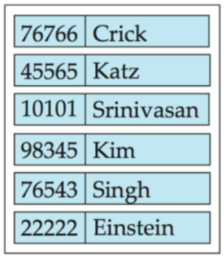
Define the following terms.

Entity

- An entity is a thing or object or person in the real world that is distinguishable from all other object.
- E.g. book, student, employee, college etc…

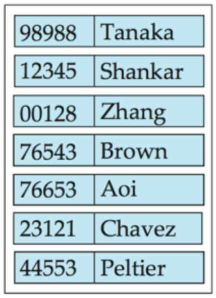
Entity sets

- An entity set is a set of entities of same type that share the same properties or attributes.
- E.g. the set of all students in a college can be defined as entity set student.



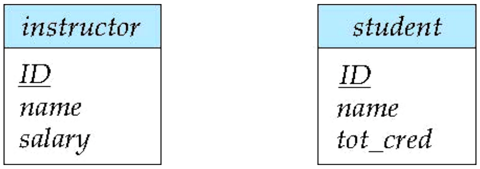
76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor



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

student

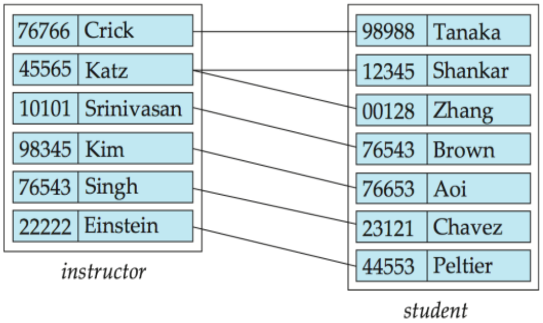


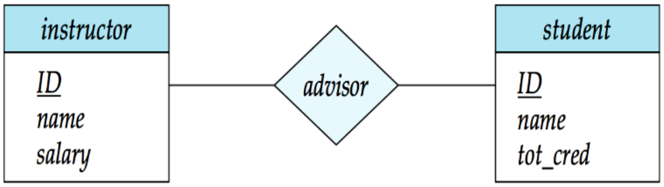
Relationship

- Relationship is an association (connection) between several entities.
- Relationship between 2 entities is called binary relationship.
- E.g. book is issued by student where book and student are entities and issue is relation.

Relationship set

- Relationship set is a set of relationships of the same type.





Degree of Relationship set:

- The number of entity-sets that participate in a relationship-set is called its degree.
- Binary relationship: degree = 2
- Ternary relationship: degree = 3

Attributes

- Attributes are properties hold by each member of an entity set.
- E.g. entity is student and attributes of students are enrollmentno, name, address, cpi etc …

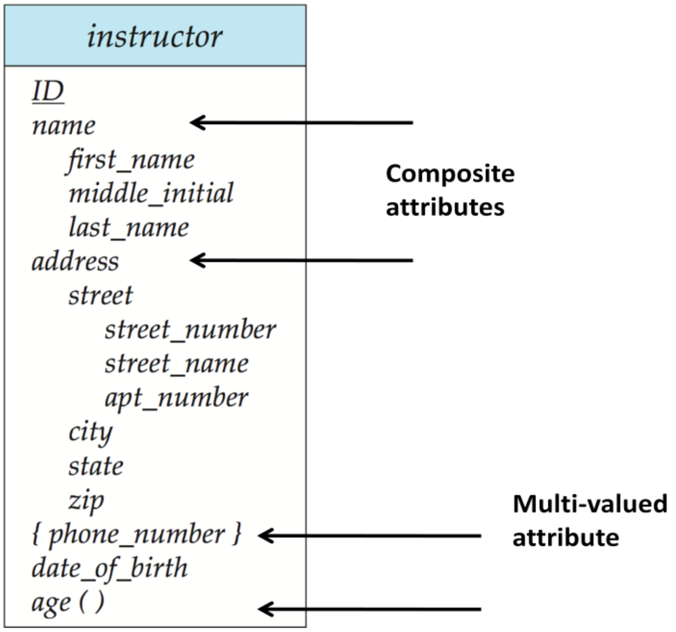
Types of attributes:

- **Simple attribute:** It cannot be divided into subparts. E.g. cpi, rollno

  Composite attribute: It can be divided into subparts. E.g. address, name
- **Single valued attribute:** It has single data value. E.g. enrollmentno, birthdate

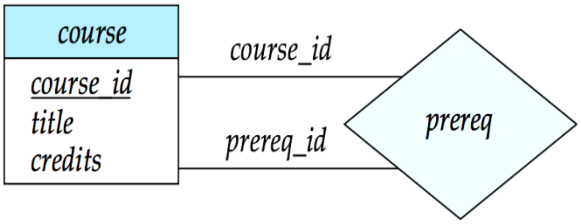
  Multi valued attribute: It has multiple data value. E.g. phoneno (may have multiple phones)
- **Stored attribute:** It's value is stored manually in database. E.g. birthdate

- **Derived attribute**: It's value is derived or calculated from other attributes. E.g. age (can be calculated using current date and birthdate).



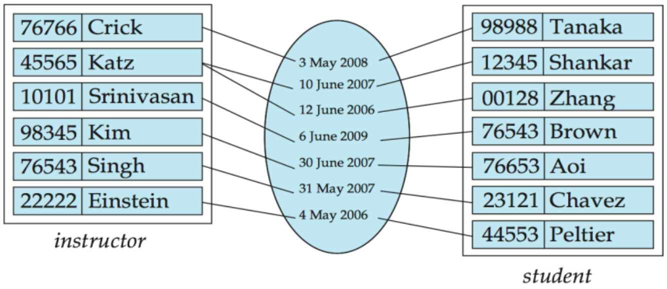
Entity Roles:

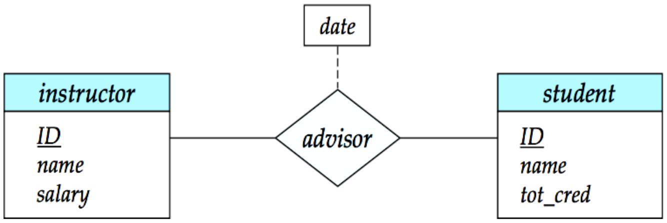
- Entity sets of a relationship need not be distinct
- The function that an entity plays in a relationship is called that entity's role.
- The labels "course_id" and "prereq_id" are called roles.



Descriptive attributes

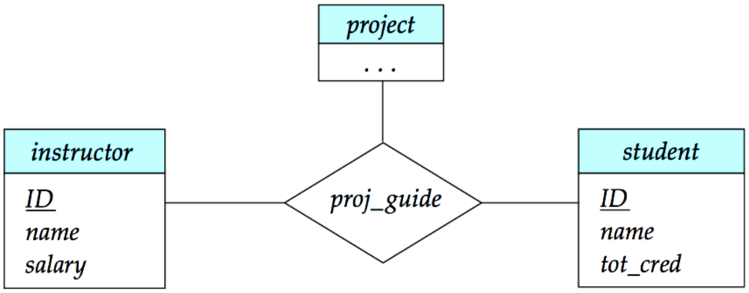
- A relationship may also have attributes like an entity. These attributes are called descriptive attributes.
- E.g. the advisor relationship set between entity sets instructor and student may have the attribute date which tracks when the student started being associated with the advisor.





n-ary relationship set:

- Most relationship sets are binary i.e. only two entity sets participate in relationship.
- There are occasions when it is more convenient to relate more than two entity sets.
- If three entity sets are related then it is known as ternary relationship.
- E-R Diagram with a Ternary Relationship



Recursive relationship set

- The same entity set participates in a relationship set more than once then it is called recursive relationship set.
- E.g. an employee entity participated in relationship under with department entity as an employee as well manager also.

What are the various phases of design process?

The task of creating a database application is a complex , involving design of the database schema, design of the programs that access and update the data, and design of a security scheme to control access to data.

Design process can be categorized into three phases:

1st phase: The initial phase of database design is to characterize fully the data needs of the prospective database users. The outcome of this phase is a specification of user requirements.

2nd phase: 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.

The schema developed at this conceptual-design phase provides a detailed overview of the enterprise. The entity-relationship model is typically used to represent the conceptual design.

3rd phase: Outcome of this phase is fully developed conceptual schema that describes 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. Ex : Update, search, delete etc.

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

Logical Design

- Design of the database schema.
- What attributes should we record in the database?
- What relation schemas should we have and how should the attributes be distributed among the various relation schemas?

Physical Design

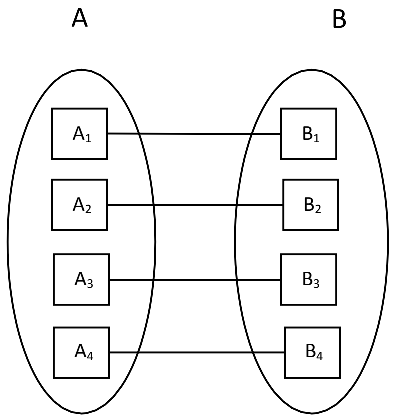
- Deciding on the physical layout of the database
- Ex. File organization, Index structures etc.

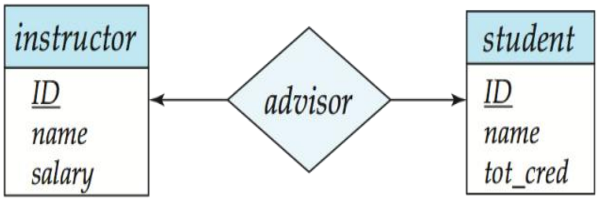
Explain various mapping cardinality (cardinality constraint).

Mapping cardinality (cardinality constraints)

- It represents the number of entities of another entity set which are connected to an entity using a relationship set.
- It is 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

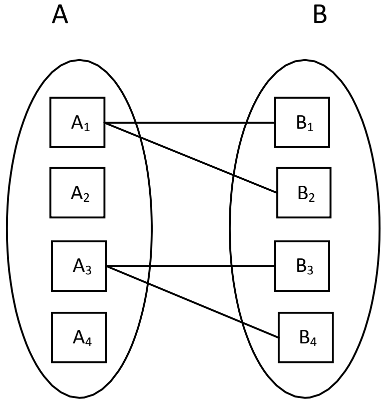
One-to-one relationship

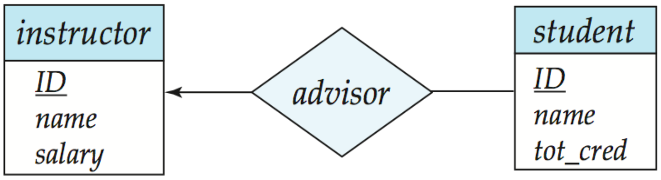




- An entity in A is associated with at most (only) one entity in B and an entity in B is associated with at most (only) one entity in A.
- A student is associated with at most one instructor via the relationship advisor.

One-to-many relationship



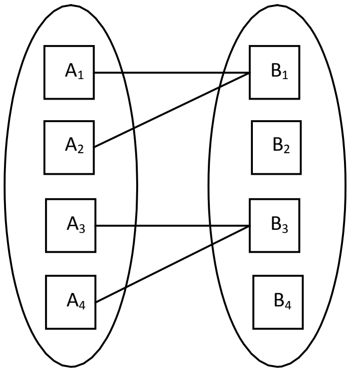


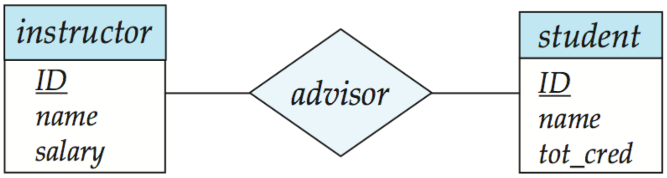
- An entity in A is associated with any number (zero or more) of entities in B and an entity in B is associated with at most one (only) entity in A.
- In the one-to-many relationship an instructor is associated with several (including 0) students via advisor but a student is associated with at most one instructor via advisor.

Many-to-one relationship

- An entity in A is associated with at most (only) one entity in B and an entity in B is associated with any number (zero or more) of entities in A.

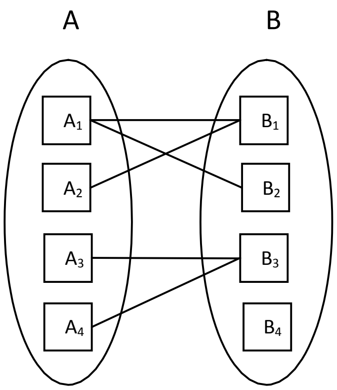
- In a many-to-one relationship an instructor is associated with at most one student via *advisor*, and a student is associated with several (including 0) instructors via *advisor.*

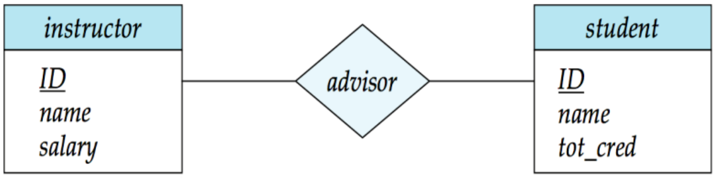




Many-to-many relationship

- An entity in A is associated with any number r (zero or more) of entities in B and an entity in B is associated with any number (zero or more) of entities in A.
- An instructor is associated with several (possibly 0) students via advisor and same for students.

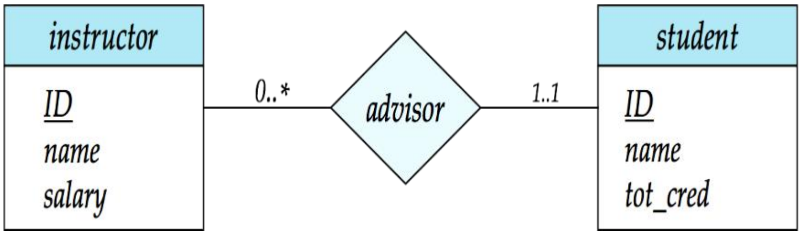




Notation for Expressing More Complex Constraints

- 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



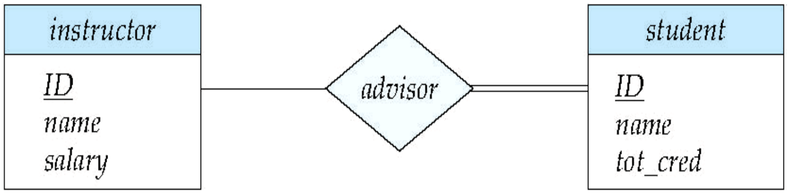
Explain various participation constraints.

Participation constraints

- It specifies the participation of an entity set in a relationship set.
- There are two types participation constraints
  - ✓ Total participation
  - ✓ Partial participation

Total participation

- In total participation every entity in the entity set participates in at least one relationship in the relationship set.
- It specifies that every entity in super class must be member of some of its sub class.
- E.g. participation of student in advisor relation is total i.e every student must have an associated instructor



Partial participation

- In partial participation some entities may not participate in any relationship in the relationship set.
- It specifies that an entity may not belong to any sub class.
- E.g. participation of *instructor* in *advisor* is partial; it is indicated by single line.

Explain various key constraints over entity and relationship set.

- Entities in an entity-set must be uniquely distinguishable using their values
- E-R model also includes the notion of keys:
  - ✓ **Superkey:** a set of one or more attributes that can uniquely identify an entity.
  - ✓ **Candidate key:** a minimal superkey .
  - ✓ **Primary key:** a candidate key chosen by Database designer as the primary means of accessing entities.
- Keys are a property of the entity-set
- They apply to all entities in the entity-set as well as to the relationship sets.

Given:

- *R* is a relationship-set with no descriptive attributes
- Entity-sets $E_1, E_2, \ldots, E_n$ participate in *R*
- *primary_key*(E_i) denotes set of attributes in E_i that represent the primary key of E_i
- A relationship instance in *R* is identified by

  ***primary_key*(E_1) ∪ *primary_key*(E_2) ∪ … ∪ *primary_key*(E_n)**
- This is a superkey for the relation R.
- The primary key of relation R depends upon the mapping cardinality.
- If *R* also has descriptive attributes $\{a_1, a_2, \ldots\}$, a relationship instance is described by:

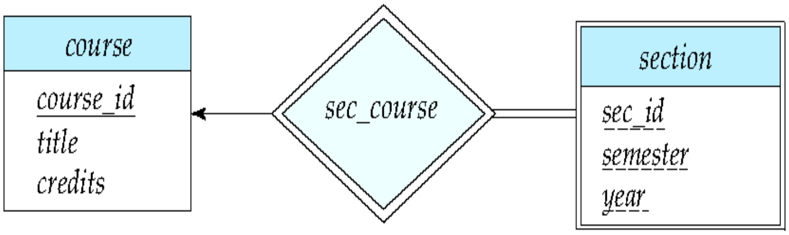
  ***primary_key*(E_1) ∪ *primary_key*(E_2) ∪ … ∪ *primary_key*(E_n) ∪ $\{a_1, a_2, \ldots\}$**
- Superkey for above is :

  ***primary_key*(E_1) ∪ *primary_key*(E_2) ∪ … ∪ *primary_key*(E_n)**

Explain weak entity set with the help of example.

Weak entity set

- An entity set that does not have a primary key is called weak entity set.
- The existence of a weak entity set depends on the existence of a strong entity set.
- Weak entity relationship set is indicated by double diamond.
- **The discriminator** (partial key) of a weak entity set is the set of attributes that distinguishes between all the entities of a weak entity set.
- **The primary key of a weak entity set is created by combining the primary key of the strong entity set on which the weak entity set is existence dependent and the weak entity set's discriminator.**
- We underline the discriminator attribute of a weak entity set with a dashed line.
- E.g. in below fig. there are two entities course and section in which curse is strong entity set and section is weak entity set.
- section entity has sec_id + semester + year which is discriminator.
- course entity has course_id as primary key.
- So primary key for section is (course_id, sec_id, semester ,year).



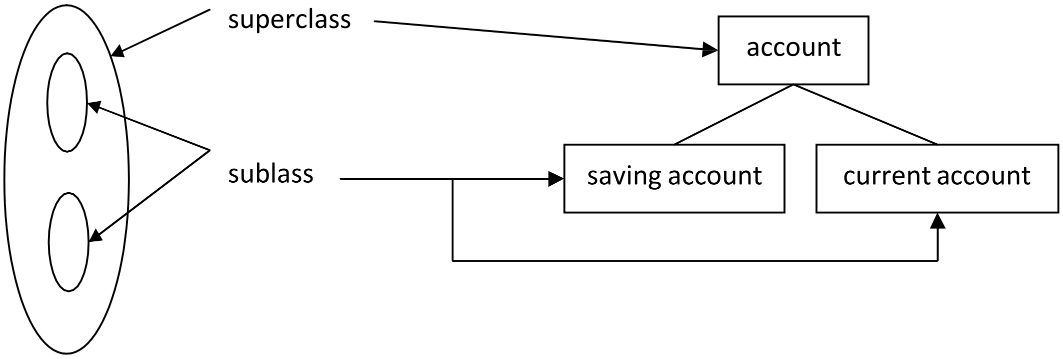
Explain the Superclass and Subclass in E-R diagram with the help of example.

Superclass

- A superclass is an entity from which another entity can be derived.
- A superclass is a generic entity set which has a relationship with one or more subclasses.
- For example, an entity set account has two subsets saving_account and current_account. So an account is superclass.
- Each member of subclass is also a member of superclass. So any saving account or a current account is a member of entity set account.

Subclass

- A subclass is an entity that is derived from another entity.
- A class is a subset of entities in an entity set which has attributes distinct from those in other subset.
- For example, entities of the entity set account are grouped in to two classes saving_account and current_account. So saving_account and current_account are subclasses.



Explain the difference between Specialization and Generalization in E-R diagram.

Specialization	Generalization
It will work in Top-down approach.	It will work in Bottom-up approach
The process of creating sub-groupings within an entity set is called specialization.	The process of creating groupings from various entity sets is called generalization.
Specialization is a process of taking a sub set of higher level entity set to form a lower-level entity set.	Generalization is a process of taking the union of two or more lower-level entity sets to produce a higher-level entity set.
Specialization starts from a single entity set; it creates different low-level entity set using some different features.	Generalization starts from the number of entity sets and creates high-level entity set using some common features.

Explain Specialization and Generalization in E-R diagram with example.

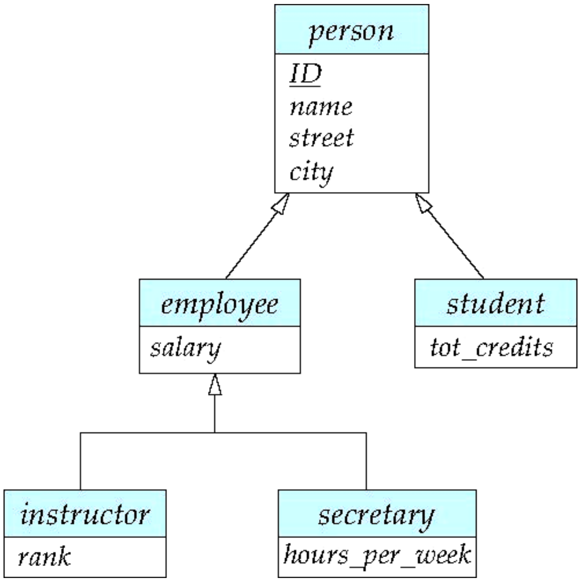
Specialization

- A top-down design process that creates subclasses based on some different characteristics of the entities in the superclass.
- An entity set may include sub groupings of entities that are distinct in some way from other entities in the set.
- For example, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set.
- Consider an entity set person, with attributes ID, name, street and city.
- A person may be further classified as one of the following:
  - ✓ Student
  - ✓ Employee
- For example, employee entities may be described by attribute **salary** and student entities may be described further by the attributes **tot_credit**.
- The process of designating sub groupings within an entity set is called specialization. The specialization of person allows us to distinguish among persons according to whether they are employees or customers.
- Now again, employees may be further classified as one of the following:
  - ✓ instructor
  - ✓ secretary
- Each of these employee types is described by a set of attributes that includes all the attributes of entity set employee plus additional attributes.
- In terms of an E-R diagram, specialization is depicted by a **ISA** relationship.
- **ISA** stands for “is a” and represents, for example, that an employee “is a” person.
- The ISA relationship may also be referred to as a superclass -subclass relationship.

There are two types of specialization:

- **Overlapping –** Entity set belongs to more than one specialized entity set .

  Ex. employee and student
- **Disjoint** – Entity set belongs to at most one specialized entity set Ex. instructor and secretary



Generalization

- A bottom-up design process that combines number of entity sets that have same features into a higher-level entity set.
- The design process proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher level entity set on the basis of common features.
- The database designer may have to first identify a student entity set with the attributes ID, name, street, city, and tot_credit, and an employee entity set with the attributes ID, name, street, city, and salary.
- But student entity set and the employee entity set have some attributes common. This commonality can be expressed by generalization, which is a containment relationship that exists between a higher level entity set and one or more lower level entity sets.
- In our example, person is the higher level entity set and student & employee are lower level entity sets.
- Differences in the two approaches may be characterized by their starting point and overall goal.

Inheritance:

- Attributes of higher-level entity-sets are inherited by lower-level entity-sets
- Relationships involving higher-level entity-sets are also inherited by lower-level entity-sets!
- Subclass can also inherit from multiple superclasses

Explain types of constraints on specialization and Generalization.

- There are mainly three categories of constraints apply to a specialization/generalization:

Membership Constraints:

It describes which entities can be members of a given lower level entity set'

- **Condition defined:**
  - ✓ Membership is specified by a predicate
  - ✓ If an entity satisfies a lower-level entity-set's predicate then it is a member of that lower-level entity-set .
  - ✓ If all sets refers to the same predicate over single attribute then it is called attribute-defined specialization.

- ✓ Ex. All students are evaluated on the basis of student_type attribute : graduate or undergraduate.

- **User defined:**
  - ✓ Entities may simply be assigned to lower-level entity sets by a database user.
  - ✓ No explicit predicate governs membership
  - ✓ Generally used when an entity's membership could change in the future

Disjointness Constraints :

Describes relationship between members of the subclasses and indicates whether member of a superclass can be a member of one, or more than one, subclass.

- **Disjoint constraint**
  - ✓ It specifies that the subclasses of the specialization must be disjointed (an entity can be a member of only one of the subclasses of the specialization).
  - ✓ Specified by 'd' in EER diagram or by writing disjoint.
- **Non-disjoint (overlapping)**
  - ✓ It specifies that is the same entity may be a member of more than one subclass of the specialization.
  - ✓ Specified by 'o' in EER diagram or by writing overlapping.

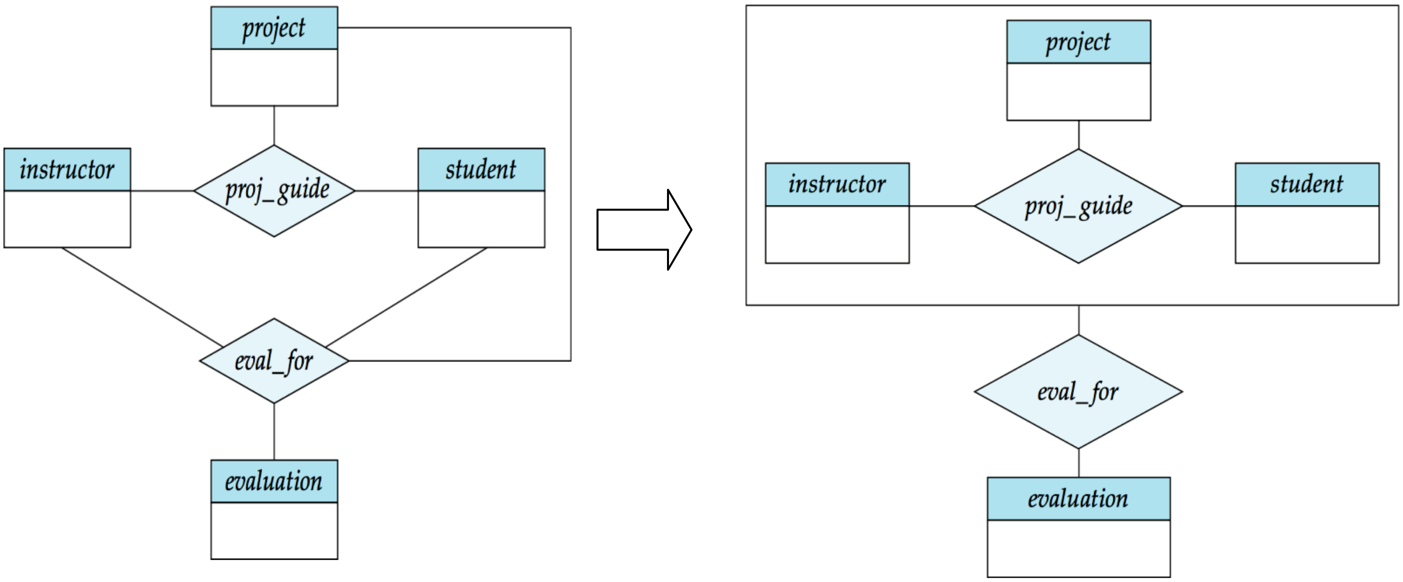
Participation (completeness) Constraint

Determines whether every member in super class must participate as a member of a subclass.
It may be total (mandatory) or partial (optional).

- **Total (mandatory)**
  - ✓ Total specifies that every entity in the super class must be a member of some subclass in the specialization/ generalization.
  - ✓ Specified by a double line in ER diagram.
- **Partial (optional)**
  - ✓ Partial specifies that every entity in the super class not belong to any of the subclasses.
  - ✓ Specified by a single line in ER diagram.
  - ✓ Based on these two different kinds of constraints, a specialization or generalization can be one of four types
    - Total, Disjoint
    - Total, Overlapping
    - Partial, Disjoint
    - Partial, Overlapping.

Explain aggregation in E-R diagram.

- The E-R model cannot express relationships among relationships.
- When would we need such a thing at that time aggregation is used.
- Consider a database with information about students who work on a particular project and Suppose we want to record evaluations of a student by a guide on a project.



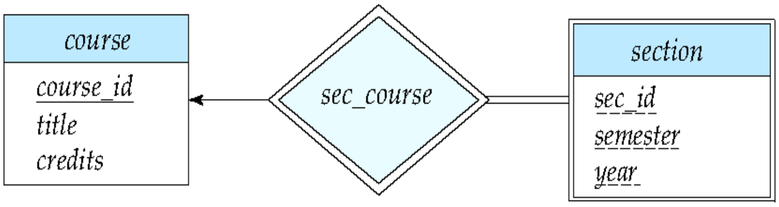
- Relationship sets eval_for and proj_guide represent overlapping information.
- We can Eliminate this redundancy via aggregation
  - ✓ Treat relationship as an abstract entity
  - ✓ Allows relationships between relationships
  - ✓ Abstraction of relationship into new entity
- Now Relationship sets proj_guide and eval_for could be combined into a single set.
- Aggregation is an abstraction through which relationships are treated ashigher-level entities

---

Reduction to Relation Schema

- Entity sets and relationship sets can be expressed uniformly as relation schemas that represent the contents of the database.
- 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:



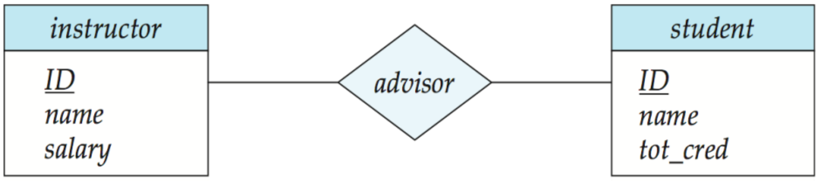
A strong entity set reduces to a schema with the same attributes in the entity set

course(course_id, title, credits)

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:



Let R be a relationship set, let a1, a2, . . . , am be the set of attributes formed by the union of the primary keys of each of the entity sets participating in R, and let the descriptive attributes (if any) of R be b1, b2, . . . , bn.We represent this relationship set by a relation schema called R with one attribute for each member of the set:

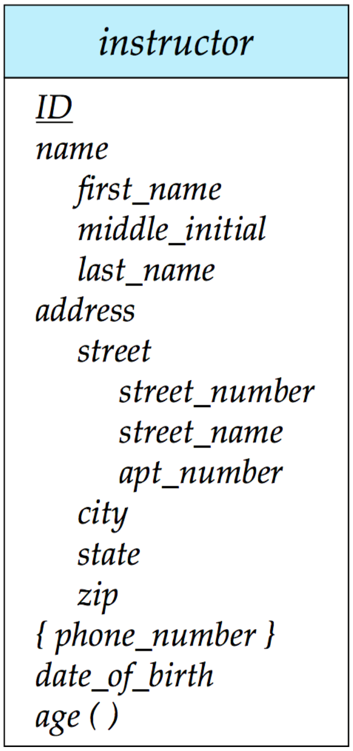
{a1, a2, . . . , am} ∪ {b1, b2, . . . , bn}

- For a binary many-to-many relationship, the union of the primary-key attributes from the participating entity sets becomes the primary key.
- For a binary one-to-one relationship set, the primary key of either entity set can be chosen as the primary key. The choice can be made arbitrarily.
- For a binary many-to-one or one-to-many relationship set, the primary key of the entity set on the "many" side of the relationship set serves as the primary key.
- Avove is many to many : ***advisor* = (*s_id, i_id*)**

Representation of Entity Sets with Composite Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
- Example: given entity set instructor
- 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)



Representation of Entity Sets with 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.

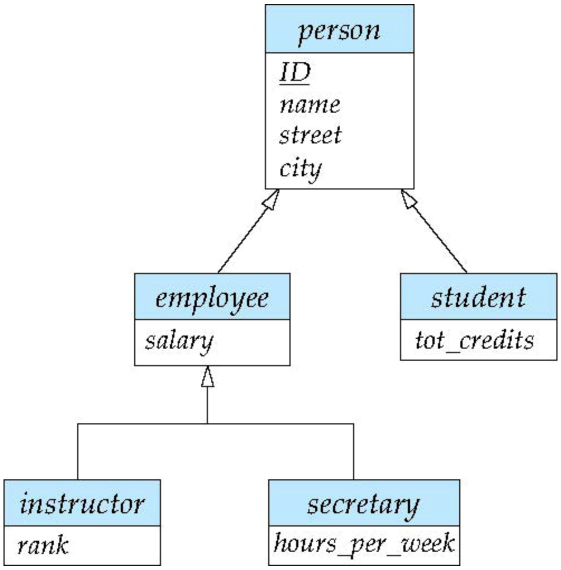
  (**22222, 456-7890) and (22222, 123-4567)**

Representing Specialization via Schemas:

Method 1:

- Form a schema for the higher-level entity
- Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes
- Drawback: getting information about, an employee requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

schema	attributes
person	ID, name, street, city
student	ID, tot_cred
Employe	ID, salary



Method 2:

- Form a schema for each entity set with all local and inherited attributes
- Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

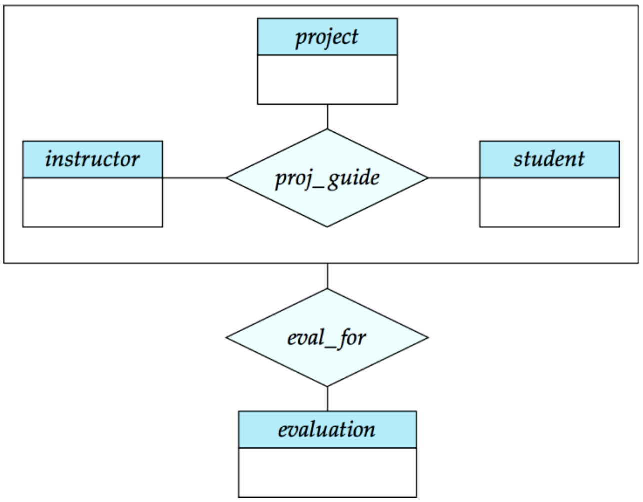
schema	attributes
person	ID, name, street, city
student	ID, name, street, city, tot_cred
employee	ID, name, street, city, salary

Representing Aggregation via Schemas:

To represent aggregation, create a schema containing:

- Primary key of the aggregated relationship,
- The primary key of the associated entity set
- Any descriptive attributes
- The schema **eval_for** is:

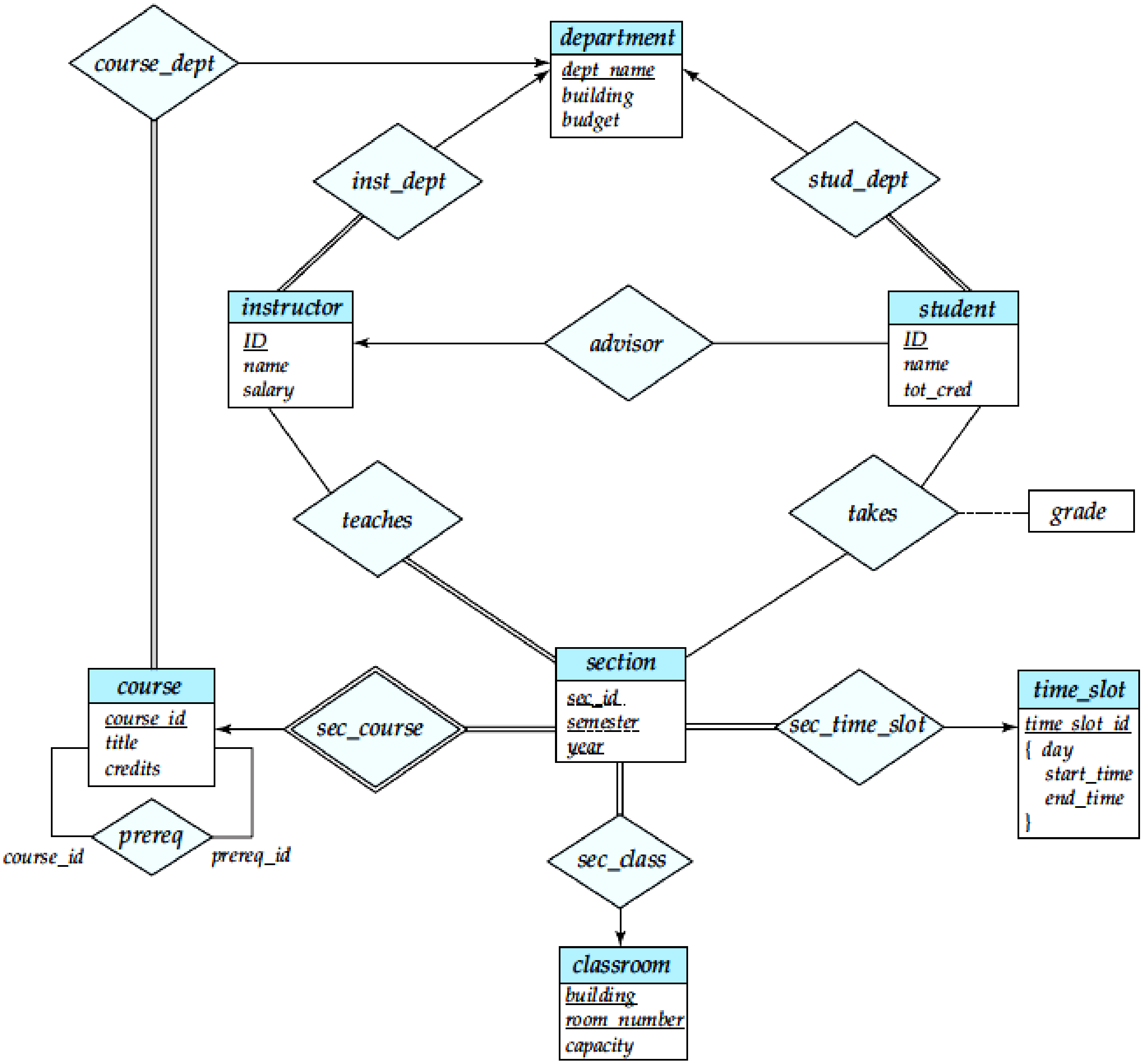
  eval_for (s_ID, project_id, i_ID, evaluation_id)
- The schema **proj_guide** is redundant.



Example E-R diagram : University

Representation of entity sets:

- *student (ID, name, tot_cred)*
- *classroom (building, room_number, capacity)*
- *department (dept_name, building, budget)*
- *course (course_id, title, credits)*
- *instructor (ID, name, salary):*
- *section (course_id, sec_id, semester, year)*
- *time_slot (time_slot_id, day, start_ime, end_time)*



Representation of Relationship sets:

- *teaches (ID, course_id, sec_id, semester, year)*
- *takes (ID, course_id, sec_id, semester, year, grade)*
- *prereq (course_id, prereq_id)*
- *advisor (s_ID, i_ID)*
- *sec_course (course _id, sec _d, semester, year)*
- *sec_time_slot (course_id, sec_id, semester, year, time_slot_id)*
- *sec_class (course_id, sec_id, semester, year, building, room_number)*
- *inst_dept (ID, dept_name)*
- *stud_dept (ID, dept_name)*
- *course_dept (course_id, dept_name)*

Redundancy of schemas:

The schema for the relationship set linking a weak entity set to its corresponding strong entity set is redundant and does not need to be present in a relational database design ased upon an E-R diagram.

Example : section (course_id, sec_id, semester, year)

sec_course (course _id, sec _d, semester, year) ← *remove this relation*

Combination of schemas

- Consider a **many-to-one** relationship set AB from entity set A to entity set B.
- Suppose further that the participation of A in the relationship is **total**
- Using our relational-schema construction algorithm outlined previously, we get three schemas: A, B, and AB
- Then we can **combine the schemas A and AB to form a single schema** consisting of the union of attributes of both schemas.

Examples :

- *inst_dept (ID, dept_name)* → *instructor(ID, name, dept name, salary).*
- *stud_dept (ID, dept_name)* → *Student (ID,name, dept name, tot cred)*
- *course_dept (course_id, dept_name)* → *course(course_id, title, dept_name, credits)*
- *sec_time_slot (course_id, sec_id, semester, year, time_slot_id)* →
  section(course _id, sec_id, semester, year, building, room_number)
- *sec_class (course_id, sec_id, semester, year, building, room_number)* →
  section(course_id, sec_id, semester, year, building, room_number,time _slot _id)

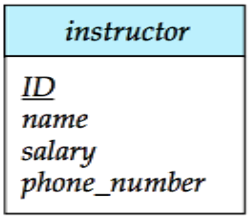
What are the Entity-Relationship Design Issues ?

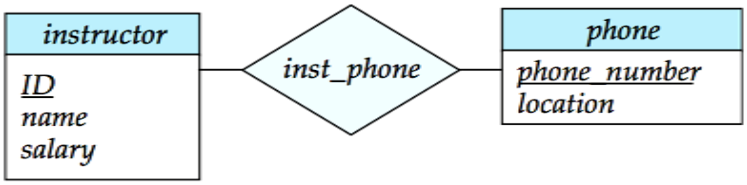
1. Use of Entity Sets versus Attributes

Consider the entity set instructor with the additional attribute phone number. It can easily be argued that a phone is an entity in its own right with attributes phone number and location; the location may be the office or home where the phone is located, with mobile (cell) phones perhaps represented by the value "mobile."

If we take this point of view, we do not add the attribute phone number to the instructor. Rather, we create:

- A phone entity set with attributes phone number and location.
- A relationship set inst phone, denoting the association between instructors and the phones that they have.

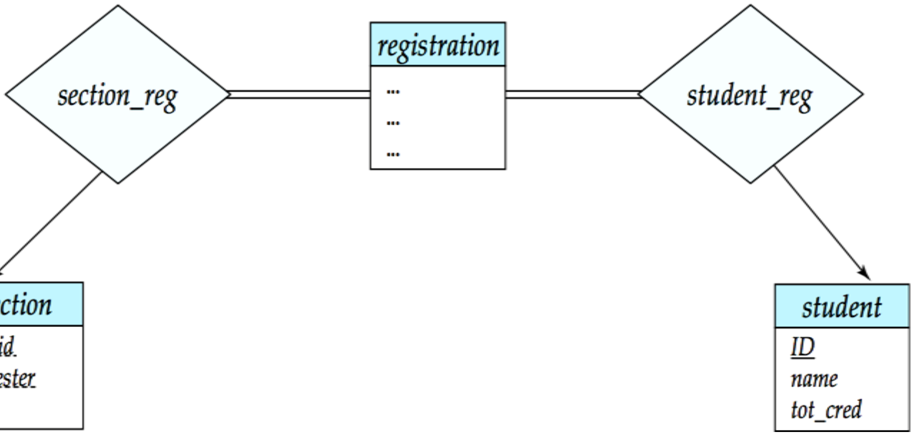




2. Use of Entity Sets versus Relationship Sets:

It is not always clear whether an object is best expressed by an entity set or a relationship set. we used the takes relationship set to model the situationwhere a student takes a (section of a) course.

An alternative is to imagine that there is a course-registration record for each course that each student takes. Then, we have an entity set to represent the course-registration record. Let us call that entity set registration.

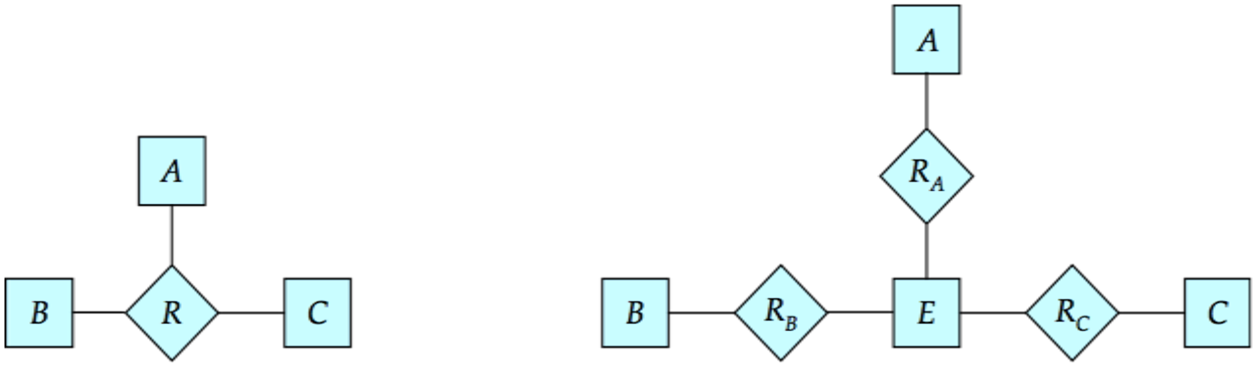


3. Binary versus n-ary Relationship Sets

An n-ary relationship several entities participate in a single relationship. Some relationships that appear to be non-binary may be better represented using binary relationships. For example, a ternary relationship parents, relating a child to his/her father and mother.

Using two binary relationships allows partial information (e.g., only mother being known)

- But there are some relationships that are naturally non-binary
- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.



(a)

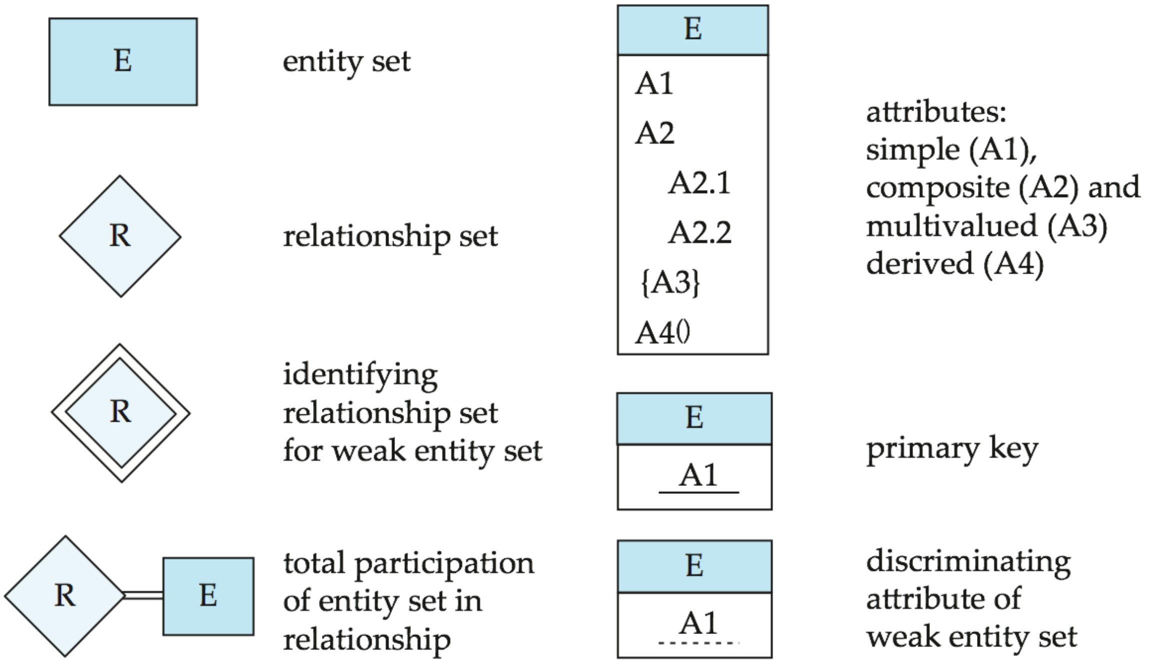
(b)

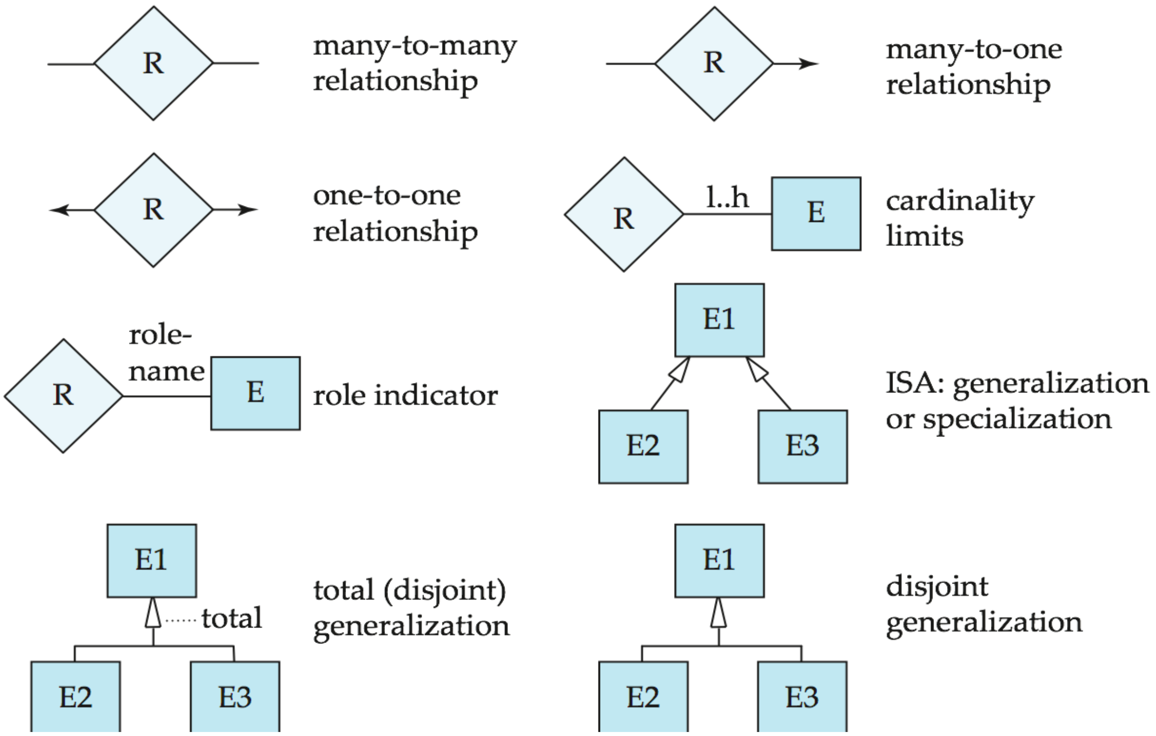
4. Placement of Relationship Attributes

The cardinality ratio of a relationship can affect the placement of relationship attributes. Thus, attributes of one-to-one or one-to-many relationship sets can be associated with one of the participating entity sets, rather than with the relationship set.

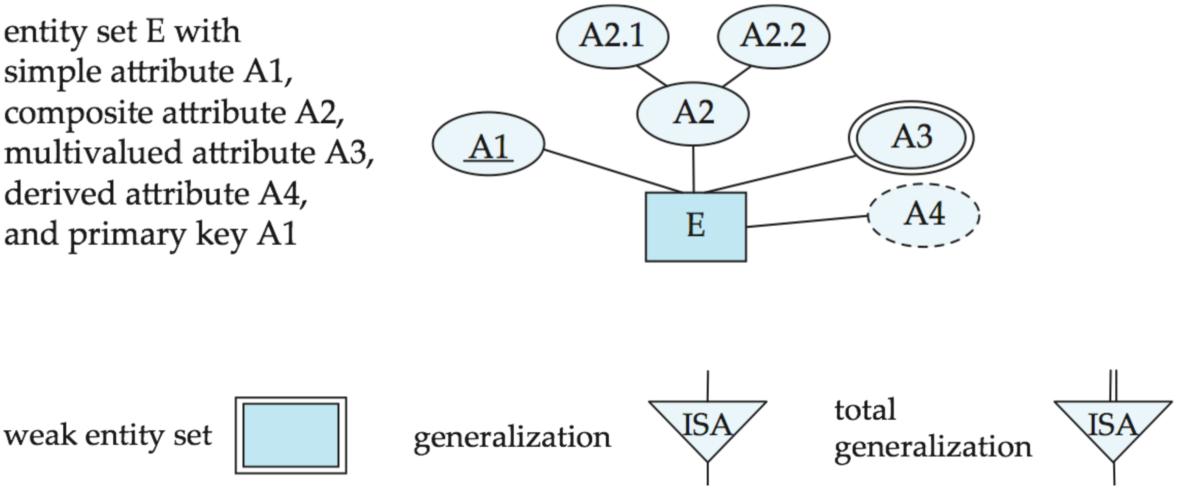
The designer may choose to retain date as an attribute of advisor or as one of the attribute of the instructor.

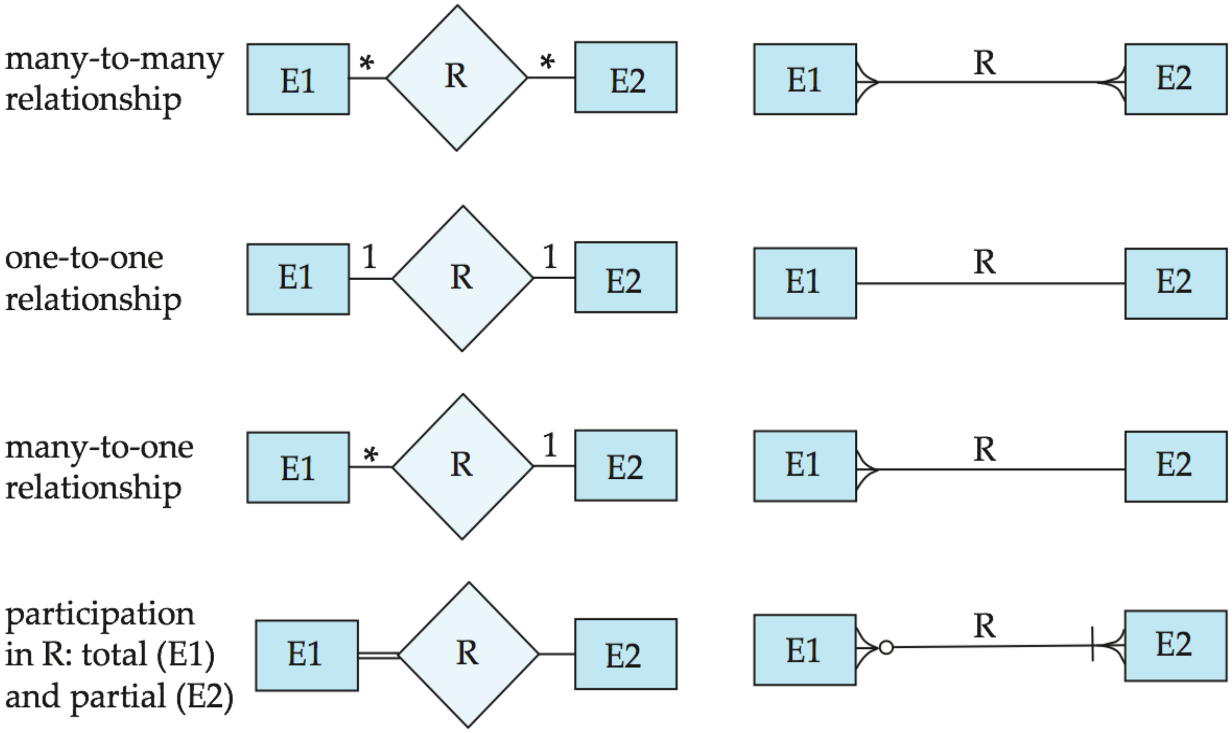
Summary of Symbols Used in E-R Notation:





Alternative ER Notations:




many-to-many relationship
E1
*
R
*
E2
E1
R
E2
one-to-one relationship
E1
1
R
1
E2
E1
R
E2
many-to-one relationship
E1
*
R
1
E2
E1
R
E2
participation in R: total (E1) and partial (E2)
E1
R
E2
E1
R
E2