

ENTITY-RELATIONSHIP MODEL

E- R DATA MODELING

- An **entity** is an object that exists and is distinguishable from other objects.
 - Example: specific person, company, event, plant
- Entities have **attributes**
 - Example: people have *names* and *addresses*
 -
- 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

ATTRIBUTES

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

- Example:

instructor = (ID, name, street, city, salary)

course= (course_id, title, credits)

- **Domain** – the set of permitted values for each attribute

- Attribute types:

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

TYPES OF ATTRIBUTES

Simple Attribute: Attribute that consist of a single atomic value.

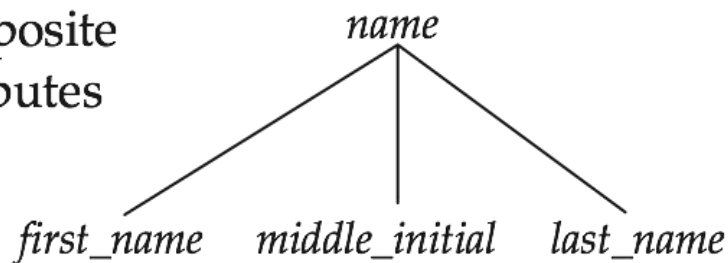
Example: Salary

Composite Attribute : Attribute value not atomic.

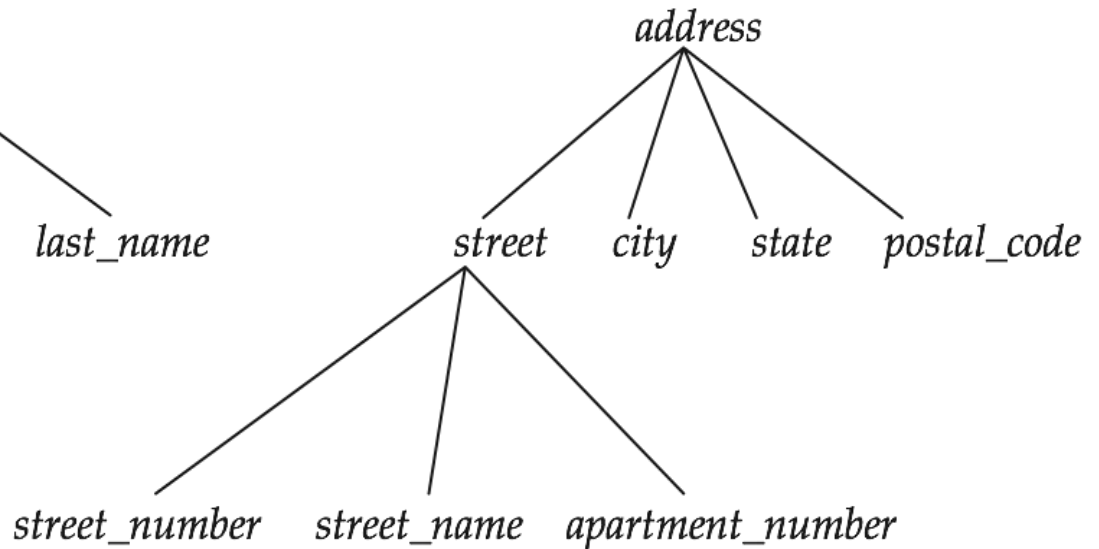
Example : Address : 'House_no:City:State

Name : 'First Name: Middle Name: Last Name'

composite
attributes



component
attributes



TYPES OF ATTRIBUTES

Single Valued Attribute: Attribute that hold a single value

Example1: City

Example2: Customer id

Multi Valued Attribute: Attribute that hold multiple values.

Example1: A customer can have multiple phone numbers, email id's etc

Example2: A person may have several college degrees

Derived Attribute: An attribute that's value is derived from a stored attribute.

Example : age, and it's value is derived from the stored attribute Date of Birth.

ENTITY SETS *INSTRUCTOR* AND *STUDENT*

instructor_ID instructor_name

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

instructor

student-ID student_name

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

student

RELATIONSHIP SETS

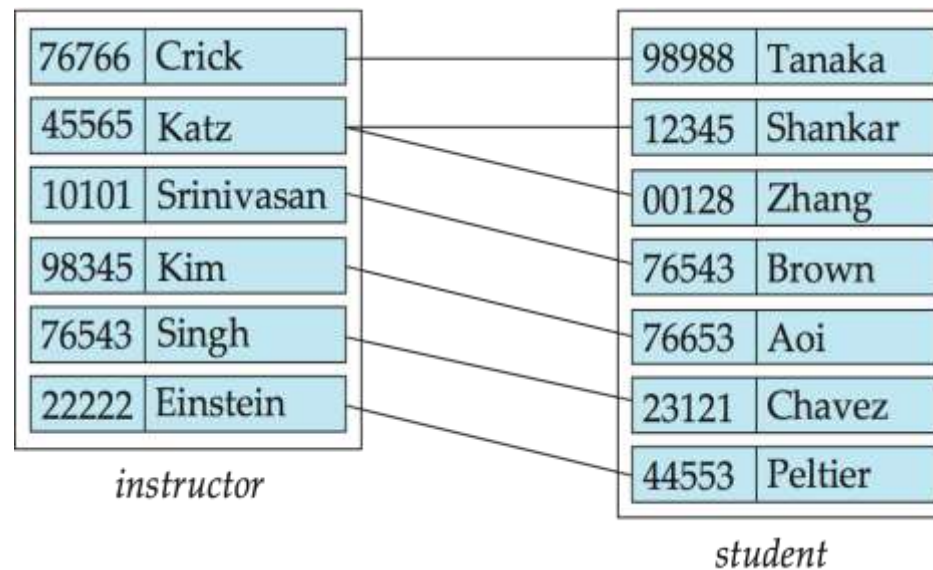
- A **relationship** is an association among several entities

Example:

44553 (Peltier)
student entity

advisor
relationship set

22222 (Einstein)
instructor entity



ENTITY-RELATIONSHIP DIAGRAMS



o Representing entities

- we represent an entity by a named rectangle
- use a singular noun, or adjective + noun
- refer to one instance in naming

∞



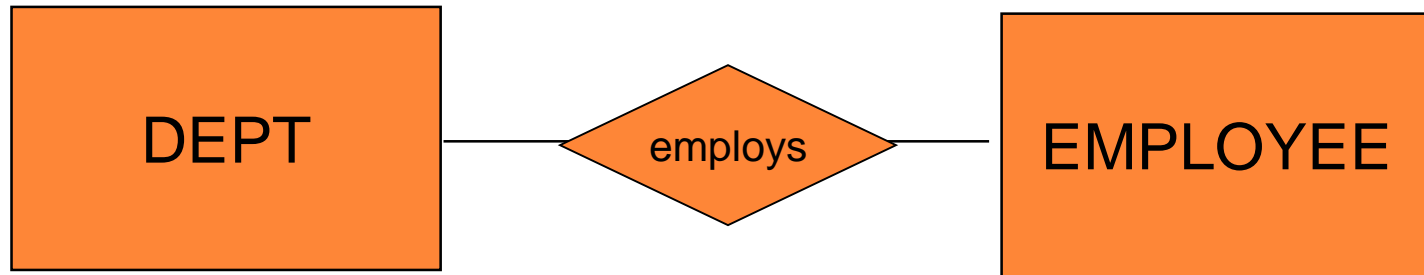
CUSTOMER

PART-TIME
EMPLOYEE

ENTITY-RELATIONSHIP DIAGRAMS



- Representing relationship



ENTITY-RELATIONSHIP DIAGRAMS

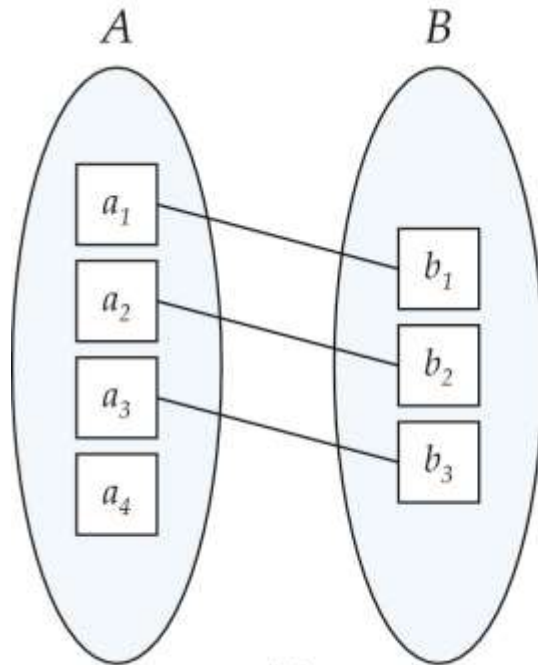


◦ Types of Relationships

- Three types of relationships can exist between entities
- One-to-one relationship (1:1): One instance in an entity (parent) refers to one and only one instance in the related entity (child).
- One-to-many relationship (1:M): One instance in an entity (parent) refers to one or more instances in the related entity (child)

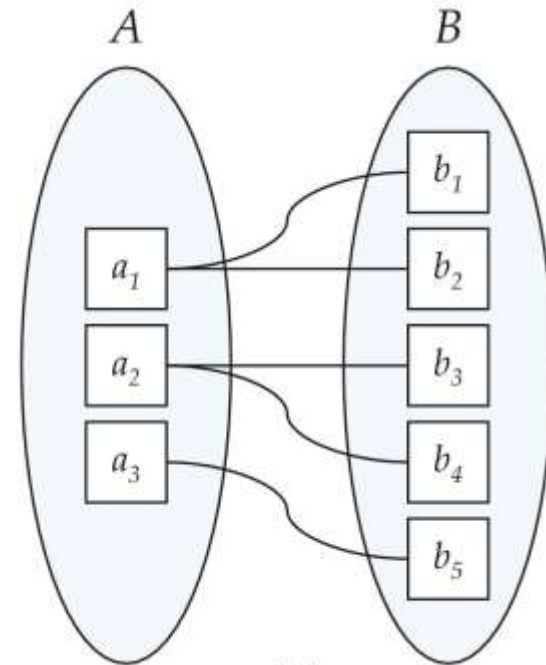


ENTITY-RELATIONSHIP DIAGRAMS



(a)

One to one



(b)

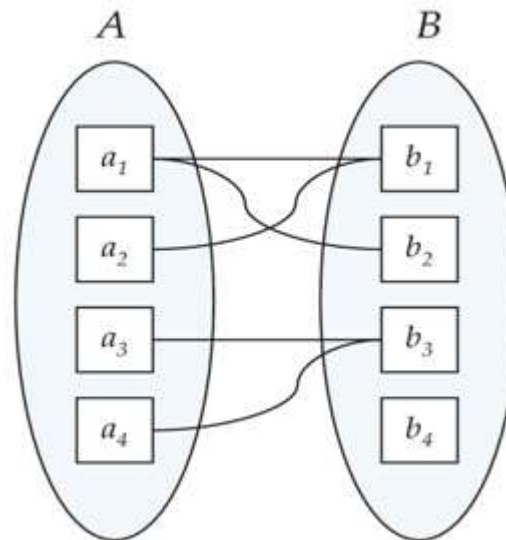
One to many



ENTITY-RELATIONSHIP DIAGRAMS

Types of Relationships

- Many-to-many relationship (M:N): exists when one instance of the first entity (parent) can relate to many instances of the second entity (child), and one instance of the second entity can relate to many instances of the first entity.



Many to many



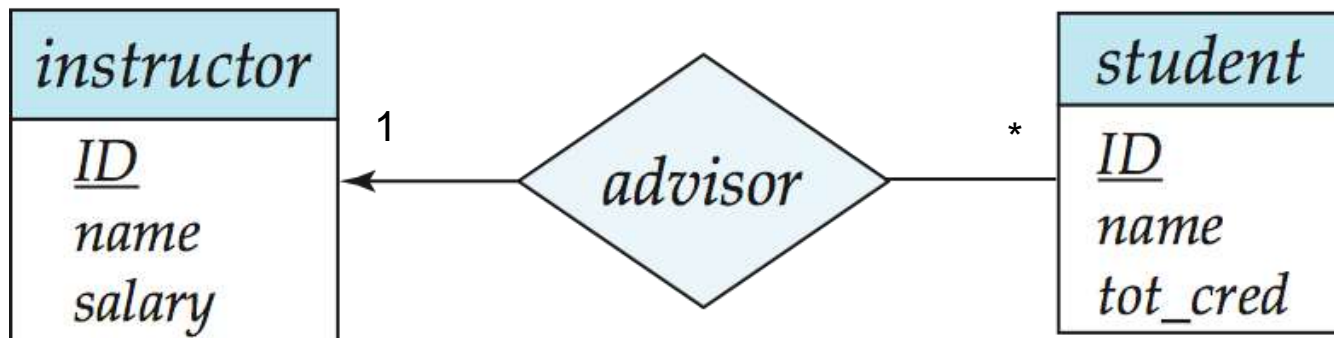
CARDINALITY CONSTRAINTS

- We express cardinality constraints by drawing either a directed line (\rightarrow), signifying “one,” or an undirected line ($—$), signifying “many,” between the relationship set and the entity set.
- Or, by numbering each entity. * or, m for many.
- One-to-one relationship:
 - 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*



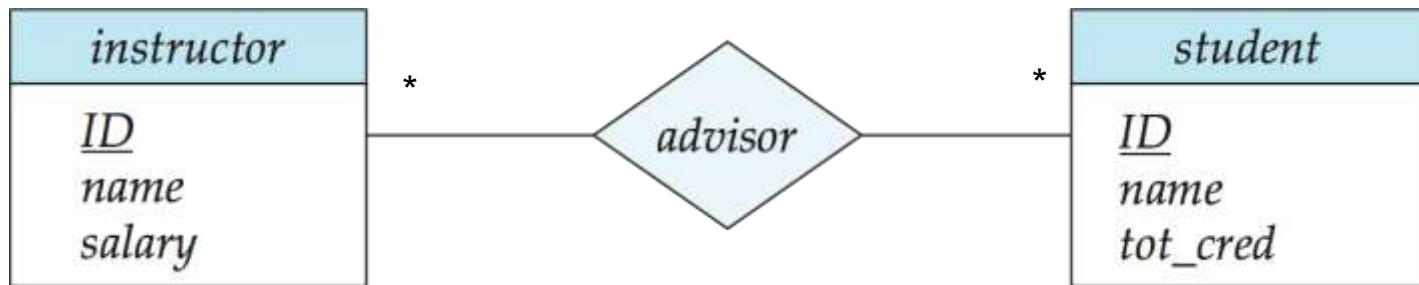
ONE-TO-MANY RELATIONSHIP

- 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-MANY RELATIONSHIP

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

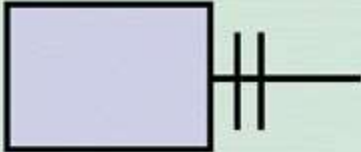





ENTITY-RELATIONSHIP DIAGRAMS



- Crow's foot notation: A type of cardinality notation. It is called crow's foot notation because of the shapes, which include circles, bars, and symbols, that indicate various possibilities.
- A single bar indicates one, a double bar indicates one and only one, a circle indicates zero, and a crow's foot indicates many.

ENTITY-RELATIONSHIP DIAGRAMS

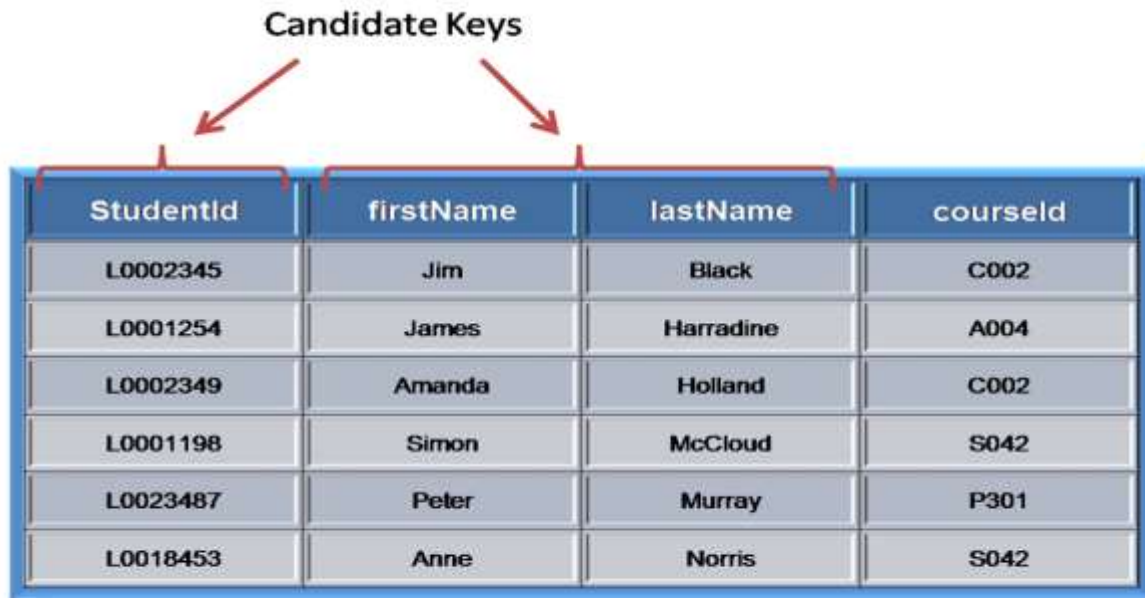
SYMBOL	MEANING	UML REPRESENTATION
	One and only one	1
	One or many	1..*
	Zero, or one, or many	0..*
	Zero, or one	0..1

Crow's foot notation is a common method of indicating cardinality. The four examples show how you can use various symbols to describe the relationships between entities.

DIFFERENT TYPES OF KEYS

- A **candidate key** of an entity set is a minimal super key
 - *ID* is candidate key of *instructor*
 - *course_id* is candidate key of *course*

Candidate Keys



StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

PRIMARY KEY

- A primary key is a candidate key that is most appropriate to be the main reference key for the table. As its name suggests, it is the primary key of reference for the table and is used throughout the database to help establish relationships with other tables.
- **The primary key must contain unique values, must never be null and uniquely identify each record in the table**

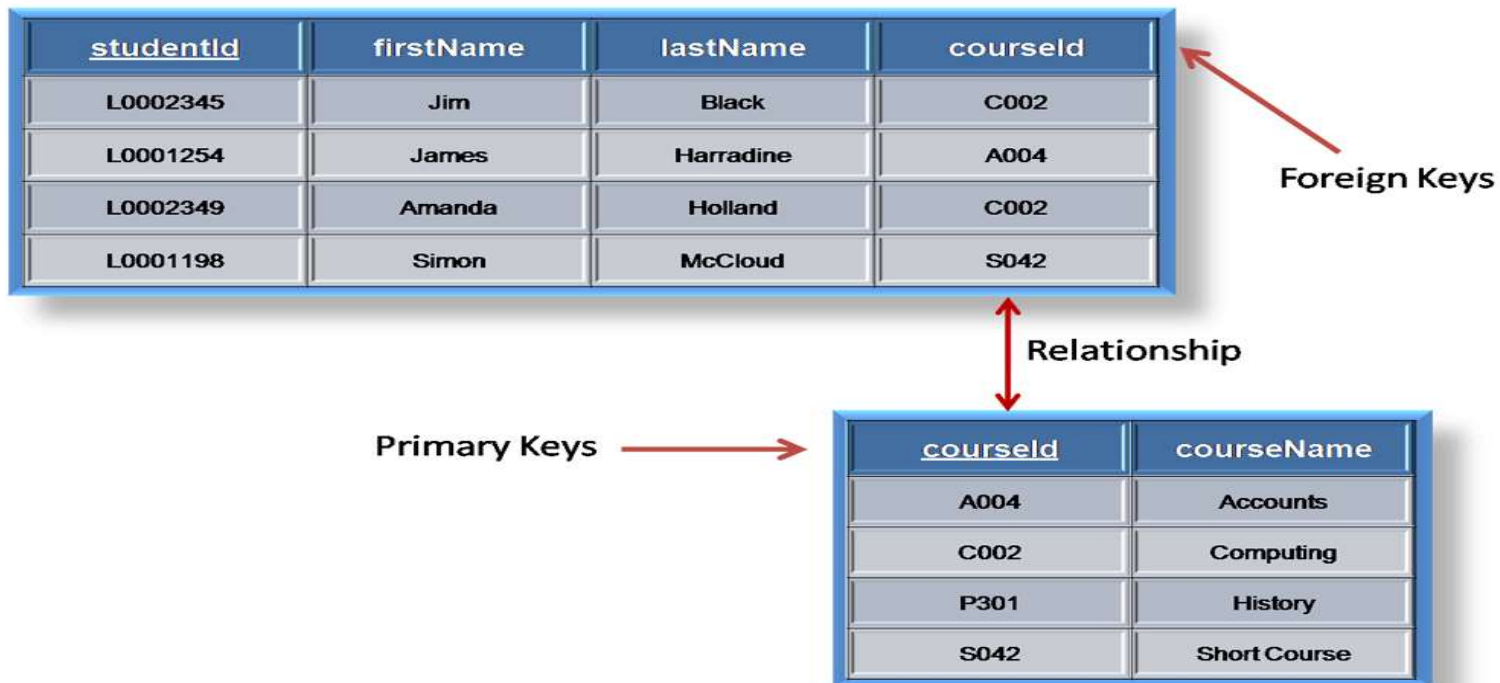
Primary Keys



<u>StudentId</u>	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

FOREIGN KEY

- A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second. In other words, if we had a table A with a primary key X that linked to a table B where X was a field in B, then X would be a foreign key in B



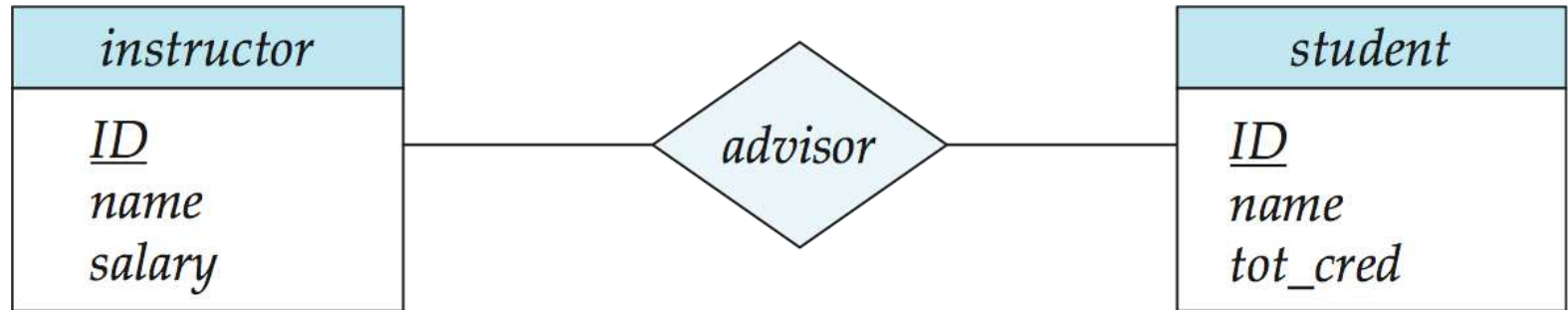
DIFFERENT TYPES OF KEYS

A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.

Example:

- {Student ID, FirstName }
- {Student ID, LastName }
- {Student ID, FirstName, LastName }

E-R DIAGRAMS

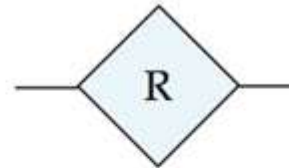


- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle. Or , as oval shape along with the rectangle.
- Underline indicates primary key attributes

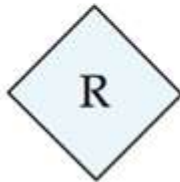
SUMMARY OF SYMBOLS USED IN E-R NOTATION



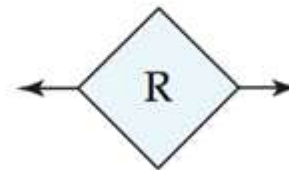
entity set



many-to-many
relationship



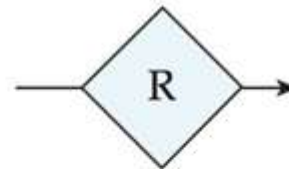
relationship set



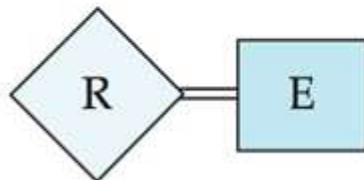
one-to-one
relationship



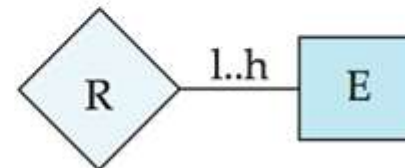
identifying
relationship set
for weak entity set



many-to-one
relationship



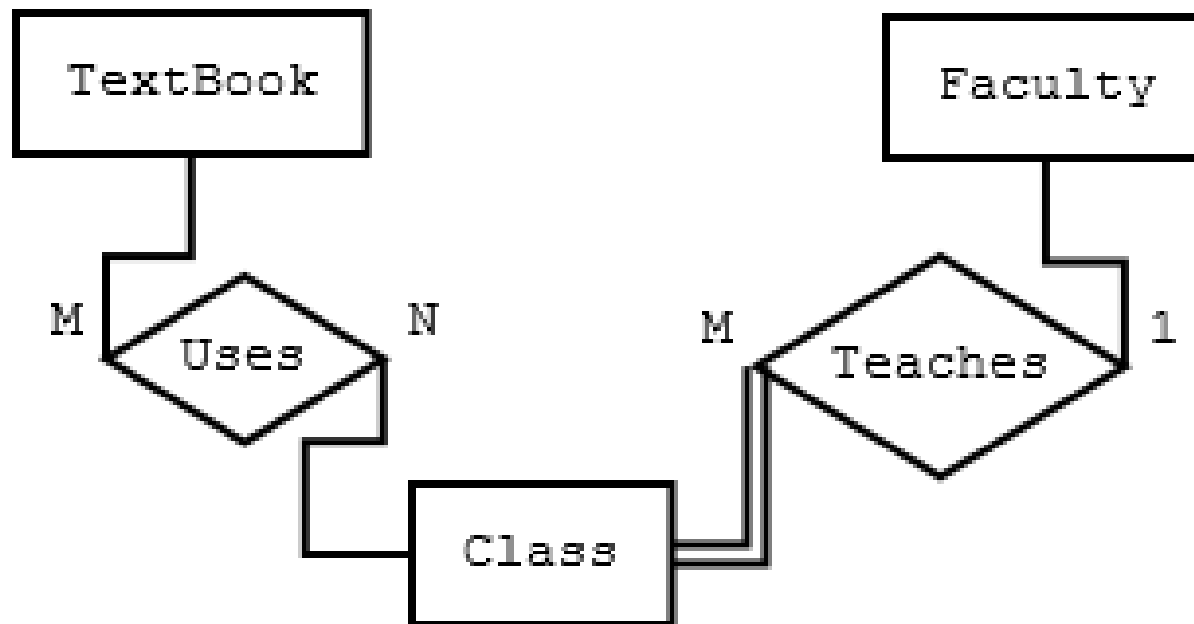
total participation
of entity set in
relationship



cardinality
limits

TOTAL PARTICIPATION OF ENTITY SET

- E.g., A *Class* entity cannot exist unless related to a *Faculty* member entity



WEAK ENTITY SETS

An entity set that does not have a primary key is referred to as a **weak entity set**

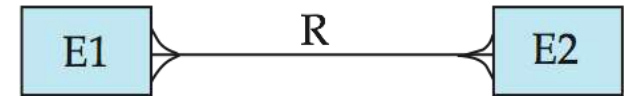
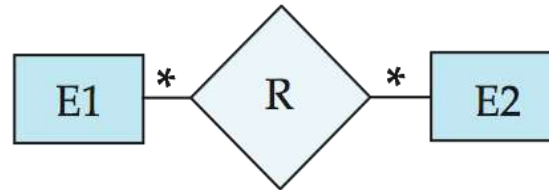
- We underline the discriminator of a weak entity set with a dashed line.
- We put the identifying relationship of a weak entity in a double diamond.
- Primary key for *section* – (*course_id*, *sec_id*, *semester*, *year*)



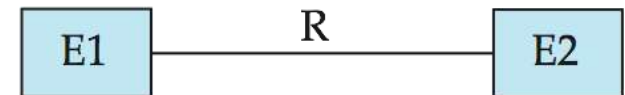
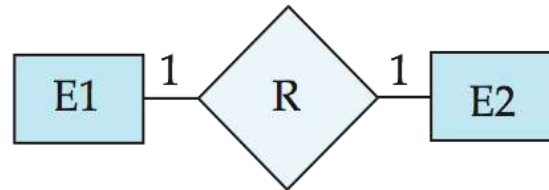
SUMMARY OF SYMBOLS USED IN E-R NOTATION

Crows feet notation

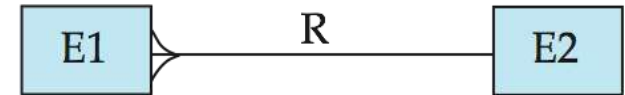
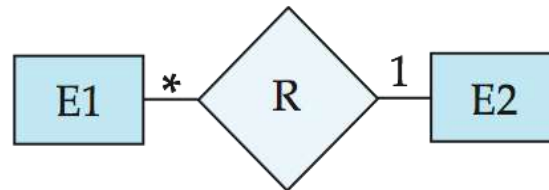
many-to-many
relationship



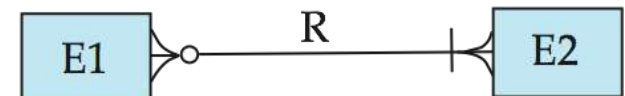
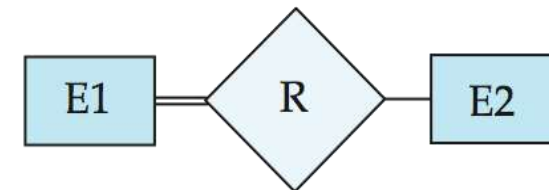
one-to-one
relationship



many-to-one
relationship

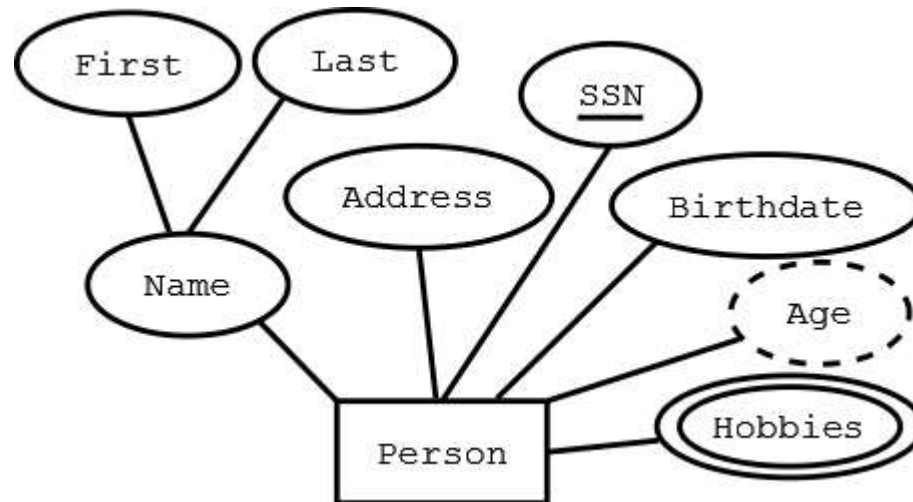


participation
in R: total (E1)
and partial (E2)



SUMMARY OF SYMBOLS USED IN E-R NOTATION

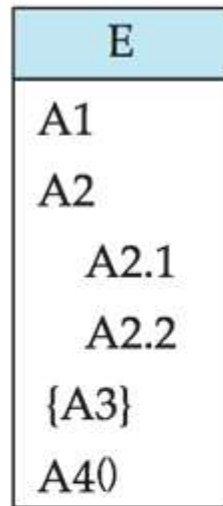
o Representing attributes



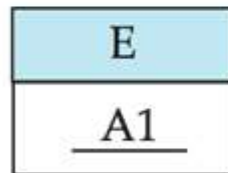
- **Rectangle** -- Entity
- **Ellipses** -- Attribute (underlined attributes are [part of] the primary key)
- **Double ellipses** -- multi-valued attribute
- **Dashed ellipses**-- derived attribute, e.g. age is derivable from birthdate and current date.

SUMMARY OF SYMBOLS USED IN E-R NOTATION

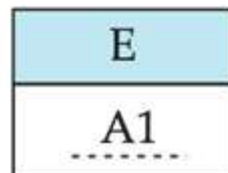
o Representing attributes



attributes:
simple (A1),
composite (A2) and
multivalued (A3)
derived (A4)

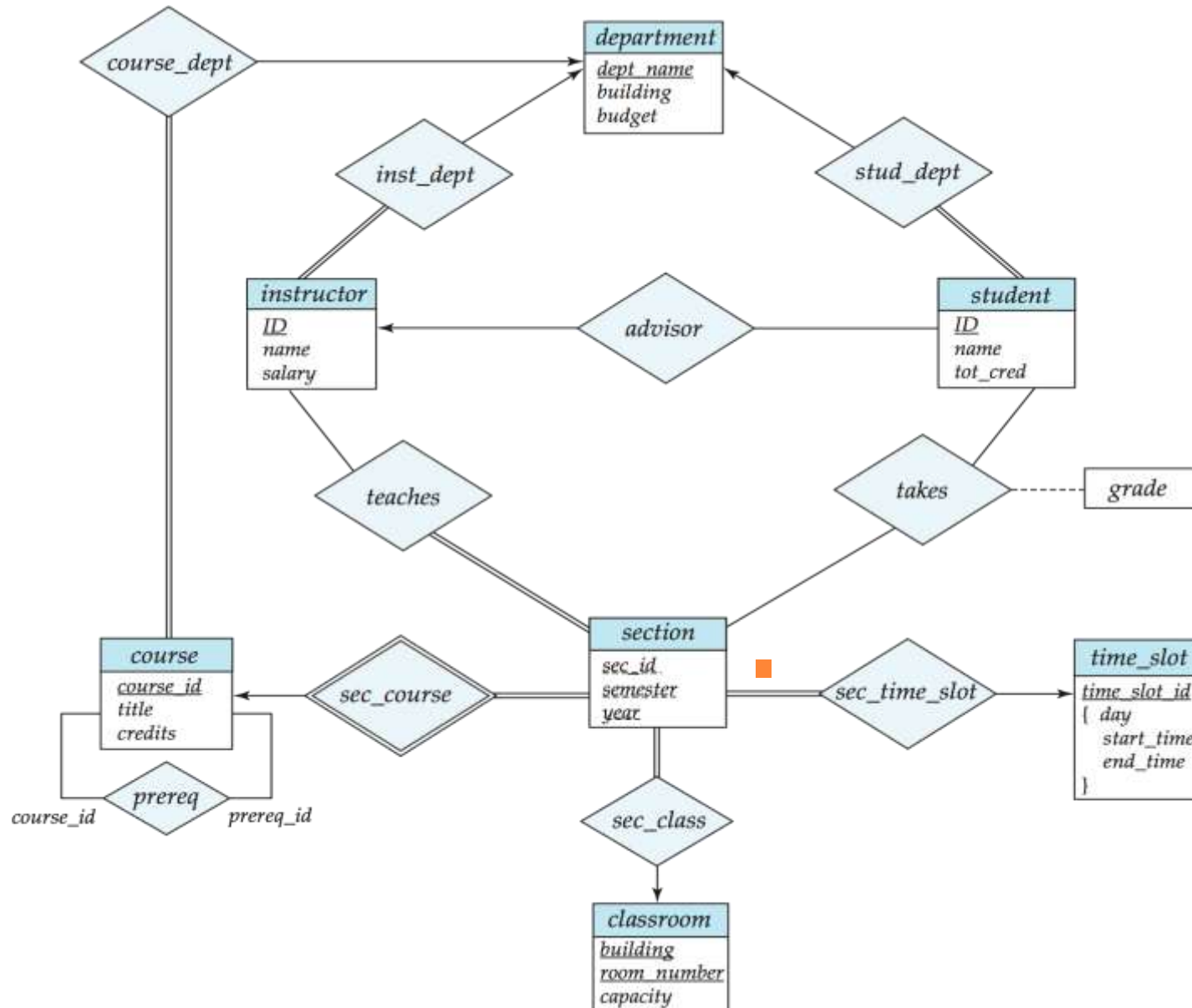


primary key



discriminating
attribute of
weak entity set

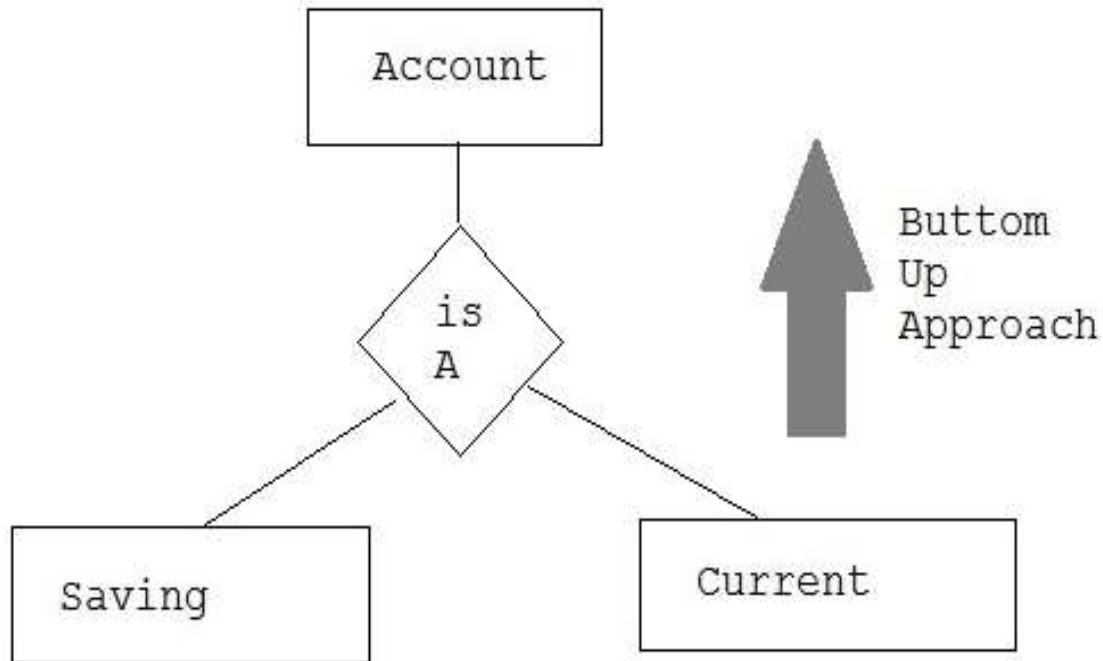
E-R DIAGRAM FOR A UNIVERSITY



Extended ER Features: Generalization

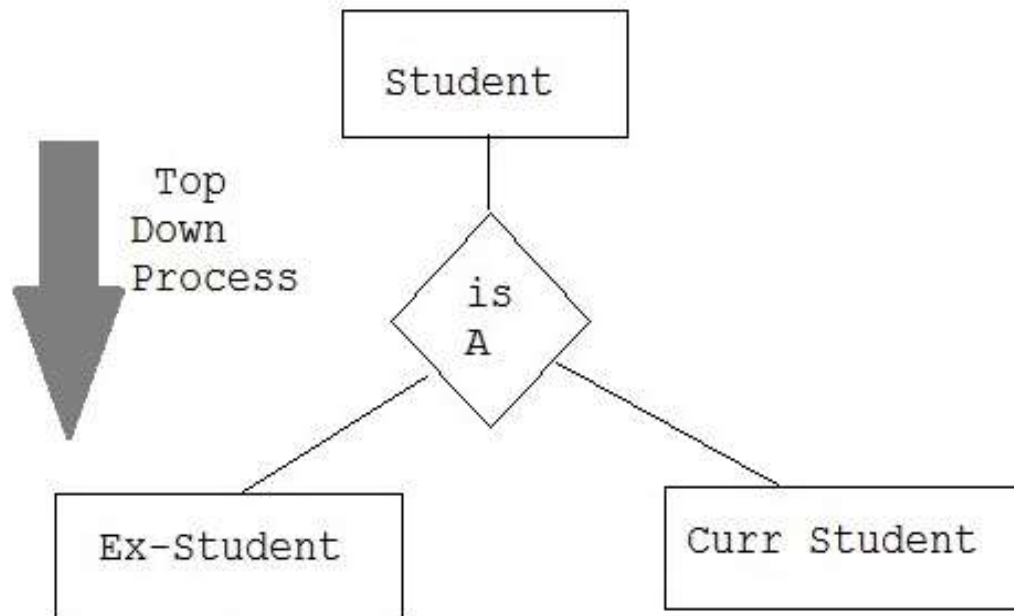
Generalization

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entity to make further higher level entity.



Specialization

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, some higher level entities may not have lower-level entity sets at all.



EXTENDED ER FEATURES: GENERALIZATION

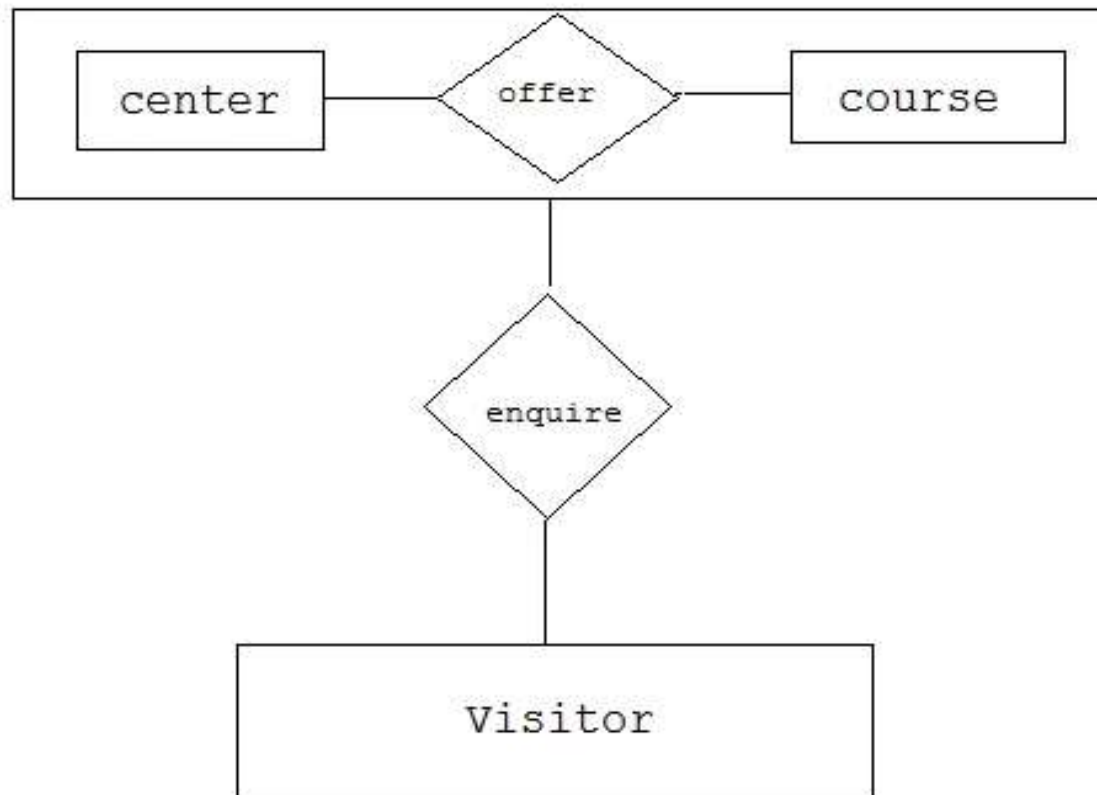
- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization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

SPECIALIZATION AND GENERALIZATION (CONT.)

- Can have multiple specializations of an entity set based on different features.
- E.g., *permanent_employee* vs. *temporary_employee*, in addition to *instructor* vs. *secretary*
- Each particular employee would be
 - a member of one of *permanent_employee* or *temporary_employee*,
 - and also a member of one of *instructor*, *secretary*
- The ISA relationship also referred to as **superclass - subclass** relationship

Aggregation

Aggregation is a process when relation between two entity is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.

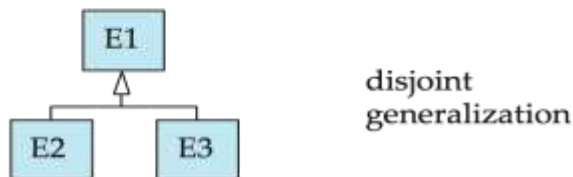
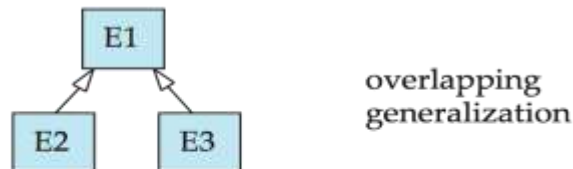
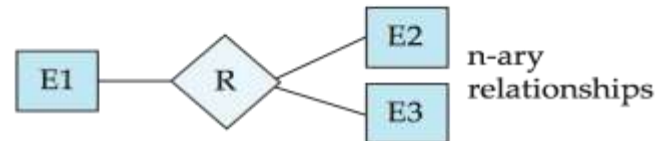
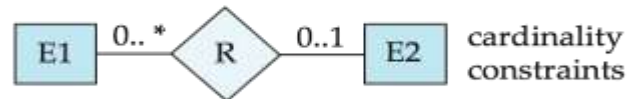
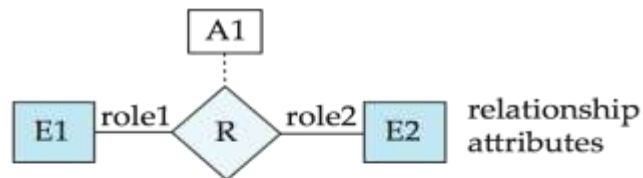
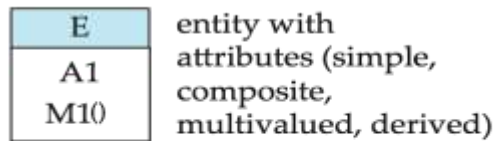


UML

- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.

FIGURE 7.26

ER Diagram Notation



Equivalent in UML

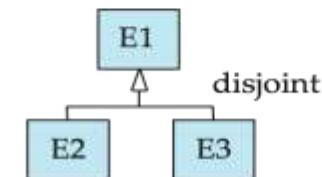
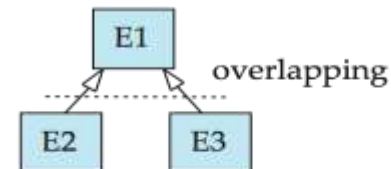
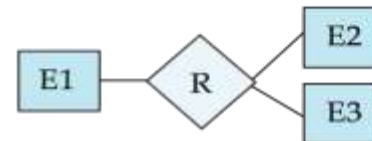
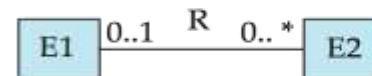
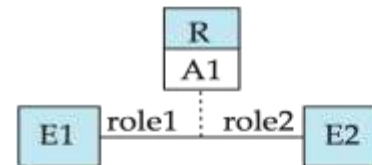
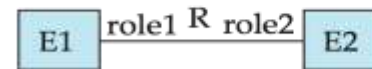
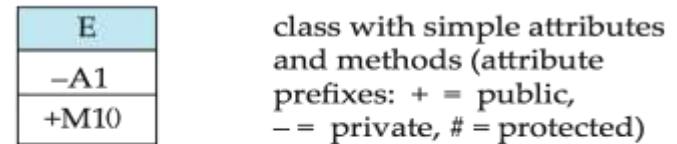
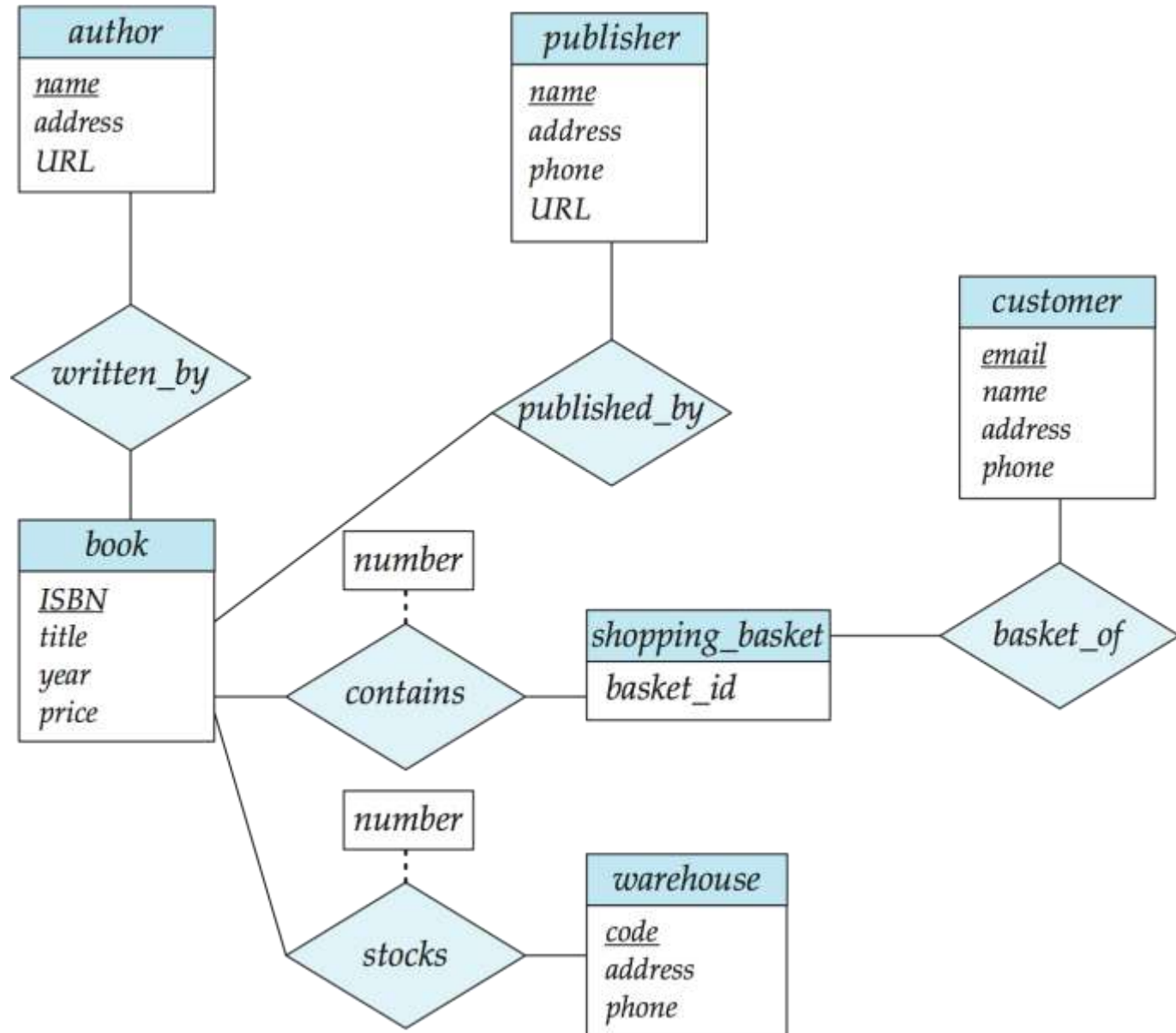


FIGURE 7.29



Thank you