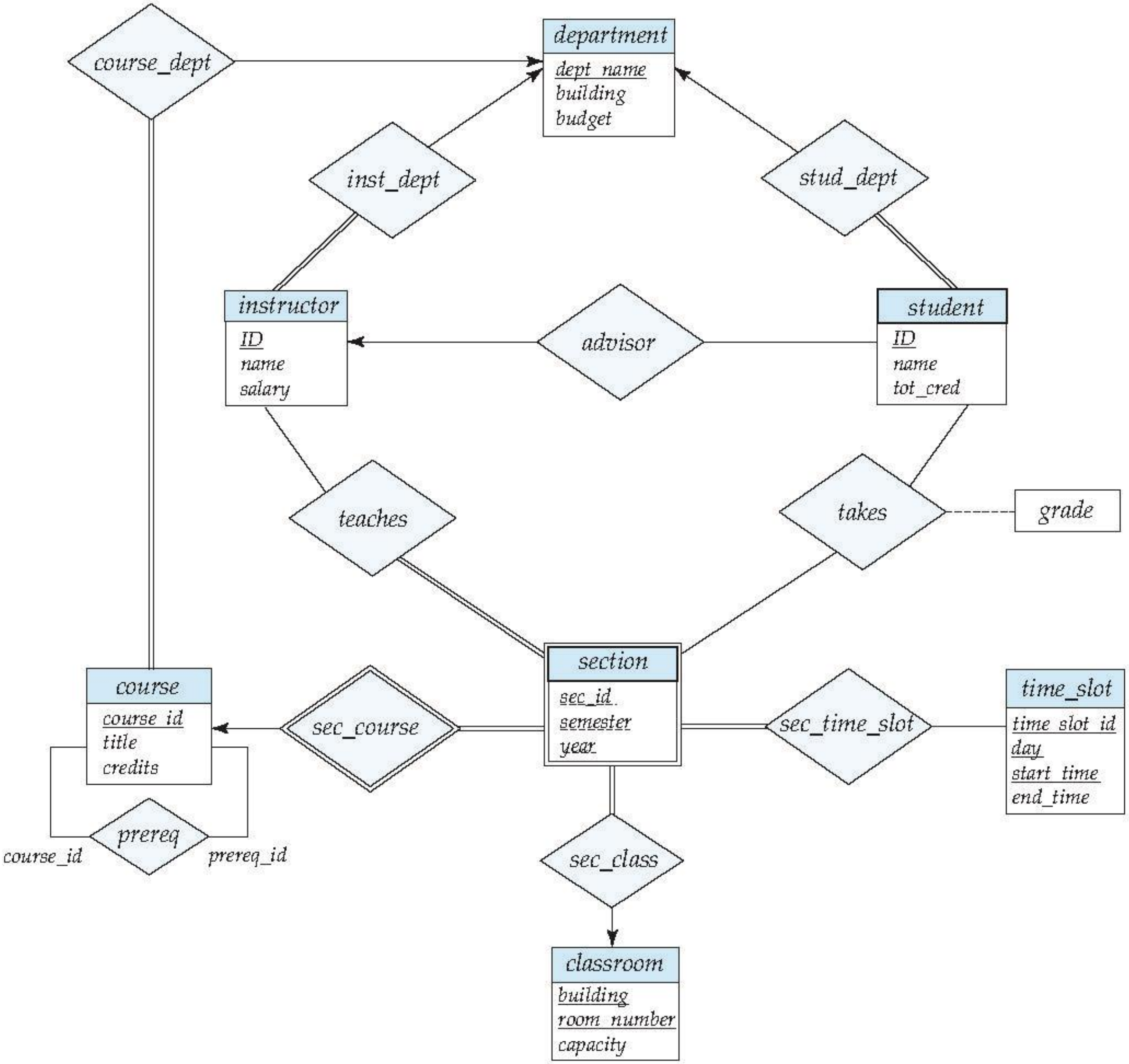


- How else could we represent composite attributes?
  - A. As ordinary attributes
  - B. Entity set
  - C. Weak entity set
  - D. All of these will work
  - E. None of these will work

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 it is the primary key for the entity set *department*, it replicates information present in the relationship and is therefore redundant in the entity set *instructor* and needs to be removed.
- If we remove dept_name from Instructor, becomes
  A. Instructor becomes a weak entity set
  B. Department becomes a weak entity set
  C. Neither is a weak entity set
  D. Need to know primary key for instructor

E-R Diagram for a University Enterprise



PURDUE
UNIVERSITY

CS34800
Information Systems

Entity-Relationship Model

Prof. Chris Clifton

21 September 2016

Indiana
Center for
Database
Systems
™

Exam 9/26 In Class

- You are allowed one 8.5x11 or A4 note sheet
  - You can use both sides
- No electronic aids
  - *If we see a device with a camera, your exam is forfeit*
- Do not discuss the exam with anyone until solutions are posted!
  - We have a couple of students with off-site job interviews Monday who will be taking the exam when they return

Project/Assignment updates

- Assignment 2:
  - Correction to question B(c)
  - Solutions will be posted shortly after the deadline
    No late work accepted
- I will hold office hours Saturday 11:30-4:30 in LWSN 2142F
  - Please look at the Assignment 2 solutions before coming in
- Project 1:
  - Solutions to be posted Thursday afternoon
  - No late work accepted after solutions posted
    You'll already have lost 50% from the late penalty
- Solutions are not necessarily the only correct solution!

Classroom Design Exercise

Imagine we are creating a database for a dorm, which includes a cooperative kitchen.

- We want to record certain information about each resident. What?
- Not all residents belong to the kitchen coop. Those that do interact in various ways:
  1. They take turns at various jobs: preparer, cleanup, buyer (for supplies). No one should have two jobs on one day.
  2. They may or may not be vegetarian. Each meal must have at least one vegetarian entreé.
  3. They pay fees to the coop.
- For each meal, there is a menu. Each menu item requires certain ingredients, which must be on hand.

If There's Time…

Suppose we only need to have a vegetarian choice for a given meal if there is at least one vegetarian taking that meal. How would we modify the database schema?

CS34800
Information Systems

Entity-Relationship to Relational

Prof. Chris Clifton

23 September 2016

Relational Database Design

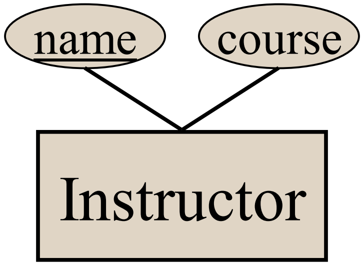
- Goals
  - – Capture all the data
    - Nothing lost
  - – Represent only the data
    - Prevent data not matching the real world
- How?
  - – Division of data into relations
  - – Key constraints

Relational Design

Simplest approach (not always best): convert each E.S. to a relation and each relationship to a relation.

Entity Set → Relation

E.S. attributes become relational attributes.



Becomes:

`Instructor(name, course)`

Keys in Relations

An attribute or set of attributes *K* is a *key* for a relation *R* if we expect that in no instance of *R* will two different tuples agree on all the attributes of *K*.

- Indicate a key by underlining the key attributes.
- Example: If `name` is a key for `Instructor`:

`Instructor(`<u>`name`</u>`, course)`

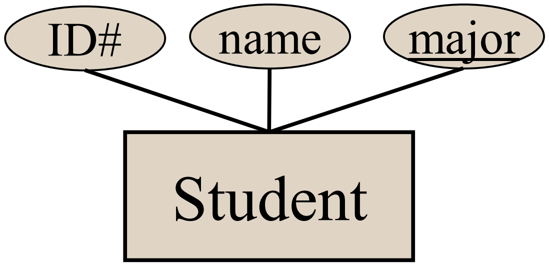
In Database Design, a Key:

A. Is needed in order to access an entity
B. Is the only way to select on an attribute
C. Uniquely identifies each entity in an entity set
D. Consists of a single attribute
E. Is used to authenticate to the database

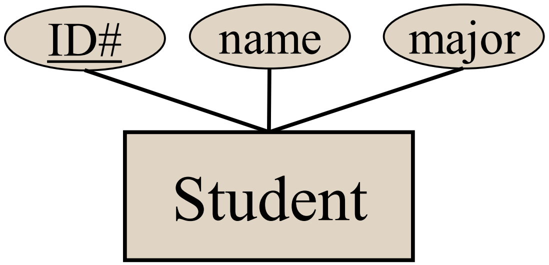
Which are valid key assignments?

- A, B, or C
- D: Two or more are correct
- E: None are correct

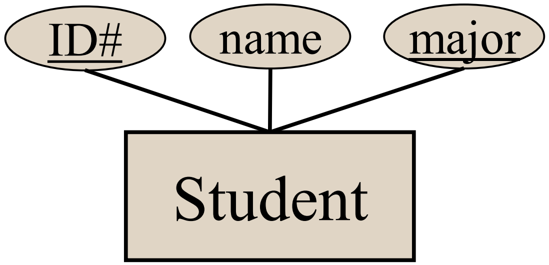
A



B



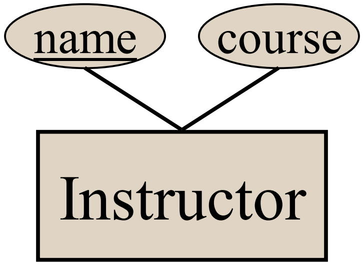
C



Types of keys:

- Candidate key
  - Any valid key (guaranteed to be unique)
- Primary key
  - A candidate key chosen as the one to use
- Superkey
  - Key that can have attributes removed and still be a key

Relational Design



Becomes:

Instructor(name, course)

SQL:

```
CREATE TABLE Instructor (
    name varchar2(40),PRIMARY KEY,
    course varchar2(9)
);
```

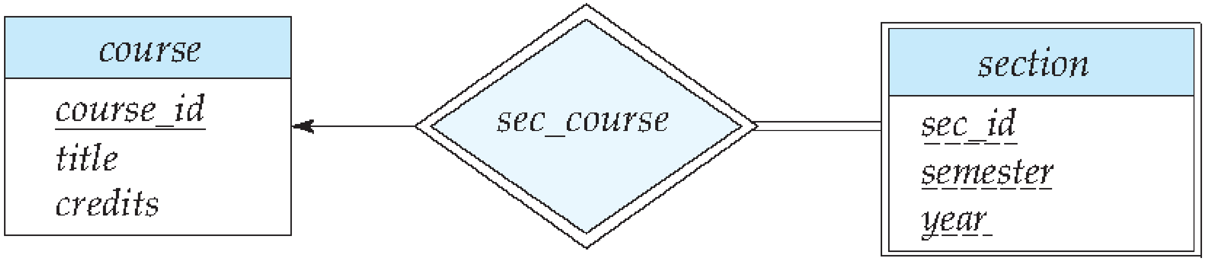
Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes

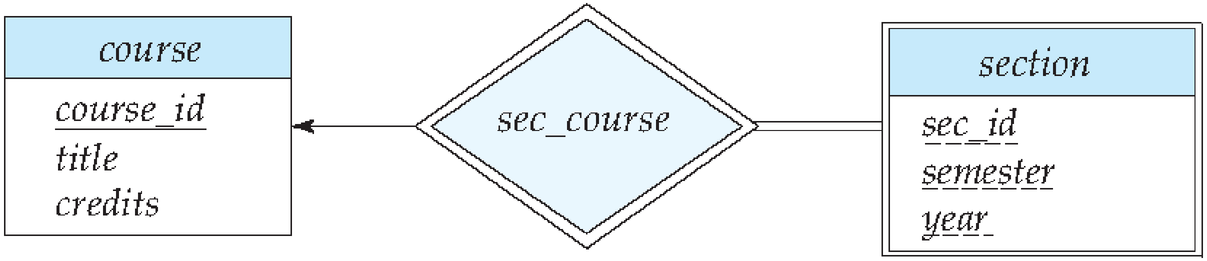
  student(<u>ID</u>, name, tot_cred)

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

  section (<u>course_id, sec_id, sem, year</u>)



Weak Entity Sets in SQL



section (course_id, sec_id, sem, year)

```
CREATE TABLE section (
  course_id varchar2(9),
  sec_id number(5),
  semester char(1),
  year number(4),
  PRIMARY KEY (course_id, sec_id, semester, year),
  FOREIGN KEY (course_id) REFERENCES course(course_id)
);
```

E/R Relationships → Relations

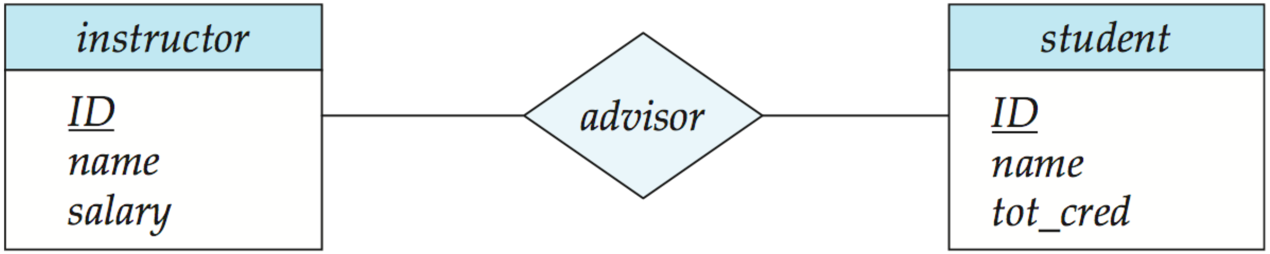
Relation has attribute for *key* attributes of each E.S. that participates in the relationship.

- Add any attributes that belong to the relationship itself.
- Renaming attributes OK.
  - Essential if multiple roles for an E.S.

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.
- Example: schema for relationship set *advisor*

advisor = (<u>*s_id, i_id*</u>)

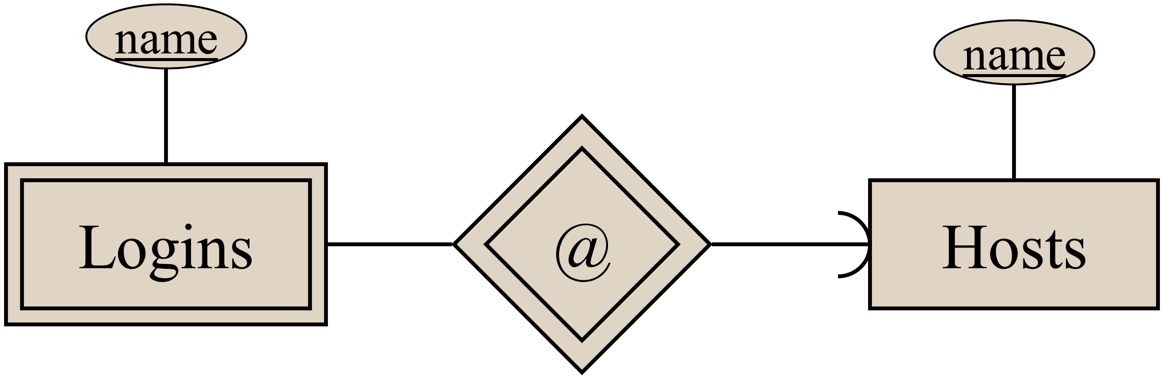


```
CREATE TABLE advisor(
  S_ID number(10), REFERENCES student(ID),
  I_ID number(10), REFERENCES instructor(ID),
  PRIMARY KEY (S_ID, I_ID),
);FOREIGN KEY S_ID REFERENCES student(ID),
  FOREIGN KEY I_ID REFERENCES instructor(ID)
);
```

Weak Entity Sets, Relationships → Relations

- Relation for a weak E.S. must include its full key (*i.e*., attributes of related entity sets) as well as its own attributes.
- A supporting (double-diamond) relationship yields a relation that is actually redundant and should be deleted from the database schema.

Example



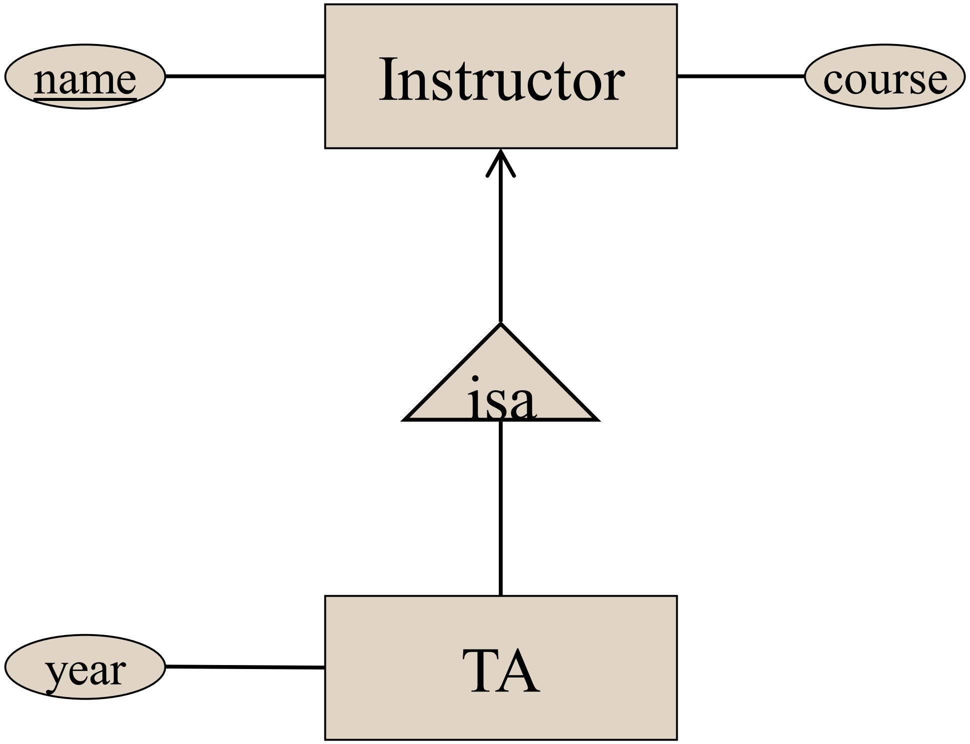
`Hosts(hostName)`

`Logins(loginName, hostName)`

`At(loginName, hostName, hostName2)`

- In `At`, `hostName` and `hostName2` must be the same host, so delete one of them.
- Then, `Logins` and `At` become the same relation; delete one of them.
- In this case, `Hosts`' schema is a subset of `Logins`' schema. Delete `Hosts`?

Generalization / Specialization



Representing Inheritance: Which is correct?

A. OO style

Name	**course**
Clifton	CS34800

Name	**Course**	**year**
Pradhan	CS34800	4

B. ER style

Name	**course**
Clifton	CS34800
Pradhan	CS34800

Name	**year**
Pradhan	4

C. Using nulls

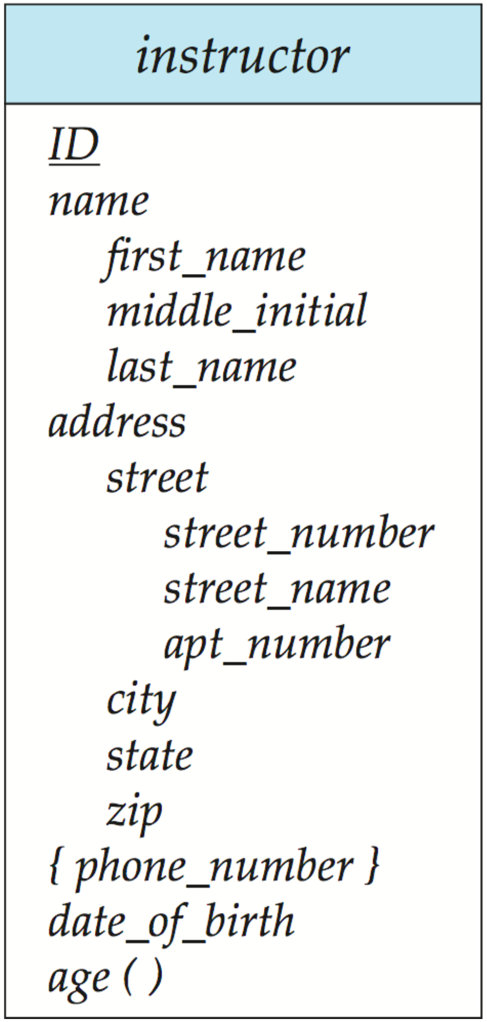
Name	**Course**	**year**
Clifton	CS34800	⊥
Pradhan	CS34800	4

Subclasses → Relations

Three approaches:

1. Object-oriented: each entity is in one class. Create a relation for each class, with all the attributes for that class.
   - Don’t forget inherited attributes.
2. E/R style: an entity is in a network of classes related by *isa*. Create one relation for each E.S.
   - An entity is represented in the relation for each subclass to which it belongs.
   - Relation has only the attributes attached to that E.S. + key.
3. Use nulls. Create one relation for the root class or root E.S., with all attributes found anywhere in its network of subclasses.
   - Put `NULL` in attributes not relevant to a given entity.

Representation of Entity Sets with Composite 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 ( )*

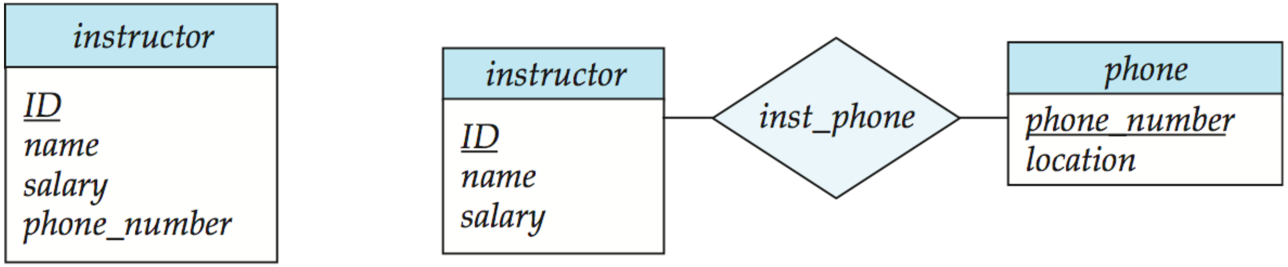
- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* 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)

Representation of Entity Sets with 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= (<u>*ID*</u>, <u>*phone_number*</u>)
- 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)

Entities vs. Attributes

- Use of entity sets vs. attributes

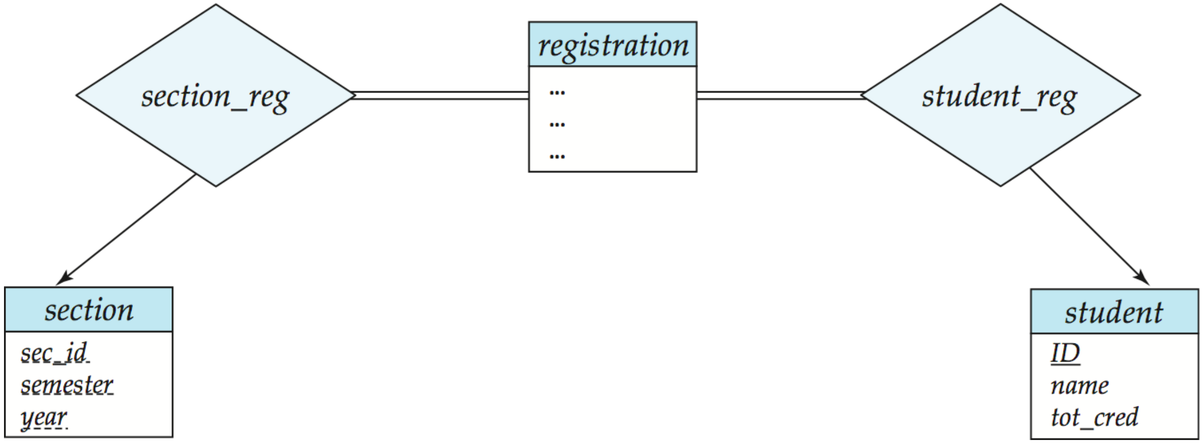


- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Entities vs. Relationship sets

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

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



- **Placement of relationship attributes**

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

Binary Vs. Non-Binary Relationships

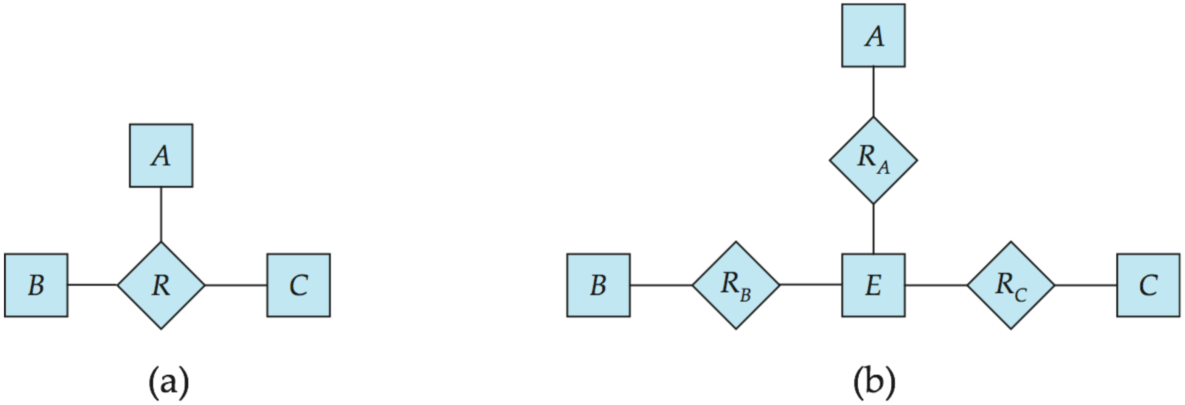
- Although it is possible to replace any non-binary (*n*-ary, for *n* > 2) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- Some relationships that appear to be non-binary may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

Converting Non-Binary Relationships to Binary Form

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

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

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



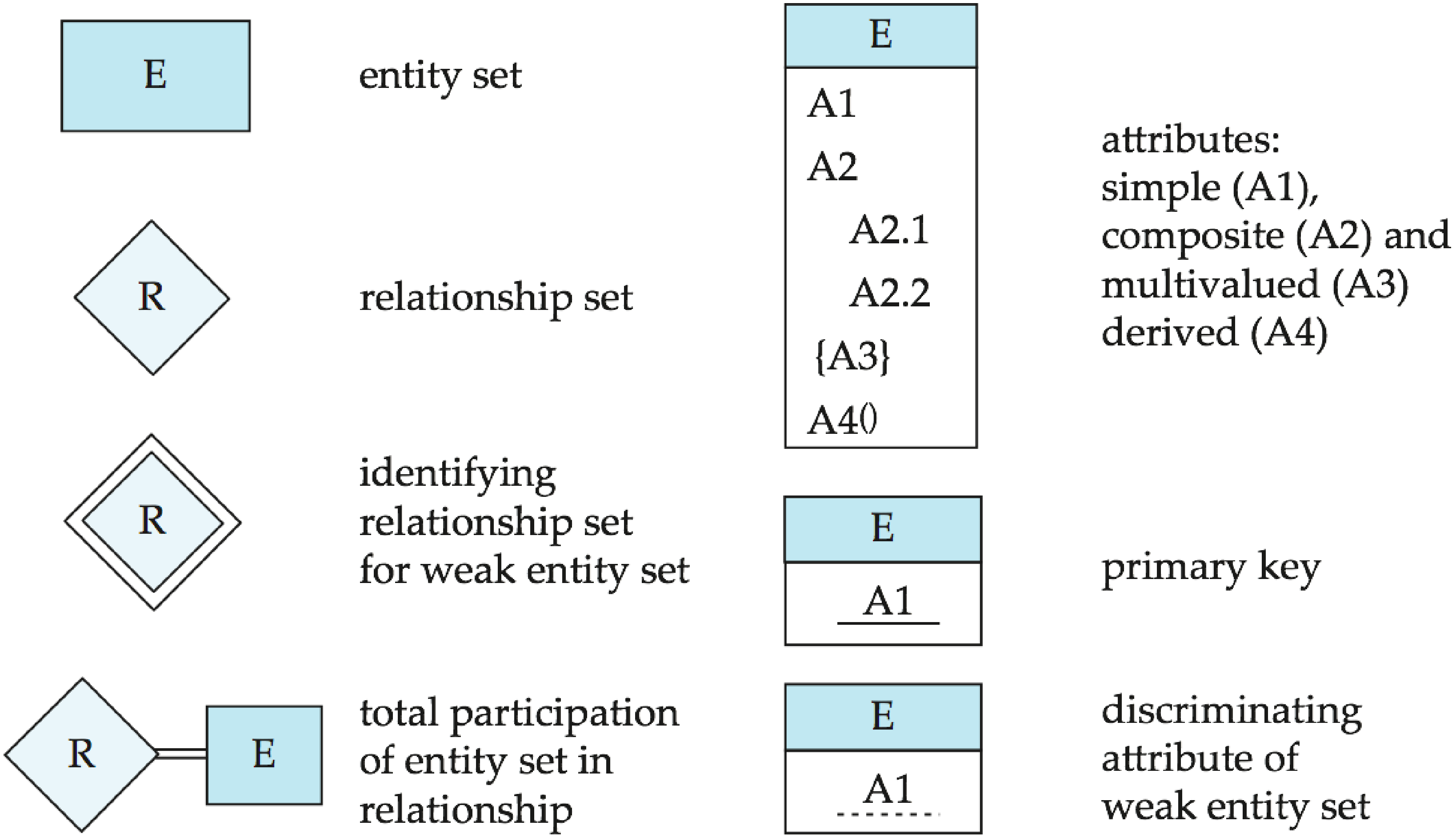
Converting Non-Binary Relationships (Cont.)

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

E-R Design Decisions

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

Summary of Symbols Used in E-R Notation



entity set

relationship set

identifying
relationship set
for weak entity set

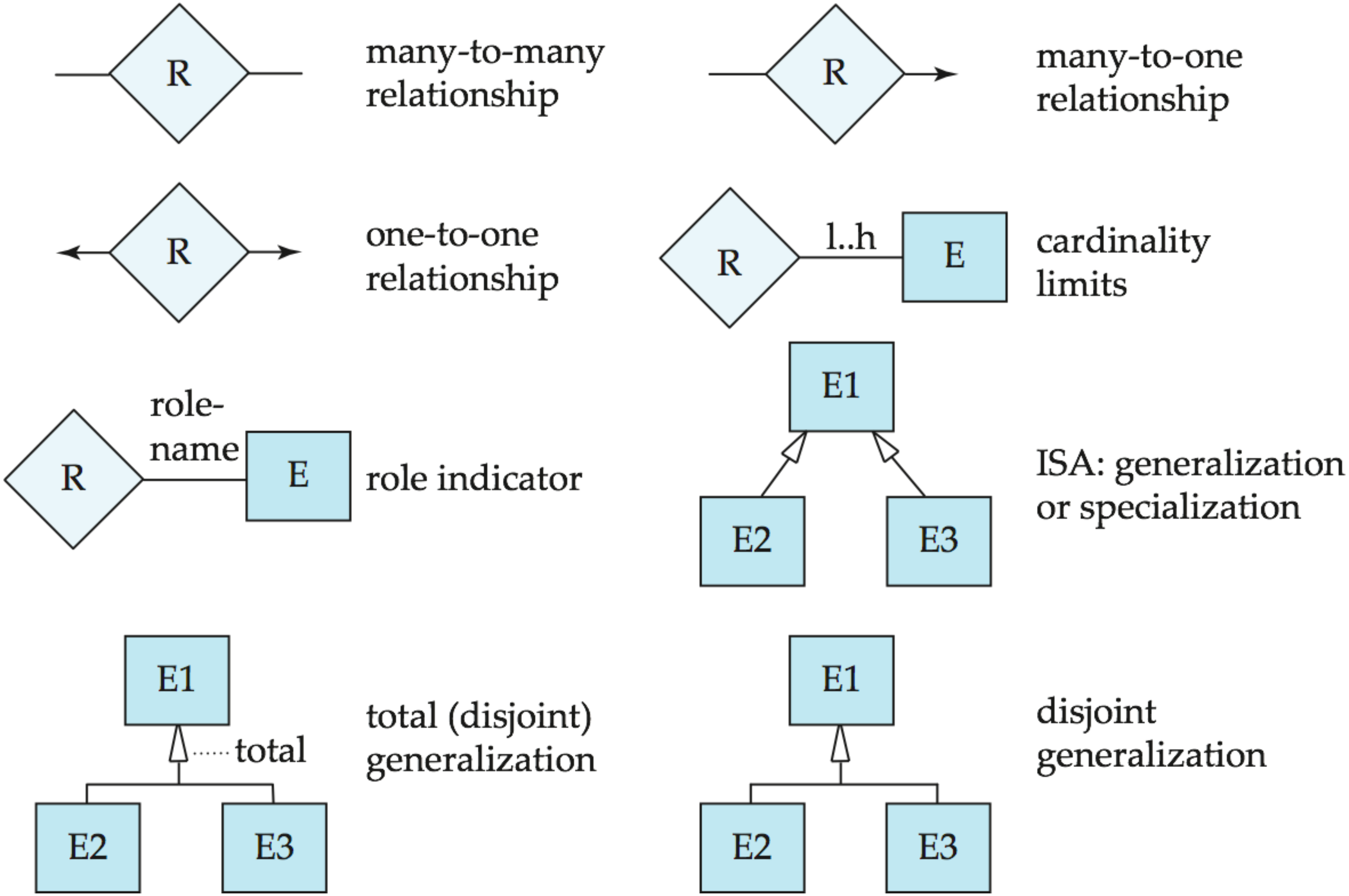
total participation
of entity set in
relationship

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

primary key

discriminating
attribute of
weak entity set

Symbols Used in E-R Notation (Cont.)



many-to-many relationship

many-to-one relationship

one-to-one relationship

cardinality limits

role indicator

ISA: generalization or specialization

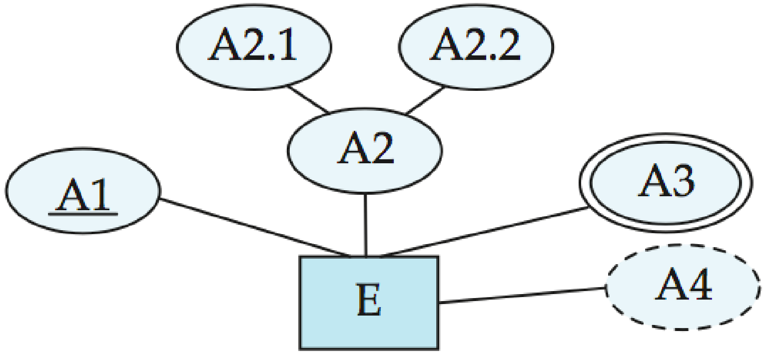
total (disjoint) generalization

disjoint generalization

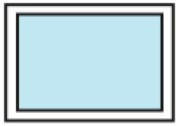
Alternative ER Notations

- Chen, IDE1FX, …

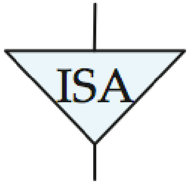
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1



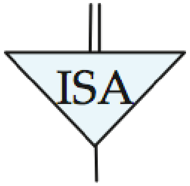
weak entity set



generalization

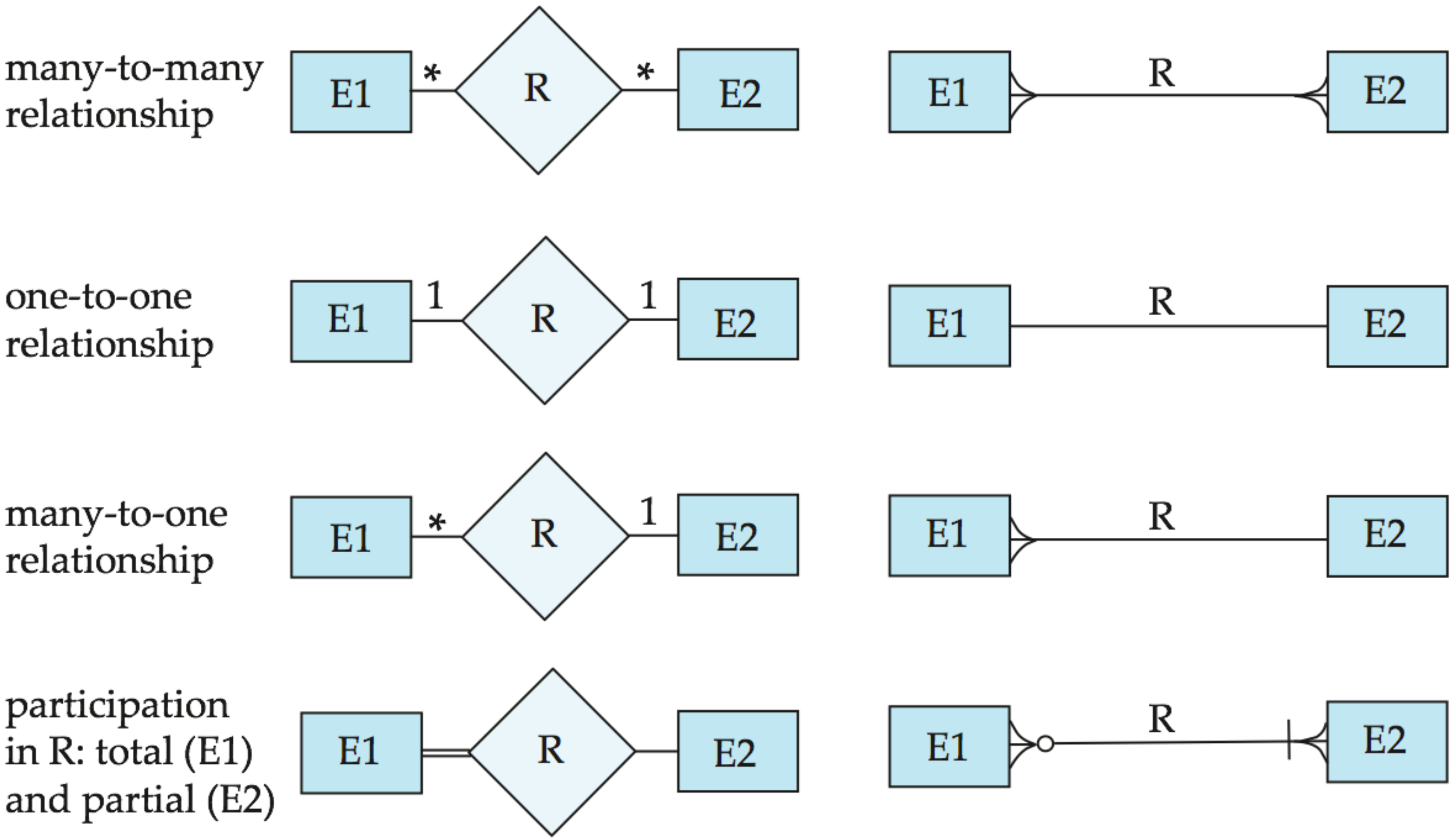


total
generalization



Alternative ER Notations

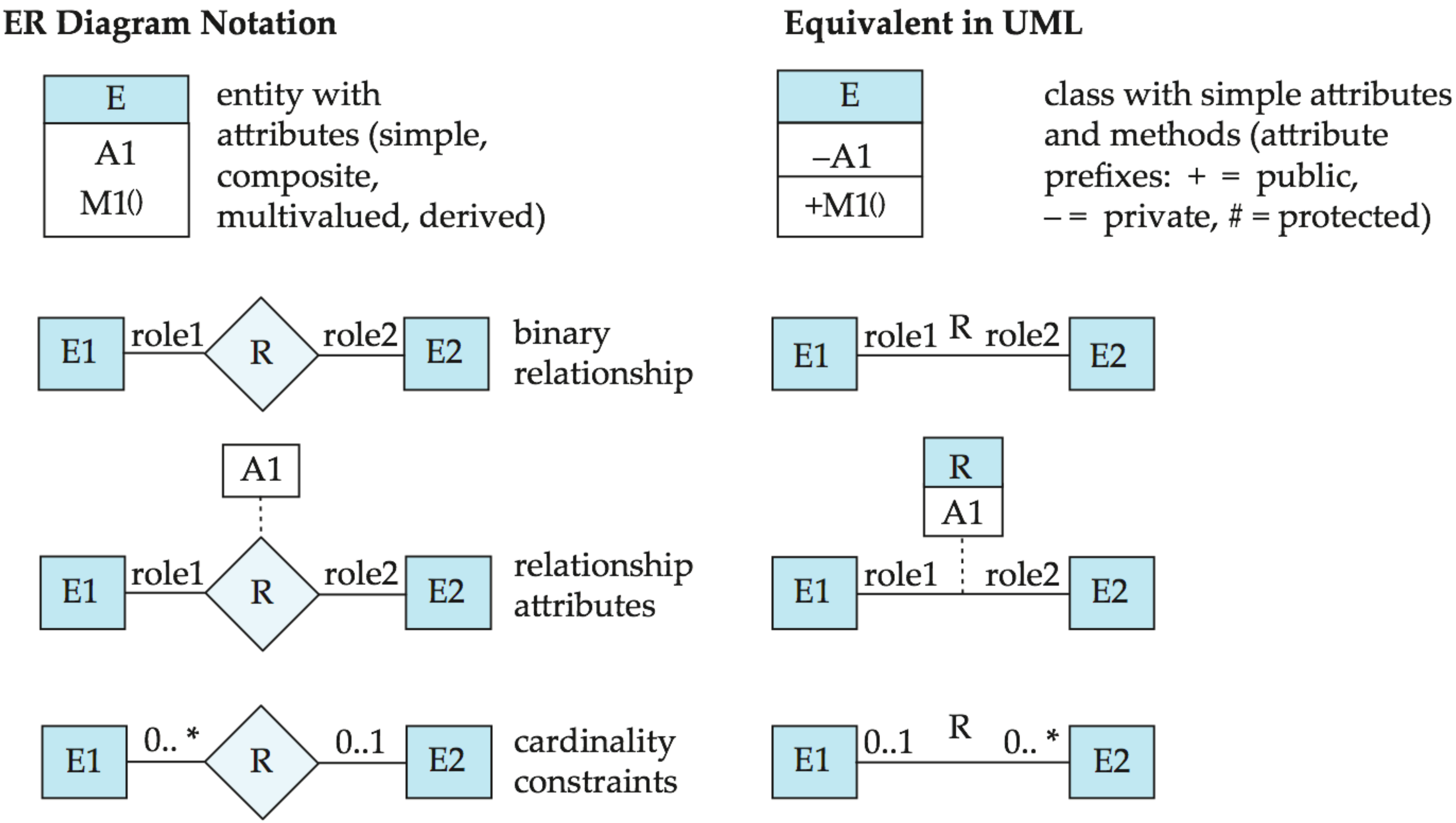
Chen **IDE1FX (Crows feet notation)**



UML

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

ER vs. UML Class Diagrams



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

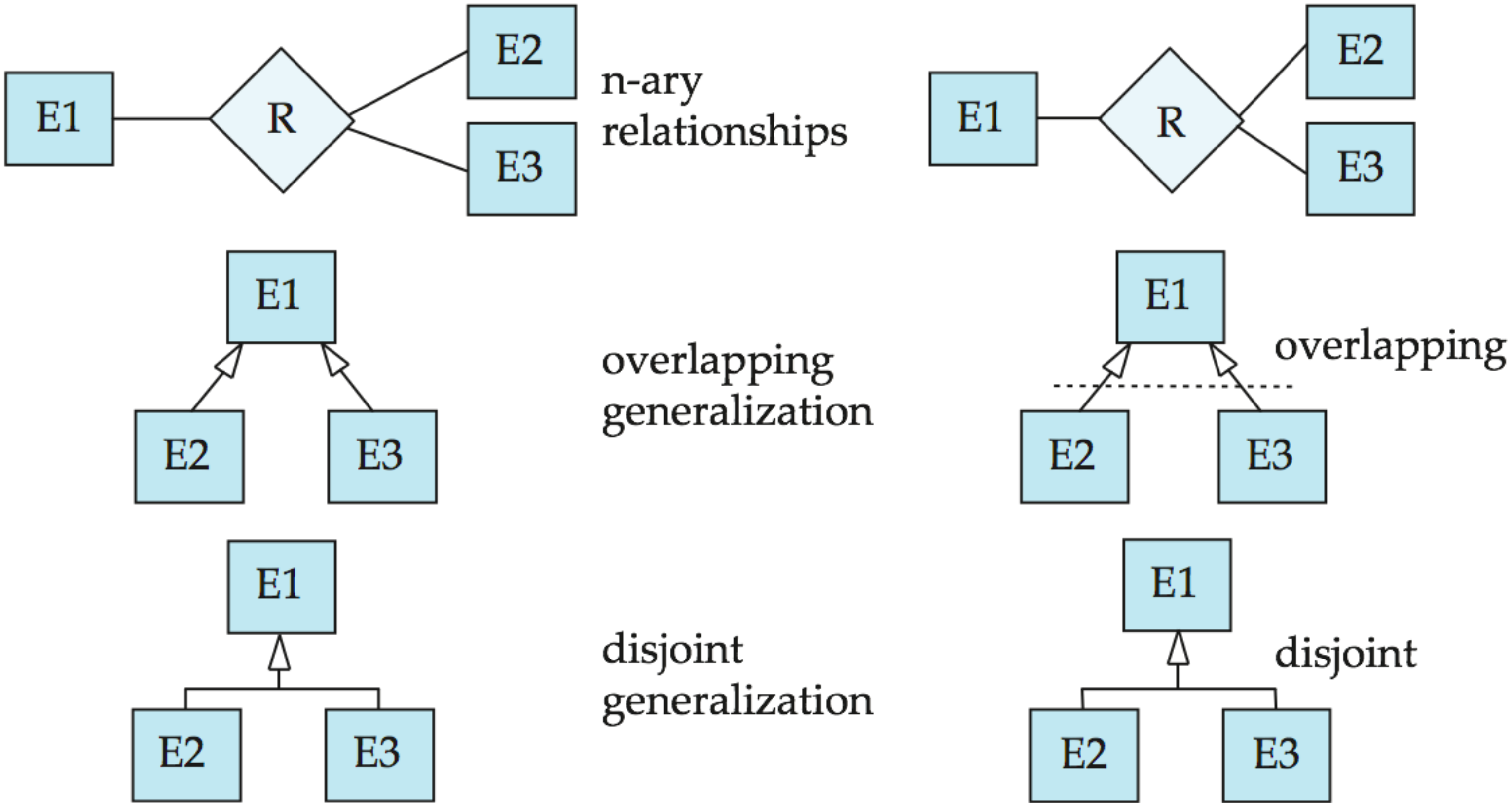
relationship attributes

cardinality constraints

*Note reversal of position in cardinality constraint depiction

ER vs. UML Class Diagrams

ER Diagram Notation **Equivalent in UML**



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

UML Class Diagrams (Cont.)

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