Database Design: Entity-Relationship Model

Database design

- How you get from user requirements to full database
- **Conceptual design**
  - capture user needs (data that needs to be stored, what questions they want to ask)
  - model the data and how they relate to each other
  - use a modeling system that is easy to explain to users (E-R diag.)
- **Logical design**
  - implement the actual relational schema
  - implement the relevant query code
- **Physical design**
  - optimize the database (files, indices, etc.)
  - make low-level decisions on how the data are stored
- Note: the steps above may go through a number of iterations

Modeling

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

  Example:
  44553 (Peltier) *advisor* 22222 (Einstein)
  student entityrelationship set *instructor* entity

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

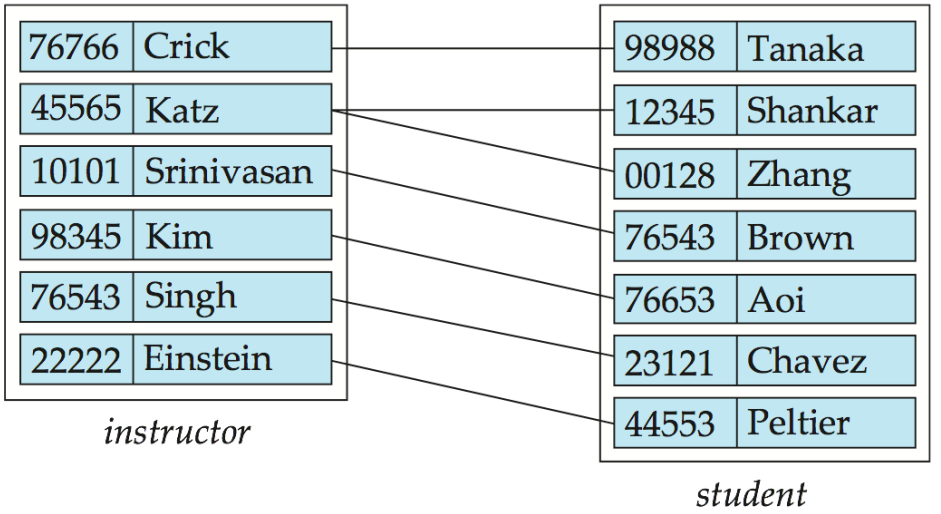
  $$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  – Example:

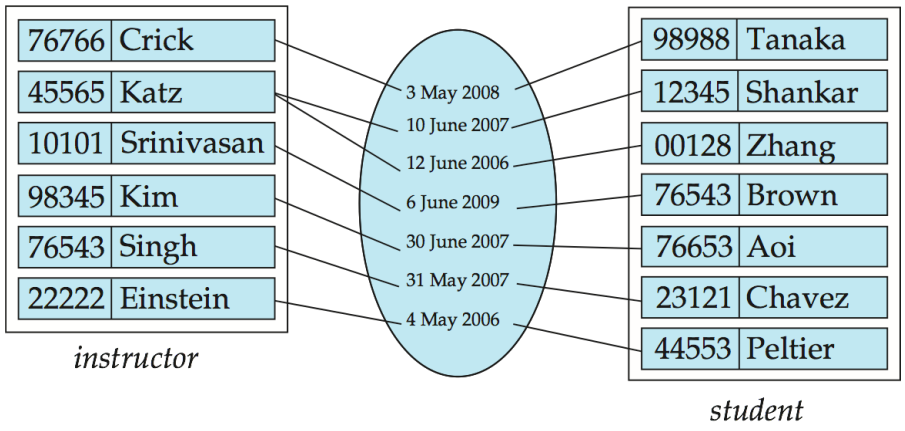
  $(44553,22222) \in$ *advisor*

Relationship Set *advisor*



Relationship Sets (Cont.)

- An **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

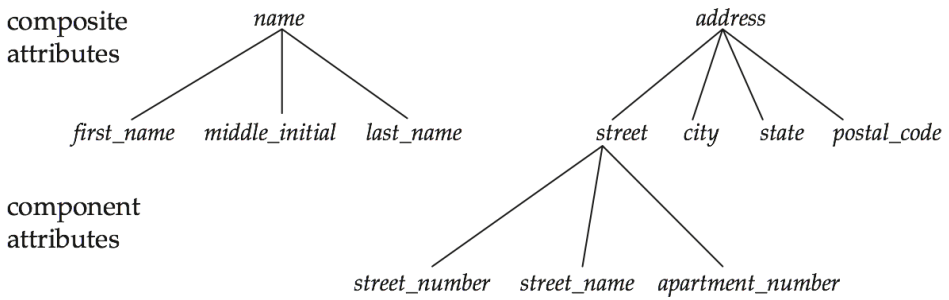
- **binary relationship**
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)
  - Example: *students* work on research *projects* under 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

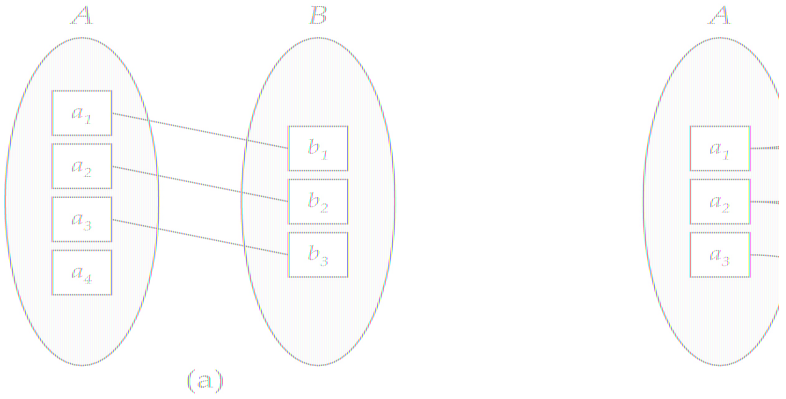
Composite Attributes



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

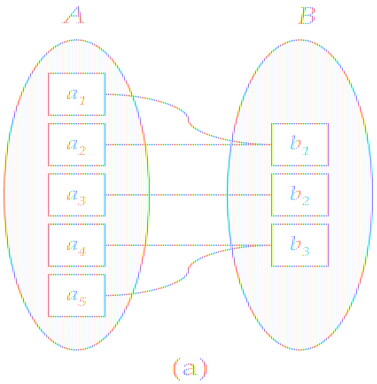


One to one

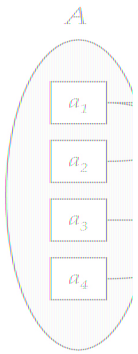
One to many

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

- 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...example

Student
SSN
UMID
Full Name
Address
Department
Major
Year

Attributes	Super-key?	Candidate key?	Primary key?
SSN, UMID, FN, AD, DEP, MAJ, YR	☺	⊘	⊘
SSN	☺	☺	
UMID	☺	☺	
FN	⊘	⊘	⊘
FN, AD	☺	☺	
FN, AD, DEP, MAJ, YR	☺	⊘	
FN, YR	⊘	⊘	⊘
FN, DEP, MAJ, YR	☺	☺	

Keys for Relationship Sets

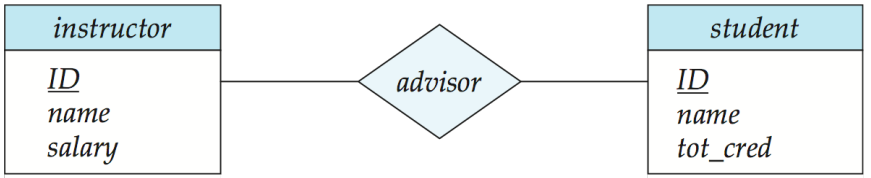
- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (*s_id, i_id*) is the super key of *advisor*
  - *NOTE: this means* ***a pair of entity sets can have at most one relationship in a particular relationship set.***
    - Example: if we wish 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
- Need to consider semantics of relationship set in selecting the *primary key* in case of more than one candidate key

Redundant Attributes

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

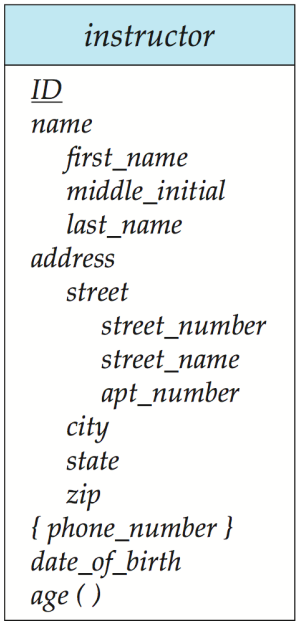
  and a relationship
  - *inst_dept* relating *instructor* and *department*
- Attribute *dept_name* in entity *instructor* is redundant since there is an explicit relationship *inst_dept* which relates instructors to departments
  - The attribute replicates information present in the relationship, and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

E-R Diagrams

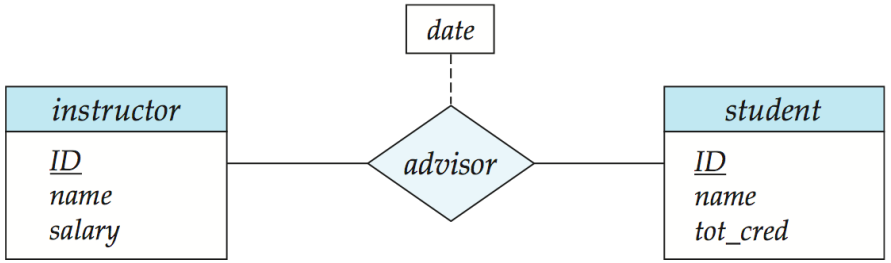


- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes

Entity With Composite, Multivalued, and Derived Attributes

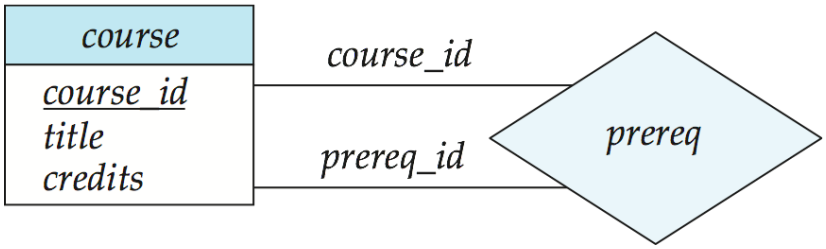


Relationship Sets with Attributes



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**.

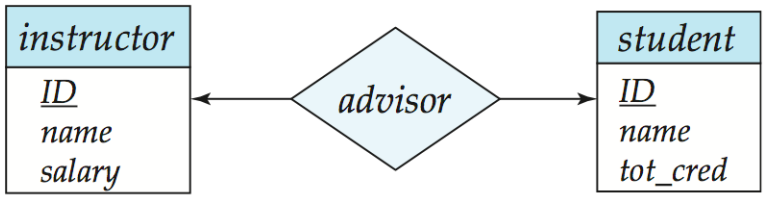


Cardinality Constraints

- 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:
  - 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*

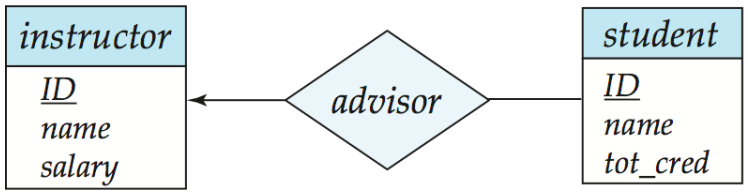
One-to-One Relationship

- 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*



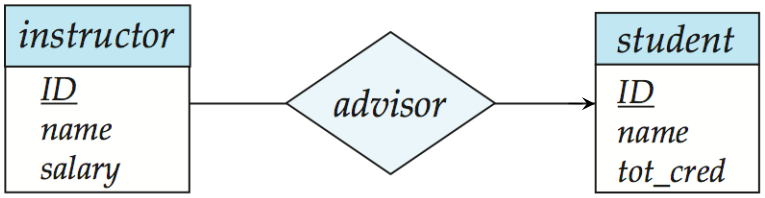
One-to-Many Relationship

- 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,



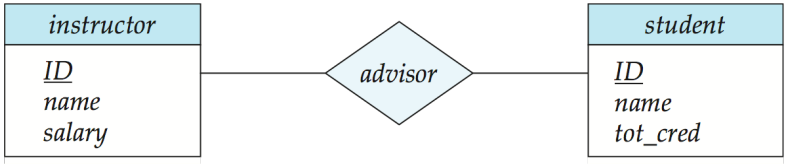
Many-to-One Relationships

- 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*



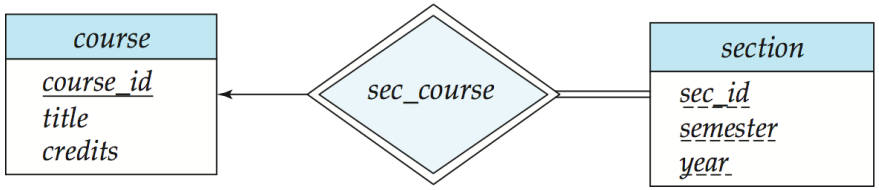
Many-to-Many Relationship

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



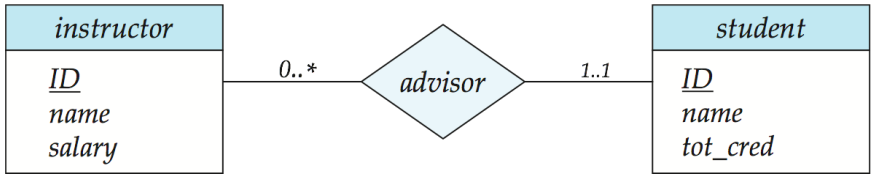
Participation of an Entity Set in a Relationship Set

- 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

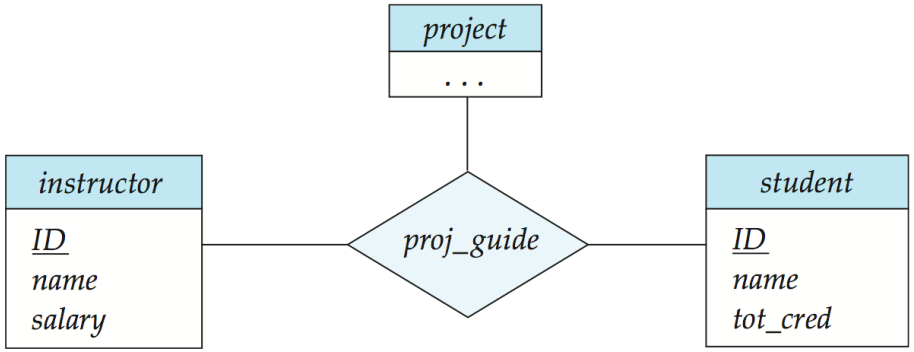


Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints



E-R Diagram with a Ternary Relationship



Cardinality Constraints on Ternary Relationship

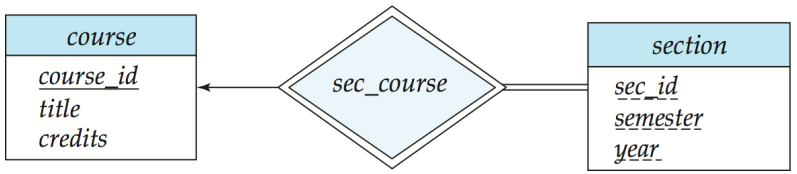
- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g., an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.
  - 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.

Weak Entity Sets (Cont.)

- We underline the discriminator of a weak entity set with a dashed line.
- We put the identifying relationship of a weak entity in a double diamond.
- Primary key for *section* – (*course_id, sec_id, semester, year*)

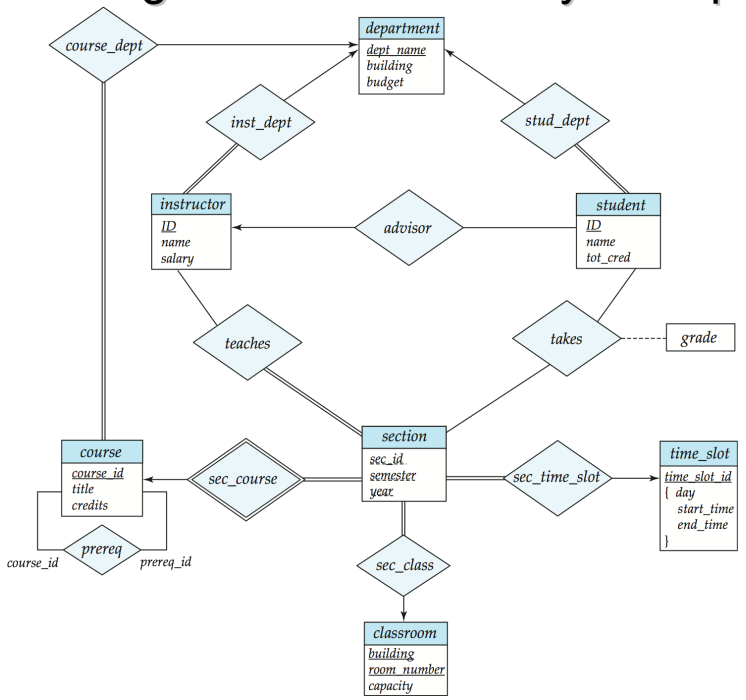


Weak Entity Sets (Cont.)

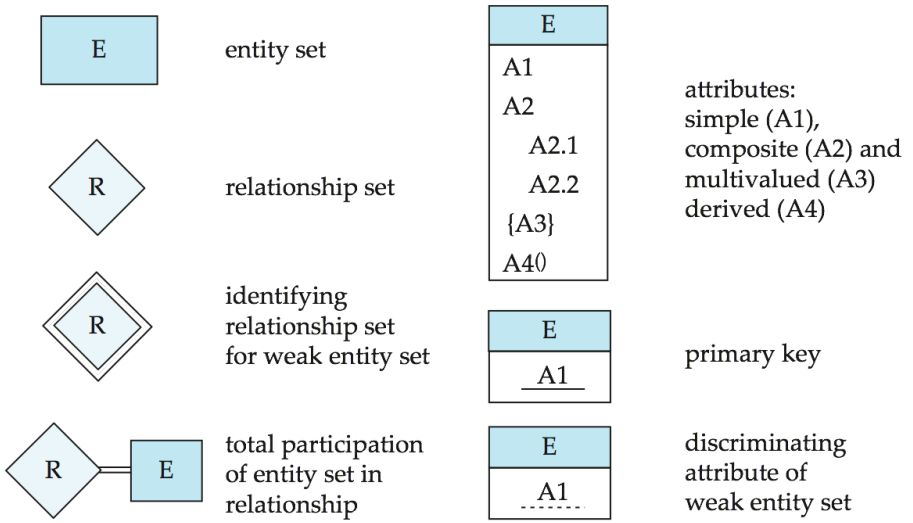
- Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.
- If *course_id* were explicitly stored, *section* could be made a strong entity, but then the relationship between *section* and *course* would be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

E-R diagram for insurance company?

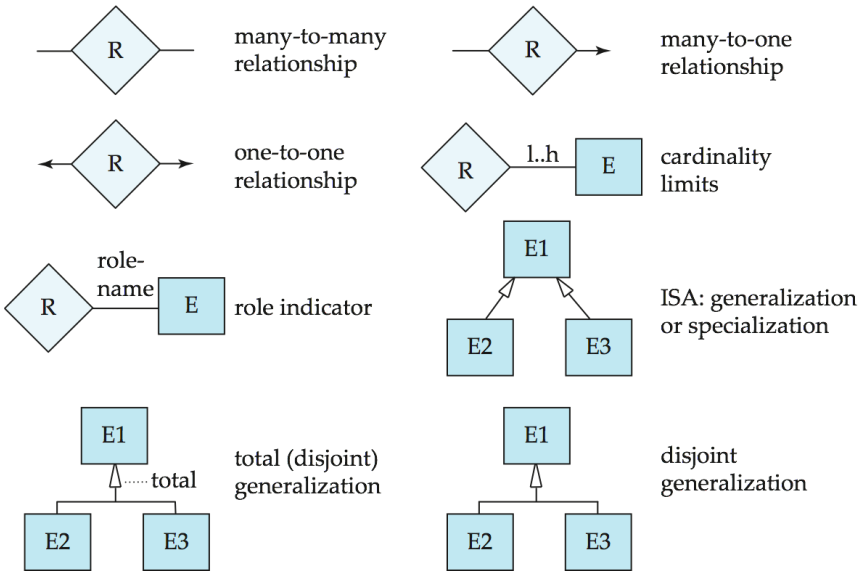
E-R Diagram for a University Enterprise



Summary of Symbols Used in E-R Notation



Symbols Used in E-R Notation (Cont.)



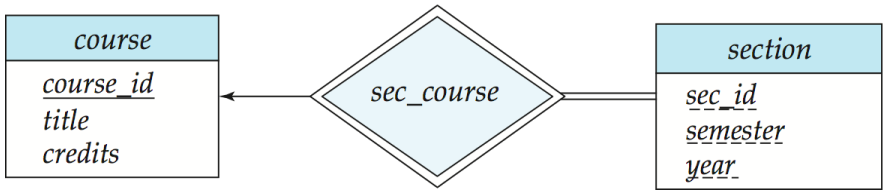
Reduction to Relational Schemas

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 Entity Sets With Simple Attributes

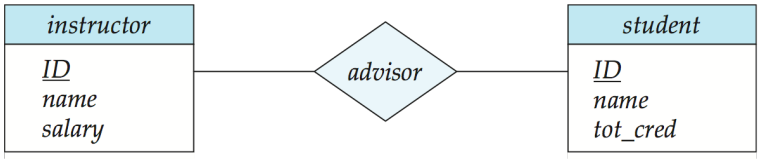
- A strong entity set reduces to a schema with the same attributes
  student(ID, name, tot_cred)
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
  section (course_id, sec_id, sem, year)

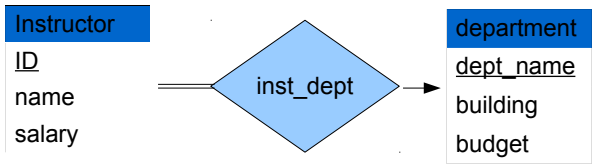


Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

  advisor = (*s_id, i_id*)

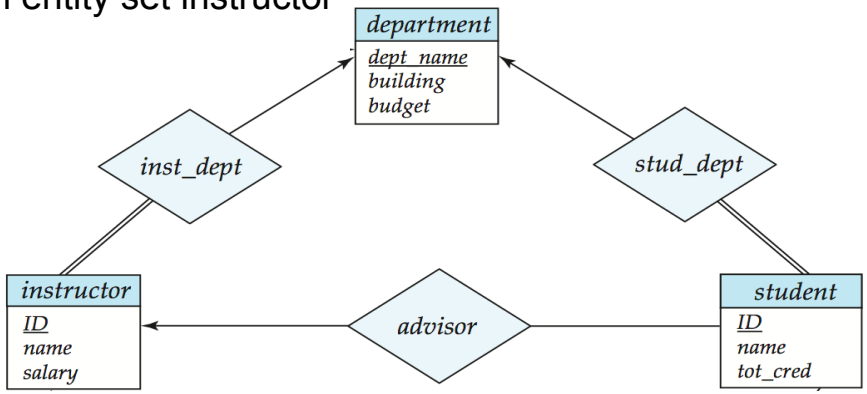




Instructor(ID, dept_name, name, salary)

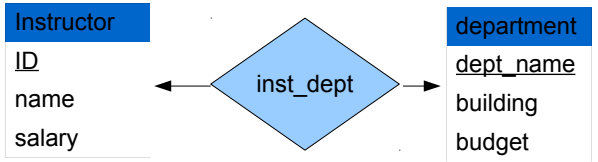
Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the “many” side, containing the primary key of the “one” side
- 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



Instructor(ID, dept_name, name, salary)
or
department(ID, inst_name, building, budget)

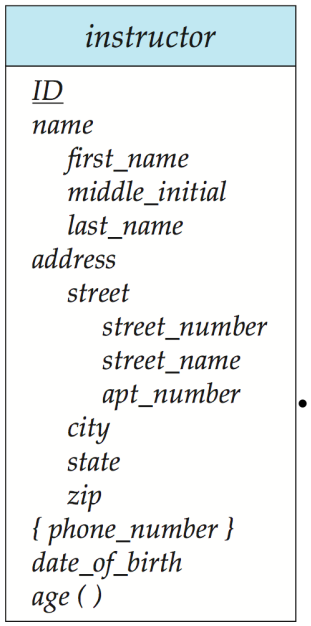
Redundancy of Schemas (Cont.)

- 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

  i.e. the approach in the previous slides does not work need to represent relationship as a separate table

- 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

Composite and Multivalued Attributes



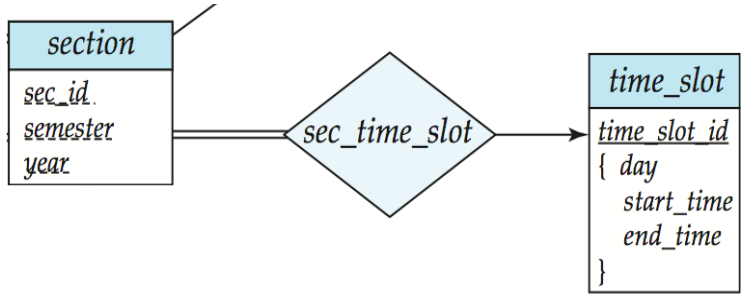
- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name*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

- 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)

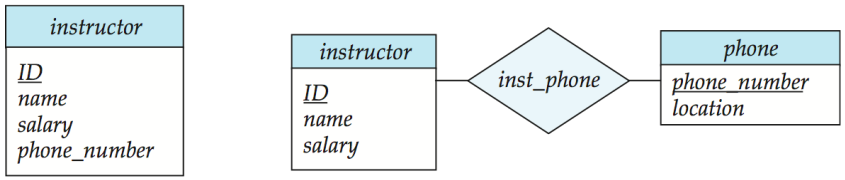
Multivalued Attributes (Cont.)

- 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*)



Design Issues

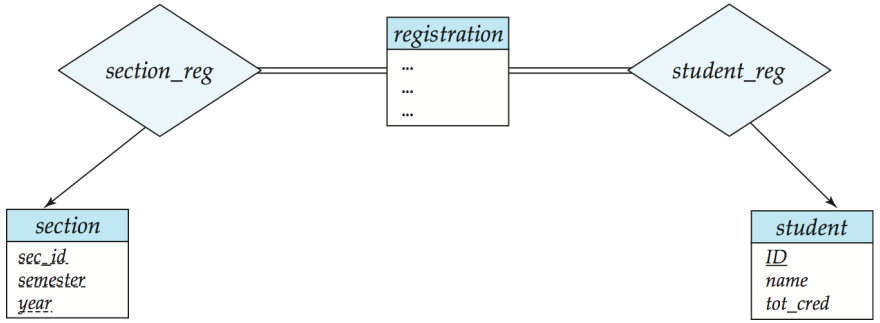
- **Use of entity sets vs. attributes**



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

Design Issues

- **Use of entity sets vs. relationship sets**
  Possible guideline is to designate a relationship set to describe an action that occurs between entities



Design Issues

- **Binary versus n-ary relationship sets**
  Although it is possible to replace any nonbinary (*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.
- **Placement of relationship attributes**
  e.g., attribute *date* as attribute of *advisor* or as attribute of *student*

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 know)
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*

Converting Non-Binary Relationships to Binary Form

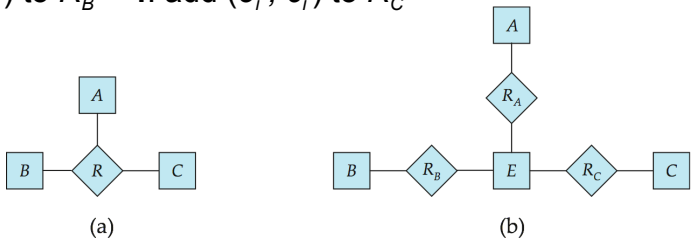
- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:

1. R_A, relating *E* and *A* **2.** R_B, relating *E* and *B* **3.** R_C, relating *E* and *C*

  - Create a special identifying attribute for *E*
  - 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 identified by the three relationship sets