KOREA TECH
KOREA UNIVERSITY OF TECHNOLOGY & EDUCATION

Database System

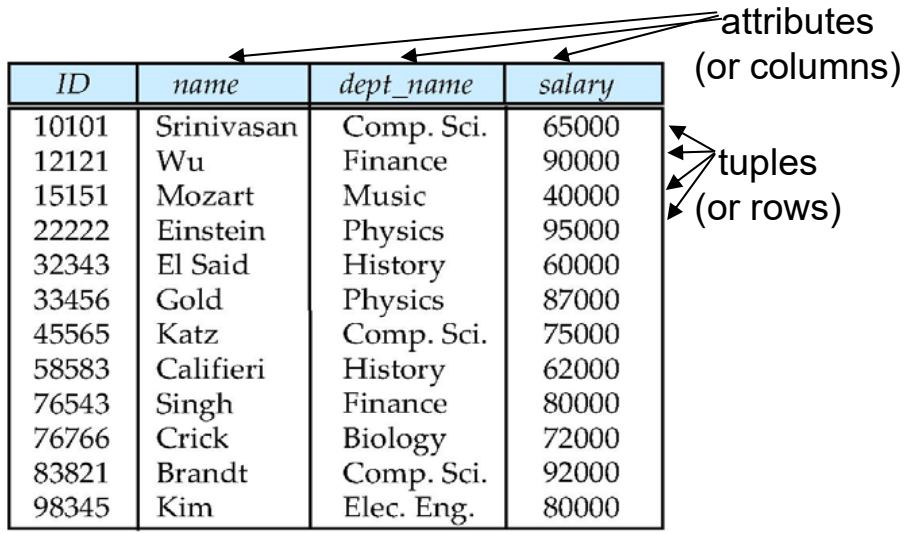
Relational Database

Muhammad Tariq Mahmood
tariq@koreatech.ac.kr
School of Computer Science and Engineering
Korea University of Technology and Education

Structure of Relational database

- A relational database consists of a collection of **relations/tables**, each of which is assigned a unique name.

Instructor



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Prereq

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

course

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Structure of Relational database

- The set of allowed values for each attribute is called the **domain** (*D)* of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain
- The null value causes complications in the definition of many operations

Relation Schema and Instance

- A_1, A_2, …, A_n are *attributes*

- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  Example:

  $instructor = (ID, name, dept\_name, salary)$

- Given sets D_1, D_2, …. D_n , a **relation** ***r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

- Thus, a relation is a set of n–tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

  - The current values (**relation instance**) of a relation are specified by a table
  - An element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Design Phases

- **Conceptual Design**
  - develop a conceptual schema based on the requirement specification
  - Conceptual Schema ->(Entity-Relationship Model)
- **Logical Design**
  - Define the tables (entities) and fields (attributes). Identify primary and foreign key for each table. Define relationships between the tables.
  - Conceptual Schema Reduction to Relation Schemas (ER Model → Logical Model)
- **Physical Design**
  - Develop physical data structures; specify file organization, and data storage etc.

6

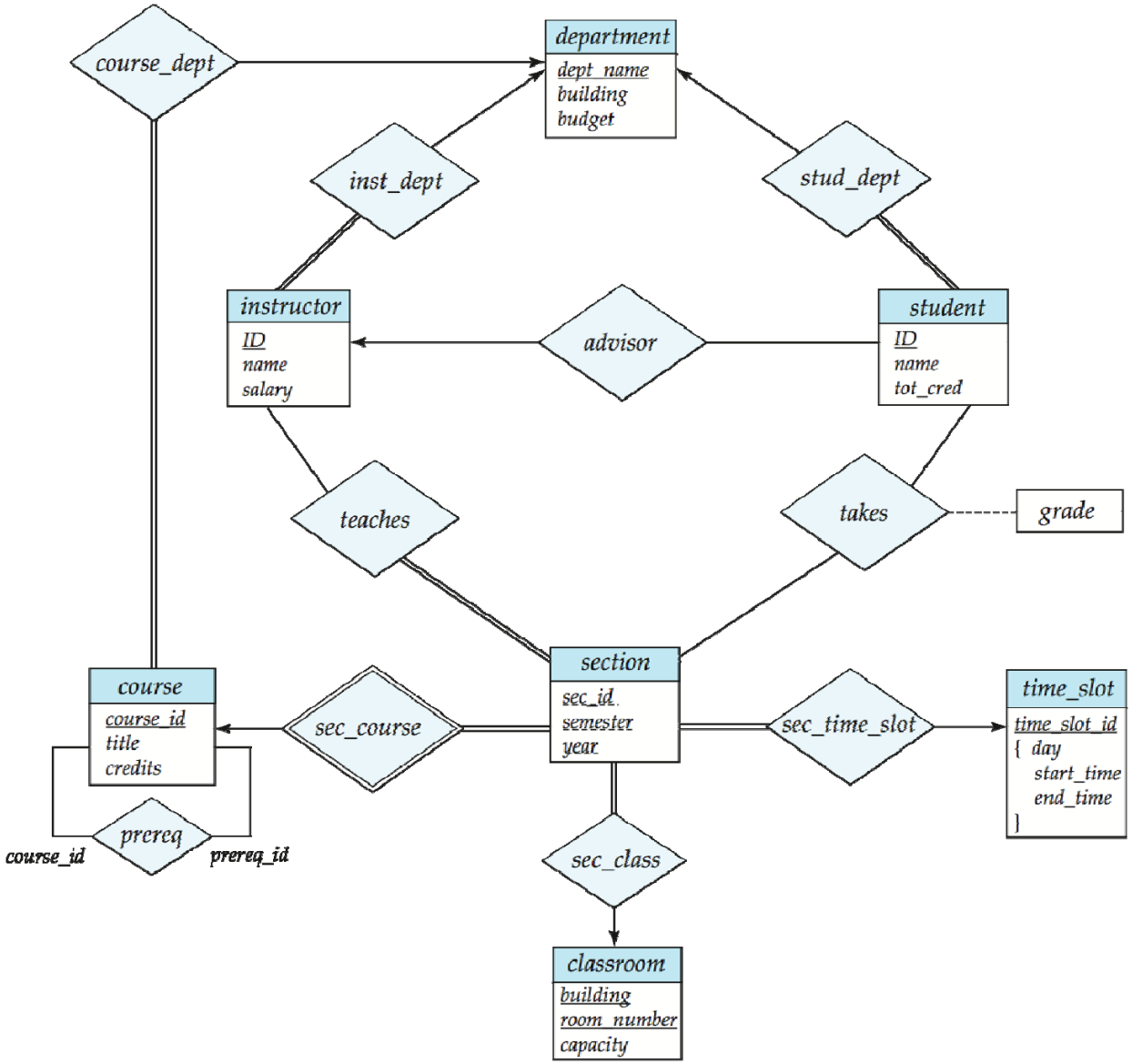
Conceptual Design
The Entity-Relationship Model

- The **entity-relationship** (**E-R**) data model represents the overall logical structure of a database.

- The E-R model is useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

- E-R Model employ three basic concepts/constructs
  - Entity sets
  - Relationship sets
  - Attributes

Conceptual Design
The Entity-Relationship Model



Entity

- An **entity** is a “thing” or “object” in the real world that is distinguishable from all other objects.
- Entity is basic building block of the E-R model. The term entity is used in three different meanings or for three different terms and that are:
  - Entity type
  - Entity instance
  - Entity set
- **Entity type**
  - The entity type can be defined as a name/label assigned to items/objects that exist in an environment and that have similar properties.
  - It could be person, place, event or even concept.
- **Entity instance**
  - It is a particular object belonging to a particular entity type

Entity

- **Entity Set**
  - A group of entity instances of a particular entity type is called an entity set. For example, all employees of an organization form an entity set. Like all students, all courses, all of them form entity set of different entity types

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

instructor

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

student

Relationship Sets

- A **relationship** is an association among several entities.
- A **relationship set** is a set of relationships of the same type.
  - Formally, it is a mathematical relation on $n \geq 2$ (possibly nondistinct) entity sets. If $E1, E2, \ldots, En$ are entity sets, then a relationship set R is a subset of $\{(e1, e2, \ldots, en) \mid e1 \in E1, e2 \in E2, \ldots, en \in En\}$ where $(e1, e2, \ldots, en)$ is a relationship.
- Example:

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

- The association between entity sets is referred to as participation; that is, the entity sets E1, E2, . . . , En participate in relationship set R.

- A **relationship instance** in an E-R schema represents an association between the named entities in the real-world enterprise that is being modeled.

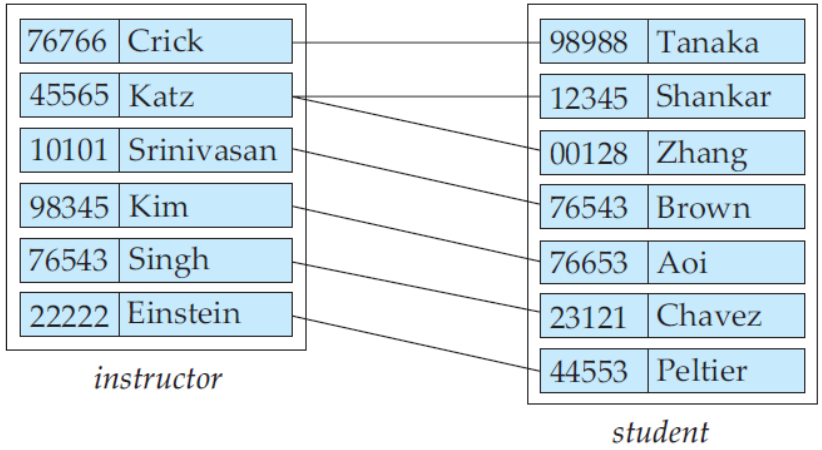


Figure 7.2 Relationship set *advisor*.

Attribute Sets

- An entity is represented by a set of **attributes.**
  - Example:
    instructor = (*ID, name, street, city, salary*)
    course= (*course_id, title, credits*)

- Attributes are also descriptive properties possessed by each member of an entity set.

- Formally, an attribute of an entity set is a function that maps from the entity set into a domain.

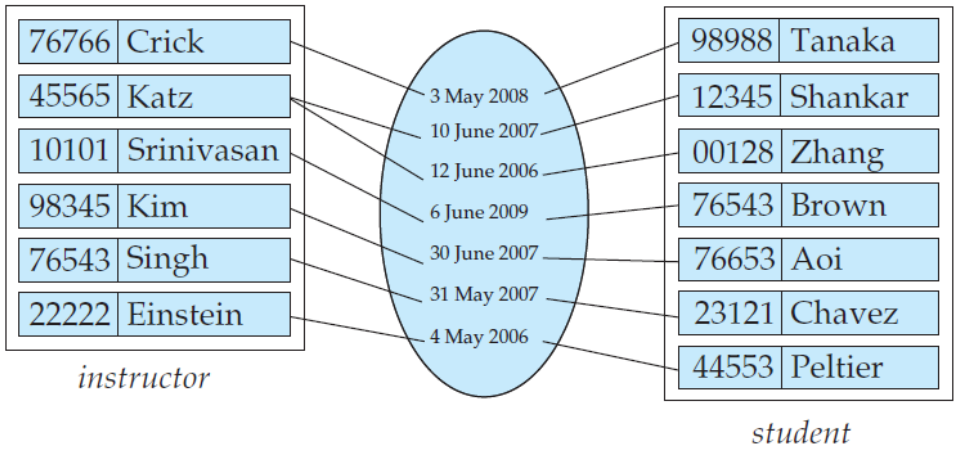


Figure 7.3 *date* as attribute of the *advisor* relationship set.

Attribute Sets

- **Simple** and **composite** attributes.
  - **Simple** attributes can not be divided into subparts
  - **Composite** attributes can be divided into subparts
- **Derived** attribute.
  - The value for this type of attribute can be derived from the values of other related attributes or entities.
- **Single-valued** and **multivalued** attributes.

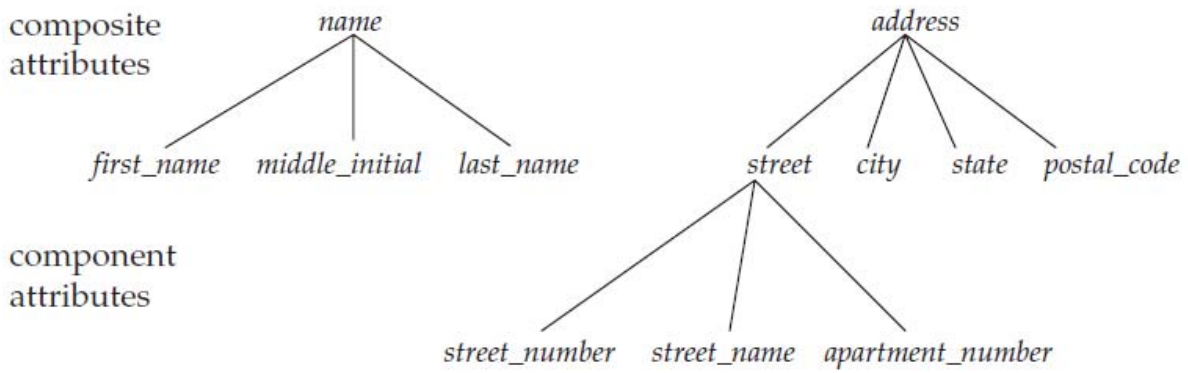


Figure 7.4 Composite attributes instructor *name* and *address*.

Constraints

- An E-R enterprise schema may define certain constraints to which the contents of a database must conform.
  - mapping cardinalities
  - participation constraints
  - Keys

- **Mapping cardinalities**,
  - Mapping cardinalities or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.
  - Mapping cardinalities are most useful in describing **binary relationship** sets

Types of Relationships

- For a binary relationship set *R* between entity sets *A* and *B*, the mapping cardinality must be one of the following:
  - **One-to-one.** An entity in *A* is associated with *at most* one entity in *B*, and an entity in *B* is associated with *at most* one entity in *A*.
  - **One-to-many.** An entity in *A* is associated with any number (zero or more) of entities in *B*. An entity in *B*, however, can be associated with *at most* one entity in *A*.

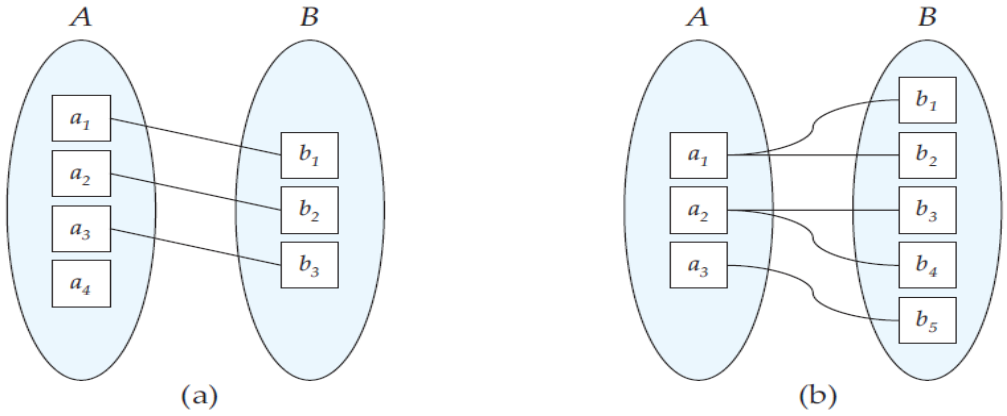


Figure 7.5 Mapping cardinalities. (a) One-to-one. (b) One-to-many.

Types of Relationships

- **Many-to-one.** An entity in *A* is associated with *at most* one entity in *B*. An entity in *B*, however, can be associated with any number (zero or more) of entities in *A*.

- **Many-to-many.** An entity in *A* is associated with any number (zero or more) of entities in *B*, and an entity in *B* is associated with any number (zero or more) of entities in *A*.

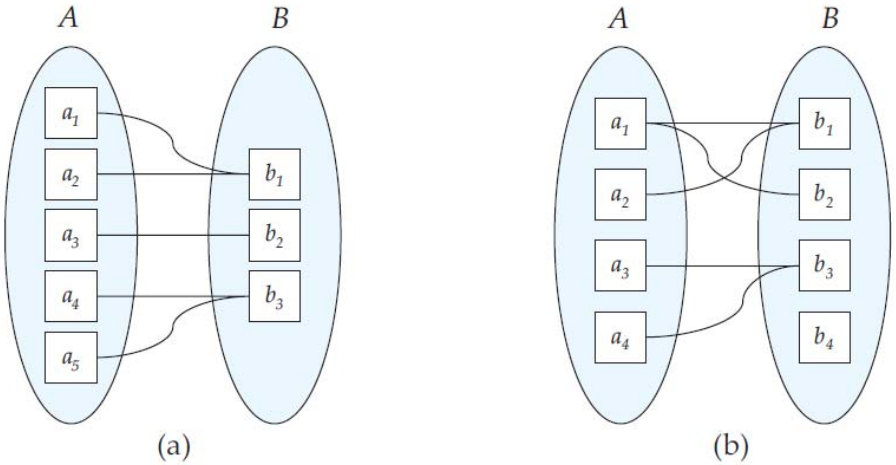


Figure 7.6 Mapping cardinalities. (a) Many-to-one. (b) Many-to-many.

Participation Constraints

- The participation of an entity set *E* in a relationship set *R* is said to be **total** if every entity in E participates in at least one relationship in R.
- If only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be **partial.**

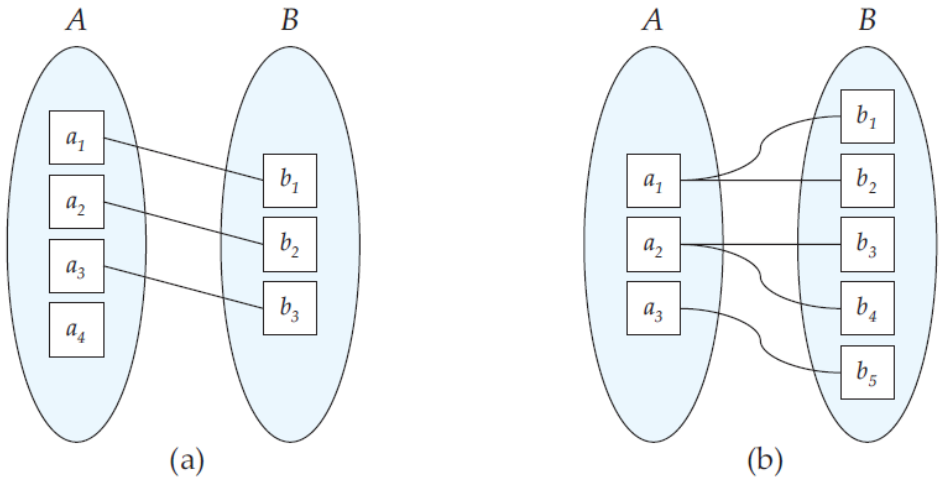


Figure 7.5 Mapping cardinalities. (a) One-to-one. (b) One-to-many.

Keys

- A **key** for an entity is a set of attributes that suffice to distinguish entities from each other.
- 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**.

Entity-Relationship Diagrams
Basic Structure

- An E-R diagram consists of the following major components:
  - Rectangles divided into two parts represent entity sets. The first part, which in this textbook is shaded blue, contains the name of the entity set. The second part contains the names of all the attributes of the entity set.
  - Diamonds represent relationship sets.
  - Undivided rectangles represent the attributes of a relationship set. Attributes that are part of the primary key are underlined.
  - Lines link entity sets to relationship sets.
  - Dashed lines link attributes of a relationship set to the relationship set.
  - Double lines indicate total participation of an entity in a relationship set.
  - Double diamonds represent identifying relationship sets linked to weak entity sets

Entity-Relationship Diagrams
Basic Structure

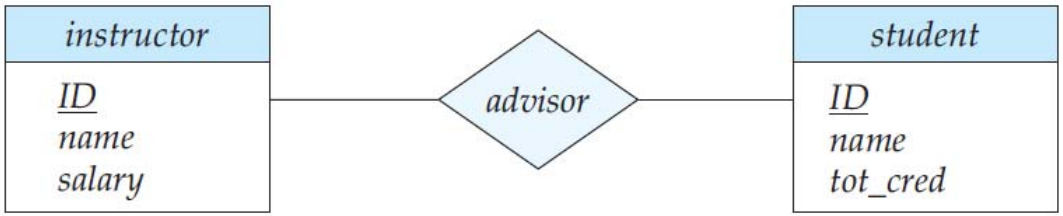


Figure 7.7 E-R diagram corresponding to instructors and students.

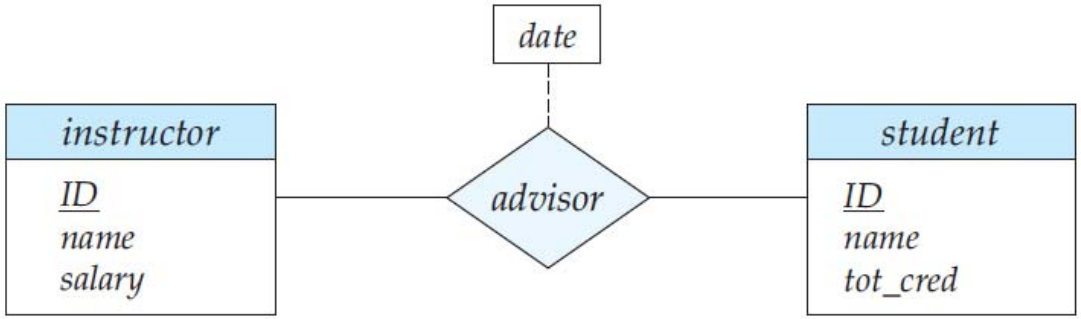


Figure 7.8 E-R diagram with an attribute attached to a relationship set.

Entity-Relationship Diagrams
Mapping Cardinality

- 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:
    - We draw a directed line from the relationship set *advisor* to both entity sets *instructor* and *student* .
    - This indicates that an instructor may advise at most one student, and a student may have at most one advisor.
  - One-to-many:
    - We draw a directed line from the relationship set *advisor* to the entity set *instructor* and an undirected line to the entity set *student*.
    - This indicates that an instructor may advise many students, but a student may have at most one advisor.
  - Many-to-one:
    - We draw an undirected line from the relationship set *advisor* to the entity set *instructor* and a directed line to the entity set *student*.
    - This indicates that an instructor may advise at most one student, but a student may have many advisors.
  - Many-to-many:
    - We draw an undirected line from the relationship set *advisor* to both entity sets *instructor* and *student* .
    - This indicates that an instructor may advise many students, and a student may have many advisors.

Entity–Relationship Diagrams
Mapping Cardinality

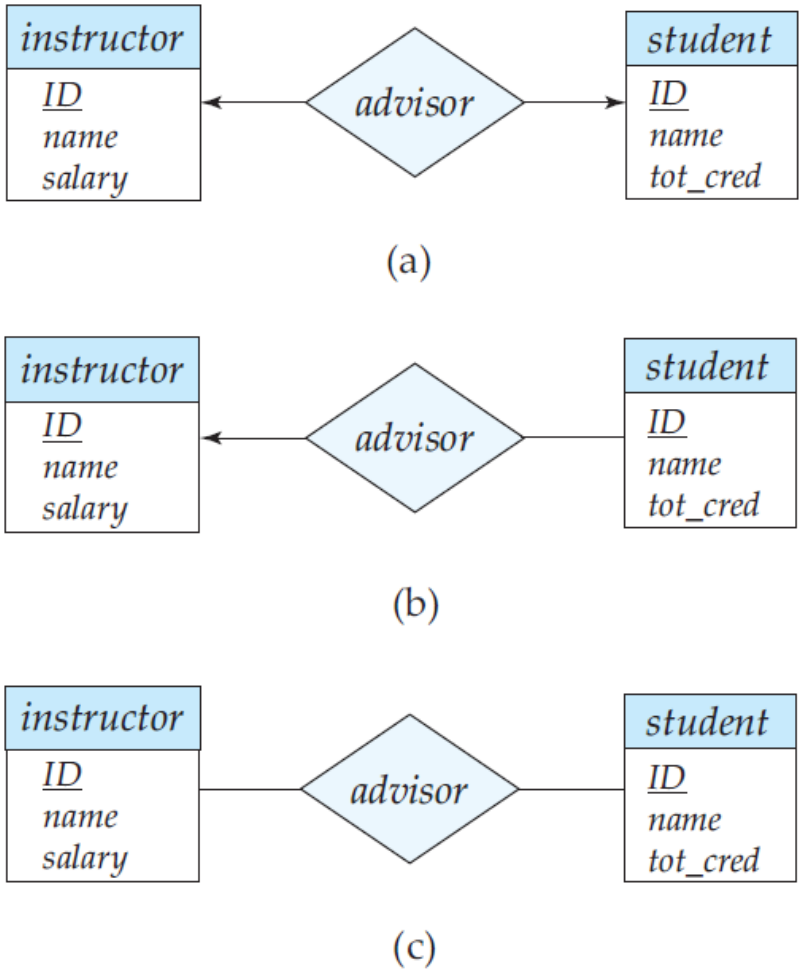
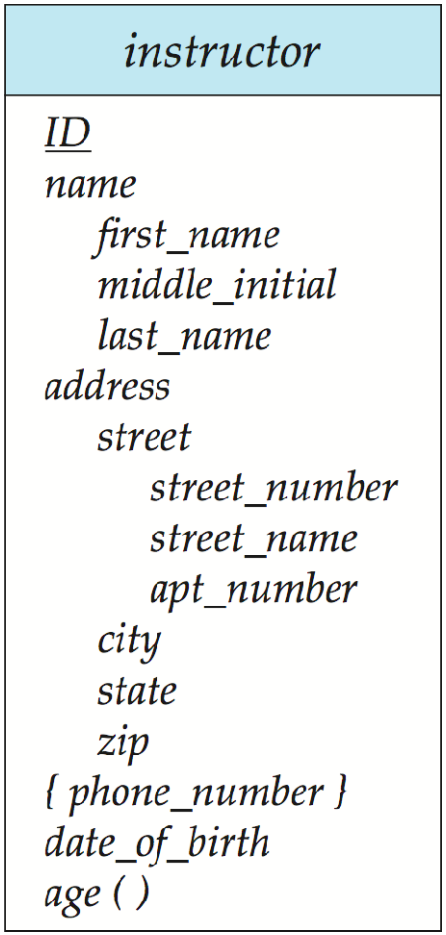


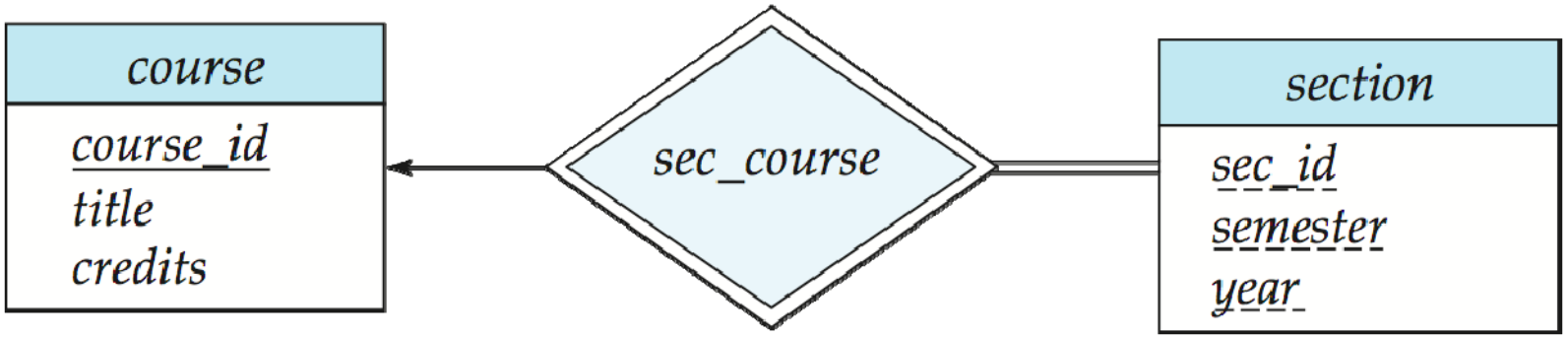
Figure 7.9 Relationships. (a) One-to-one. (b) One-to-many. (c) Many-to-many.

Entity With Composite, Multivalued, and Derived Attributes



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



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

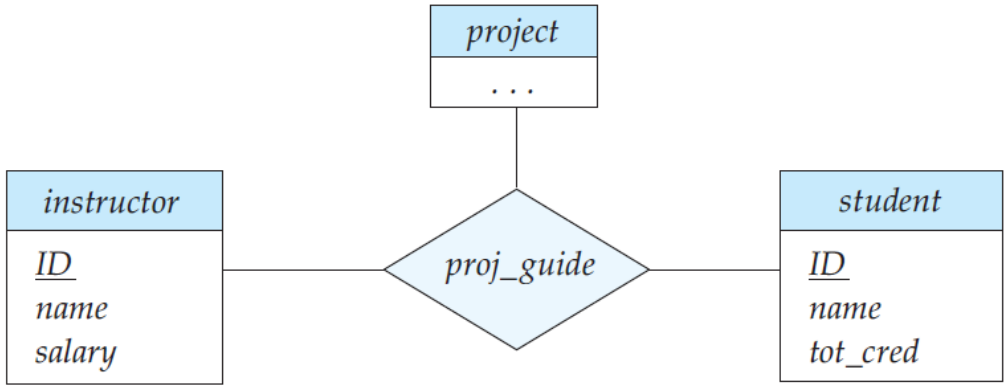
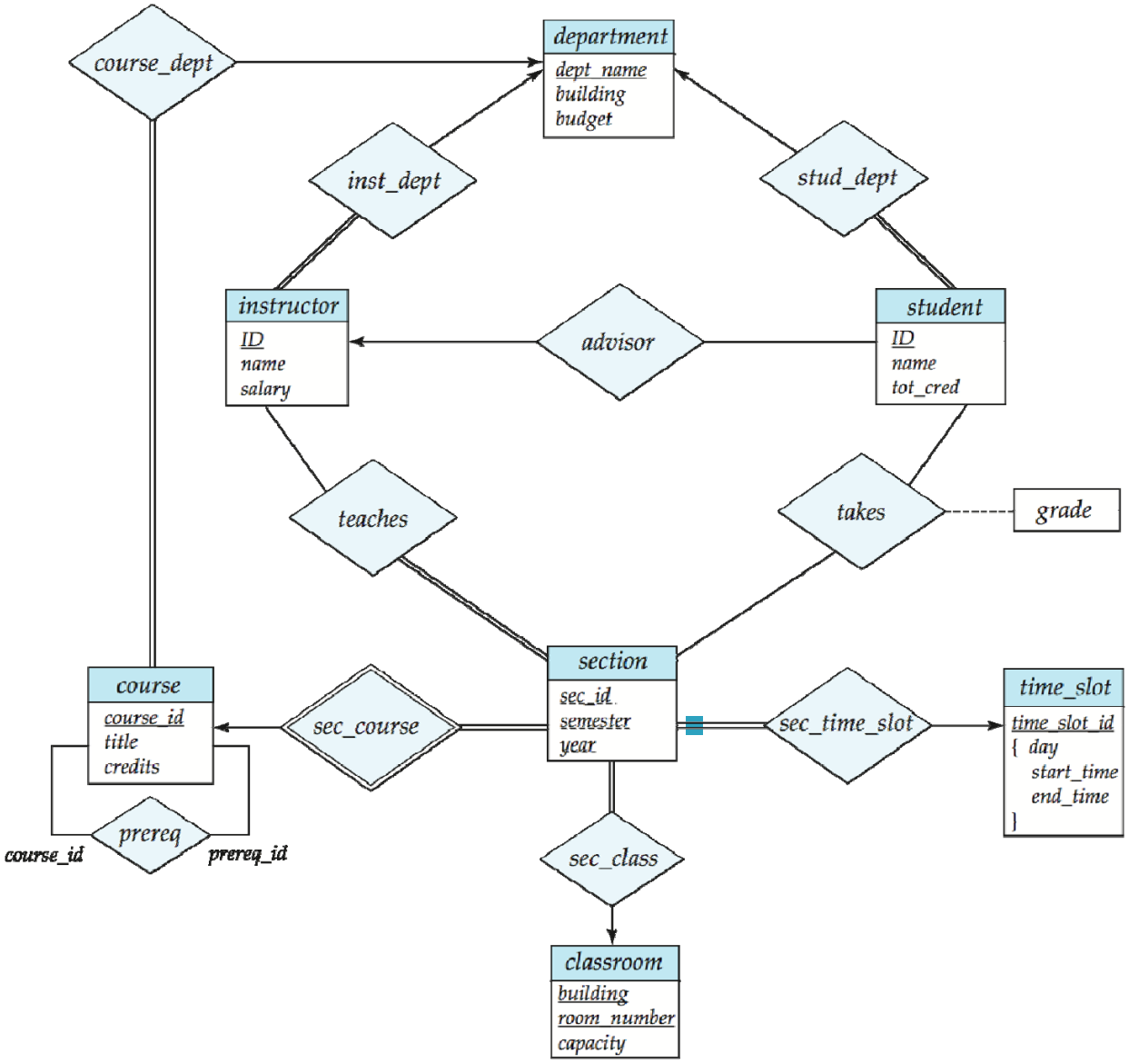


Figure 7.13 E-R diagram with a ternary relationship.

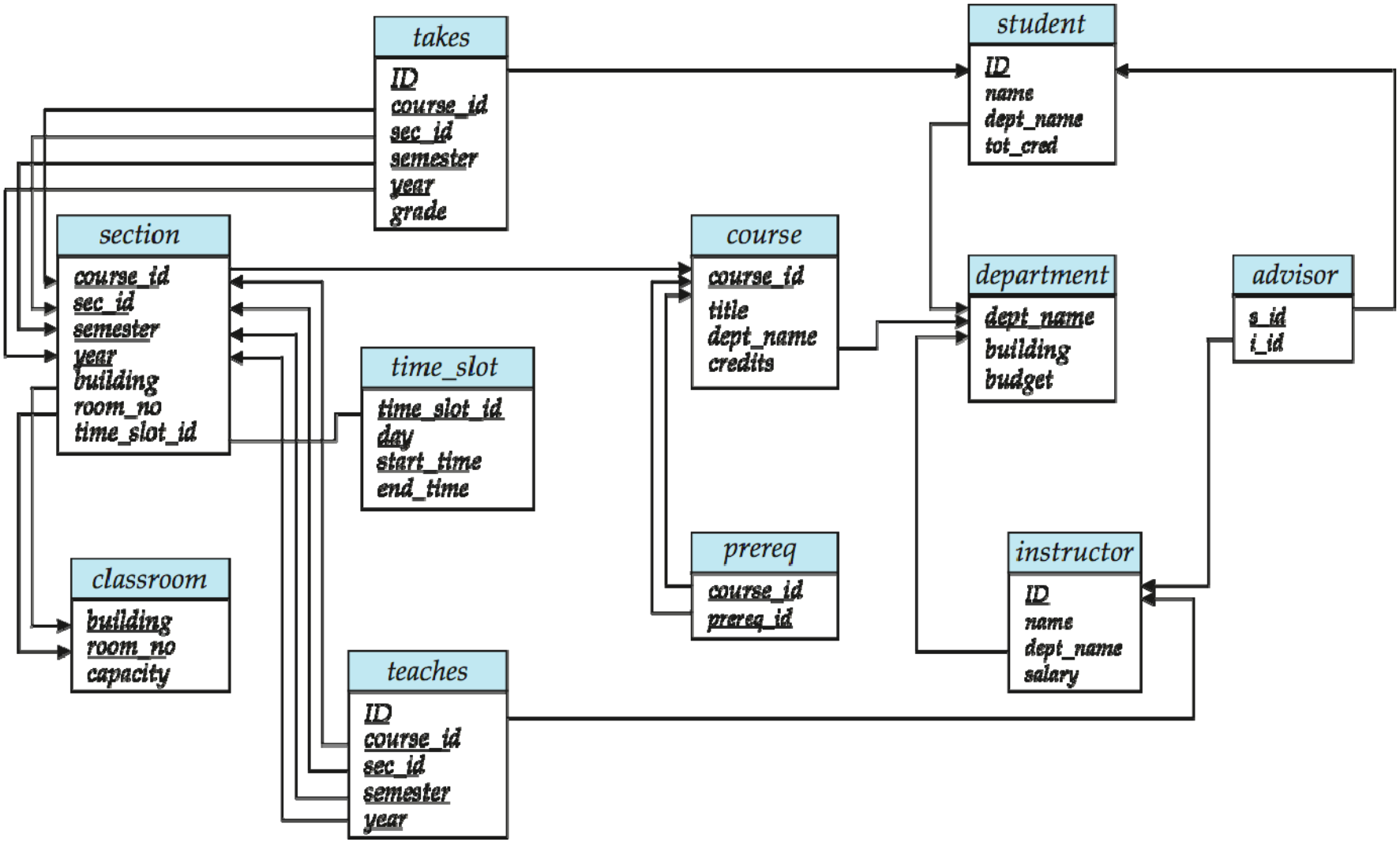
Weak Entity Sets

- 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.

E-R Diagram for a University Enterprise



Reduction to Relation Schemas

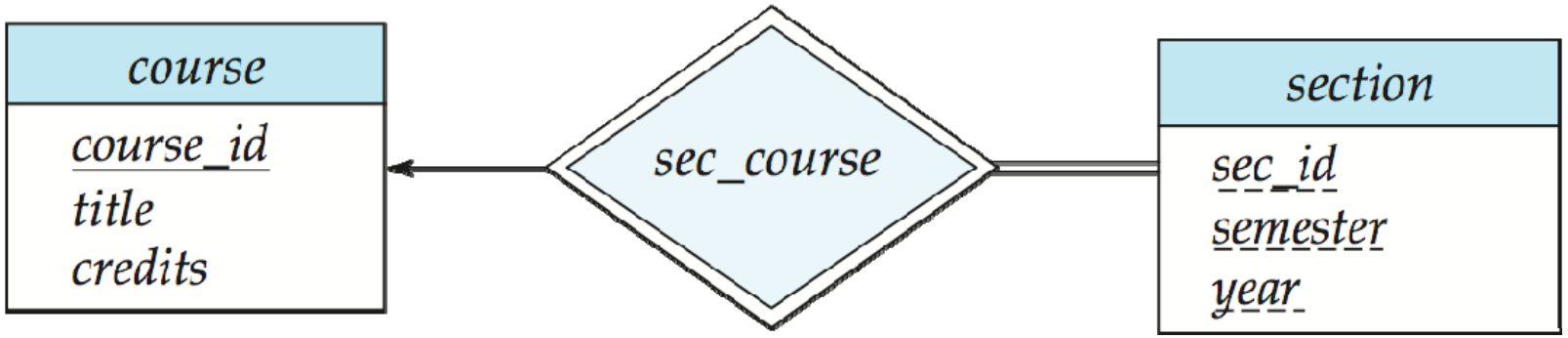


Reduction to Relation Schemas

- 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

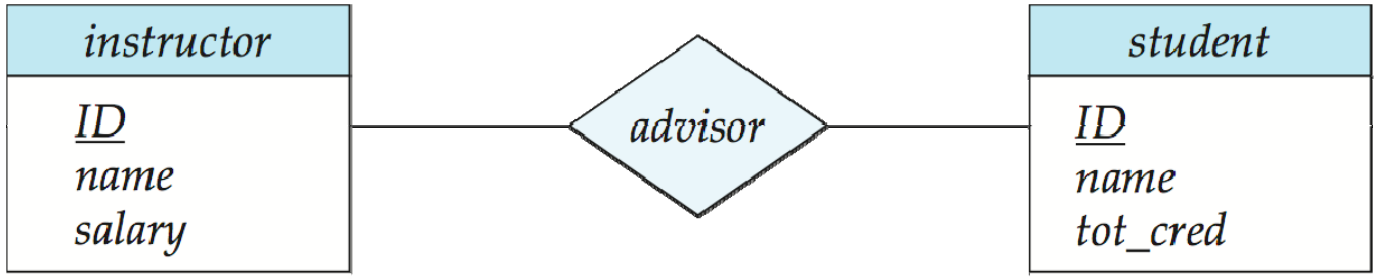
- A strong entity set reduces to a schema with the same attributes
  student(ID, name, tot_cred)
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
  section (course_id, sec_id, sem, year)



Representing Relationship Sets

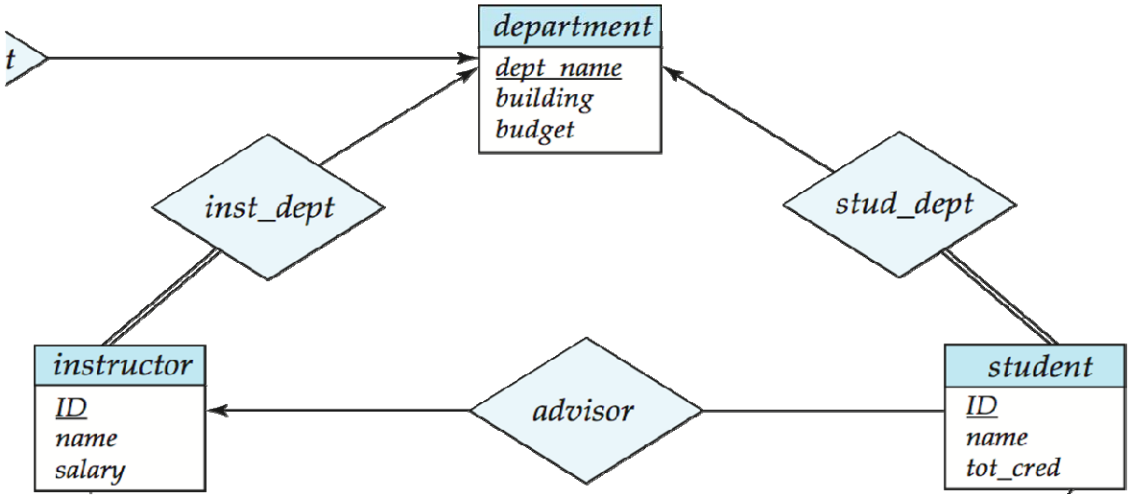
- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

  advisor = (*s_id, i_id*)



Representing Relationship Sets

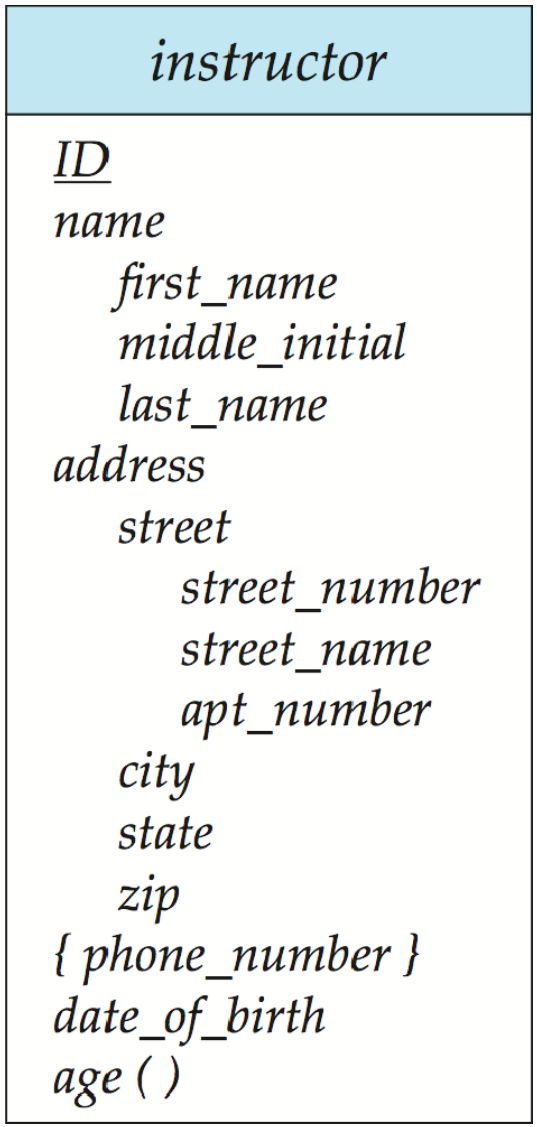
- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
- Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the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



Representing Relationship Sets

- 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

Composite and Multivalued Attributes



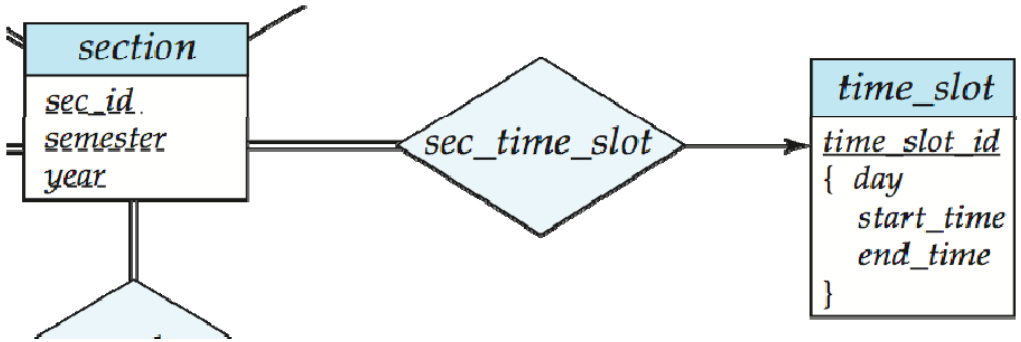
- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* 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*)
  - Caveat: *time_slot* attribute of *section* (from *sec_time_slot*) cannot be a foreign key due to this optimization



Normalization

- Optimum database structure is achieved by eliminating redundancies, inefficiencies, update and deletion anomalies
- Data normalization is a progressive process
- The steps in the normalization process are called normal forms
- Each normal form progressively improves the database design and makes it more efficient

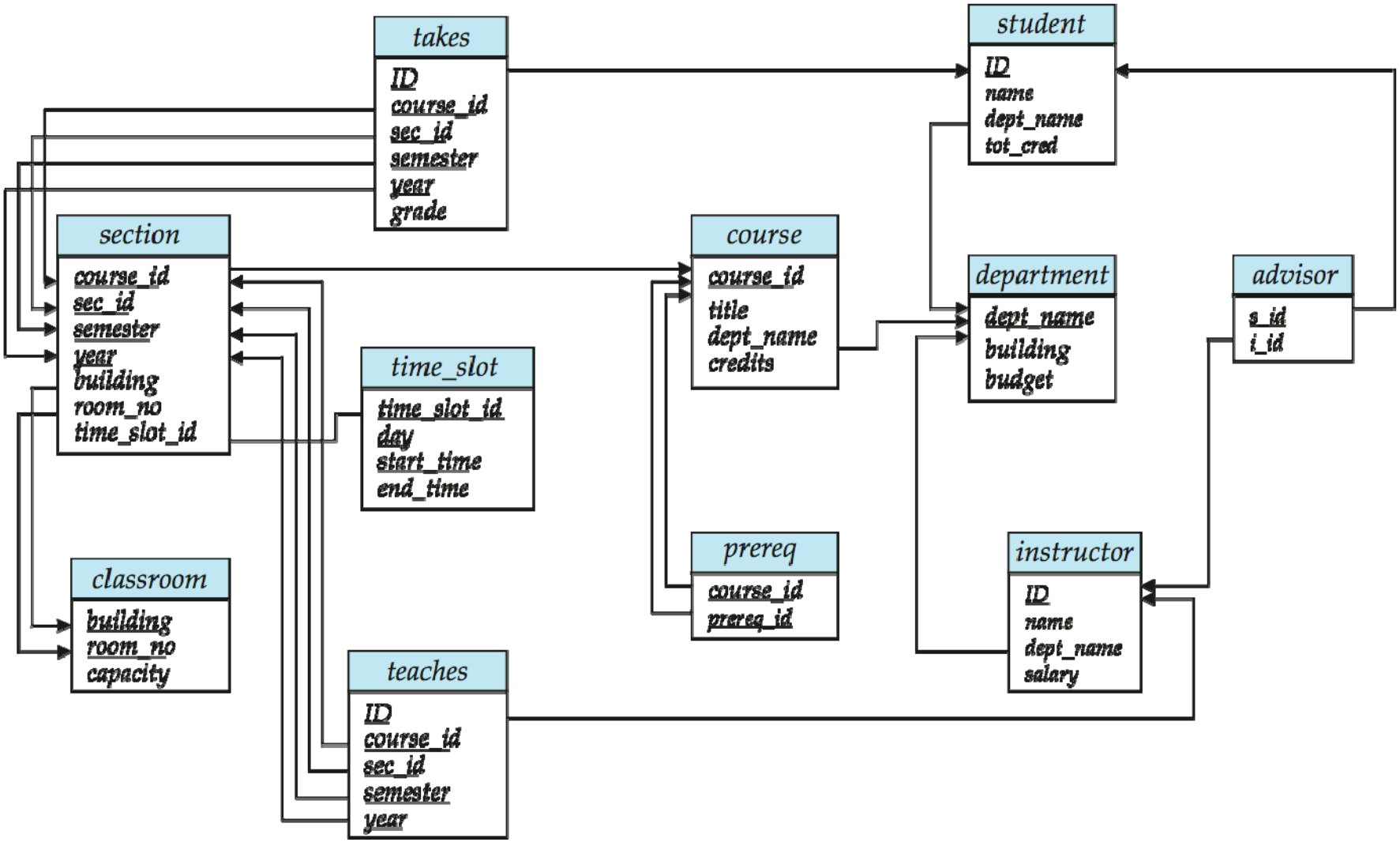
Normal Forms

- *First normal form (1NF) implies*
  - that the values in each column of a table are atomic, that is there are no repeating groups of data

- *Second normal form (2NF) implies*
  - that the table is already in 1NF and every non-key column is fully dependent upon the primary key

- *Third normal (3NF) form implies*
  - that the table is already in 2NF and every non-key column is non- transitively dependent upon the primary key. In other words all non-key columns are mutually independent, but at the same time they are fully dependent upon the primary key only

Schema Diagram for University Database



Schema Diagram for University Database

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

Exercise – I

Consider a MOVIE database in which data is recorded about the movie industry. The data requirements are summarized as follows:

- Each movie is identified by title and year of release. Each movie has a length in minutes. Each has a production company, and each is classified under one or more genres (such as horror, action, drama, and so forth). Each movie has one or more directors and one or more actors appear in it. Each movie also has a plot outline. Finally, each movie has zero or more quotable quotes, each of which is spoken by a particular actor appearing in the movie.
- Actors are identified by name and date of birth and appear in one or more movies. Each actor has a role in the movie.
- Directors are also identified by name and date of birth and direct one or more movies. It is possible for a director to act in a movie (including one that he or she may also direct).
- Production companies are identified by name and each has an address. A production company produces one or more movies.

Exercise – II

Design a database for a world-wide package delivery company (e.g., DHL or FedEX). The database must be able to keep track of customers (who ship items) and customers (who receive items); some customers may do both.

Each package must be identifiable and trackable, so the database must be able to store the location of the package and its history of locations. Locations include trucks, planes, airports, and warehouses.