Chapter 6: Database Design using ERD [1]

Abu Raihan Mostofa Kamal

Professor, CSE Department
Islamic University of Technology (IUT)

September 12, 2023

[1] Slides are based on Textbook, its companion slide and other sources

Chapter Outline

Overview of the Design Process

ER Model

Mapping Cardinality

Generalization/Specialization

Design Phases

1. **The initial phase:** Characterize fully the data needs of the prospective database users. The database designer needs to interact extensively with domain experts and users to carry out this task. The outcome of this phase is a specification of user requirements.
2. **Conceptual-design phase:**The entity-relationship model is typically used to represent the conceptual design. This is an abstract-design of the system. This is independent of any specific database implementation.
3. **Implementation phase:** It has two levels:
   (a) Logical Design: It maps the high-level conceptual schema onto the implementation data model of the database system.
   (b) Physical Design: It takes care of the actual storage features such as default space, indexing and so on.

Design Phases

1. **The initial phase:** Characterize fully the data needs of the prospective database users. The database designer needs to interact extensively with domain experts and users to carry out this task. The outcome of this phase is a specification of user requirements.
2. **Conceptual-design phase:**The entity-relationship model is typically used to represent the conceptual design. This is an abstract-design of the system. This is independent of any specific database implementation.
3. **Implementation phase:** It has two levels:
   (a) Logical Design: It maps the high-level conceptual schema onto the implementation data model of the database system.
   (b) Physical Design: It takes care of the actual storage features such as default space, indexing and so on.

Design Phases

1. **The initial phase:** Characterize fully the data needs of the prospective database users. The database designer needs to interact extensively with domain experts and users to carry out this task. The outcome of this phase is a specification of user requirements.
2. **Conceptual-design phase:**The entity-relationship model is typically used to represent the conceptual design. This is an abstract-design of the system. This is independent of any specific database implementation.
3. **Implementation phase:** It has two levels:
   (a) Logical Design: It maps the high-level conceptual schema onto the implementation data model of the database system.
   (b) Physical Design: It takes care of the actual storage features such as default space, indexing and so on.

Design Phases

1. **The initial phase:** Characterize fully the data needs of the prospective database users. The database designer needs to interact extensively with domain experts and users to carry out this task. The outcome of this phase is a specification of user requirements.
2. **Conceptual-design phase:**The entity-relationship model is typically used to represent the conceptual design. This is an abstract-design of the system. This is independent of any specific database implementation.
3. **Implementation phase:** It has two levels:
    (a) Logical Design: It maps the high-level conceptual schema onto the implementation data model of the database system.
    (b) Physical Design: It takes care of the actual storage features such as default space, indexing and so on.

Design Phases

1. **The initial phase:** Characterize fully the data needs of the prospective database users. The database designer needs to interact extensively with domain experts and users to carry out this task. The outcome of this phase is a specification of user requirements.
2. **Conceptual-design phase:**The entity-relationship model is typically used to represent the conceptual design. This is an abstract-design of the system. This is independent of any specific database implementation.
3. **Implementation phase:** It has two levels:
   (a) Logical Design: It maps the high-level conceptual schema onto the implementation data model of the database system.
   (b) Physical Design: It takes care of the actual storage features such as default space, indexing and so on.

Bad and Good Design

In designing a database schema, we must ensure that we avoid two major problems:

1. **Redundancy:** A bad design may repeat information. The biggest problem with such redundant representation of information is that the copies of a piece of information can become inconsistent if the information is updated.
2. **Incompleteness:** A bad design may make certain aspects of the enterprise difficult or impossible to model. It introduces bad business logic.

Design Approaches

- **Entity Relationship Model** (will be covered in this chapter).
  - ✓ Collection of entities and relationships
  - ✓ Entity is an object (with a number of attributes), while relationship defines how entities are related.
  - ✓ Represented diagrammatically by an entity-relationship diagram
- **Normalization Theory:** Formal method to test if the design is good or bad. (will be covered in Chapter 7)

Overview of ER Model

Entity Sets

- **Entity:** 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, courses
- 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.

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

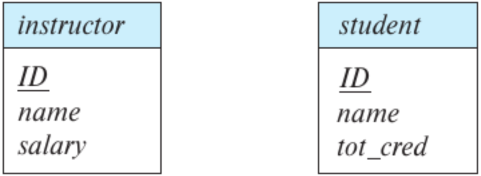


Figure: E-R diagram for entity sets instructor and student

Relationship Sets

- A relationship is an association among several entities

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

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

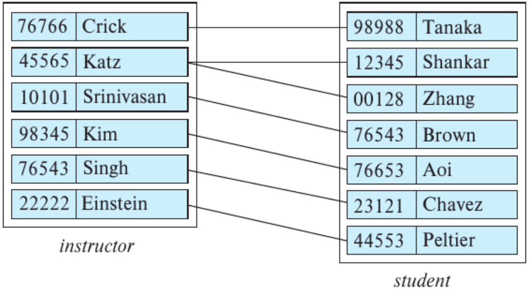
$$\{(e_1, e_2, \ldots, e_n) | e_1 \subseteq E_1, e_2 \subseteq E_2, \ldots, e_n \subseteq E_n\}$$

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

Example: (44553,22222)$\subseteq$ 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 as follows:



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

Figure: Relationship set: Example

Represent Relationship Sets

- **Diamonds** represent relationship sets.

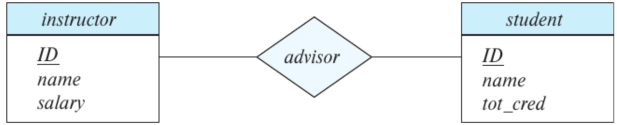


Figure: Diamond is the Relationship symbol

Relationship Sets: Additional Attribute

- 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

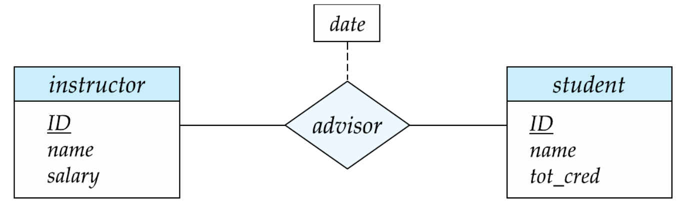


Figure: Additional Attribute in the Relationship

Relationship Sets: Roles

- Entity sets of a relationship need not be distinct, each relation may be used multiple times with different roles.
- For instance, `course_id` and `prereq_id` are called **roles**
- In fact, here same attribute (i.e. `course_id`) is used for 2 purposes.

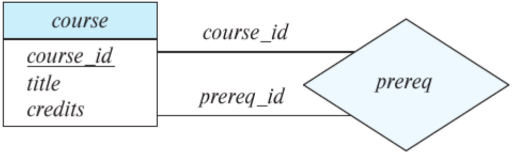


Figure: Roles in the Relationship

Relationship Sets: Roles (Cont.)

- In this type of relationship set, sometimes called a recursive relationship set.
- The records of one entity refers to itself. For instance, course entity here will have at least two attributes: (i) CourseID and (ii)PreReqID .
- In actual implementation, such Roles are achieved by self-referencing.

Degree of a Relationship Set

- The number of entity sets that participate in a relationship set is the degree of the relationship set.
- Most of the relationship sets in a database system are binary.
- Occasionally, however, relationship sets involve more than two entity sets.
- Example: Consider the entity projects that represents all the research projects of the university. Each project can have multiple associated students and multiple associated instructors. It becomes a Ternary Relationship.

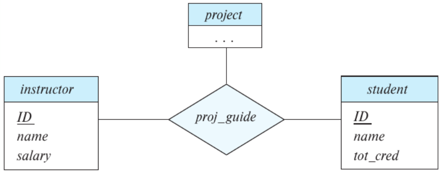


Figure: 3 Entities in Relationship

Complex Attributes

In the E-R model, attributes can be grouped into as follows:

- Simple and composite attributes
- Single-valued and multivalued attributes.
- Derived attributes.

Simple and Composite Attributes

- Simple attribute has no subparts. While Composite attribute **has** subparts (i.e. Name= first name, last name).

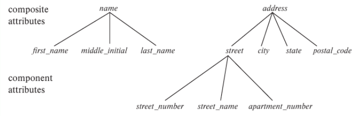


Figure: Composite Attribute

- Composite attribute may appear as a hierarchy. (each level is called component attribute)
- Composite attribute is preferable if different components are useful at different occasions. For instance, if we need some report of employees according to last name for HR section, while another list with first name is needed for accounts section.

Simple and Composite Attributes: Implementation

- In Oracle Database, it is implemented as TYPE as OBJECT as a user-defined datatype containing sub-parts.
- Oracle Database requires you to use a table alias to qualify any dot-notational reference to subprograms or attributes of objects.
- Use of a table alias is optional when referencing top-level attributes of an object table directly, without using the dot notation.

Single-valued and Multi-valued attributes

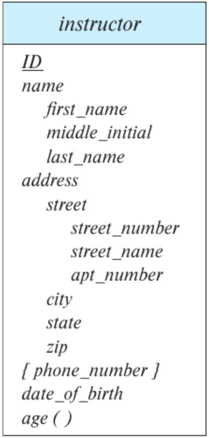
- The attributes in our examples all have a single value for a particular entity. For instance, the student ID attribute for a specific student entity refers to only one student ID. Such attributes are said to be single valued.
- An attribute may have zero, one or more values against one record, then it is called multi-valued attribute. Example: one citizen may have more than one phone number.

Derived attributes

- The value for this type of attribute can be derived from the values of other related attributes or entities.
- For instance, age can be derived from date of birth.
- The value of a derived attribute is not stored but is computed when required.

3 categories presented are based on the concept of Domain of attribute, which defines the permissible values and number of values for that attribute.

Complex Attributes in ER Diagram



instructor

ID
name
first_name
middle_initial
last_name
address
street
street_number
street_name
apt_number
city
state
zip
{ phone_number }
date_of_birth
age ( )

Figure: E-R diagram with composite, multivalued, and derived attributes

Composite Attributes: Implementation (Cont.)

1. Step 1: Create the type first (using Object concept)

```
create type nametype as object (
fname varchar2(10),
middlename varchar2(10),
lastname varchar2(10)
);
```

2. Step 2: Create table with user-defined type as an attribute

```
create table emp
(id number primary key,
name nametype,
address varchar2(20),
constraint cnamenn check (name.fname IS NOT NULL)
);
```

Composite Attributes: Implementation (Cont.2)

1. Step 3: Data insert and select

```
insert into emp values(101,
nametype('Abdur','Rahim','Mia'),'Uttara');


% Select can be done this way:

select id, e.name.fname, address from emp e;

select id, name, address from emp;
```

Multi-valued Attributes: Implementation

1. Create attribute using varray, then create table and finally insert data

```
CREATE OR REPLACE TYPE vmobiles AS VARRAY(10) OF VARCHAR2(20);

CREATE TABLE students_multiple (id number, name varchar2(20), phone vmobiles);

insert into students_multiple values(1, a ,vmobiles( 0001 ));

insert into students_multiple values(2, b ,vmobiles( 0002 , 0003 , 0004 ));

insert into students_multiple values(3, c ,vmobiles(NULL));
```

Multi-valued Attributes: Implementation (Select)

1. You will have the use the TABLE operator in the case of a VARRAY

```
--this is how you need to select
--You will have the use the TABLE operator in the case of a VARRAY

select id,name,  e.*
FROM STUDENTS_MULTIPLE s, TABLE(s.phone)e;


select * from
students_multiple;
```

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:
  - (a) One to one
  - (b) One to many
  - (c) Many to one
  - (d) Many to many

Mapping Cardinality: One to One

- **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.

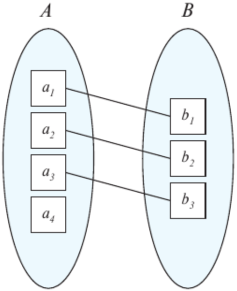


Figure: One to One Mapping

Mapping Cardinality: One to Many

- **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.

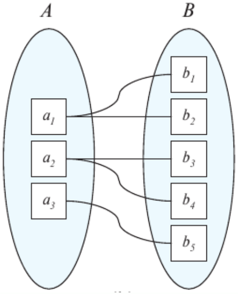


Figure: One to Many Mapping

Mapping Cardinality: Many to One

- **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.

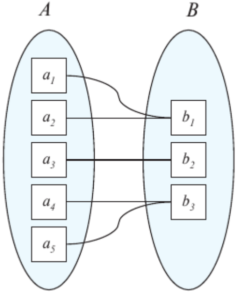


Figure: Many to One Mapping

Mapping Cardinality: Many to Many

- **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.

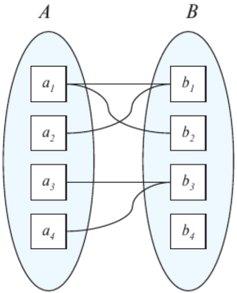


Figure: Many to Many Mapping

Cardinality Constraints in ER Diagram

- A directed line ⟶ is used to specify "one"
- An un-directed ———— line specify "many" which is either zero or more
- **Example:** One-to-one relationship between an instructor and a student. A student is associated with at most one instructor via the relationship advisor

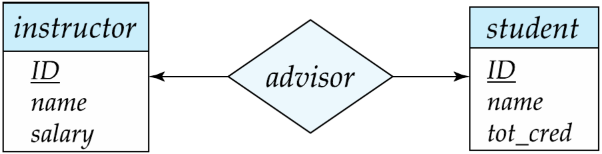


Figure: One to One Mapping

ER Diagram for One-to-Many

- 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

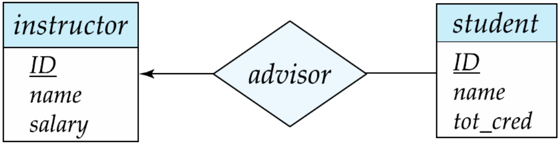


Figure: One to Many Mapping

ER Diagram for Many-to-One

- 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

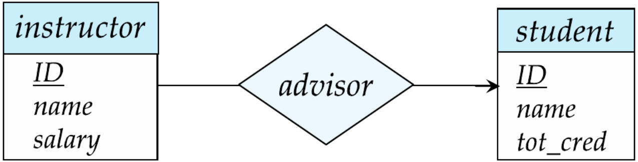


Figure: Many to One Mapping

ER Diagram for Many-to-Many

- Suppose we are allowing joint-supervision, students work in a group.
  - ✓ An instructor is associated with several (possibly 0) students via advisor
  - ✓ A student is associated with several (possibly 0) instructors via advisor

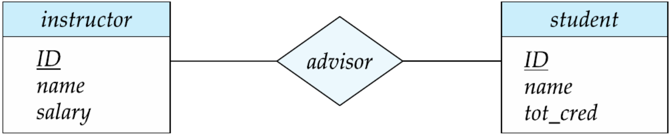


Figure: Many to Many Mapping

Total and Partial Participation

- **Total Participation[1] (indicated by double line):** every entity in the entity set participates in at least one relationship in the relationship set.

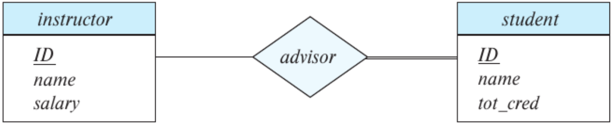


Figure: Total Participation

Example: participation of student in advisor relation is total which implies 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 which implies some instructors may not supervise any student

[1] For Total Participation NOT NULL constraint is used along with Foreign Key

Max..Min Cardinality: Alternative Representation

- E-R diagrams also provide a way to indicate more complex constraints on the number of times each entity participates in relationships in a relationship set.
- 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 of the entity set in the relationship set.
- 0..* means zero or more

Max..Min Cardinality: Precautions

- This min..max format could be misleading if not properly explained.

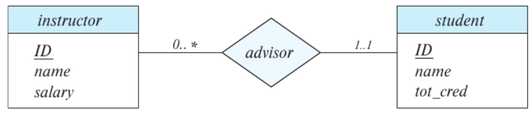


Figure: Min..Max format in cardinality

- The limit 0.. * on the line between advisor and instructor indicates that an instructor can have zero or more students.
- The limit 1..1 on the line between student and advisor indicates that one student must have exactly one instructor as advisor.
- So, the relationship advisor is **one-to-many** from instructor to student. And participation of student in advisor is **total**.

Max..Min Cardinality: Precautions

- This min..max format could be misleading if not properly explained.

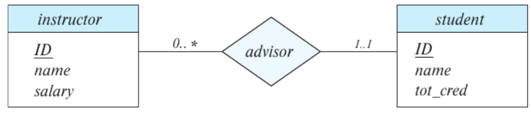


Figure: Min..Max format in cardinality

- The limit 0.. * on the line between advisor and instructor indicates that an instructor can have zero or more students.
- The limit 1..1 on the line between student and advisor indicates that one student must have exactly one instructor as advisor.
- So, the relationship advisor is **one–to–many** from instructor to student. And participation of student in advisor is **total**.

Max..Min Cardinality: Precautions

- This min..max format could be misleading if not properly explained.

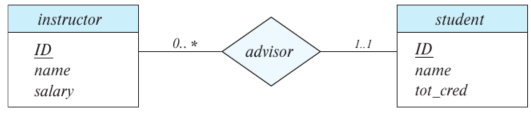


Figure: Min..Max format in cardinality

- The limit 0.. * on the line between advisor and instructor indicates that an instructor can have zero or more students.
- The limit 1..1 on the line between student and advisor indicates that one student must have exactly one instructor as advisor.
- So, the relationship advisor is one–to–many from instructor to student. And participation of student in advisor is total.

Max..Min Cardinality: Precautions

- This min..max format could be misleading if not properly explained.

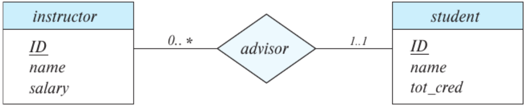


Figure: Min..Max format in cardinality

- The limit 0.. * on the line between advisor and instructor indicates that an instructor can have zero or more students.
- The limit 1..1 on the line between student and advisor indicates that one student must have exactly one instructor as advisor.
- So, the relationship advisor is **one-to-many** from instructor to student. And participation of student in advisor is **total**.

Weak and Strong Entity Sets

- Strong Entity is independent of any other entity in the schema, in other words, it has sufficient attributes to form the primary key.
  - ✓ The strong entity is represented by a single rectangle.
  - ✓ The relationship between two strong entities is represented by a single diamond.
- A weak entity is an entity set that does not have sufficient attributes for Unique Identification of its records. (no primary key)

Weak Entity Sets

- It has no primary key. The existence of a weak entity set depends on the existence of a strong entity set
- It has partial key, which is combined with the primary key of the its strong entity set on the other-side of the relation to distinguish each record.
- Note: In most cases the design is flawed, a better design is possible where both will be strong entity sets.

Weak Entity Sets (Cont.)

- A double rectangle is used for representing a weak entity set
- The double diamond symbol is used for representing the relationship between a strong entity and a weak entity which is known as identifying relationship

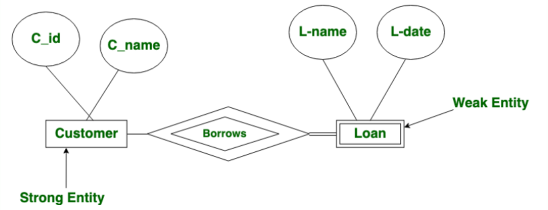


Figure: Weak Entity Set[1]

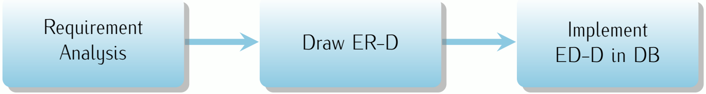
[1] image source:https://www.geeksforgeeks.org/

ER Diagram : Implementation

- In theory, there are 4 types of mappings, but we can eliminate one (since 1 to many and many to 1 becomes identical if we change the direction)
- First we will look at some realistic examples in each group before we implement them.
- Now the task goes in the following steps:

ER Diagram : Implementation

- In theory, there are 4 types of mappings, but we can eliminate one (since 1 to many and many to 1 becomes identical if we change the direction)
- First we will look at some realistic examples in each group before we implement them.
- Now the task goes in the following steps:



ER Diagram : Real-life Example

- One to One: Passport and Driving License
- One to Many: Department and students
- Many to Many: Students and Courses (taken)

ER Diagram: Implementation

- Suppose we have entity sets:
  - ✓ `student(ID (pk), name, tot_cred, dept_name)`
  - ✓ `department(dept_name(pk), building, budget)`
- We model the fact that each student has an associated department using a relationship set `stud_dept`
- The attribute `dept_name` in student below replicates information present in the relationship and is therefore redundant and it must be removed.
- BUT: when converting back to tables, in some cases the attribute gets reintroduced.

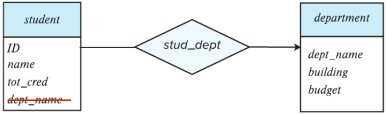


Figure: Redundant Attribute in ER-D

ER Diagram: Implementation (Cont.)

- **One to Many:** In many part use foreign key referencing the first entity.
- **One to One:** In any part (that depends on the context) use foreign key with unique constraint.
- **Many to Many:** Here a third entity (i.e. table) is needed termed as Junction Table which is formed by two foreign keys of the entities. Additional attributes may appear here.
- To ensure Total Participation in the first entity (for 1–1,1–m), additional mechanism is needed while, for Total Participation in the second entity, use NOT NULL constraint along with other constraint such as Foreign Key. (this principle is not applicable for m–m mapping)
- Remember: Total Participation may introduce additional business logic in many cases which may be impractical.

ER Diagram: Implementation– Practice

Consider the following scenario:

IUT has a number of departments. In each department, students are admitted. Each student must have ID, Name,Address. In each semester, a number of courses are offered. Each student can take a maximum of 48 courses.

- Your task is to draw the ER-D and implement it using DDLs.

Constraints on Generalization/Specialization in Database Design

Based on: Attribute of higher-level entity determines lower-level entity membeship

- **Condition-defined:** Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person. Since all the lower-level entities are evaluated on the basis of the same attribute (in this case, on age), this type of generalization is said to be attribute-defined.
- **User-defined:** Not constrained by a membership condition; rather, the database user assigns entities to a given entity set. For instance, let us assume that, after 3 months of employment, university employees are assigned to one of four work teams.

Constraints on Generalization/Specialization in Database Design

Based on: Attribute of higher-level entity determines lower-level entity membeship

- **Condition-defined:** Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person. Since all the lower-level entities are evaluated on the basis of the same attribute (in this case, on age), this type of generalization is said to be attribute-defined.
- **User-defined:** Not constrained by a membership condition; rather, the database user assigns entities to a given entity set. For instance, let us assume that, after 3 months of employment, university employees are assigned to one of four work teams.

Constraints on Generalization/Specialization in Database Design

Based on: Attribute of higher-level entity determines lower-level entity membeship

- **Condition-defined:** Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person. Since all the lower-level entities are evaluated on the basis of the same attribute (in this case, on age), this type of generalization is said to be **attribute-defined.**
- **User-defined:** Not constrained by a membership condition, rather, the **database user assigns** entities to a given entity set. **For instance,** let us assume that, after 3 months of employment, university employees are assigned to one of four work teams.

Constraints on Generalization/Specialization in Database Design

Based on: Attribute of higher-level entity determines lower-level entity membeship

- **Condition-defined:** Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person. Since all the lower-level entities are evaluated on the basis of the same attribute (in this case, on age), this type of generalization is said to be **attribute-defined.**
- **User-defined:** Not constrained by a membership condition, rather, the database user assigns entities to a given entity set. For instance, let us assume that, after 3 months of employment, university employees are assigned to one of four work teams.

Constraints on Generalization/Specialization in Database Design

Based on: Attribute of higher-level entity determines lower-level entity membeship

- **Condition-defined:** Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person. Since all the lower-level entities are evaluated on the basis of the same attribute (in this case, on age), this type of generalization is said to be **attribute-defined.**
- **User-defined:** Not constrained by a membership condition, rather, the **database user assigns** entities to a given entity set. **For instance,** let us assume that, after 3 months of employment, university employees are assigned to one of four work teams.

Constraints on Generalization/Specialization in Database Design

Based on: Attribute of higher-level entity determines lower-level entity membeship

- **Condition-defined:** Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person. Since all the lower-level entities are evaluated on the basis of the same attribute (in this case, on age), this type of generalization is said to be **attribute-defined.**
- **User-defined:** Not constrained by a membership condition, rather, the **database user assigns** entities to a given entity set. **For instance,** let us assume that, after 3 months of employment, university employees are assigned to one of four work teams.

Constraints on Generalization/Specialization (cont.)

Based on: The number of branching in its lower-level entity (i.e. leaf)

- **Disjoint.** A disjointness constraint requires that an entity belong to no more than one lower-level entity set.**For example,** Student entity can satisfy only one condition for the student type attribute; an entity can be either a graduate student or an undergraduate student, **but cannot be both.**
- **Overlapping.** The same entity **may belong to more than one lower-level entity set** within a single generalization. **For instance,** consider the employee work-team example, and assume that certain employees participate in more than one work team.

Constraints on Generalization/Specialization (cont.)

Based on: The number of branching in its lower-level entity (i.e. leaf)

- **Disjoint.** A disjointness constraint requires that an entity belong to no more than one lower-level entity set.**For example,** Student entity can satisfy only one condition for the student type attribute; an entity can be either a graduate student or an undergraduate student, **but cannot be both.**
- **Overlapping.** The same entity **may belong to more than one lower-level entity set** within a single generalization. **For instance,** consider the employee work-team example, and assume that certain employees participate in more than one work team.

Constraints on Generalization/Specialization (cont.)

Based on: Completeness

- **Total generalization or specialization.** Each higher-level entity (i.e. root) must belong to a lower-level entity set (i.e. leaf)
- **Partial generalization or specialization.** Some higher-level entities may not belong to any lower-level entity set

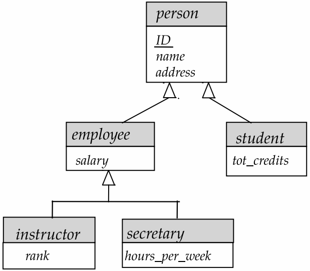
Constraints on Generalization/Specialization (cont.)

Based on: Completeness

- **Total generalization or specialization.** Each higher-level entity (i.e. root) must belong to a lower-level entity set (i.e. leaf)
- **Partial generalization or specialization.** Some higher-level entities may not belong to any lower-level entity set

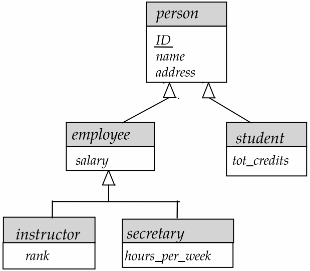
Representation of Generalization: 2 Ways

Consider the following E-R Diagram (ingore last 2 entities)



Representation of Generalization: 2 Ways

Consider the following E-R Diagram (ingore last 2 entities)



Representation of Generalization: Method 1

Method 1

- Create a schema for the higher–level entity set.
- For each lower–level entity set create it and link it with its upper–level schema.
- The primary–key attributes of the higher–level entity set become primary–key attributes of the higher–level entity set as well as all lower–level entity sets.
- We create foreign–key constraints on the lower–level entity sets.

Representation of Generalization: Method 1

Method 1

- Create a schema for the higher-level entity set.
- For each lower-level entity set create it and link it with its upper-level schema.
- The primary-key attributes of the higher-level entity set become primary-key attributes of the higher-level entity set as well as all lower-level entity sets.
- We create foreign-key constraints on the lower-level entity sets.

Representation of Generalization: Method 1

Method 1

- Create a schema for the higher-level entity set.
- For each lower-level entity set create it and link it with its upper-level schema.
- The primary-key attributes of the higher-level entity set become primary-key attributes of the higher-level entity set as well as all lower-level entity sets.
- We create foreign-key constraints on the lower-level entity sets.

Representation of Generalization: Method 1

Method 1

- Create a schema for the higher-level entity set.
- For each lower-level entity set create it and link it with its upper-level schema.
- The primary-key attributes of the higher-level entity set become primary-key attributes of the higher-level entity set as well as all lower-level entity sets.
- We create foreign-key constraints on the lower-level entity sets.

Representation of Generalization: Method 1

Method 1

- Create a schema for the higher-level entity set.
- For each lower-level entity set create it and link it with its upper-level schema.
- The primary-key attributes of the higher-level entity set become primary-key attributes of the higher-level entity set as well as all lower-level entity sets.
- We create foreign-key constraints on the lower-level entity sets.

Representation of Generalization: Method 1

Method 1

- Create a schema for the higher-level entity set.
- For each lower-level entity set create it and link it with its upper-level schema.
- The primary-key attributes of the higher-level entity set become primary-key attributes of the higher-level entity set as well as all lower-level entity sets.
- We create foreign-key constraints on the lower-level entity sets.

Representation of Generalization: Method 1 (Cont.)

Entities should look like:

person (ID , name, street, city)

employee (ID , salary)

student (ID , totalcredit)

Strength and Weakness of Method 1

Strengh: Natural in design and reduces redundancies.
Weakness: Getting information requires joining 2 entities.

Representation of Generalization: Method 1 (Cont.)

Entities should look like:

person (ID , name, street, city)

employee (ID , salary)

student (ID , totalcredit)

Strength and Weakness of Method 1

Strengh: Natural in design and reduces redundancies.
Weakness: Getting information requires joining 2 entities.

Representation of Generalization: Method 1 (Cont.)

Entities should look like:

person (ID , name, street, city)

employee (ID , salary)

student (ID , totalcredit)

Strength and Weakness of Method 1

Strengh: Natural in design and reduces redundancies.

Weakness: Getting information requires joining 2 entities.

Representation of Generalization: Method 1 (Cont.)

Entities should look like:

person (ID , name, street, city)

employee (ID , salary)

student (ID , totalcredit)

Strength and Weakness of Method 1

Strengh: Natural in design and reduces redundancies.

Weakness: Getting information requires joining 2 entities.

Representation of Generalization: Method 1 (Cont.)

Entities should look like:

person (ID , name, street, city)

employee (ID , salary)

student (ID , totalcredit)

Strength and Weakness of Method 1

Strengh: Natural in design and reduces redundancies.

Weakness: Getting information requires joining 2 entities.

Representation of Generalization: Method 1 (Cont.)

Entities should look like:

person (ID , name, street, city)

employee (ID , salary)

student (ID , totalcredit)

Strength and Weakness of Method 1

Strengh: Natural in design and reduces redundancies.

Weakness: Getting information requires joining 2 entities.

Representation of Generalization: Method 2

It can be applied if:

- If the generalization is disjoint
- If the generalization is complete

It implies:

If no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher-level entity set is also a member of one of the lower-level entity sets

Its ERD should look like:

employee (ID , name, street, city, salary)

student (ID , name, street, city, totalcredit)

Representation of Generalization: Method 2

It can be applied if:

- If the generalization is **disjoint**
- If the generalization is **complete**

It implies:

If no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher-level entity set is also a member of one of the lower-level entity sets

Its ERD should look like:

employee (ID , name, street, city, salary)

student (ID , name, street, city, totalcredit)

Representation of Generalization: Method 2

It can be applied if:

- If the generalization is **disjoint**
- If the generalization is **complete**

It implies:

If no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher-level entity set is also a member of one of the lower-level entity sets

Its ERD should look like:

employee (ID , name, street, city, salary)

student (ID , name, street, city, totalcredit)

Representation of Generalization: Method 2

It can be applied if:

- If the generalization is **disjoint**
- If the generalization is **complete**

It implies:

If no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher-level entity set is also a member of one of the lower-level entity sets

Its ERD should look like:

employee (ID , name, street, city, salary)

student (ID , name, street, city, totalcredit)

Representation of Generalization: Method 2

It can be applied if:

- If the generalization is **disjoint**
- If the generalization is **complete**

It implies:

If no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher-level entity set is also a member of one of the lower-level entity sets

Its ERD should look like:

employee (ID , name, street, city, salary)

student (ID , name, street, city, totalcredit)

Representation of Generalization: Method 2

It can be applied if:

- If the generalization is **disjoint**
- If the generalization is **complete**

It implies:

If no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher-level entity set is also a member of one of the lower-level entity sets

Its ERD should look like:

employee (ID , name, street, city, salary)

student (ID , name, street, city, totalcredit)

Method 2: Strength and weakness

- **Strength:** Data can be obtained directly, quick access (since no joining is done).
- **Weakness:** Data validation is failed (since no Foreign Key constraint can be imposed).
- **Weakness:** With **overlapping generalization**, some values would be stored multiple times, introduces **redundancies.** For instance, if a person is both an employee and a student, values for street and city would be stored twice.

Method 2: Strength and weakness

- **Strength:** Data can be obtained directly, quick access (since no joining is done).
- **Weakness:** Data validation is failed (since no Foreign Key constraint can be imposed).
- **Weakness:** With **overlapping generalization**, some values would be stored multiple times, introduces **redundancies.** For instance, if a person is both an employee and a student, values for street and city would be stored twice.

Method 2: Strength and weakness

- **Strength:** Data can be obtained directly, quick access (since no joining is done).
- **Weakness:** Data validation is failed (since no Foreign Key constraint can be imposed).
- **Weakness:** With **overlapping generalization**, some values would be stored multiple times, introduces **redundancies.** For instance, if a person is both an employee and a student, values for street and city would be stored twice.