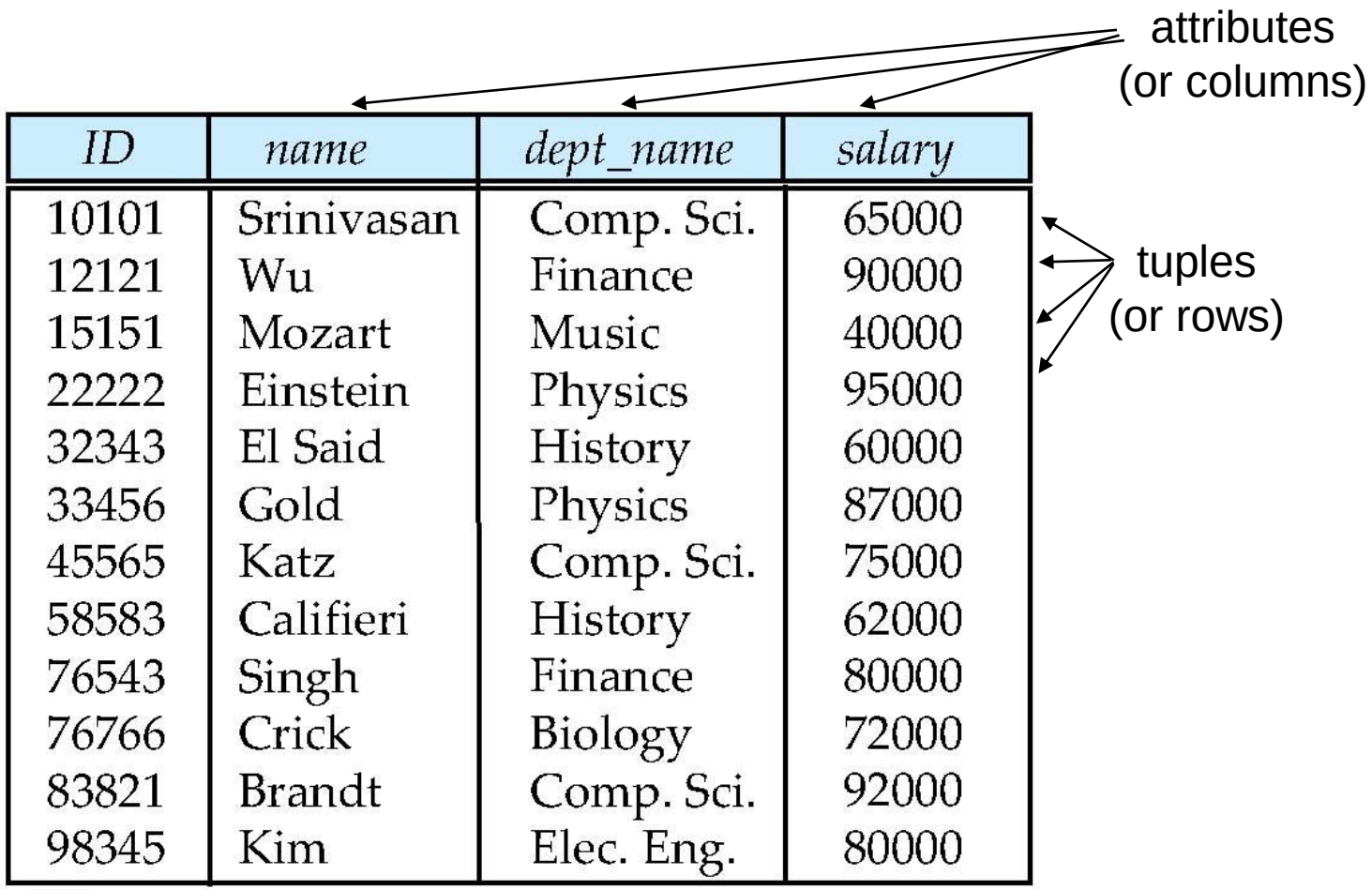
The Relational Model

Introduction

A relational database consists of a **collection of tables**, each of which is assigned a unique name. For example, consider the instructor table of the figure, which stores information about instructors. The table has four column headers: ID, name, dept name, and salary. Each row records information about an instructor, consisting of the instructor's ID, name, dept name, and salary.

In the relational model the term **relation** is used to refer to a **table**, while the term **tuple** is used to refer to a **row**. Similarly, the term **attribute** refers to a **column**.



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

Introduction

- We use the term **relation instance** to refer to a specific instance of a relation, containing a specific set of rows. The instance of instructor shown in Figure has 12 tuples, corresponding to 12 instructors.
- The data type describing the types of values that can appear in each column is represented by a ***domain*** of possible values.
- Thus, for each attribute of a relation, there is a set of permitted values, called the domain of that attribute.

Attribute Types

- Attribute values are required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. It signifies that the value is unknown or does not exist.
- The null value causes complications in the definition of many operations
- Examples of *logical* definitions of domains:
  - *Social_security_numbers* . The set of valid nine-digit Social Security numbers.
  - *Names* : The set of character strings that represent names of persons.
  - *Academic_department_names* . The set of academic department names in a university, such as Computer Science, Economics, and Physics.

Relation Schema and Instance

- When we talk about a database, we must differentiate between the **database schema**, which is the logical design of the database, and the **database instance**, which is a snapshot of the data in the database at a given instant in time.
- In general, a relation schema consists of a list of attributes and their corresponding domains.

Relation Schema and Instance

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

Example:

instructor = (*ID, name, dept_name, salary*)

- The current values (**relation instance**) of a relation are specified by a table
- The degree (or arity) of a relation is the number of attributes of its relation schema.

– A relation of degree six, which stores information about university students, would contain six attributes describing each student, as follows:

STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age)

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

Database

- A relational database consists of multiple relations
- Information about an enterprise is broken up into parts

  instructor
  student
  advisor

- Bad design:

  univ (*instructor -ID, name, dept_name, salary, student_Id*, ...)

  results in repetition of information (e.g., two students have the same instructor)

  - the need for null values (e.g., represent an student with no advisor)

Keys

- We must have a way to specify how tuples within a given relation are distinguished. This is expressed in terms of their attributes. No two tuples in a relation are allowed to have exactly the same value for all attributes.
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a **candidate key** if *K* is minimal

  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - which one? The designer of the database must choose it.

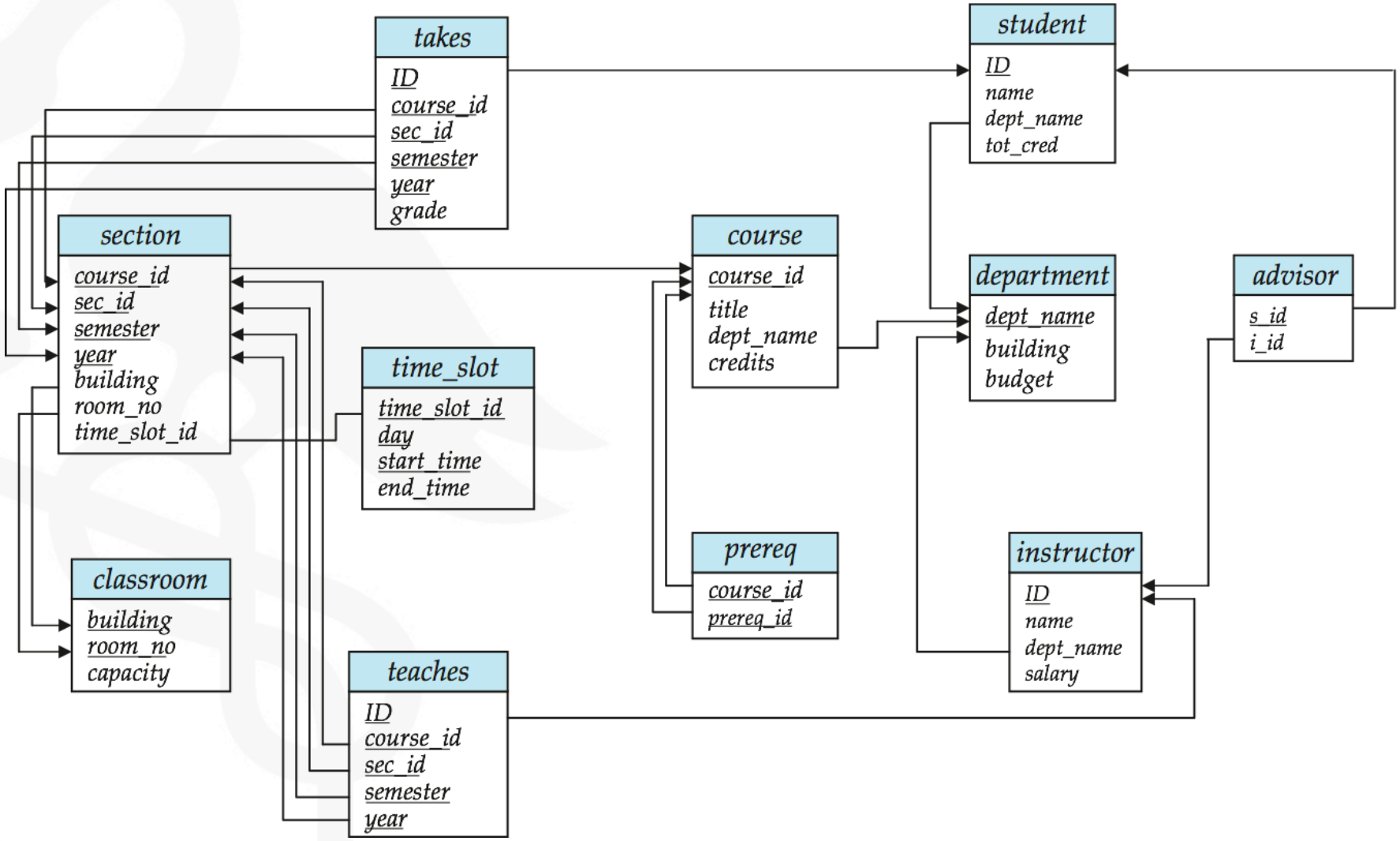
Keys

- A relation, say r1, may include among its attributes the primary key of another relation, say r2. This attribute is called a **foreign key** from r1, referencing r2. The relation r1 is also called the **referencing relation** of the foreign key dependency, and r2 is called the **referenced relation** of the foreign key

- **Foreign key** constraint: Value in one relation must appear in another

Schema Diagram for University Database

Each relation appears as a box, with the relation name at the top in blue, and the attributes listed inside the box. Primary key attributes are shown underlined. Foreign key dependencies appear as arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation.



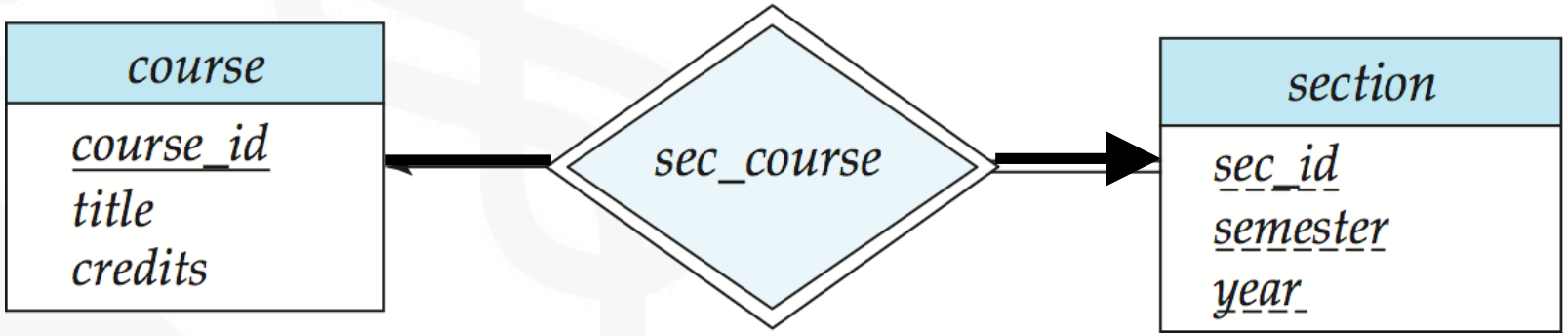
Reduction of ER Models to Relational Schemas

Reduction to Relation Schemas

- A database modeled by an E-R diagram can be represented by a collection of schemas.
- Entity sets and relationship sets can be expressed as *relation schemas* that represent the contents of the database.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

Representing Entity Sets With Simple Attributes

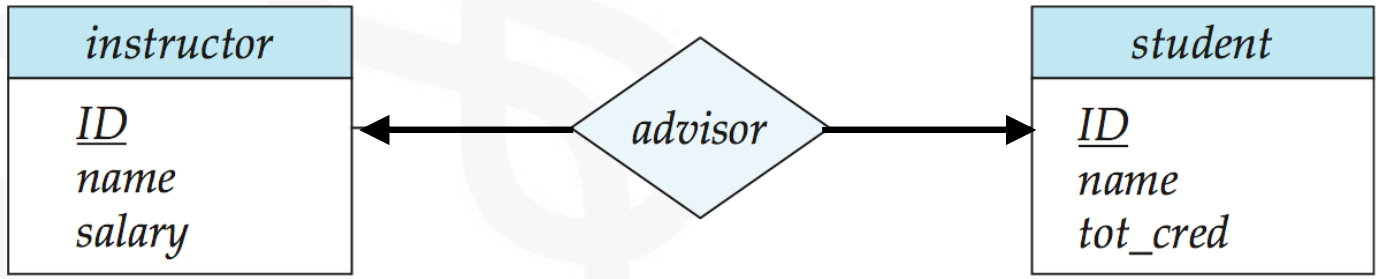
- A strong entity set reduces to a schema with the same attributes
  - *student(ID, name, tot_cred)*
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
  - *section (course_id, sec_id, sem, year)*



Representing Relationship Sets

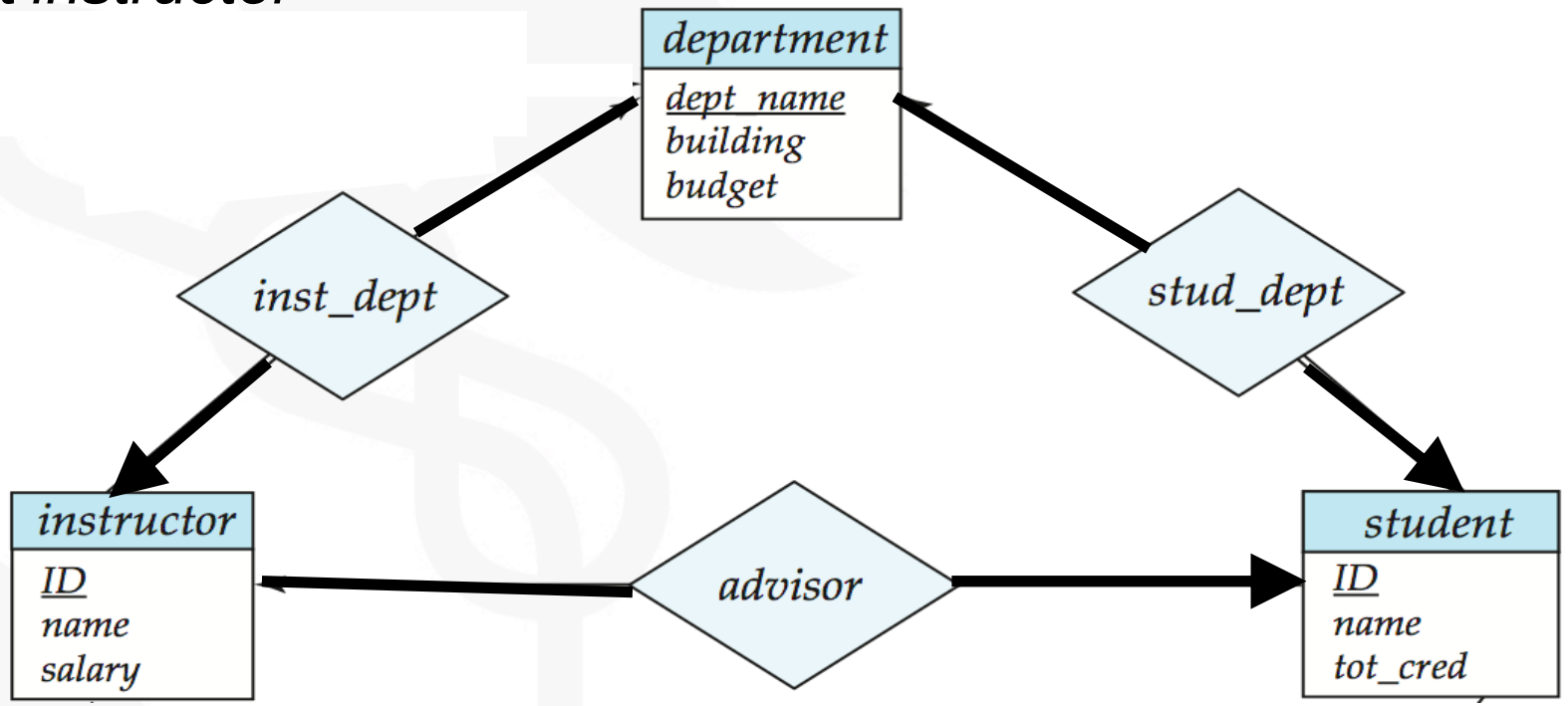
- A **many-to-many** relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

  advisor = (*s_id, i_id*)



Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the one-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*



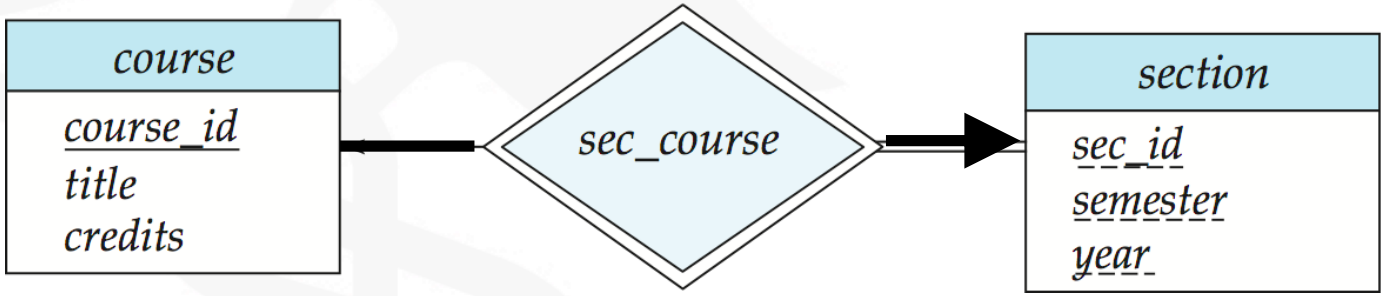
Redundancy of Schemas (Cont.)

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

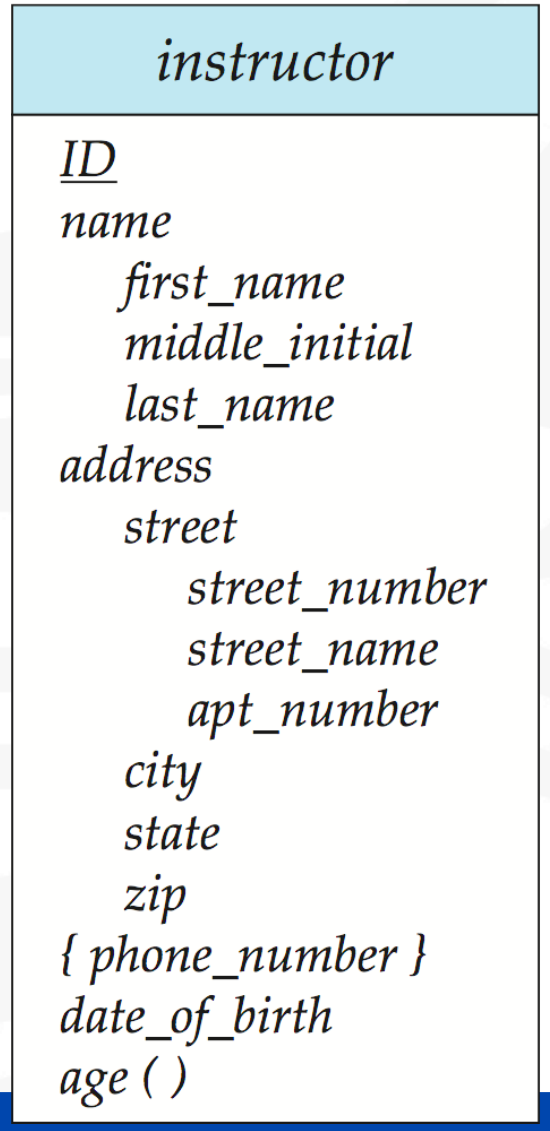
Relation 1	Relation 2	Action
(0, 1)	(0, 1)	Create a new relation
(0, 1)	(1, 1)	Propagation of PK from R2 to R1
(1, 1)	(1, 1)	Indifferent propagation

Redundancy of Schemas (Cont.)

- 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 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, middle_name* and *last_name,* the schema corresponding to the entity set has 3 attributes *first_name, middle_name* and *last_name*
- Ignoring multivalued and derived 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)*

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)

Generalization/Specialization relationships

- If specialization is not complete or it is not disjoint:
  - One table for the higher-level entity and
  - One table for each lower-level entity with its atributes plus the PK of the higher-level entity.
- Example: person, employee, client.
  - Person (SSN, name, address)
  - Employee (SSN, salary)
  - Client (SSN, email)

Generalization/Specialization relationships

- If specialization is complete and disjoint:
  - The table of the higher-level entity is not created.
  - One table for each lower-level entity with its atributes plus the atributes of the higher-level entity.
- Example: bankAccount, savingAccount, checkingAccount.
  - savingAccount (number, interestRate)
  - checkingAccount (number, overdrafts)