Entity Relationship Model

Database System Concepts, 7th Ed.

Design Phases

- Initial phase -- characterize fully the data needs of the prospective database users.
- Second phase -- choosing a data model
  - Applying the concepts of the chosen data model
  - Translating these requirements into a conceptual schema of the database.
  - A fully developed conceptual schema indicates the functional requirements of the enterprise.
    - Describe the kinds of operations (or transactions) that will be performed on the data.

Design Phases (Cont.)

- Final Phase -- Moving from an abstract data model to the implementation of the database
  - 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
  - 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*

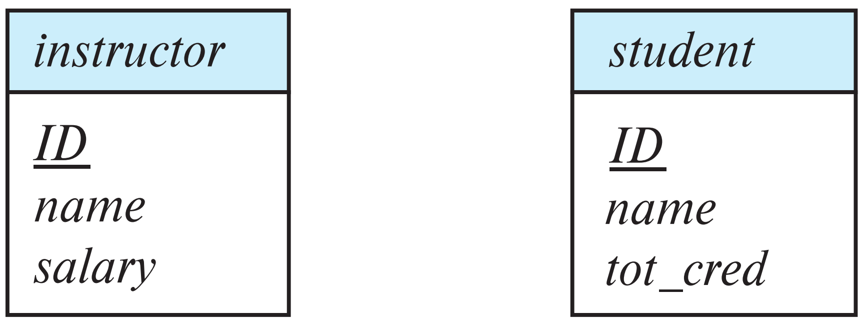
Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of attributes; i.e., descriptive properties possessed by all members of an entity set.
  - Example:

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

Representing Entity sets in ER Diagram

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



Weak Entity Sets

- Consider a *section* entity, which is uniquely identified by a *course_id*, *semester*, *year*, and *sec_id*.
- Clearly, section entities are related to course entities. Suppose we create a relationship set *sec_course* between entity sets *section* and *course*.
- Note that the information in *sec_course* is redundant, since *section* already has an attribute *course_id*, which identifies the course with which the section is related.
- One option to deal with this redundancy is to get rid of the relationship *sec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

Weak Entity Sets (Cont.)

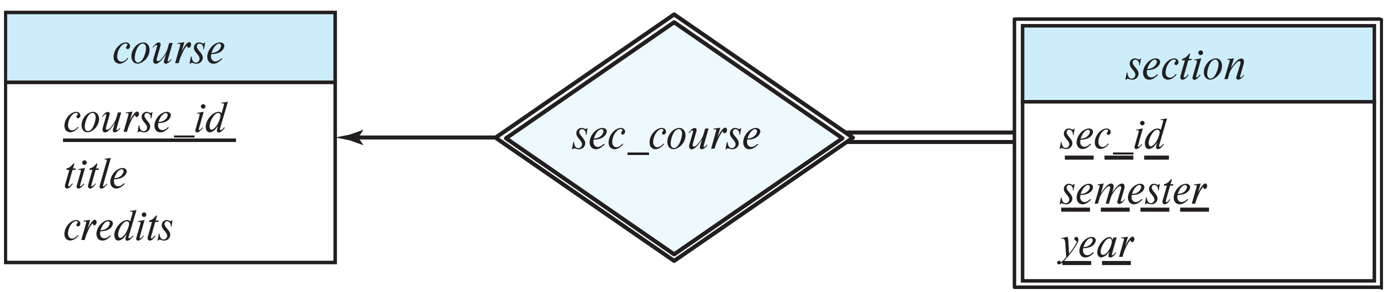
- An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes *section_id*, *year*, and *semester*.
  - However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely
- To deal with this problem, we treat the relationship *sec_course* as a special relationship that provides extra information, in this case, the *course_id*, required to identify *section* entities uniquely.
- A **weak entity set** is one whose existence is dependent on another entity, called its **identifying entity**
- Instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called **discriminator** to uniquely identify a weak entity.

Weak Entity Sets (Cont.)

- An entity set that is not a weak entity set is termed a **strong entity set**.
- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set.
- The identifying entity set is said to **own** the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.

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*)



Relationship Sets

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

  Example:

44553 (Priti)	*advisor*	22222 (Mukesh)
student entity	relationship set	*instructor* entity

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

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

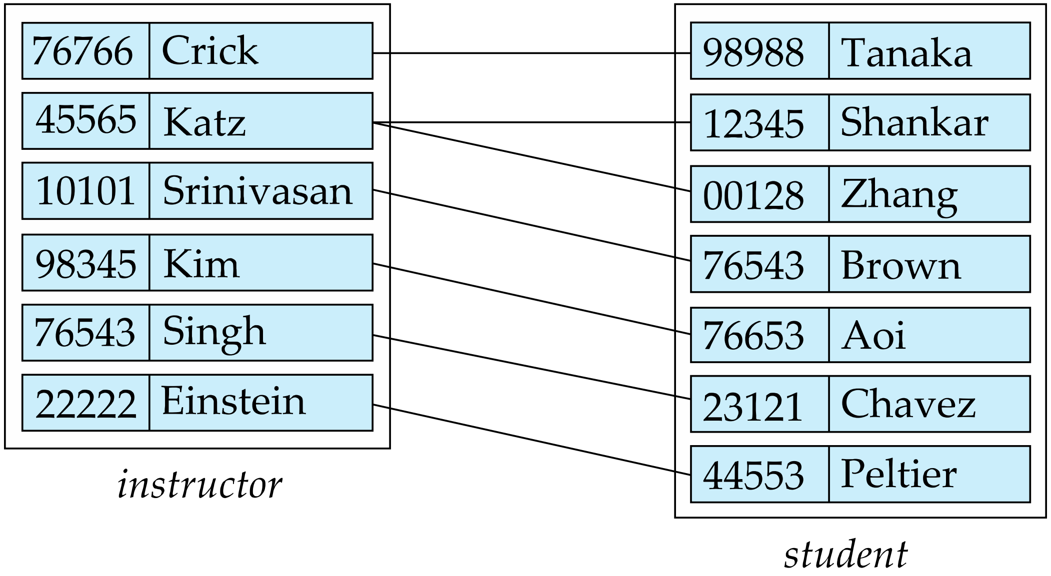
  where $(e_1, e_2, \ldots, e_n)$ is a relationship

  - Example:

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

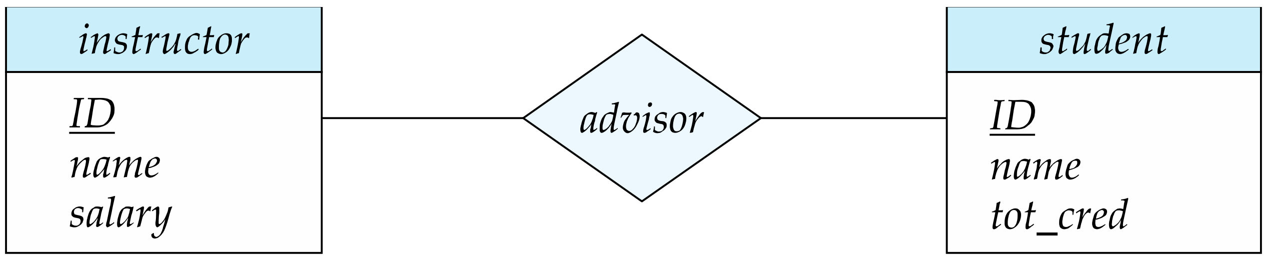
Relationship Sets (Cont.)

- Example: we define the relationship set *advisor* to denote the associations between students and the instructors who act as their advisors.
- Pictorially, we draw a line between related entities.



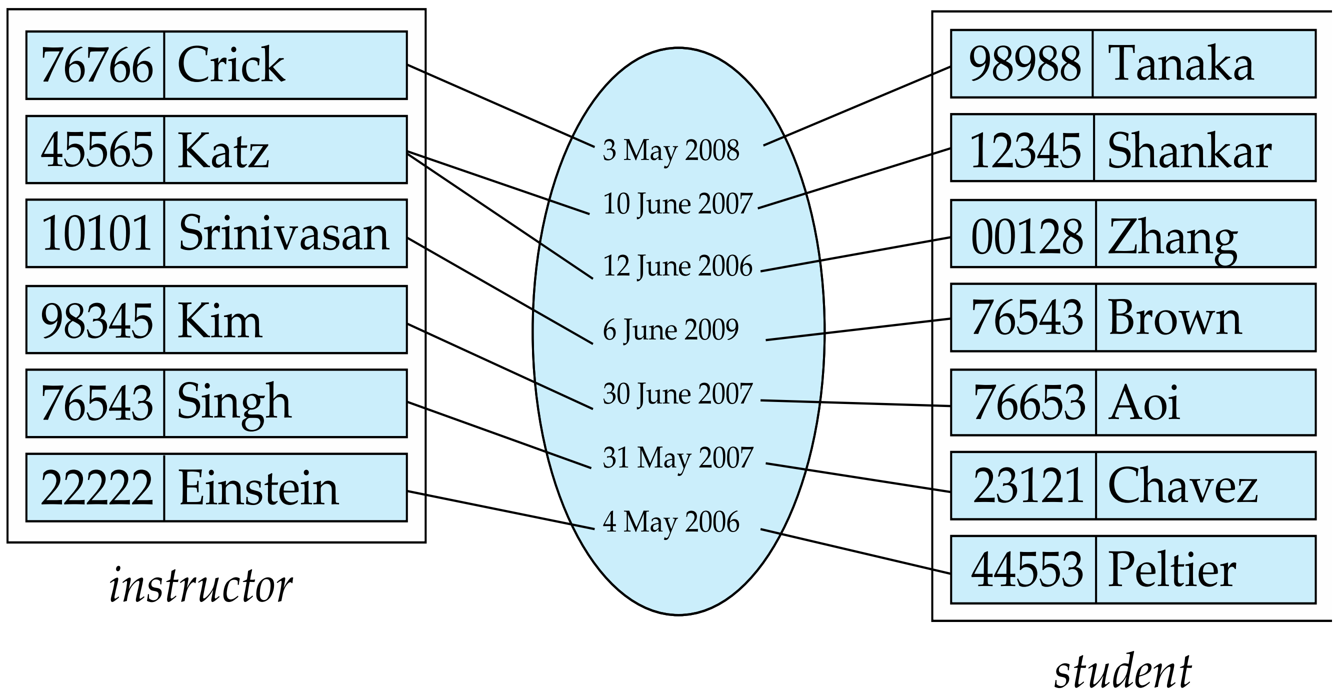
Representing Relationship Sets via ER Diagrams

- Diamond shape represent relationship sets.

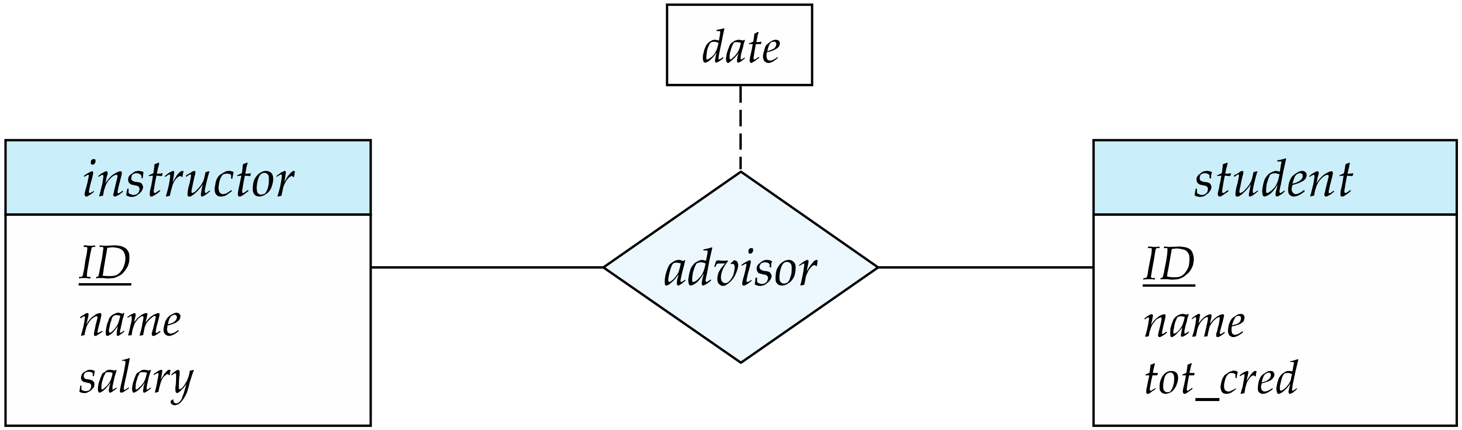


Relationship Sets (Cont.)

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

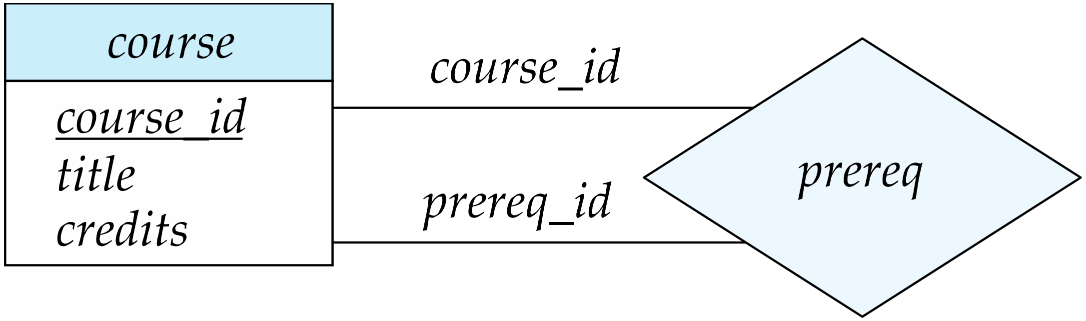


Relationship Sets with Attributes



Roles

- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a "role" in the relationship
- The labels "*course_id*" and "*prereq_id*" are called **roles**.

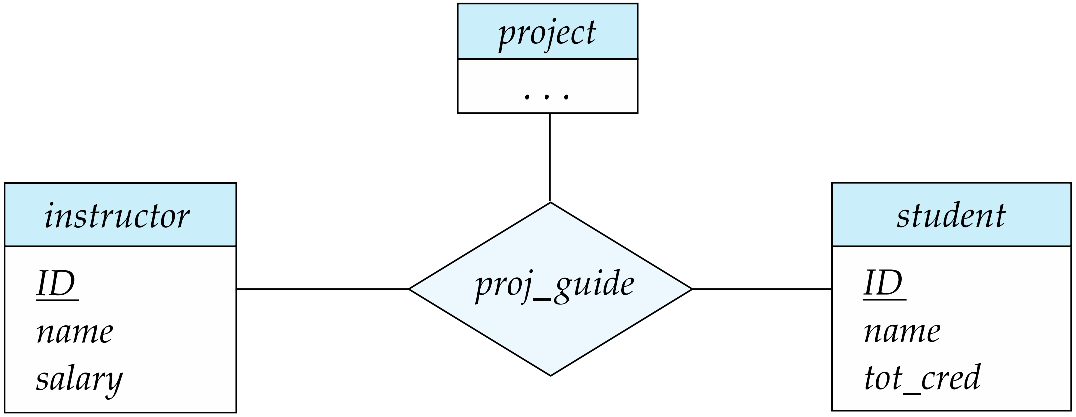


Degree of a Relationship Set

- Binary relationship
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary.
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor*, *student*, and *project*

Non-binary Relationship Sets

- There are occasions when it is more convenient to represent relationships as non-binary.
- E-R Diagram with a Ternary Relationship

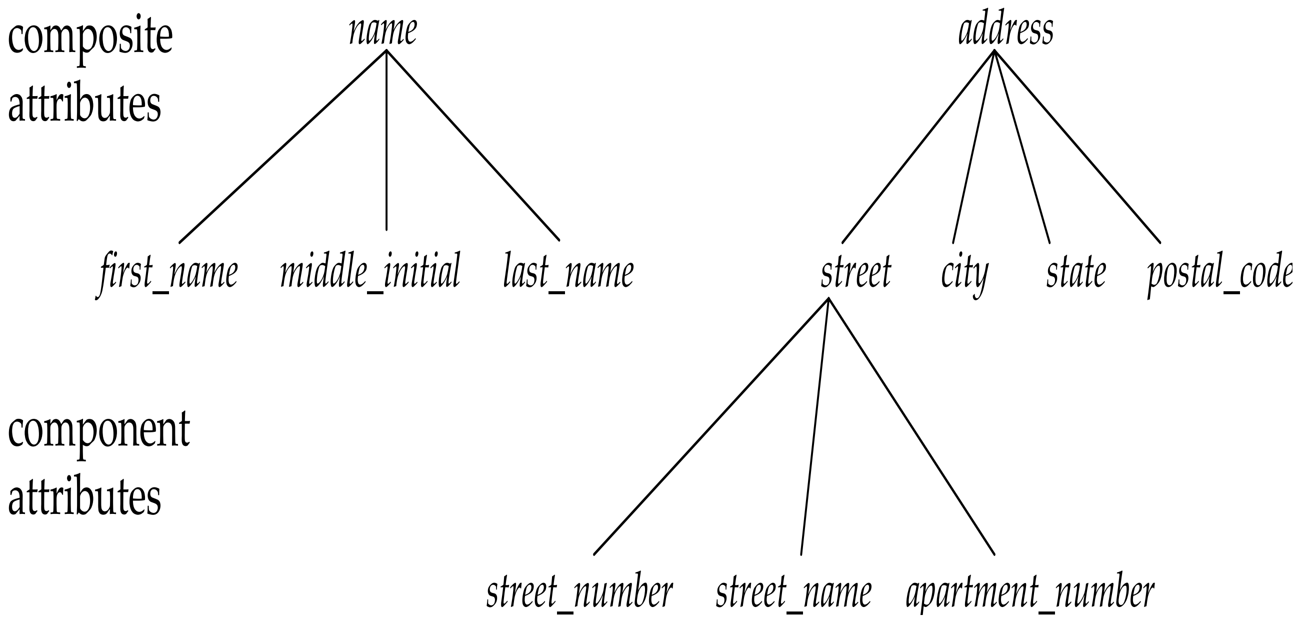


Complex Attributes

- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth
- **Domain** – the set of permitted values for each attribute

Composite Attributes

- Composite attributes allow us to divided attributes into subparts (other attributes).



Representing Complex Attributes in ER Diagram

1.Simple Attribute : Stored as atomic values in a single column

2.Composite Attribute: Stored as separate columns for each component or as a structured data type

3.Multi-Valued: Stored in a separate table with foreign key reference to the main enti

4.Derived Attribute: Not stored directly; calculated at query time

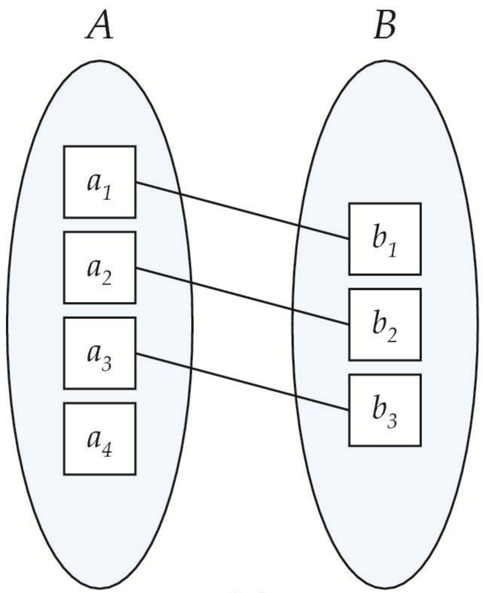
5.Complex Attribute(Combination of composite and multi-valued attributes): Requires nested tables

instructor
<u>*ID*</u> *name*   *first_name*   *middle_initial*   *last_name* *address*   *street*   *street_number*   *street_name*   *apt_number*   *city*   *state*   *zip* *{ phone_number }* *date_of_birth* *age ( )*

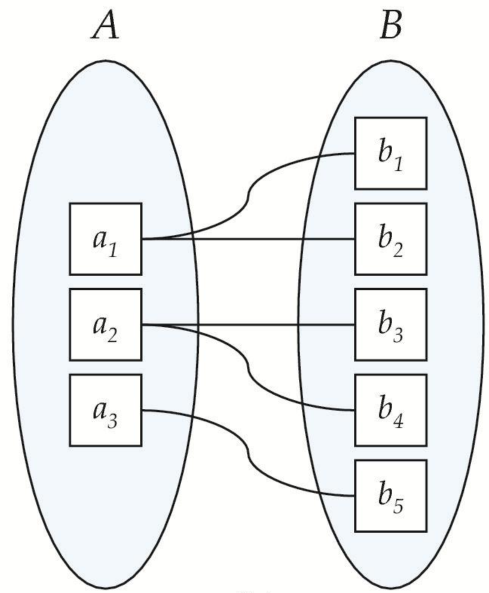
Mapping Cardinality Constraints

- 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



(a)



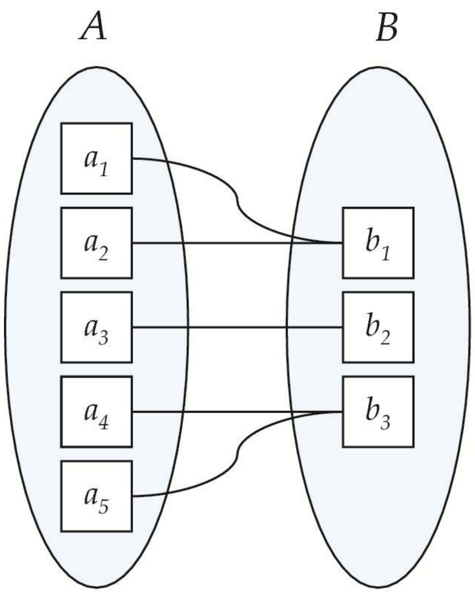
(b)

One to one

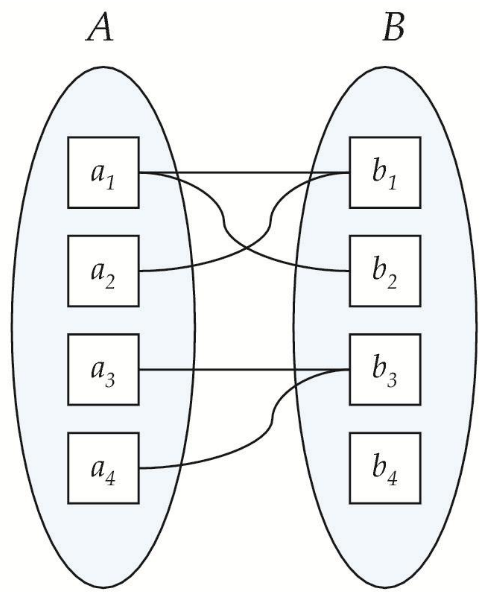
One to many

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

Mapping Cardinalities



(a)



(b)

Many to one

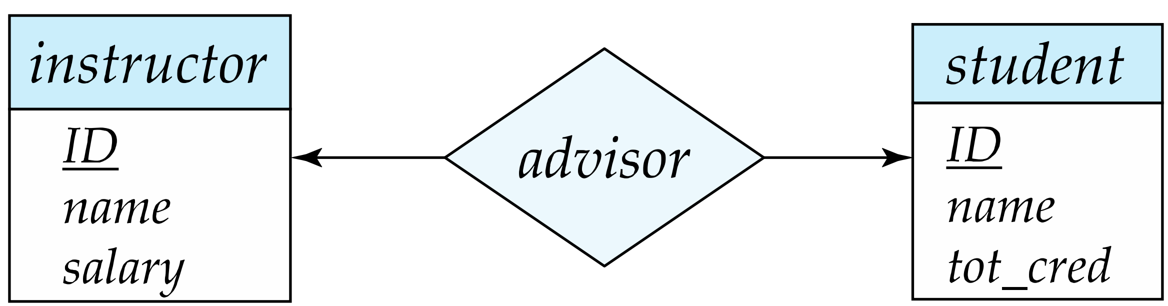
Many to many

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

Representing Cardinality Constraints in ER Diagram

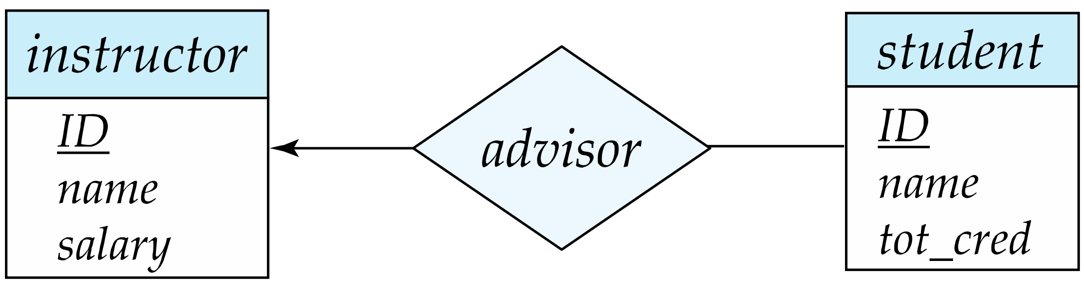
- We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.

- One-to-one relationship between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*



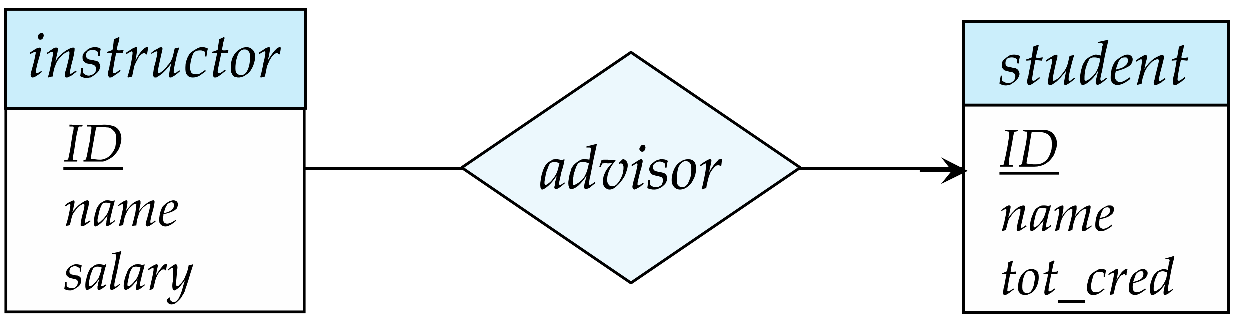
One-to-Many Relationship

- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,



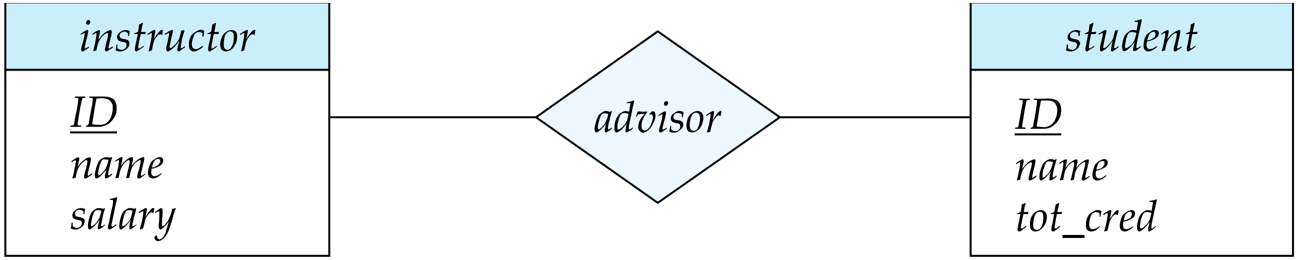
Many-to-One Relationships

- In a many-to-one relationship between an *instructor* and a *student*,
  - an instructor is associated with at most one student via *advisor*,
  - and a student is associated with several (including 0) instructors via *advisor*



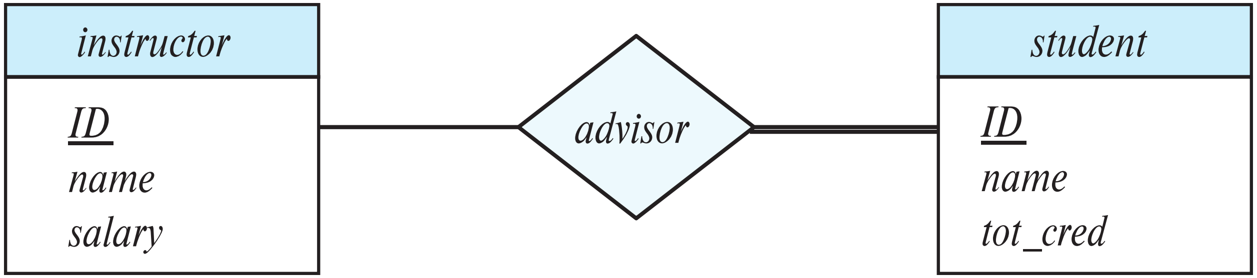
Many-to-Many Relationship

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



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 r*elation is total

  - every *student* must have an associated instructor

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

Keys

- Used to specify how entities within a given entity set are distinguished
- In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes
- Keys are applicable to relation schemas also **to identify relationships uniquely**, and thus distinguish relationships from each other

  Types:

  - Super key
  - Candidate key
  - Primary key
  - Alternate Key
  - Foreign key

Keys

- Let K ⊆ R
- *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**. Others are said to be **Alternate key.**
  - {ID} → Primary Key
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - Example – *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*

Keys

- Used to uniquely identify any record or row of data from the table.
- Used to establish and identify relationships between tables.
- **For example:**
  - Student table, ID is used as a key because it is unique for each student.
  - PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

STUDENT
ID
Name
Address
Course

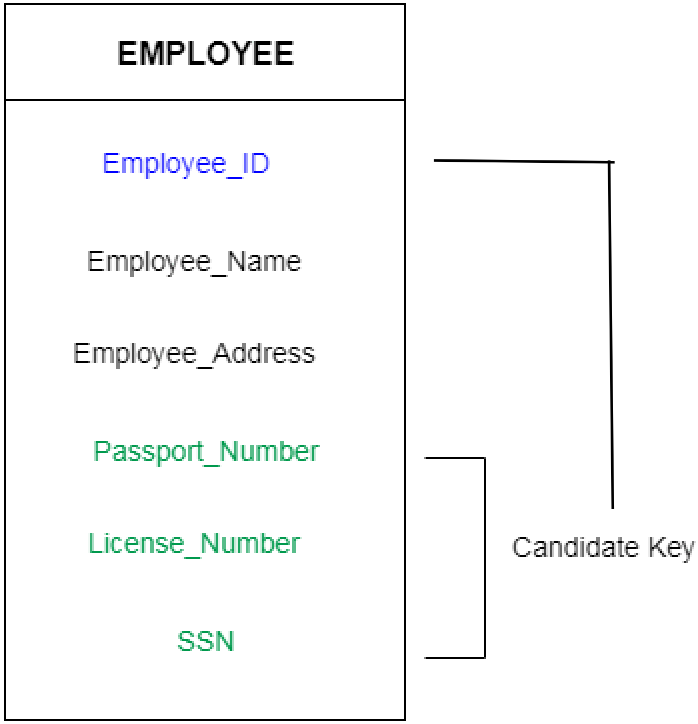
PERSON
Name
DOB
Passport_Number
License_Number
SSN

Super Key

- A set of an attribute which can uniquely identify a tuple.
- Super key is a superset of a candidate key.
- EMPLOYEE table,
  - (EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.
  - The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

Candidate Key

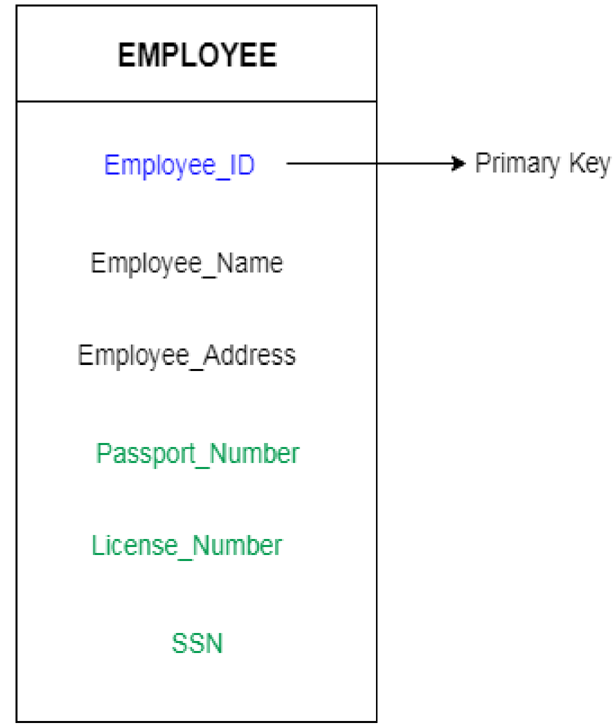
- A candidate key is an attribute or set of an attribute which can uniquely and minimally identify a tuple.
- Thus, from the number of obtained candidate keys, we can identify the appropriate primary key.
- E.g. EMPLOYEE table:
  - {employee_id}, {Pass_no}, {L_num},{SSN}



Primary Key

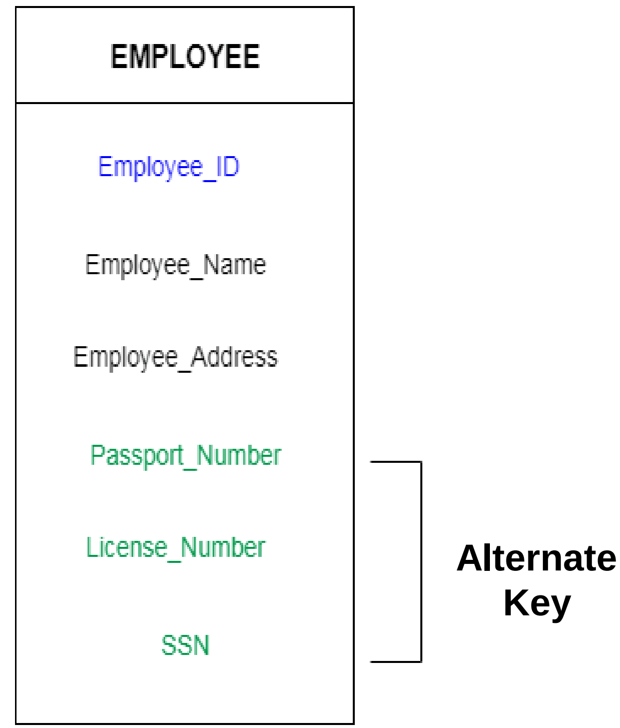
- The first key which is used to identify one and only one instance of an entity uniquely.
- An entity can contain multiple keys
- The key which is most suitable from those lists become a primary key.
- E.g. EMPLOYEE table, ID can be primary key since it is unique for each employee.

  License_Number and Passport_Number as primary key since they are also unique.
- For each entity, selection of the primary key is based on requirement and developers.



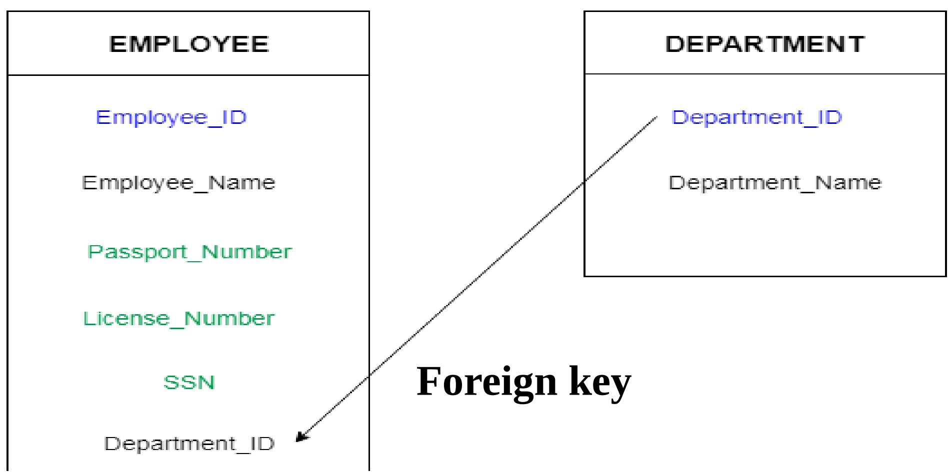
Alternate Key

- The alternative key is the key in the database table which is not selected as a primary key.
- It's an additional key similar to the primary key with which we can access data in the table.
- It can uniquely identify attributes of each element in the row of the table.
- But, because there can be multiple candidate keys all cannot be selected. So, the remaining keys which are not selected as primary keys are called alternative keys.
- E.g. EMPLOYEE table:
  - {employee_id}, {Pass_no}, {L_num},{SSN}
  - employee_id can be primary key since it is unique for each employee.
  - {Pass_no}, {L_num},{SSN} they are also unique but not selected as primary key
  - {Pass_no}, {L_num},{SSN} are Alternate key



Foreign Key

- The column of the table which is used to point to the primary key of another table.
  - E.g. In a company, every employee works in a specific department, and employee and department are two different entities.
  - The information of the Employee and Department are in the employee table.
  - Link these two tables through the primary key of one table.
- Steps:
  - Add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table.
  - Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



Student Table

Student_Number	Student_Name	Student_Phone	Subject_Number
1	Andrew	6615927284	10
2	Sara	6583654865	20
3	Harry	4647567463	10

- **The Super Keys** –

  {Student_Number} {Student_Phone} {Student_Number,Student_Name}
  {Student_Number,Student_Phone} {Student_Number,Subject_Number}
  {Student_Phone,Student_Name} {Student_Phone,Subject_Number}
  {Student_Number,Student_Name,Student_Phone}
  {Student_Number,Student_Phone,Subject_Number}
  {Student_Number,Student_Name,Subject_Number}
  {Student_Phone,Student_Name,Subject_Number}

- **The Candidate Key**

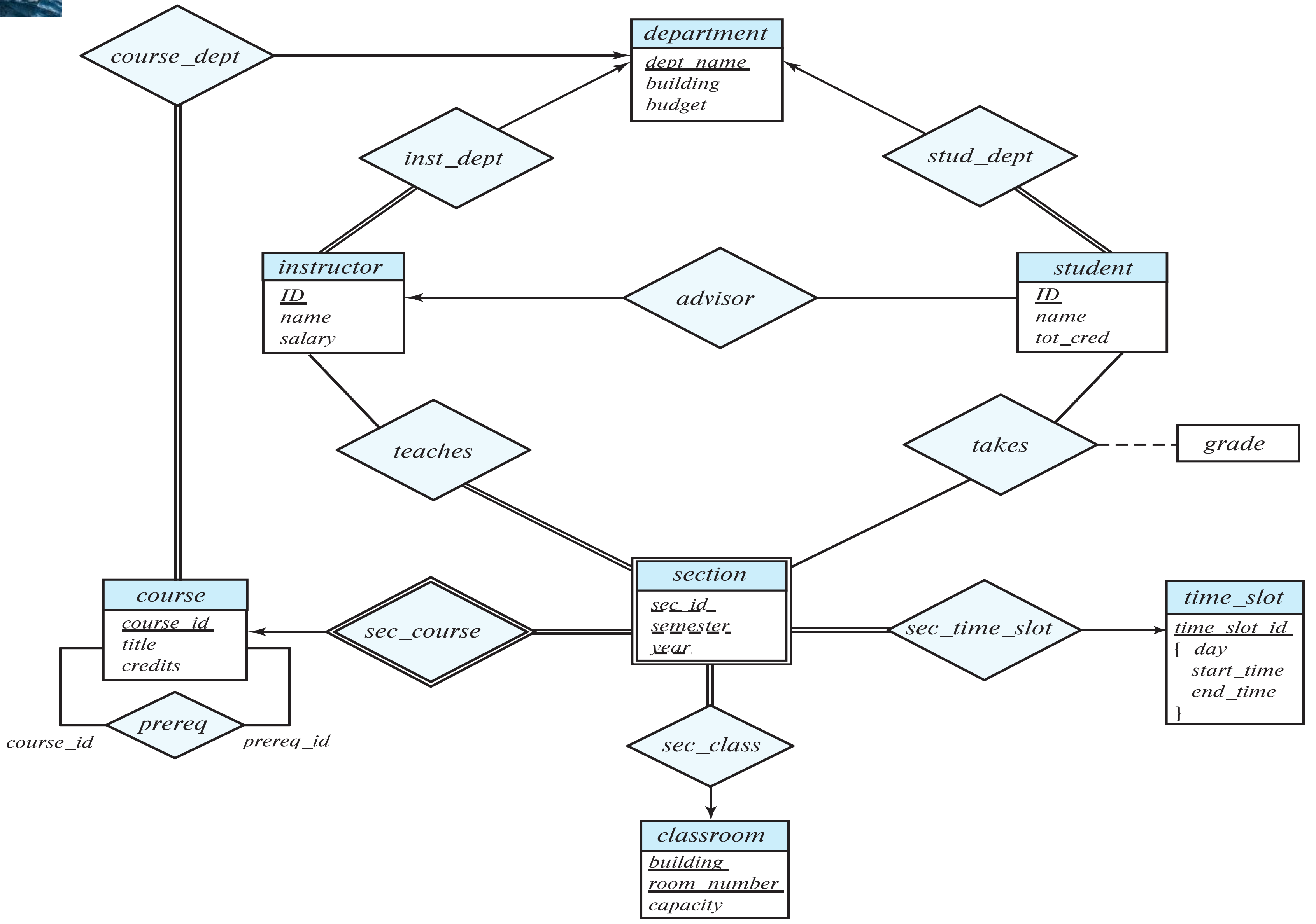
  {Student_Number}

  {Student_Phone}

  {Student_name, Subject_Number}

- **The Primary Key** : {Student_Number}

E-R Diagram for a University Enterprise



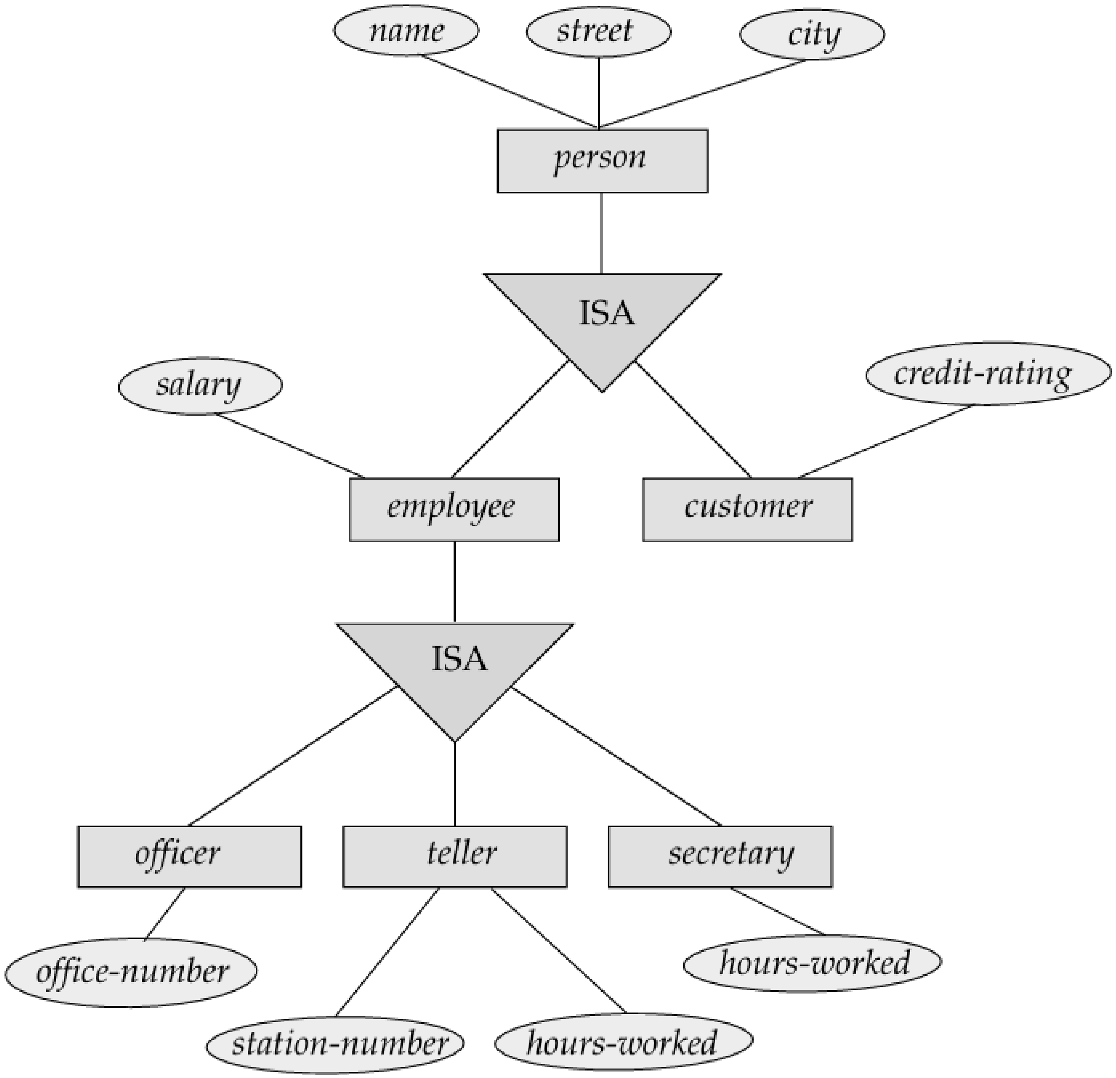
Extended E-R Features: Specialization

- Top-down design process
- Sub grouping within an entity set that are distinctive from other entities in the set
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked

Extended ER Features: Generalization

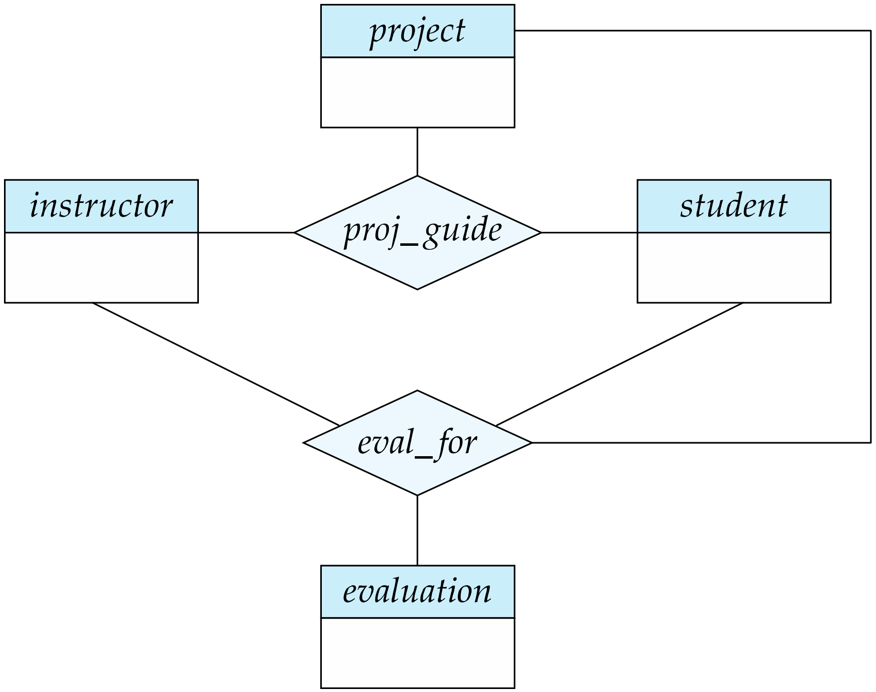
- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way
- The terms specialization and generalization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

Specialization Example



Aggregation

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



Aggregation (Cont.)

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

Aggregation (Cont.)

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