Database design

Part 1

Chapter 7: Entity-Relationship Model

Chapter 7: Entity-Relationship Model

- Design Process
- Modeling
- Mapping Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Design of the Bank Database
- Reduction to Relation Schemas
- Database Design
- UML

- Design Process
- E-R Model

DESIGN PROCESS

Design Phases

- The initial phase of database design is to characterize fully the data needs of the prospective database users
- Next, the designer chooses a data model and, by applying the concepts of the chosen data model, translates these requirements into a conceptual schema of the database
- A fully developed conceptual schema also indicates the functional requirements of the enterprise. In a "specification of functional requirements", users describe the kinds of operations (or transactions) that will be performed on the data

Design Phases (Cont.)

The process of moving from an abstract data model to the implementation of the database proceeds in two final design phases.

- Logical Design – Deciding on the database schema. Database design requires that we 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

Design Approaches

- Entity Relationship Model (covered in this chapter)
  - 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:*
- Normalization Theory (Chapter 8)
  - Formalize what designs are bad, and test for them

- Design Process
- E-R Model

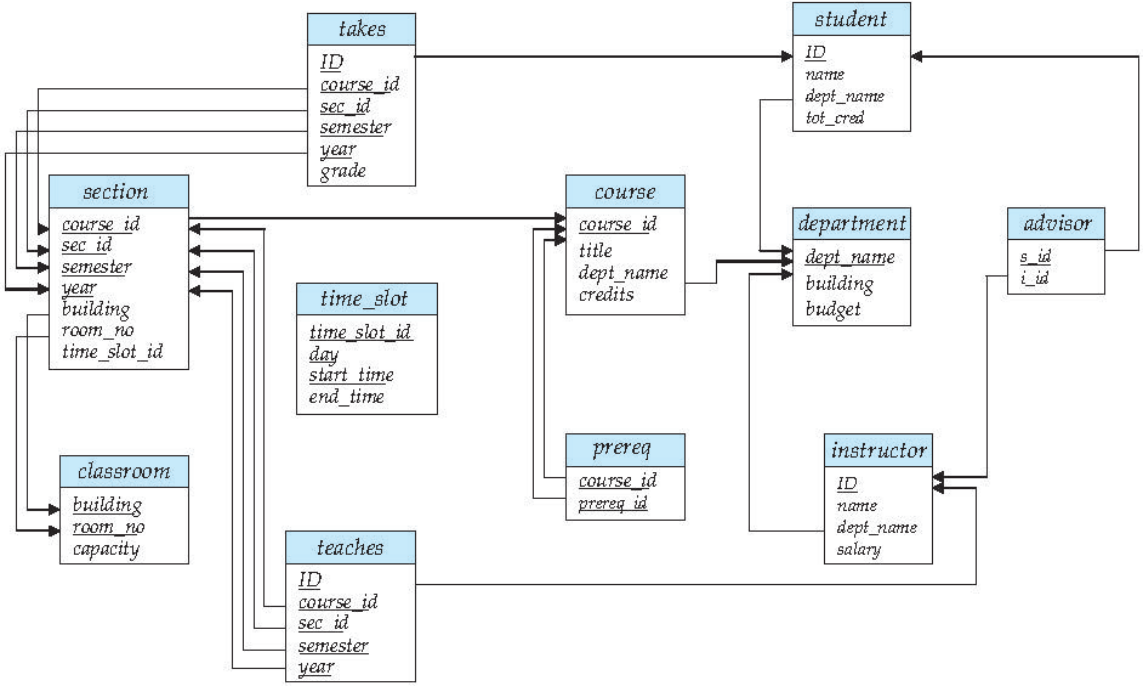
E-R MODEL

ER model – Database Modeling

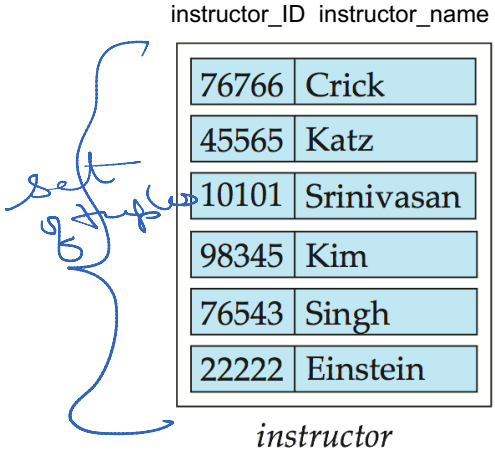
- The ER data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database
- The ER model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this usefulness, many database-design tools draw on concepts from the ER model
- The ER data model employs three basic concepts:
  - entity sets
  - relationship sets
  - attributes
- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the overall logical structure of a database graphically

Modeling

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: person, company, event, plant, Course, Department
- 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


takes
ID
course_id
sec_id
semester
year
grade
student
ID
name
dept_name
tot_cred
section
course_id
sec_id
semester
year
building
room_no
time_slot_id
course
course_id
title
dept_name
credits
department
dept_name
building
budget
advisor
s_id
i_id
time_slot
time_slot_id
day
start_time
end_time
prereq
course_id
prereq_id
instructor
ID
name
dept_name
salary
classroom
building
room_no
capacity
teaches
ID
course_id
sec_id
semester
year

Entity Sets *instructor* and *student*



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

instructor

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

student

Relationship Sets

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

  Example:

  44553 (Austin) *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

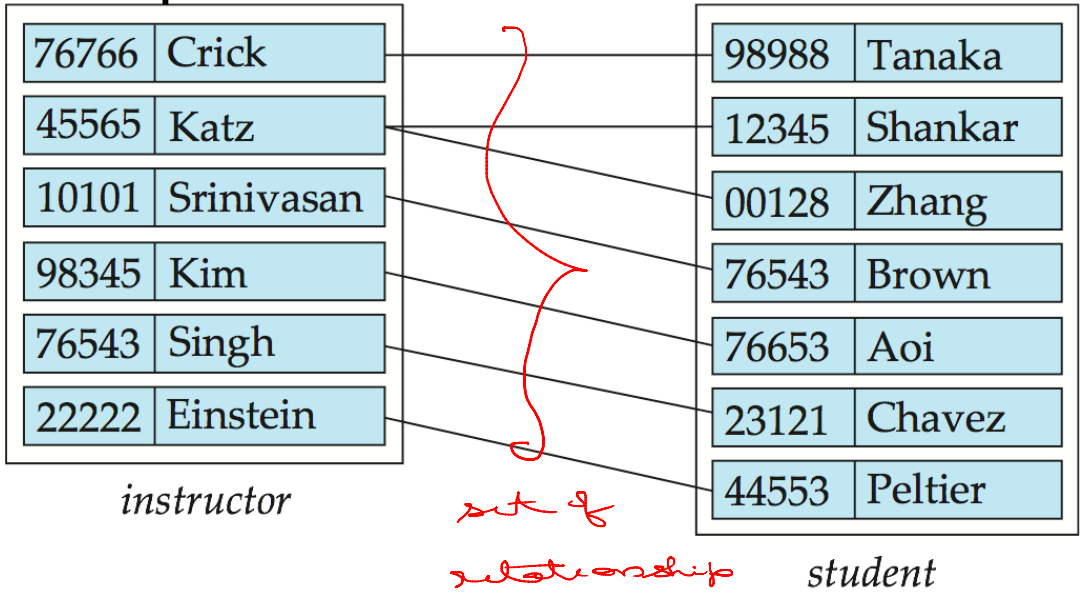
  $$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

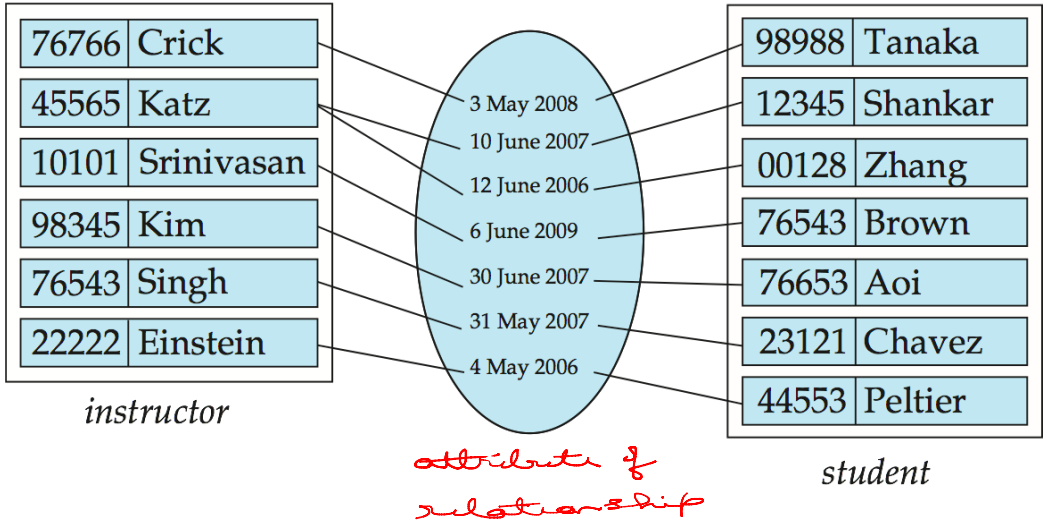
    $$(44553,22222) \in advisor$$

Relationship Set *advisor*



Relationship Sets (Cont.)

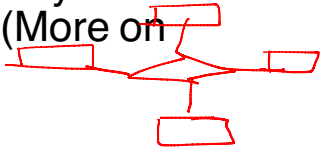
- A descriptive **attribute** can also be property of a relationship set.
- For instance, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor



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.)



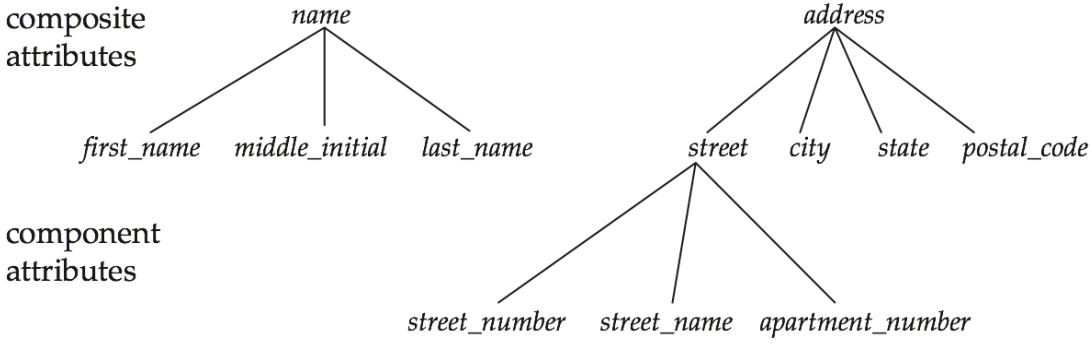
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

Attributes

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

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

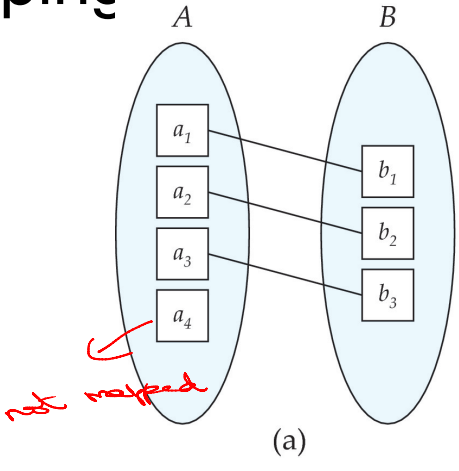
Composite Attributes



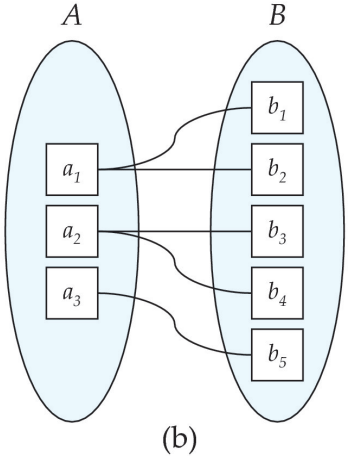
Mapping Cardinality Constraints

- 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



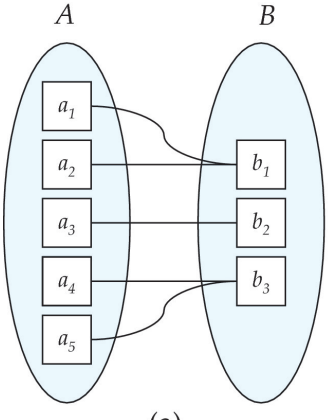
One to one



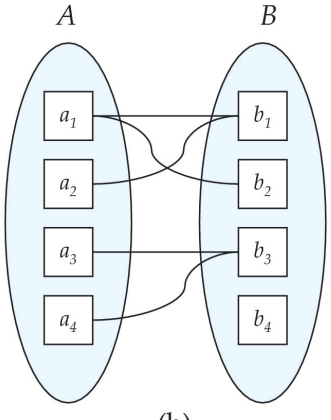
One to many

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

Keys

Key of an Entity Set

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

Redundant Attributes

- Suppose we have entity sets
  - *instructor*, with attributes including *dept_name*
  - *Department*

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

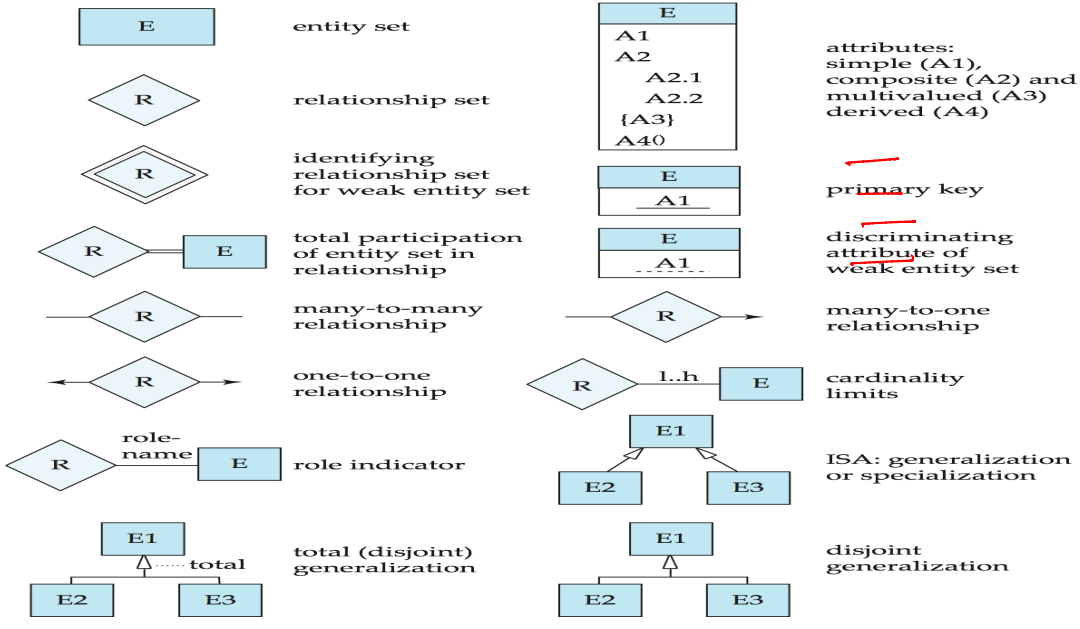
- Attribute *dept_name* in entity *instructor* is redundant since there is an explicit relationship *inst_dept* which relates instructors to departments
  - The attribute replicates information present in the relationship, and should be removed from *instructor*
  - BUT: when converting back to tables,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

ER DIAGRAM

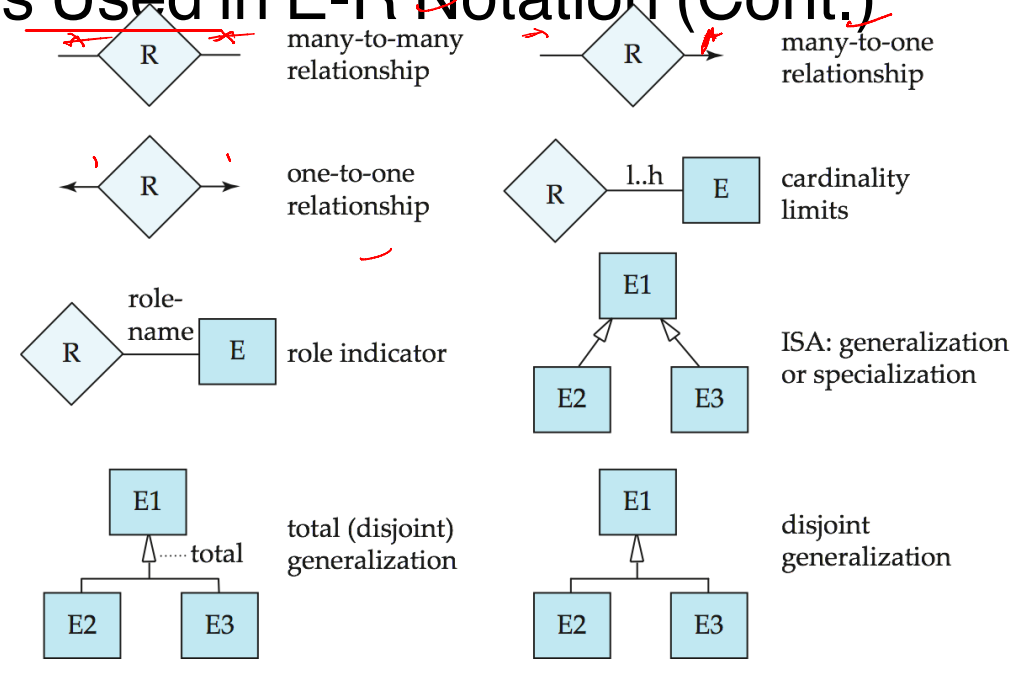
CHEN'S NOTATION

To be preferably followed

Symbols Used in E-R Notation



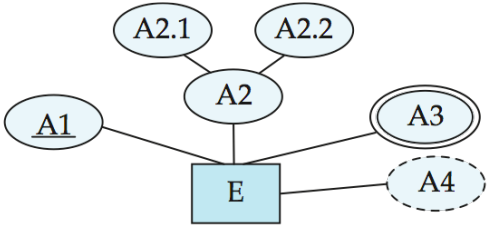
Symbols Used in E-R Notation (Cont.)

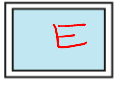


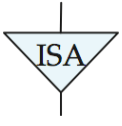
Alternative ER Notations

- Chen, IDE1FX, …

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

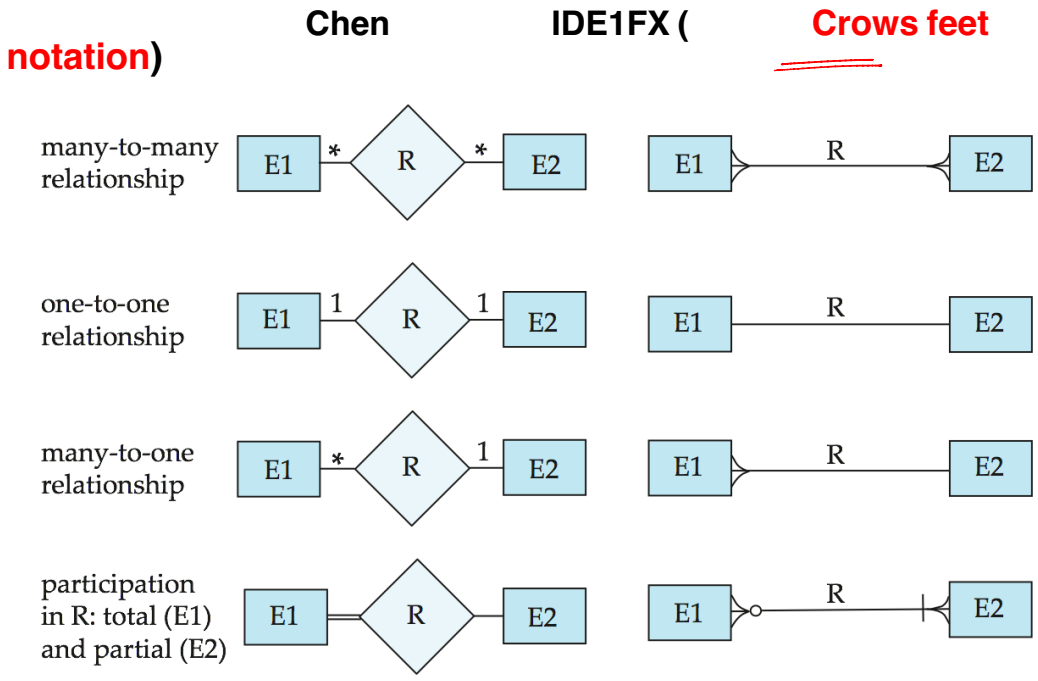


weak entity set 

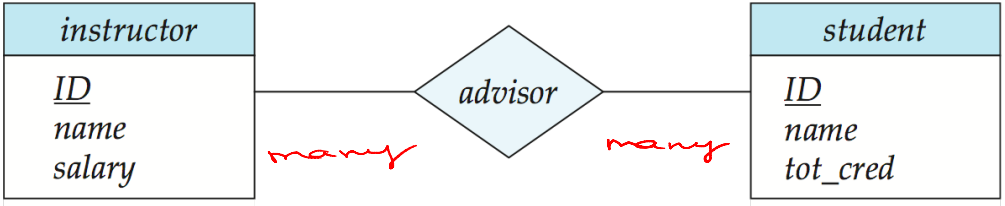
generalization 

total generalization 

Alternative ER Notations

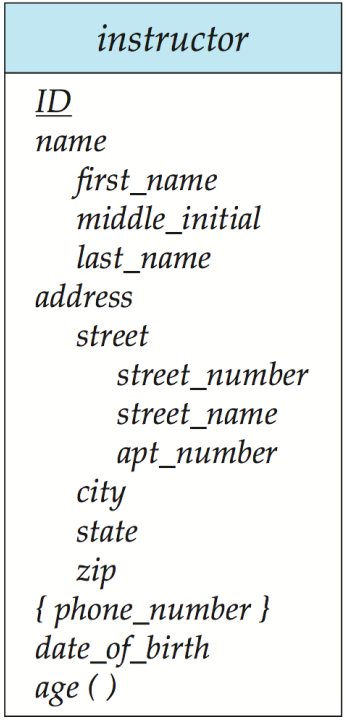


E-R Diagrams



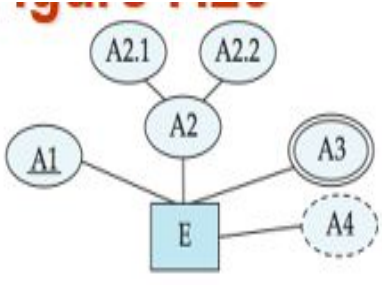
- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes

Entity With Composite, Multivalued, and Derived Attributes

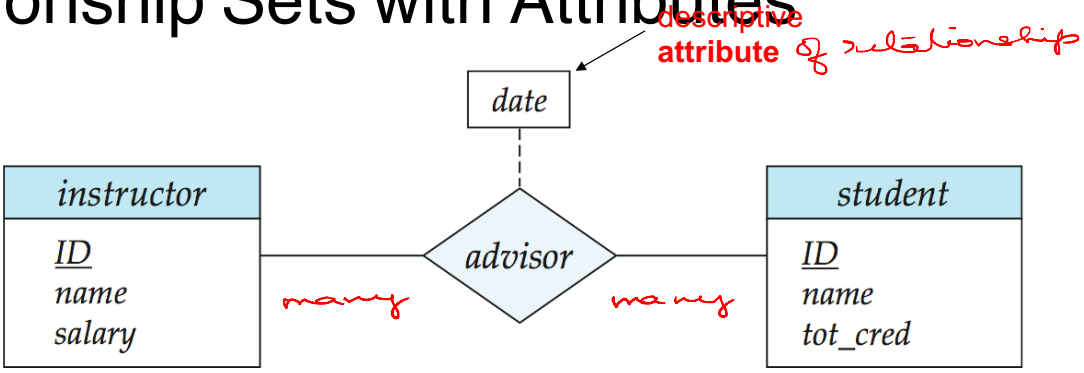


Note :- Preferable representation

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

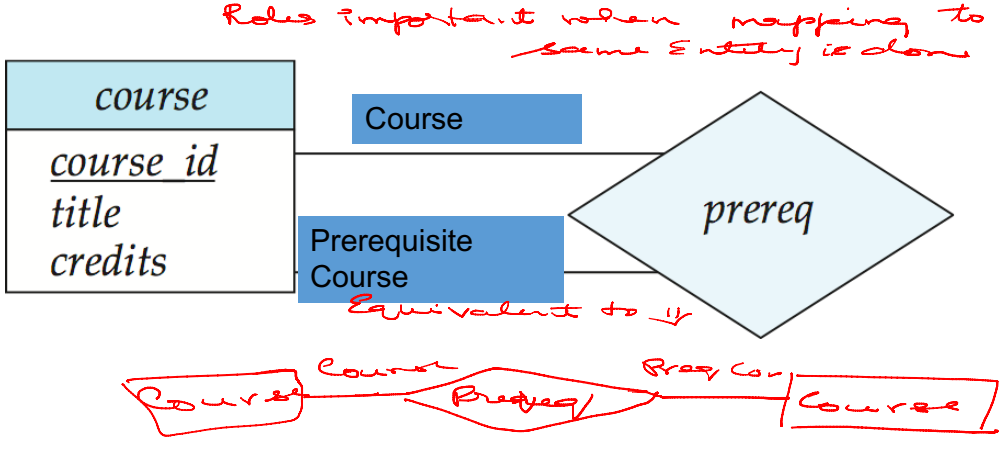


Relationship Sets with Attributes



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**.



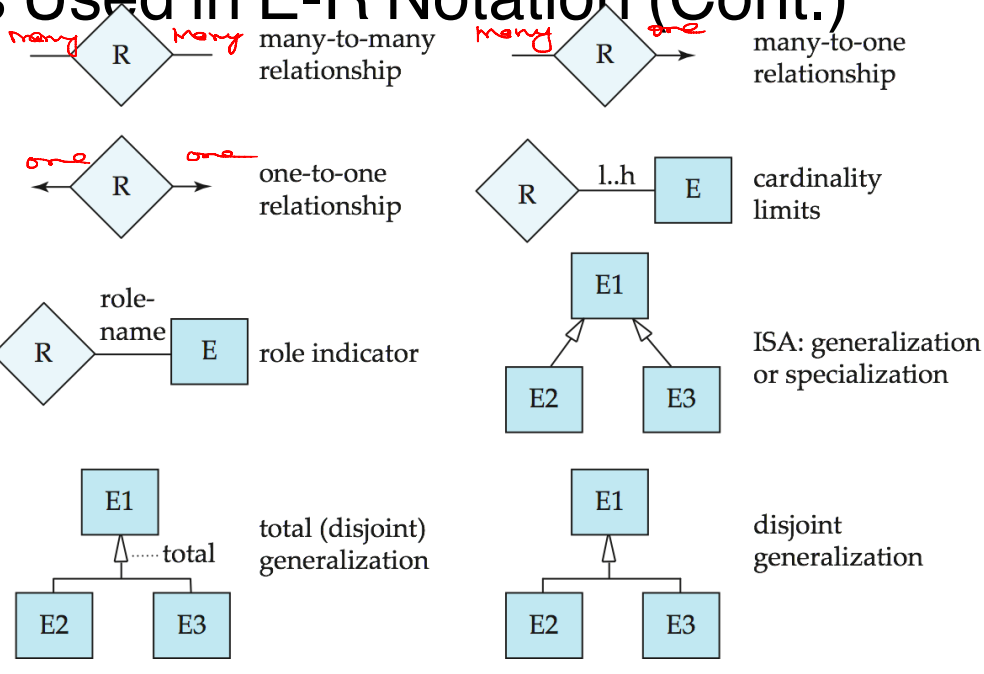
constraints

- Cardinality
- participation

Cardinality Constraints

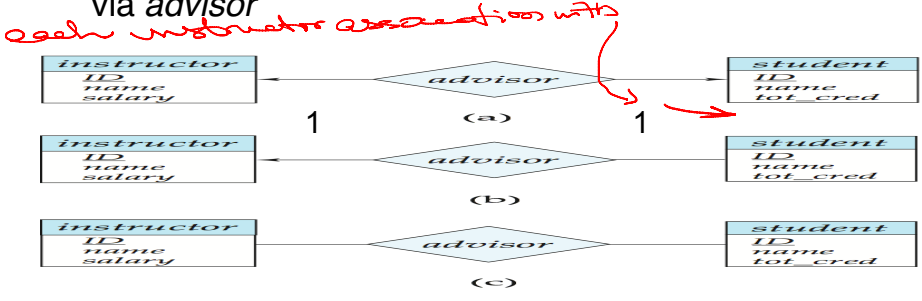
- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.

Symbols Used in E-R Notation (Cont.)



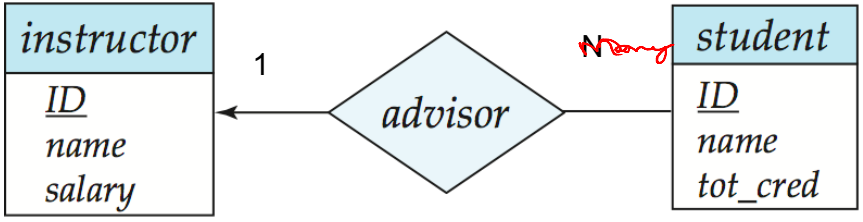
One-to-One Relationship

- 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*



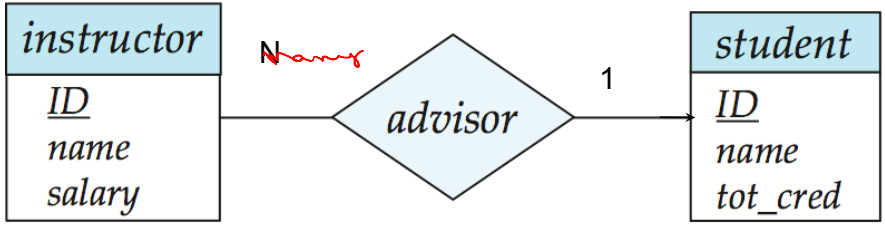
One-to-Many Relationship

- 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,



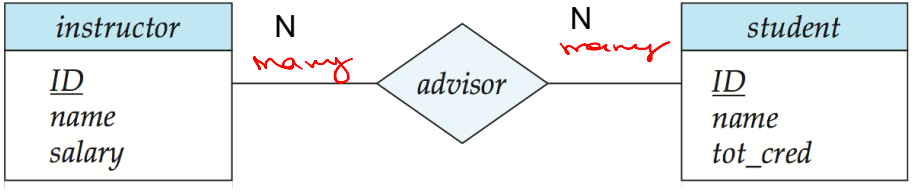
Many-to-One Relationships

- 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*



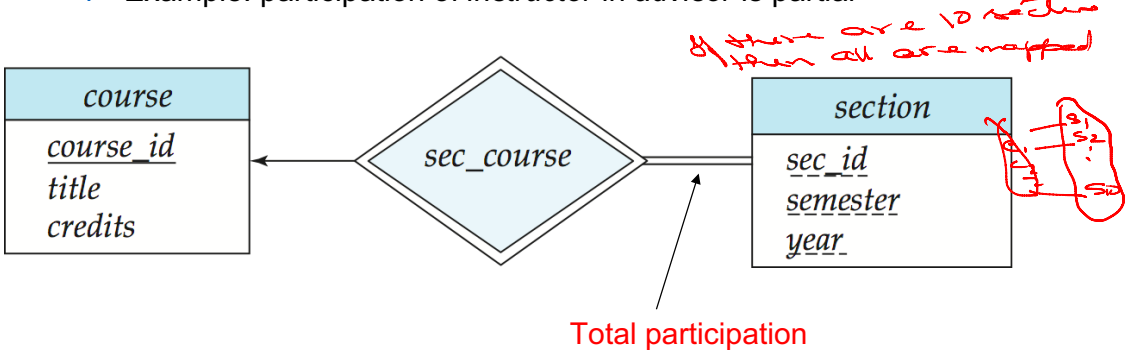
Many-to-Many Relationship

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



Participation of an Entity Set in a Relationship Set

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



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

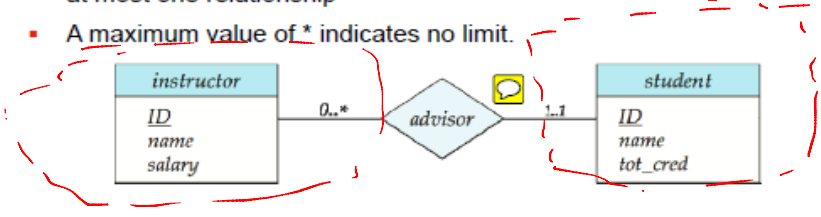
  - participation of *student* in *advisor* relation is total
    - every *student* must have an associated instructor

Keywords:-
Every ... must ...

- 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.

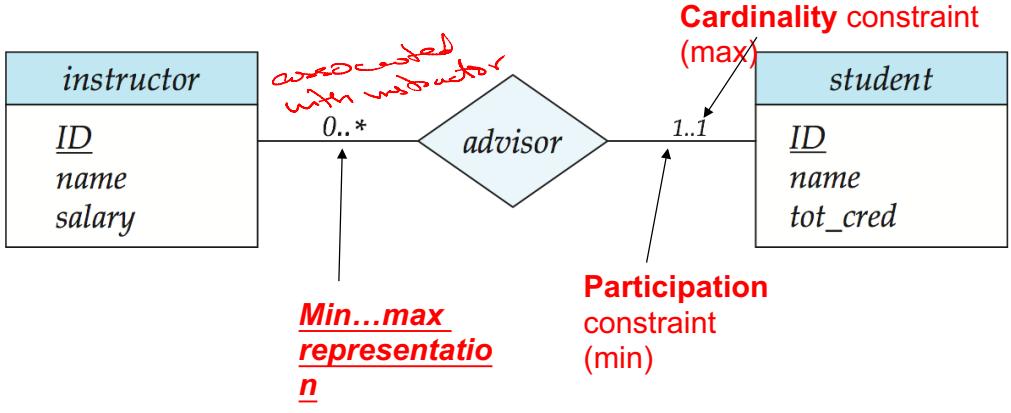


Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

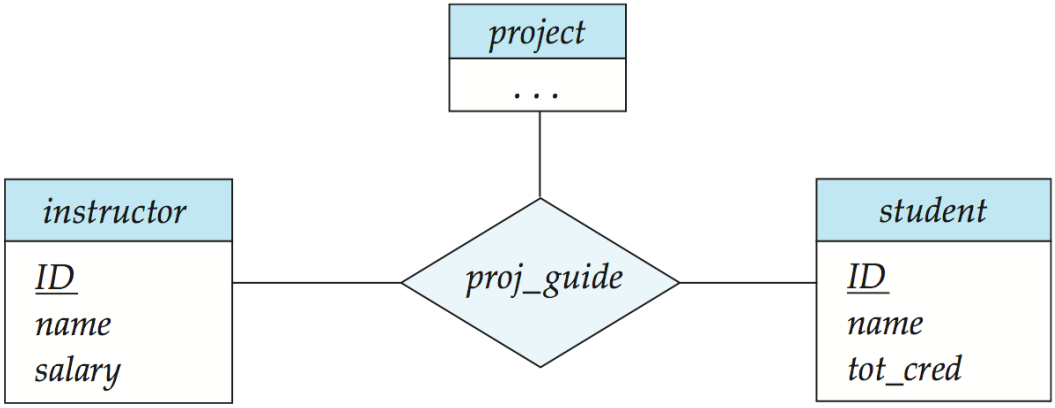
Note: Cardinality associated with the entity close to where it is written.

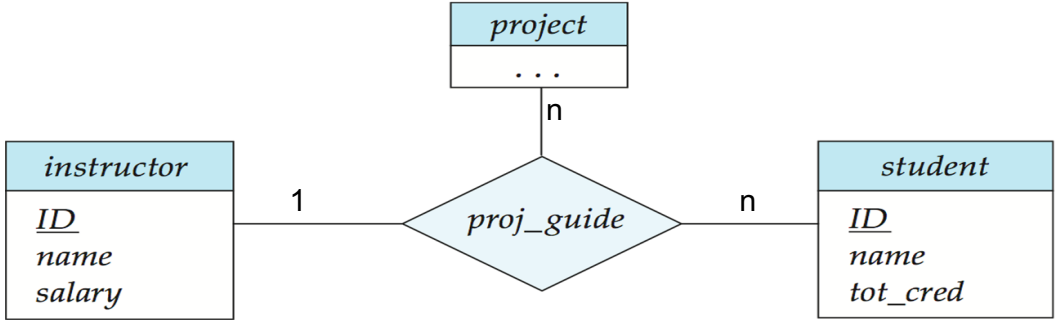
Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints



E-R Diagram with a Ternary Relationship




project
. . .
n
instructor
ID
name
salary
1
proj_guide
n
student
ID
name
tot_cred

Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g., an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.
  - E.g., a ternary relationship *R* between *A*, *B* and *C* with arrows to *B* and *C* could mean
    1. each *A* entity is associated with a unique entity from *B* and *C* or
    2. each pair of entities from (*A, B*) is associated with a unique *C* entity, and each pair (*A, C*) is associated with a unique *B*
  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow

Weak Entity Sets

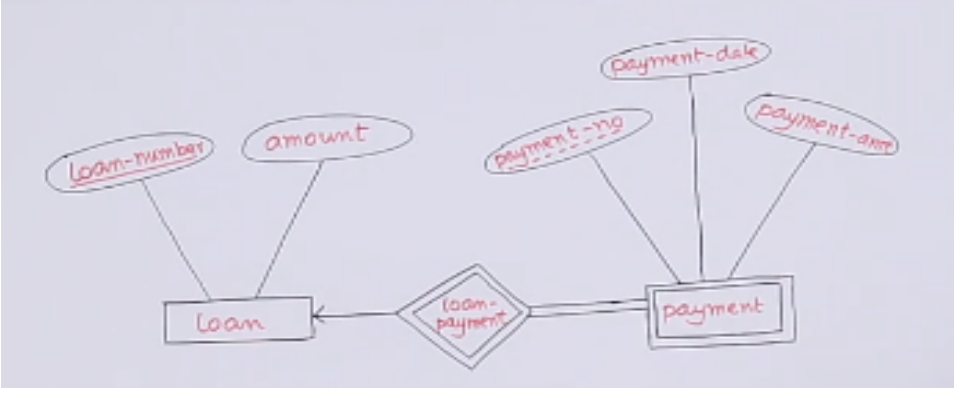
- An entity set that does not have a primary key is referred to as a **weak entity set**.
- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the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

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*)

Weak Entity Sets (Cont.)

- We underline the discriminator (*or partial key)* of a weak entity set with a dashed line.
- We put the identifying relationship of a weak

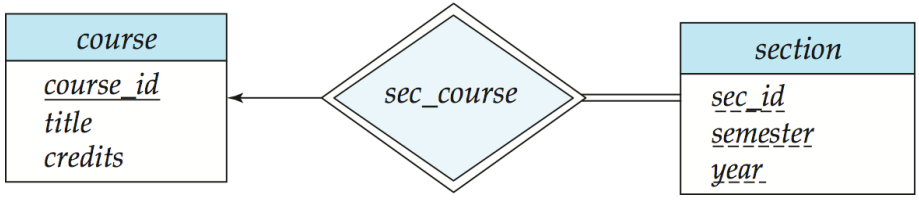


The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, **plus** the weak entity set's discriminator.

Weak Entity Sets (Cont.)

- Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.
- If *course_id* were explicitly stored, *section* could be made a strong entity, but then the relationship between *section* and *course* would be duplicated by an implicit relationship defined by the attribute *course_id* common to *course* and *section*

- Primary key for *section* – (*course_id, sec_id, semester, year*)



Removing redundant attributes in entity sets

- Once the entities and their corresponding attributes are chosen, the relationship sets among the various entities are formed. These relationship sets may result in a situation where attributes in the various entity sets are redundant and need to be removed from the original entity sets.

- the attribute *dept name* in fact gets added to the relation *instructor*, but only if each instructor has at most one associated department.
- If an instructor has more than one associated department, the relationship between instructors and departments is recorded in a separate relation *inst dept*.
- Ex2: sec- timeslot
- And so on

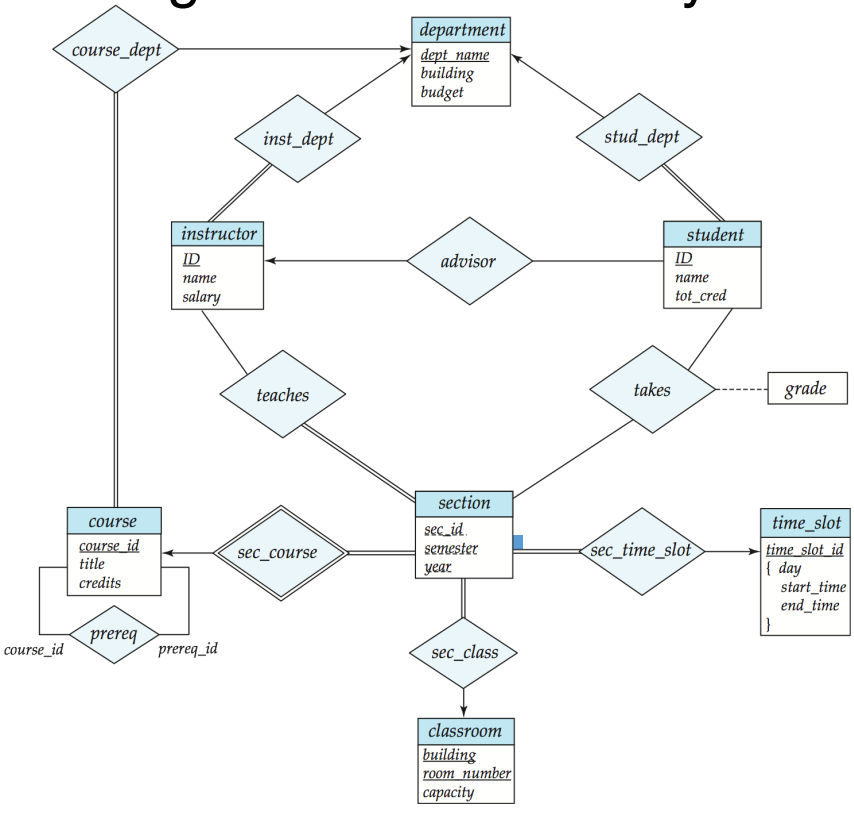
For our university example, we list the entity sets and their attributes below, with primary keys underlined:

- **classroom**: with attributes (*building*, *room number*, *capacity*).
- **department**: with attributes (*dept name*, *building*, *budget*).
- **course**: with attributes (*course id*, *title*, *credits*).
- **instructor**: with attributes (*ID*, *name*, *salary*).
- **section:** with attributes (*course id*, *sec id*, *semester*, *year*).
- **student**: with attributes (*ID*, *name*, *tot cred*).
- **time slot**: with attributes (*time slot id*, {(*day*, *start time*, *end time*) }).

The relationship sets in our design are listed below:

- **inst dept**: relating instructors with departments.
- **stud dept**: relating students with departments.
- **teaches**: relating instructors with sections.
- **takes**: relating students with sections, with a descriptive attribute g *rade*.
- **course dept**: relating courses with departments.
- **sec course**: relating sections with courses.
- **sec class**: relating sections with classrooms.
- **sec time slot**: relating sections with time slots.
- **advisor**: relating students with instructors.
- **prereq**: relating courses with prerequisite courses.

E-R Diagram for a University Enterprise



ER diagram

- **E-R diagram** can express the overall logical structure of a database graphically.
- Basic structure
- cardinality mapping
- participation
- Weak entity
- Composite
- mutivalued

Reduction to Relational Schemas

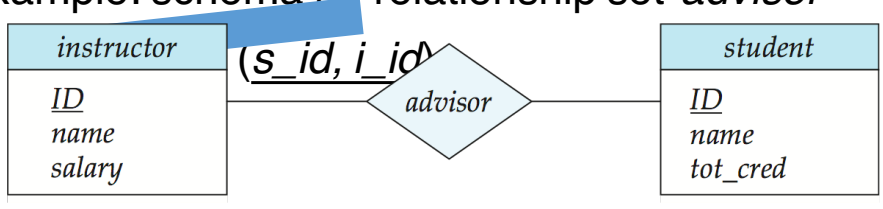
Reduction to Relation Schemas

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

Representing Relationship Sets

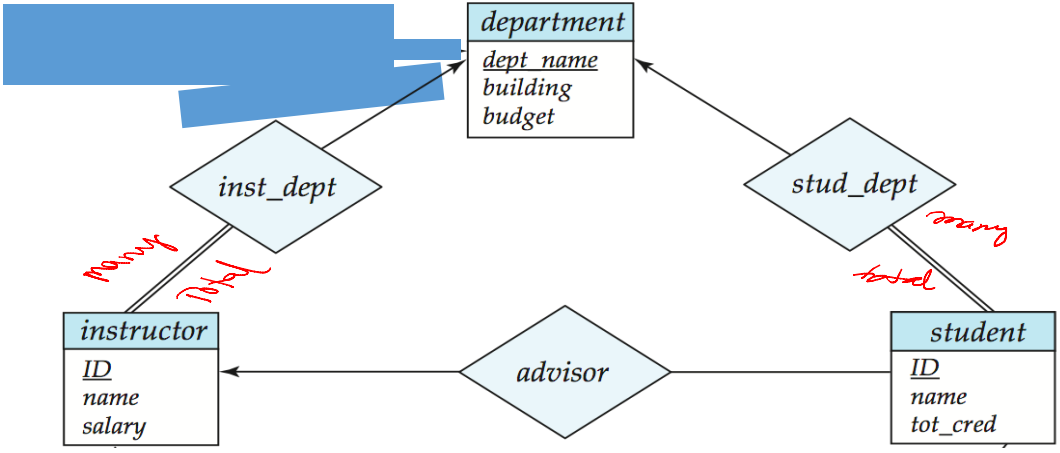
Rule 1 :-

- 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*



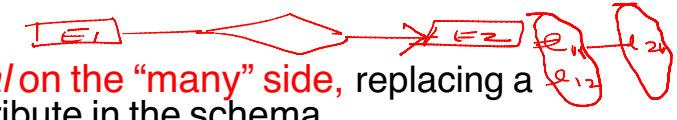
Rule 2

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
- If Partial participation: Separate table
- Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*



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



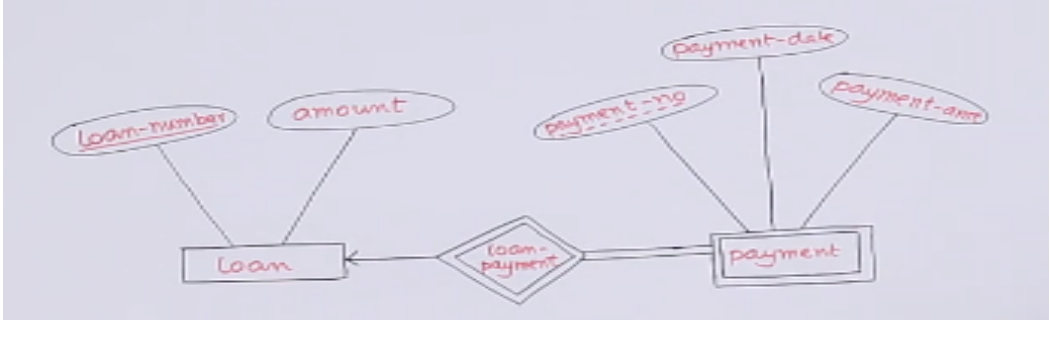
- If participation is *partial* on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values

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

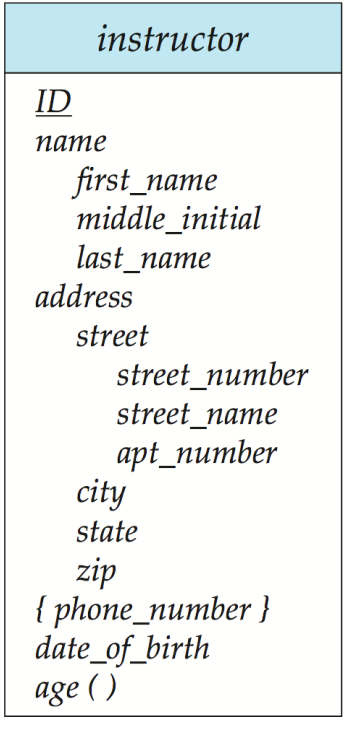
Representing Entity Sets With Simple Attributes

Rule 4:

- A strong entity set reduces to a schema with the same attributes
  student(ID, name, tot_cred)
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong



Composite and Multivalued Attributes



- 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*
- 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)*

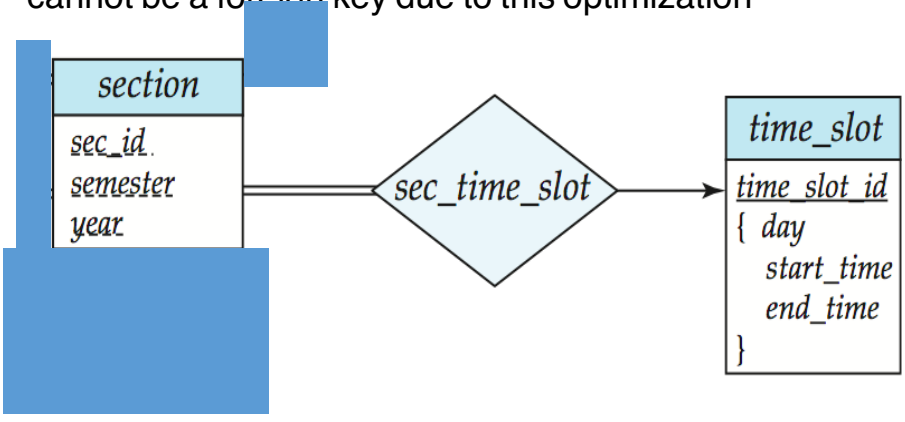
Composite and Multivalued Attributes

Rule 6

- A multivalued attribute *M* of an entity *E* is represented by a separate schema *EM*
  - Schema *EM* has attributes corresponding to the primary key of *E* and an attribute corresponding to multivalued attribute *M*
  - Example: Multivalued attribute *phone_number* of *instructor* is represented by a schema:
    inst_phone= (*ID*, *phone_number*)
  - Each value of the multivalued attribute maps to a separate tuple of the relation on schema *EM*
    - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples: (22222, 456-7890) and (22222, 123-4567)

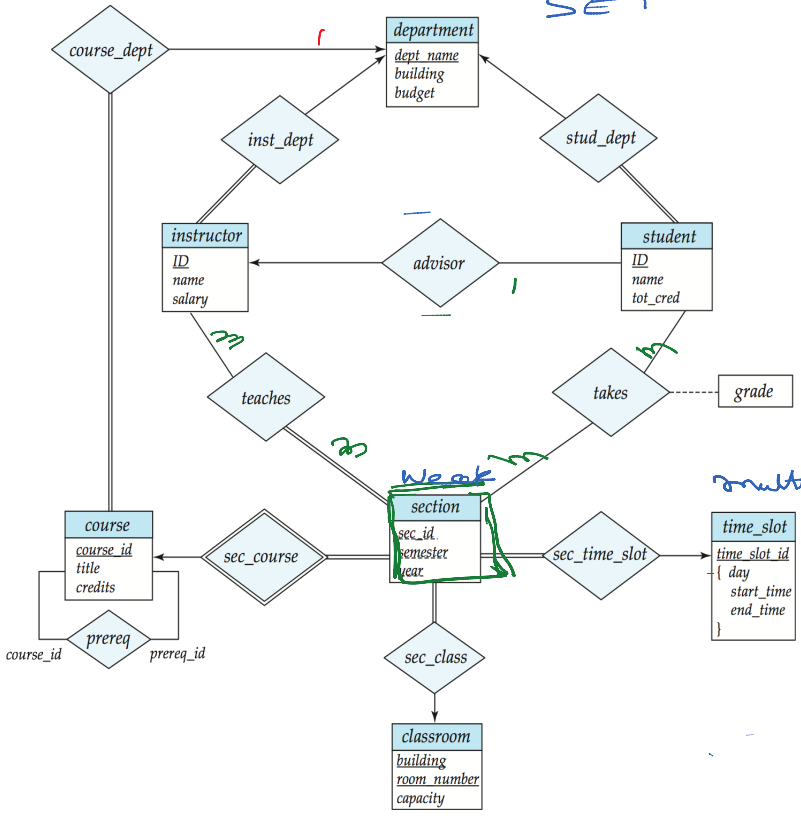
Multivalued Attributes (Cont.)

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



Summarizing the rule

- Regular entity: schema with attributes
- Weak entities: schema including strong entity PK
- Composite attribute: schema is extended
- Multivalued attribute: new schema with its individual attribute and PK of the parent table
- Based on degree
  - One to one: separate schema
  - Many to one or one to many: schema for many side include PK of one side
  - Many to many: relationship is converted to schema
- Based on participation:
  - Total participation: on many side add PK of one side
  - Partial participation: separate table


course_dept
department
dept_name
building
budget
inst_dept
stud_dept
instructor
ID
name
salary
advisor
student
ID
name
tot_cred
teaches
takes
grade
Weak
section
sec_id
semester
year
course
course_id
title
credits
sec_course
sec_time_slot
time_slot
time_slot_id
{ day
start_time
end_time
}
prereq
course_id
prereq_id
sec_class
classroom
building
room_number
capacity

dept(dname, building, budget)

student(ID, name, tot_cred, dname, IID)

instructor(IID, name, salary, dname)

classroom(building, roomno, capacity)

course(courseid, title, credit, dname)

teaches(IID, secid, sem, year, courseid)

takes(grade, ID, secid, sem, year, courseid)

time_slot(day, start time, end time)

section(courseid, secid, sem, year, building, roomno, tID);

Advisor(SID, IID)
Prereq(courseid,
prereqid);

Construct ER diagram (student work)

1. Each person has at most one e-mail address. Each e-mail address belongs to exactly one person.
2. An apartment is located in a house in a street in a city in a country.
3. Two teams play football against each other. A referee makes sure the rules are followed.
4. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.
5. Consider a company database having each employee works for only 1 department which is managed by one of them. Department has projects running under it and assigned to a team of employees. Further employees are managed by a supervisor. Company is also interested to know the details of employee's dependents.