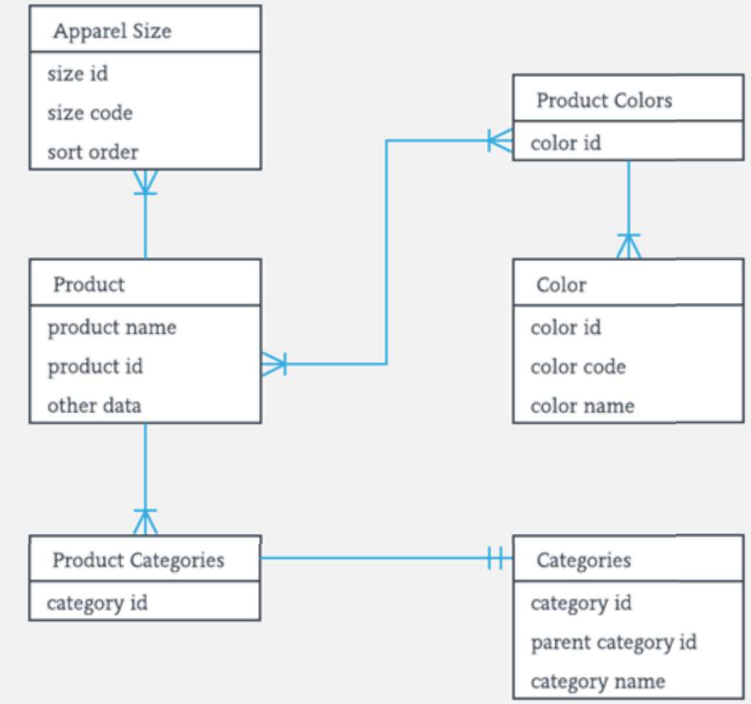
What is ER Diagrams?

- ENTITY-RELATIONSHIP DIAGRAM (ERD) displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.



Facts

- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modeling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities

Why use

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD is allowed you to communicate with the logical structure of the database to users

Components of the ER Diagram

- This model is based on three basic concepts:
  - Entities
  - Attributes
  - Relationships
- Example

For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.

Entity Name

Entity

Person,place,object,event or concept about which data is to be maintained

Example: Car, Student

Jack

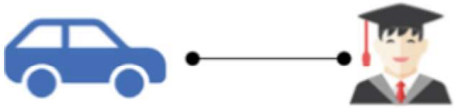
Attribute Name

Attribute

Property or characteristic of an entity

Example: Color of car Entity

Name of Student Entity



Relation

Verb Phrase

Association between the instances of one or more entity types

Example: Blue Car Belongs to Student Jack

Entity

- A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.

Examples of entities:

- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

Entity (Cont.) Notation of an Entity

- **Entity set:**

Student

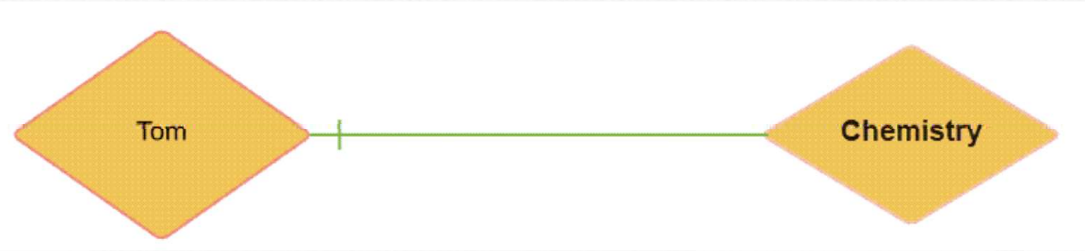
An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values. Entities are represented by their properties, which also called attributes. All attributes have their separate values. For example, a student entity may have a name, age, class, as attributes.

Example of Entities:

- A university may have some departments. All these departments employ various lecturers and offer several programs.
- Some courses make up each program. Students register in a particular program and enroll in various courses. A lecturer from the specific department takes each course, and each lecturer teaches a various group of students.

Relationship

Relationship is nothing but an association among two or more entities.
E.g., Tom works in the Chemistry department.



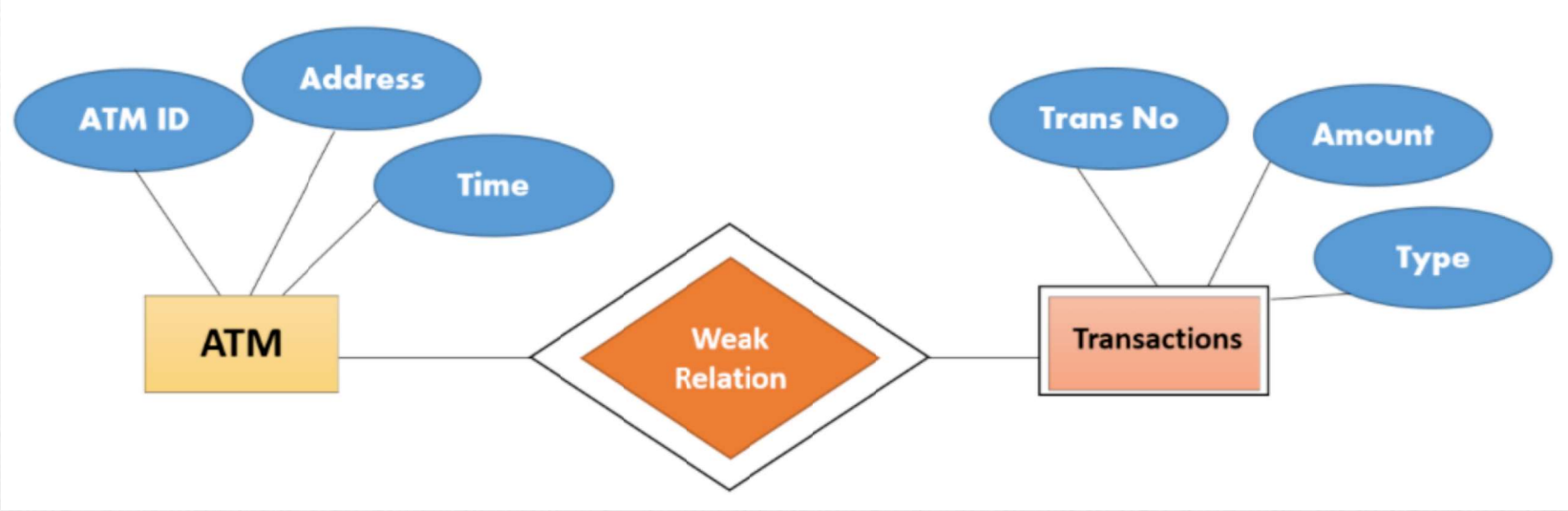
Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

For example:

- You are attending this lecture
- I am giving the lecture
- Just look entities, we can classify relationships according to relationship-types:
- A student attends a lecture
- A lecturer is giving a lecture.

Weak Entities

A weak entity is a type of entity which doesn't have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets need to have participation.



In above example, "Trans No" is a discriminator within a group of transactions in an ATM.

Weak Entity by comparing it with a Strong Entity

Strong Entity Set

- Strong entity set always has a primary key.
- It represented by a rectangle symbol.

Weak Entity Set

- It does not have enough attributes to build a primary key.
- It is represented by a double rectangle symbol.

Weak Entity by comparing it with a Strong Entity (Cont.)

Strong Entity Set

- It contains a Primary key represented by the underline symbol.
- The member of a strong entity set is called a dominant entity set.

Weak Entity Set

- It contains a Partial Key which is represented by a dashed underline symbol.
- The member of a weak entity set called as a subordinate entity set.

Weak Entity by comparing it with a Strong Entity (Cont.)

Strong Entity Set

- Primary Key is one of its attributes which helps to identify its member.
- In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.

Weak Entity Set

- In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
- The relationship between one strong and a weak entity set shown by using the double diamond symbol.

Weak Entity by comparing it with a Strong Entity

Strong Entity Set

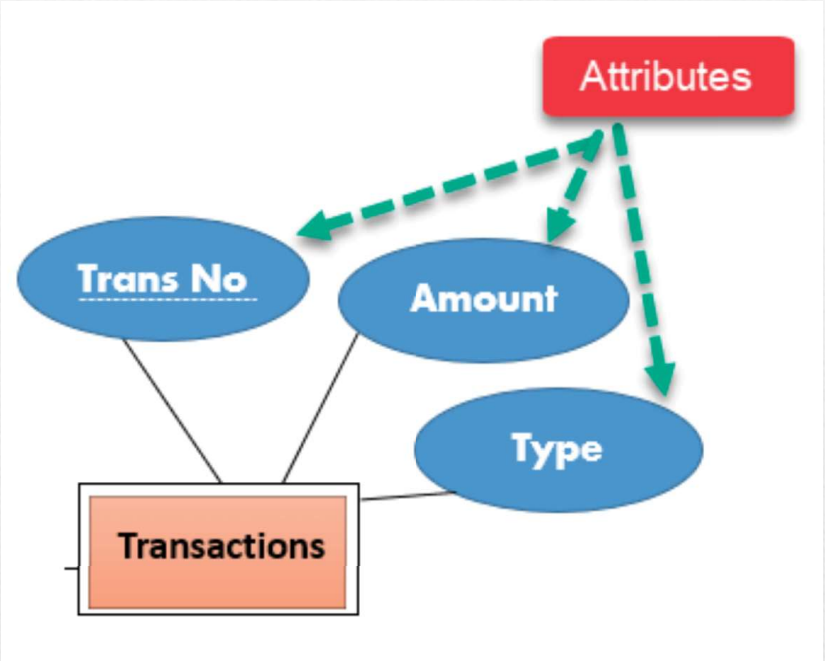
- The connecting line of the strong entity set with the relationship is single.

Weak Entity Set

- The line connecting the weak entity set for identifying relationship is double.

Attributes

- It is a single-valued property of either an entity-type or a relationship-type.
- For example, a lecture might have attributes: time, date, duration, place, etc.
- An attribute is represented by an Ellipse



Attributes (Cont.)

Types of Attributes	Description
• Simple attribute	• Simple attributes can't be divided any further. For example, a student's contact number. It is also called an atomic value.

Attributes (Cont.)

Types of Attributes	Description
• Composite attribute	• It is possible to break down composite attribute. For example, a student's full name may be further divided into first name, second name, and last name.

Attributes (Cont.)

Types of Attributes	Description
• Derived attribute	• This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee.

Attributes (Cont.)

Types of Attributes	Description
• Multivalued attribute	• Multivalued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc.

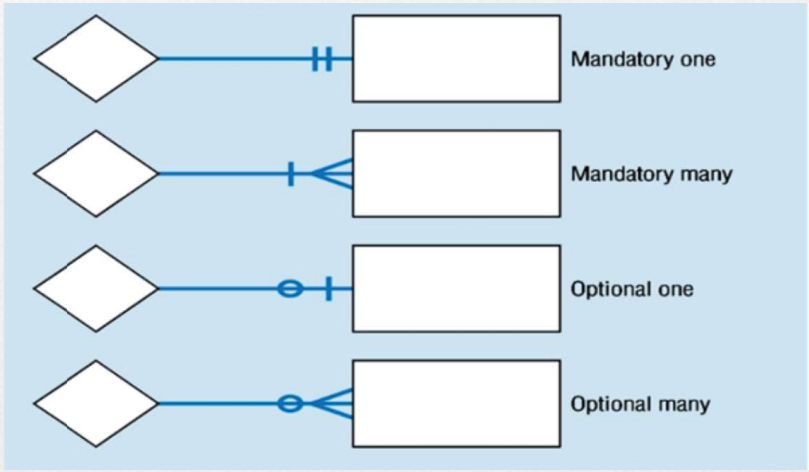
Cardinality

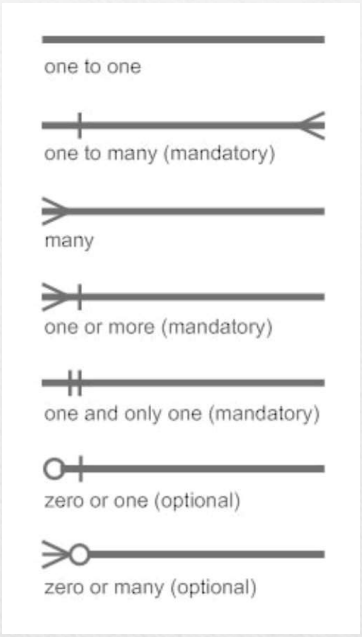
Defines the numerical attributes of the relationship between two entities or entity sets.

Different types of cardinal relationships are:

1. One-to-One Relationships
2. One-to-Many Relationships
3. May to One Relationships
4. Many-to-Many Relationships

Cardinality (Notation)

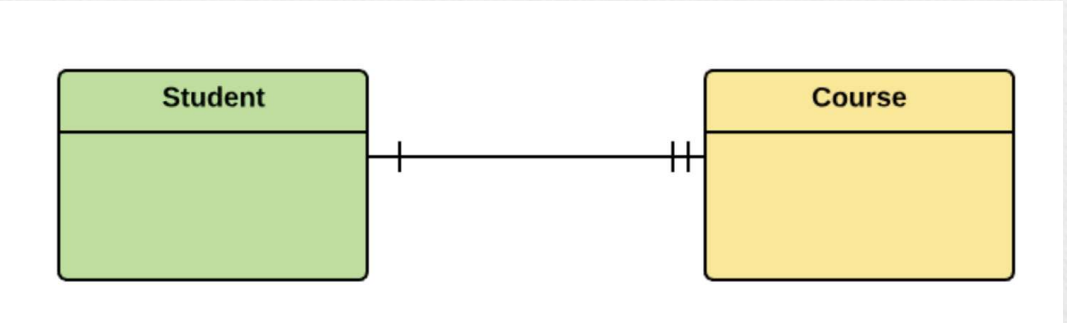




1. One-to-one

- One entity from entity set X can be associated with at most one entity of entity set Y and vice versa.

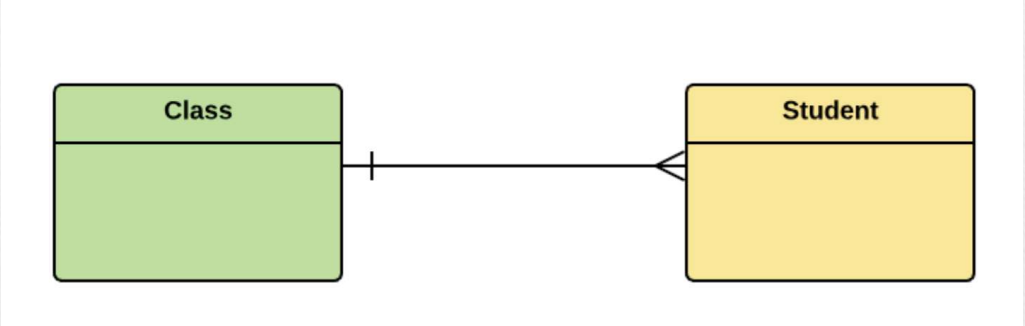
Example: One student can register for numerous courses. However, all those courses have a single line back to that one student.



2. One-to-many

- One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity.

For example, one class is consisting of multiple students.



3. Many to one

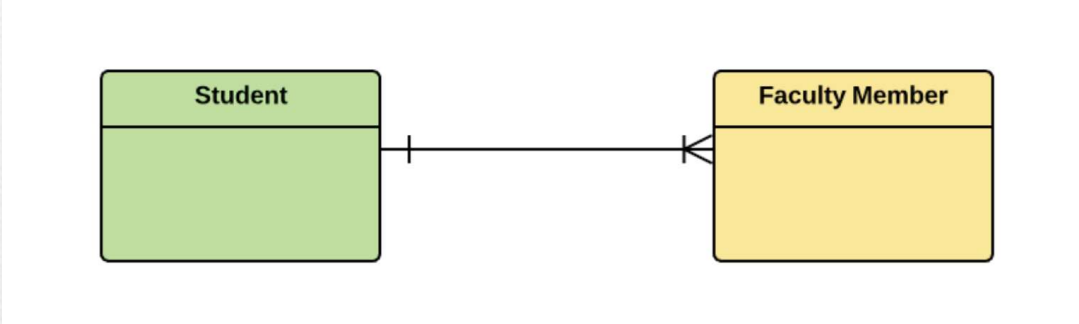
- More than one entity from entity set X can be associated with at most one entity of entity set Y. However, an entity from entity set Y may or may not be associated with more than one entity from entity set X.

For example, many students belong to the same class.

4. Many to many

- One entity from X can be associated with more than one entity from Y and vice versa.

For example, Students as a group are associated with multiple faculty members, and faculty members can be associated with multiple students.



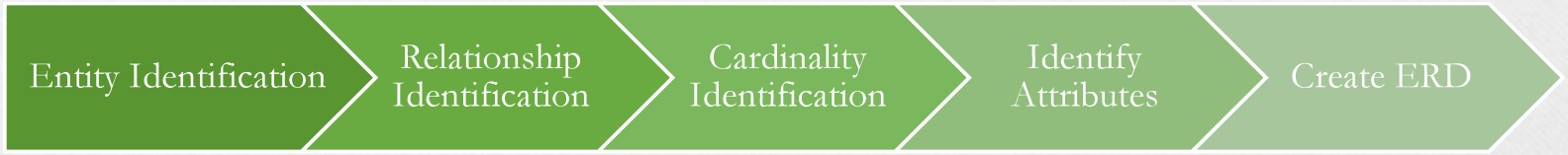
ERD Notations

ER- Diagram is a visual representation of data that describe how data is related to each other.

- **Rectangles:** This symbol represent entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

ERD Notations

Steps to create an ERD



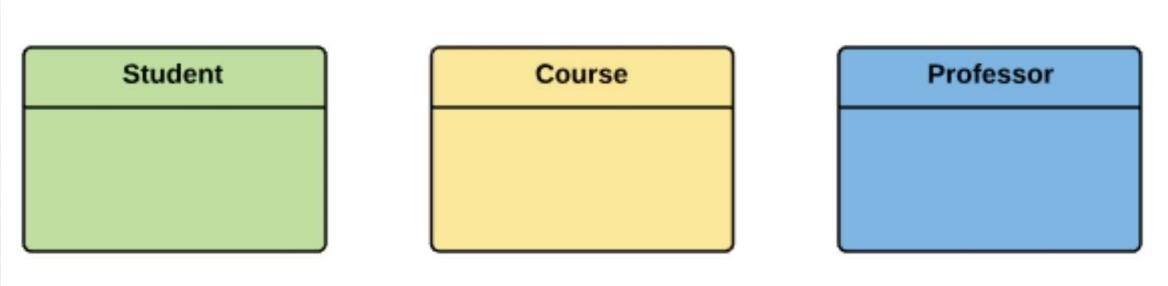
Create with example

- In a university, a Student enrolls in Courses. A student must be assigned to at least one or more Courses. Each course is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course

Step 1. Entity Identification

We have three entities

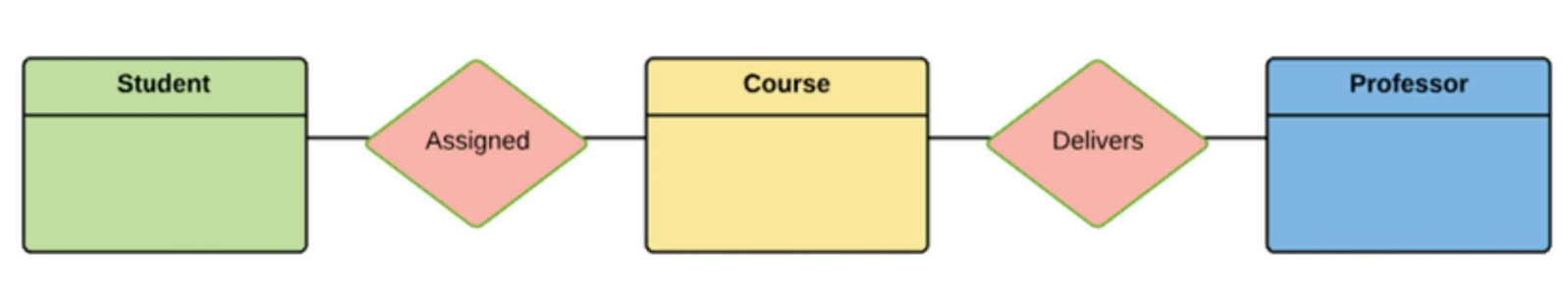
- Student
- Course
- Professor



Step 2. Relationship Identification

We have the following two relationships

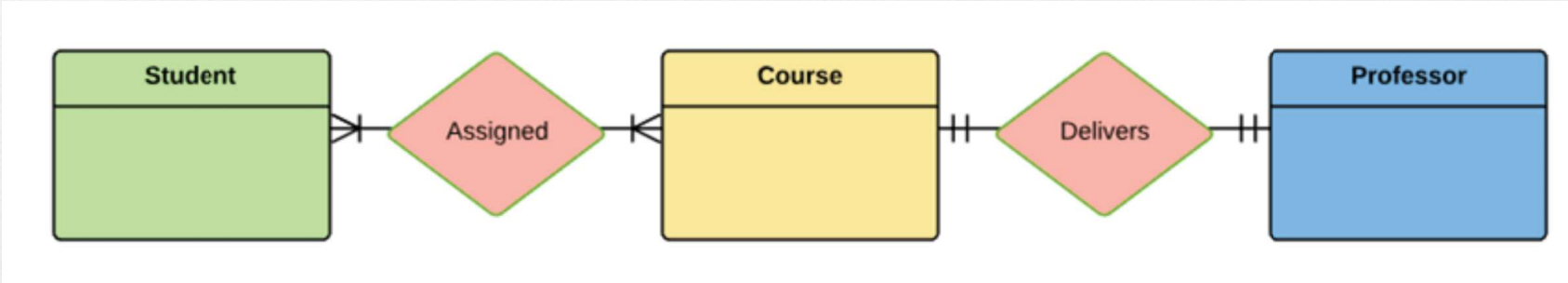
- The student is **assigned** a course
- Professor **delivers** a course



Step 3. Cardinality Identification

For them problem statement we know that,

- A student can be assigned **multiple** courses
- A Professor can deliver only **one** course



Step 4. Identify Attributes

- You need to study the files, forms, reports, data currently maintained by the organization to identify attributes. You can also conduct interviews with various stakeholders to identify entities. Initially, it's important to identify the attributes without mapping them to a particular entity.
- Once, you have a list of Attributes, you need to map them to the identified entities. Ensure an attribute is to be paired with exactly one entity. If you think an attribute should belong to more than one entity, use a modifier to make it unique.

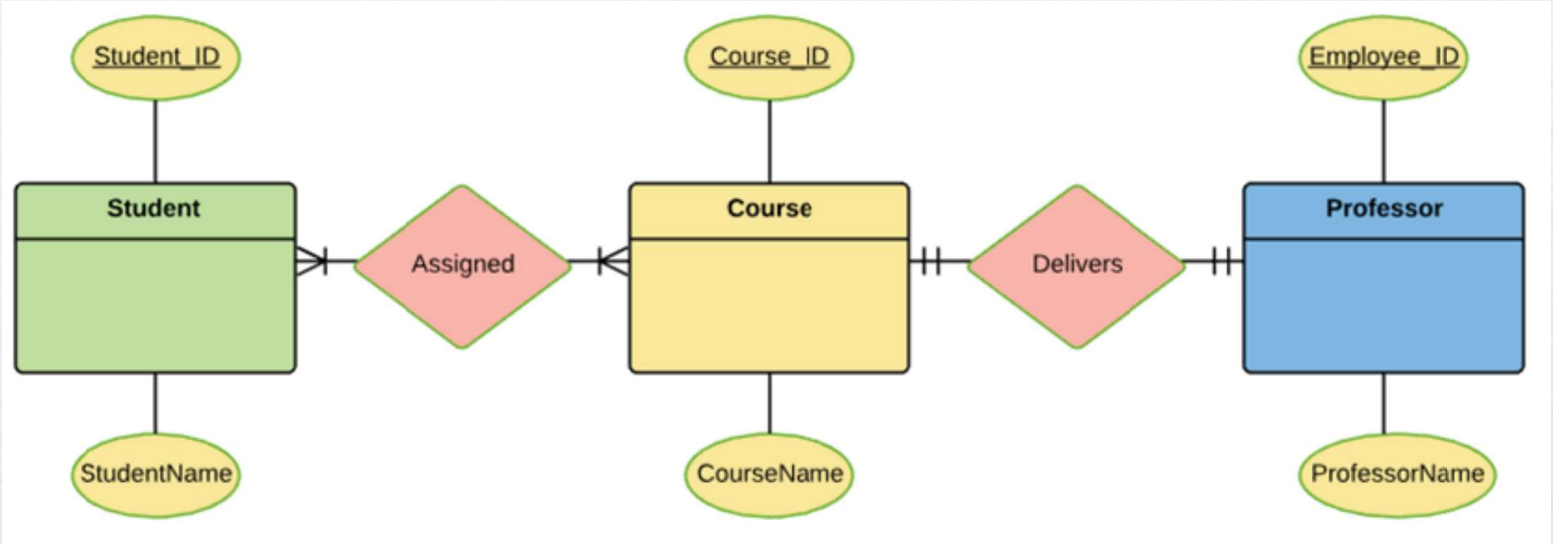
Step 4. Identify Attributes (Cont.)

Once the mapping is done, identify the primary Keys. If a unique key is not readily available, create one.

Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName

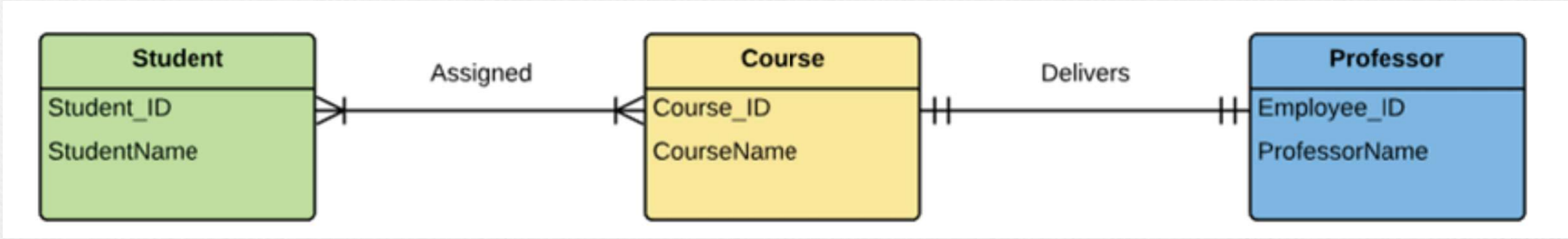
Step 4. Identify Attributes (Cont.)

For Course Entity, attributes could be Duration, Credits, Assignments, etc. For the sake of ease we have considered just one attribute.



Step 5. Create the ERD

A more modern representation of ERD Diagram



Best Practices for Developing Effective ER Diagram

- Eliminate any redundant entities or relationships
- Need to make sure that all entities and relationships are properly labeled
- There may be various valid approaches to an ER diagram. Need to make sure that the ER diagram supports all the data need to store
- Assure that each entity only appears a single time in the ER diagram
- Name every relationship, entity, and attribute are represented on diagram
- Never connect relationships to each other
- Use colors to highlight important portions of the ER diagram

Summary

- The ER model is a high-level data model diagram
- ER diagrams are a visual tool which is helpful to represent the ER model
- Entity relationship diagram displays the relationships of entity set stored in a database
- ER diagrams help to define terms related to entity relationship modeling
- ER model is based on three basic concepts: Entities, Attributes & Relationships
- An entity can be place, person, object, event or a concept, which stores data in the database

Summary

- Relationship is nothing but an association among two or more entities
- A weak entity is a type of entity which doesn't have its key attribute
- It is a single-valued property of either an entity-type or a relationship-type
- It helps you to defines the numerical attributes of the relationship between two entities or entity sets
- ER- Diagram is a visual representation of data that describe how data is related to each other
- While Drawing ER diagram you need to make sure all your entities and relationships are properly labeled.

What is ER Model?

- It is a high-level conceptual data model diagram. ER modeling helps to analyze data requirements systematically to produce a well-designed database. The Entity-Relation model represents real-world entities and the relationship between them. It is considered a best practice to complete ER modeling before implementing database.
- ER modeling helps to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing database.

History of ER models

- ER diagrams are a visual tool which is helpful to represent the ER model. It was proposed by Peter Chen in 1971 to create a uniform convention which can be used for relational database and network. He aimed to use an ER model as a conceptual modeling approach.

Entity-Relationship Model

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Design of the Bank Database
- Reduction to Relation Schemas
- Database Design

Design Process

- Requirement analysis phase
- Conceptual-design phase
- Logical-design phase
- Physical-design phase
- Database implementation
- Database run and maintenance

Modeling

- A database can be modeled as:
  - a collection of entities,
  - relationship among entities.
- 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

Entity Sets customer and loan

customer_id	customer_name	customer_street	customer_city
321-12-3123	Ram	Marg	Kathmandu
019-28-3746	Hari	West	Bhaktapur
677-89-9011	Sita	Main	Birgunj
666-66-6666	Gita	Galli	Bhairahawa
019-28-3747	Krishna	Chowk	Butwal
677-89-9012	Jones	Main	Biratnagar
666-66-6667	Salman	North	Darchula

customer

loan_number	amount
L-16	1000
L-23	1500
L-30	900
L-37	500
L-44	2000
L-51	2500
L-58	700

loan

Relationship Sets

- A relationship is an association among several entities
- Example:

Ram	depositor	A-102
customer	entity relationship set	account entity

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

  $\{(e1, e2, \ldots en) \mid e1 \in E1, e2 \in E2, \ldots, en \in En\}$

  where (e1, e2, …, en) is a relationship
- Example: (Ram, A-102) ∈ depositor

Entity Sets customer and loan

customer_id	customer_name	customer_street	customer_city
321-12-3123	Ram	Marg	Kathmandu
019-28-3746	Hari	West	Bhaktapur
677-89-9011	Sita	Main	Birgunj
666-66-6666	Gita	Galli	Bhairahawa
019-28-3747	Krishna	Chowk	Butwal
677-89-9012	Jones	Main	Biratnagar
666-66-6667	Salman	North	Darchula

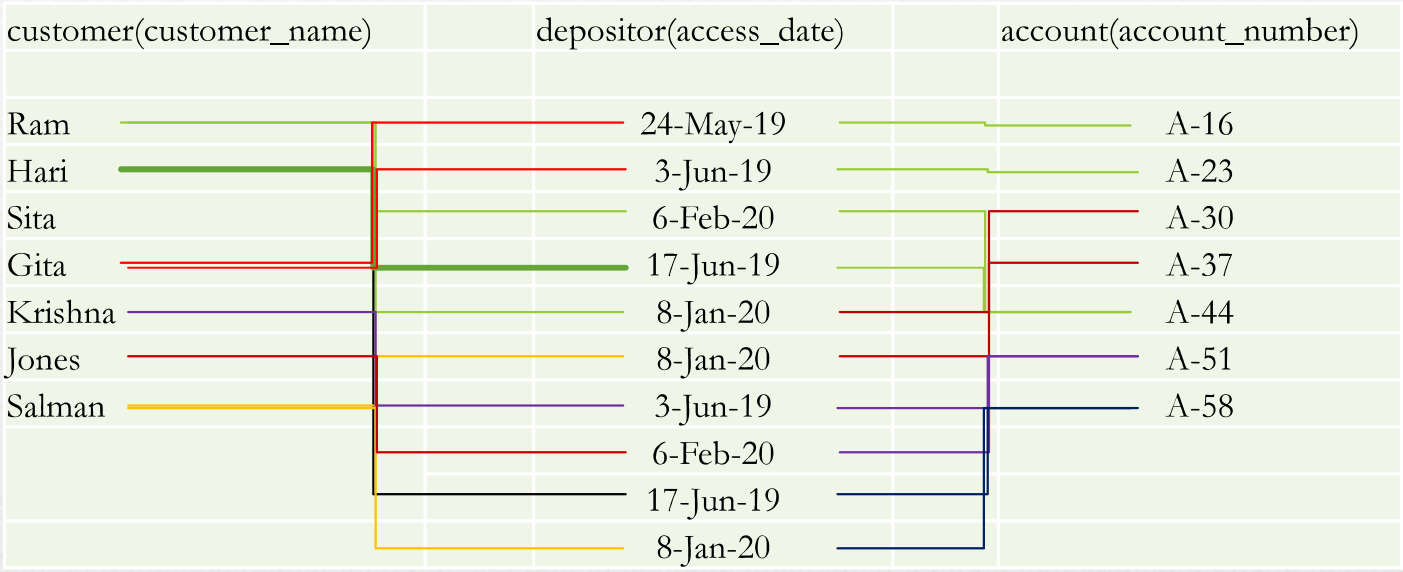
customer

loan_number	amount
L-16	1000
L-23	1500
L-30	900
L-37	500
L-44	2000
L-51	2500
L-58	700

loan

Relationship Sets (Cont.)

- An attribute can also be property of a relationship set.
- For instance, the depositor relationship set between entity sets customer and account may have the attribute access-date



Degree of a Relationship Set

- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are binary (or degree two). Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets.
  Example: Suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a ternary relationship set between entity sets employee, job, and branch
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)

Attributes

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

  Example: customer = (customer_id, customer_name, customer_street, customer_city)

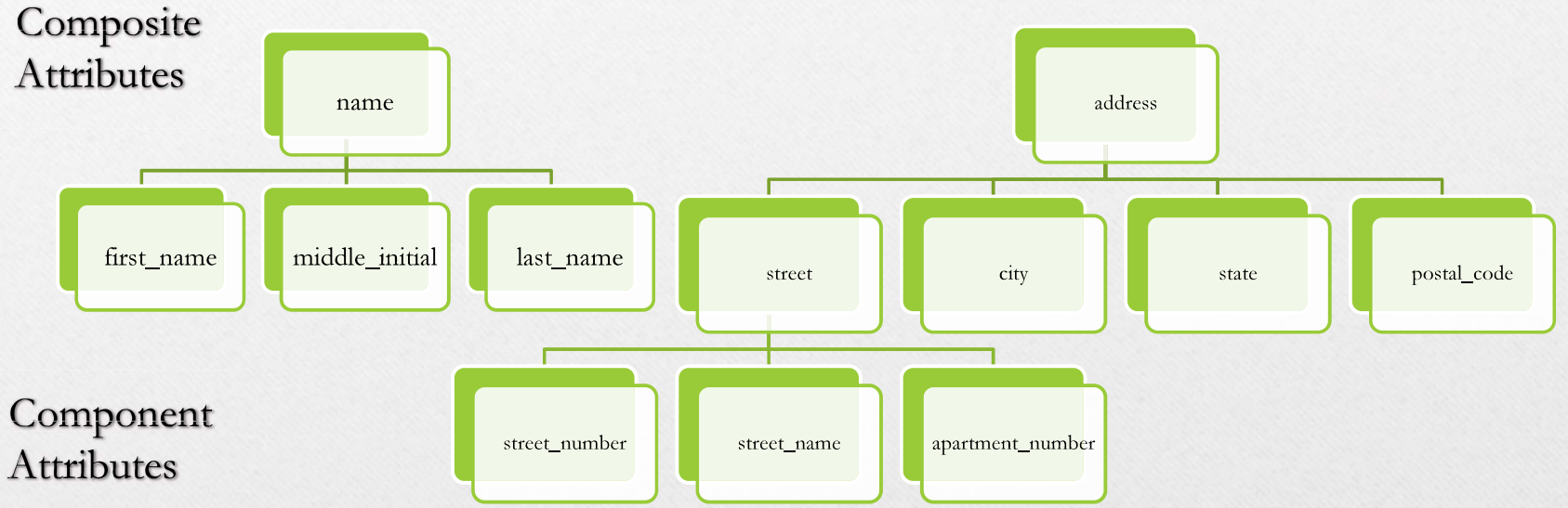
  loan = (loan_number, amount)

- Domain – the set of permitted values for each attribute
- Attribute types:
  - Simple and composite attributes.
  - Single-valued and multi-valued attributes

    Example: multivalued attribute: phone_numbers
- Derived attributes
  - Can be computed from other attributes

    Example: age, given date_of_birth

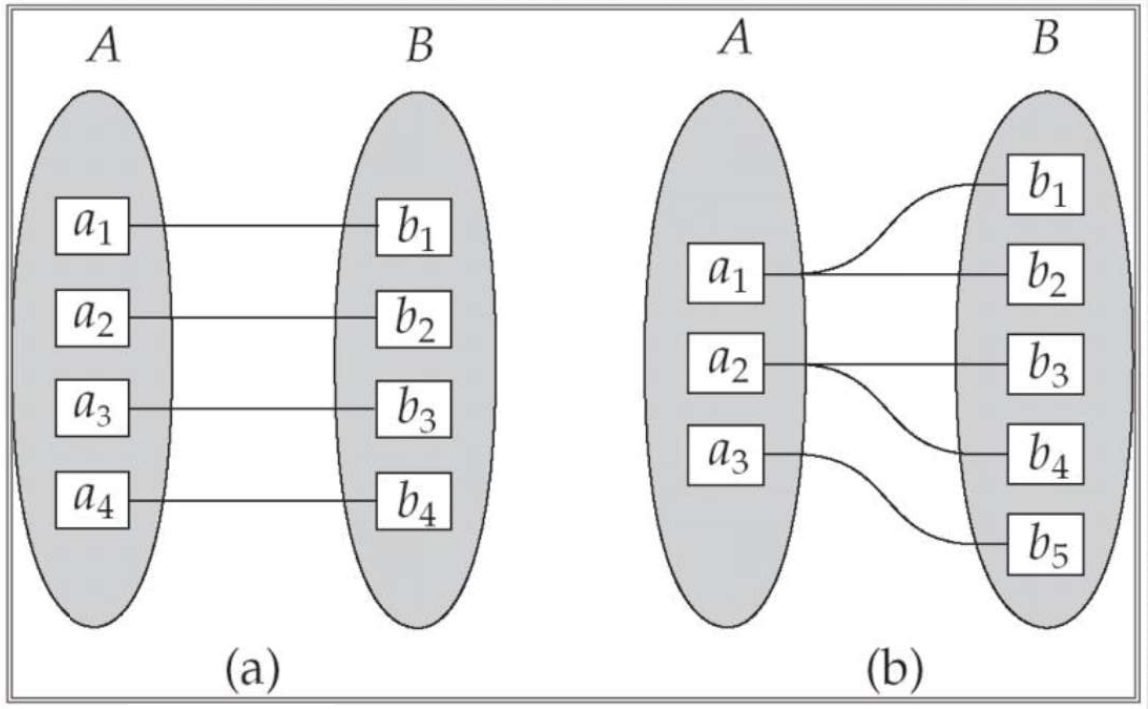
Composite Attributes



Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities

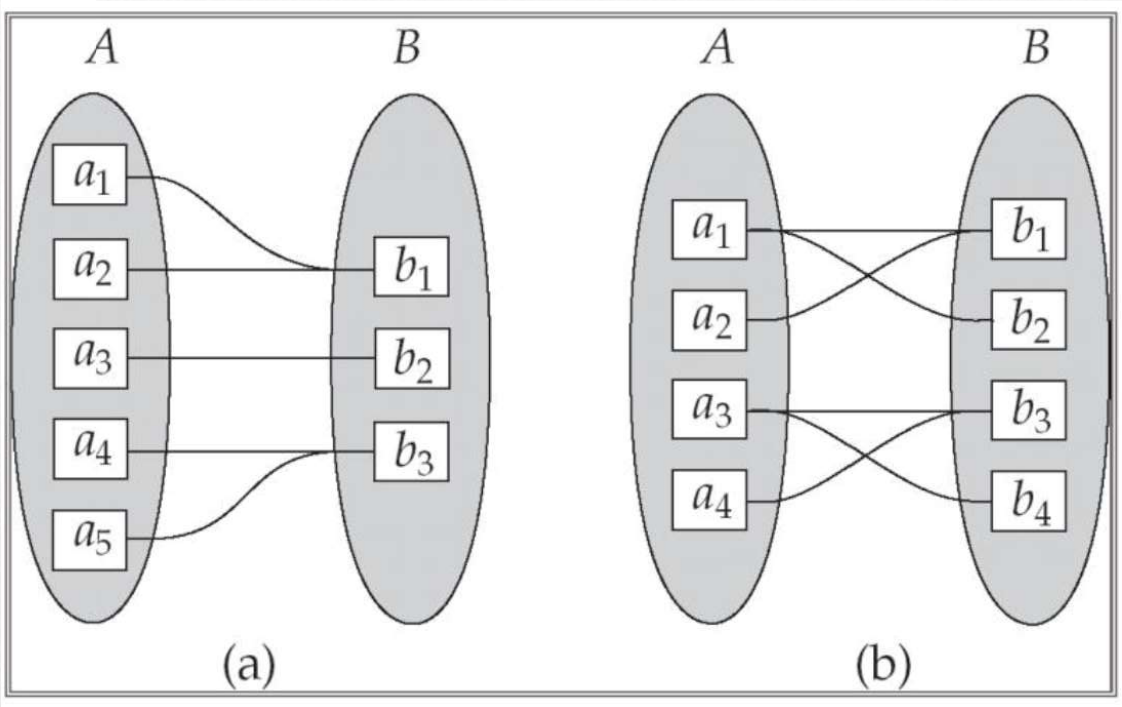


One to one

One to many

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

Mapping Cardinalities



Many to one

Many to many

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

Keys

- A super key of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A candidate key of an entity set is a minimal super key
  - Customer_id is candidate key of customer
  - account_number is candidate key of account
- Although several candidate keys may exist, one of the candidate keys is selected to be the primary key.

Keys for Relationship Sets

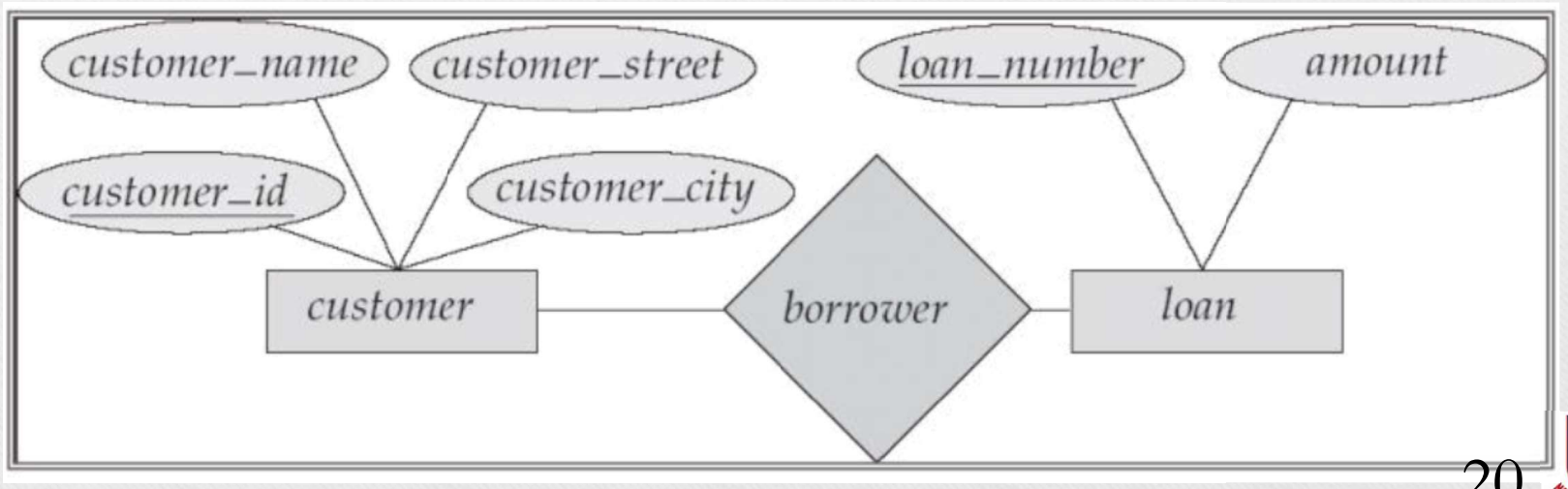
- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (customer_id, account_number) is the super key of depositor
  - NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.
    - Example: if we wish to track all access_dates to each account by each customer, we cannot assume a relationship for each access. We can use a multivalued attribute though
- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys
- Need to consider semantics of relationship set in selecting the primary key in case of more than one candidate key

From Previous Slides

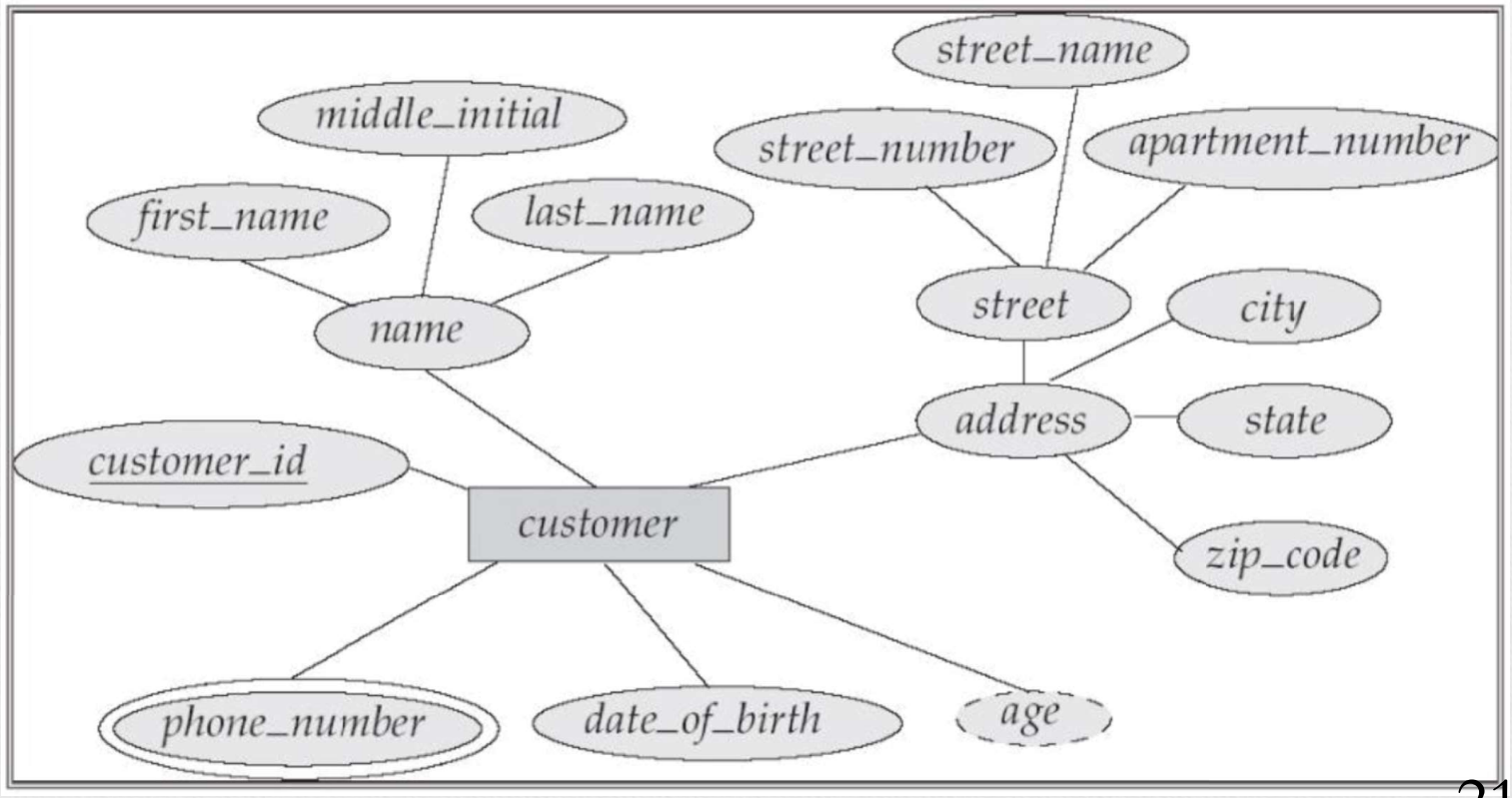
customer(customer_name)	depositor(access_date)	account(account_number)
Ram	24-May-19	A-16
Hari	3-Jun-19	A-23
Sita	6-Feb-20	A-30
Gita	17-Jun-19	A-37
Krishna	8-Jan-20	A-44
Jones	8-Jan-20	A-51
Salman	3-Jun-19	A-58
	6-Feb-20	
	17-Jun-19	
	8-Jan-20	

E-R Diagrams

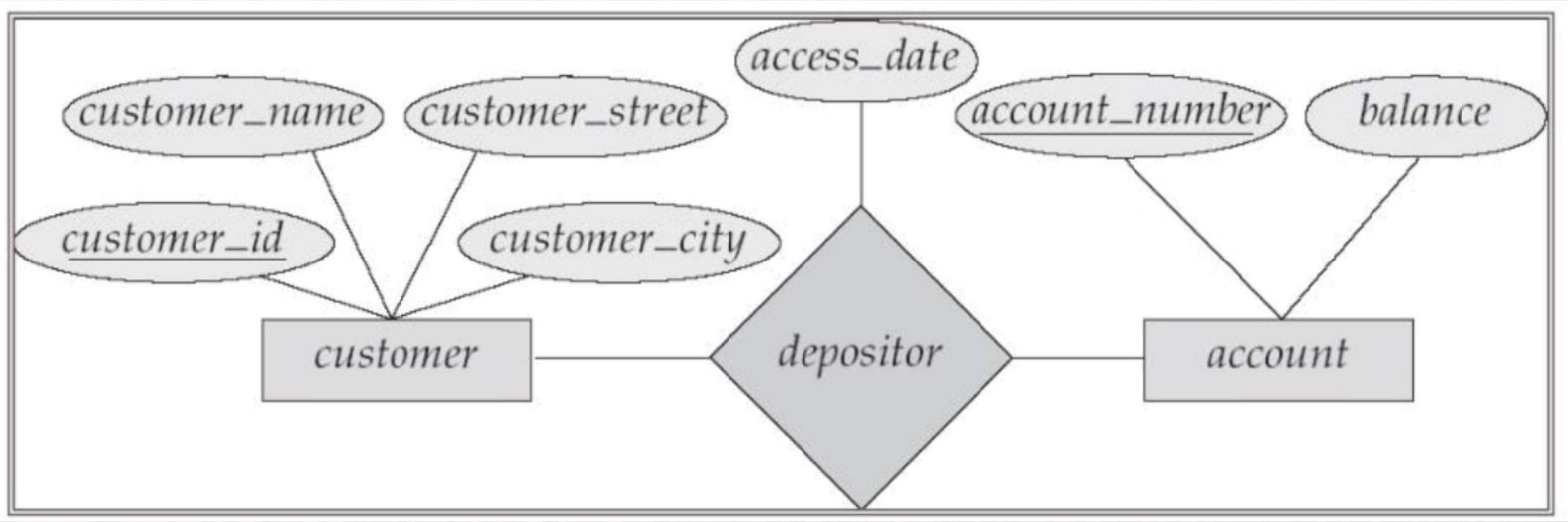
- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Ellipses represent attributes
  - Double ellipses represent multivalued attributes.
  - Dashed ellipses denote derived attributes.
- Underline indicates primary key attributes (will study later)



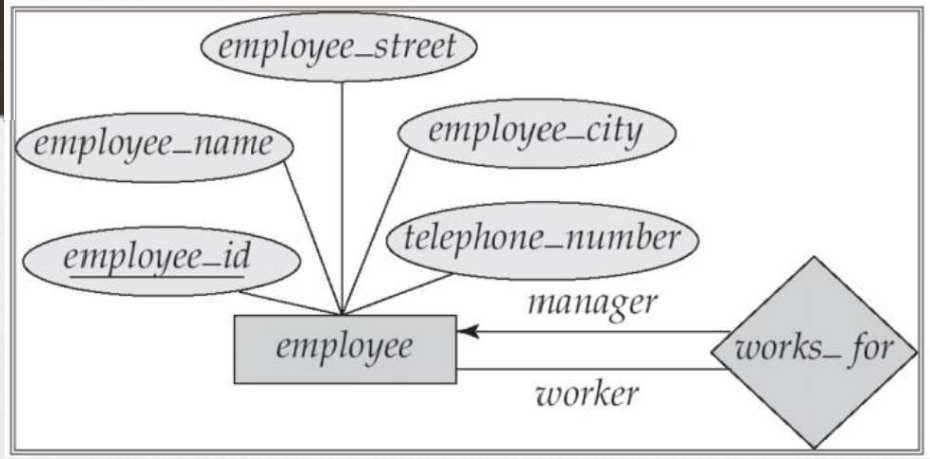
E-R Diagram With Composite, Multivalued, and Derived Attributes



Relationship Sets with Attributes

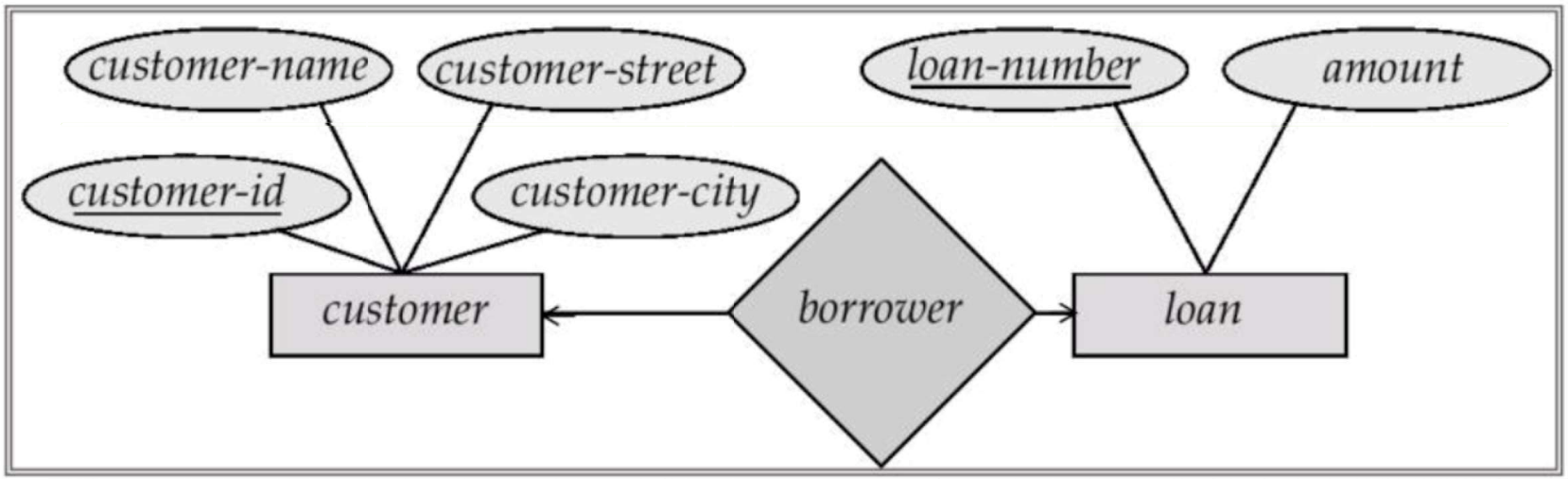


Roles



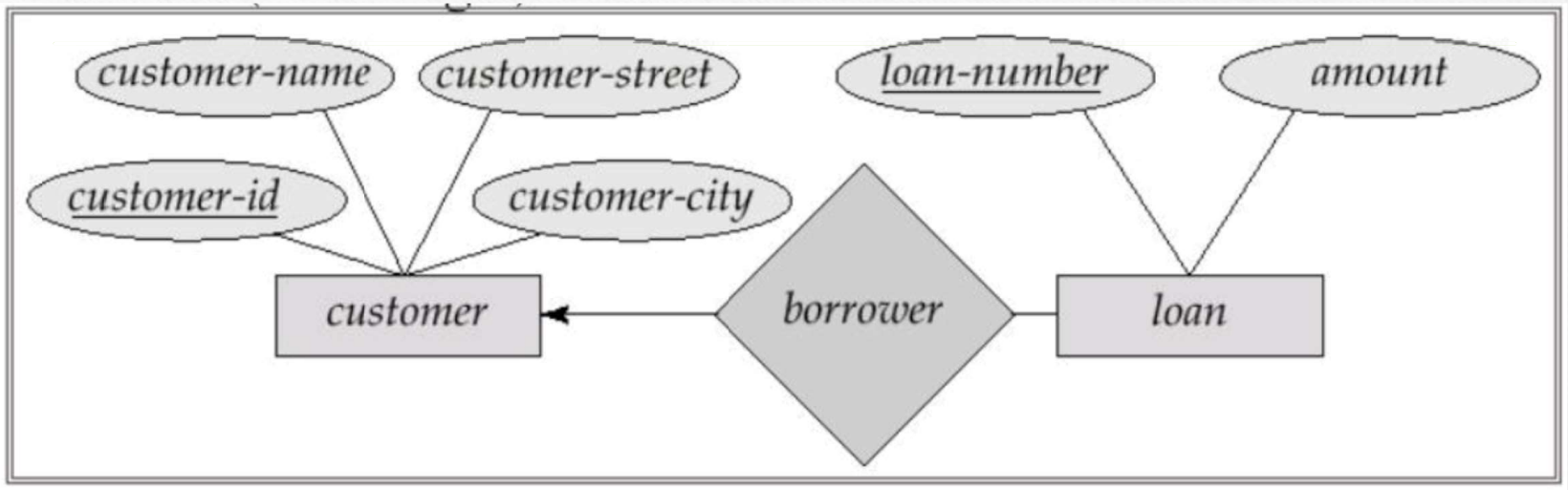
- Entity sets of a relationship need not be distinct
- The labels "manager" and "worker" are called roles; they specify how employee entities interact via the works for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship

Cardinality Constraints



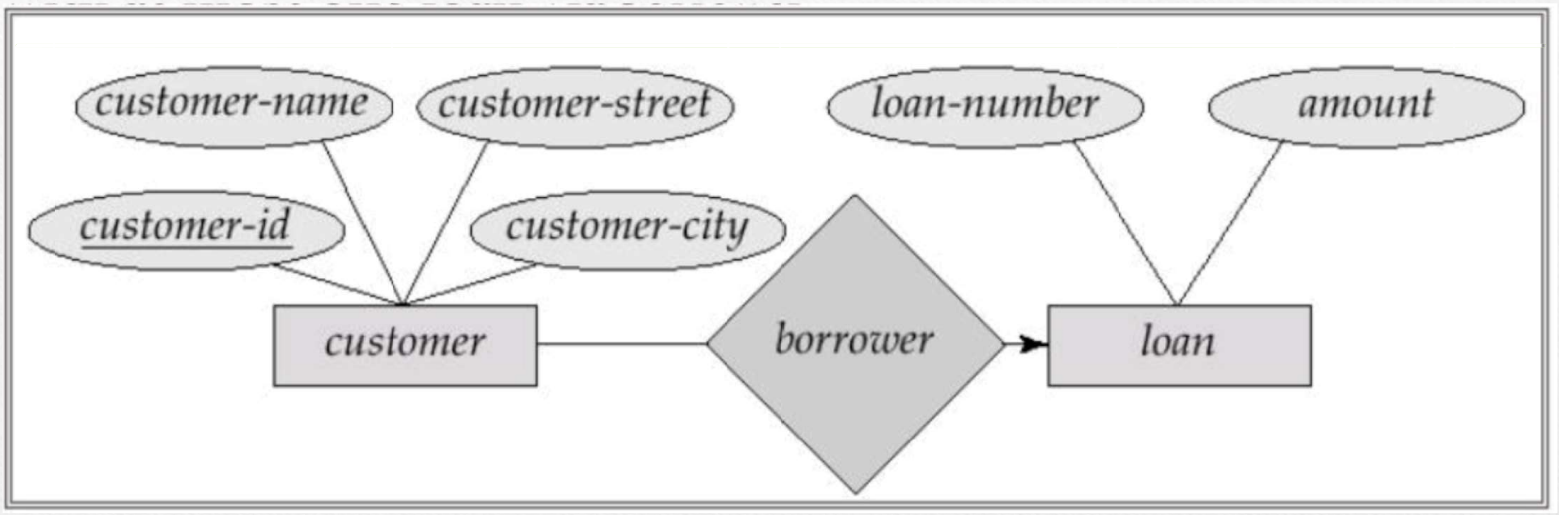
- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.
- One-to-one relationship:
  - A customer is associated with at most one loan via the relationship borrower
  - A loan is associated with at most one customer via borrower

One-To-Many Relationship



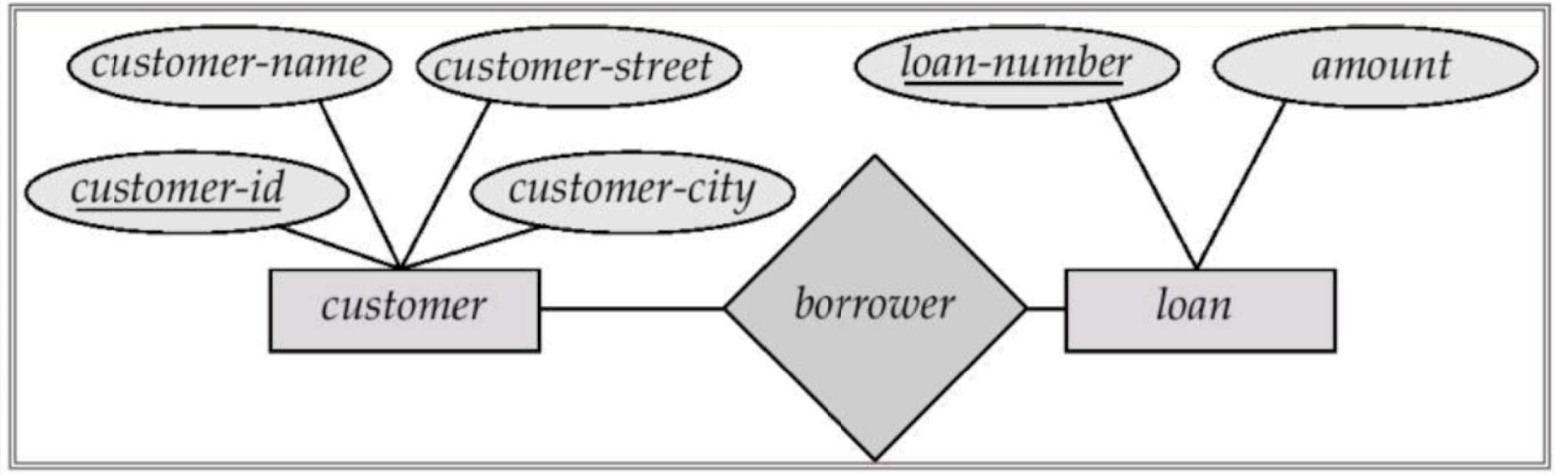
- In the one-to-many relationship a loan is associated with at most one customer via borrower, a customer is associated with several (including 0) loans via borrower

Many-To-One Relationship



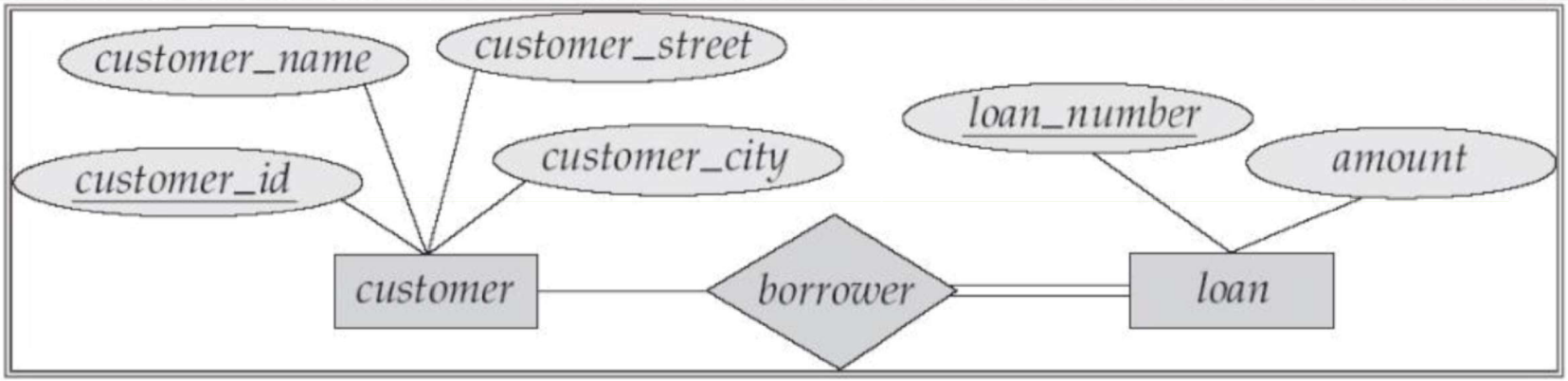
- In a many-to-one relationship a loan is associated with several (including 0) customers via borrower, a customer is associated with at most one loan via borrower

Many-To-Many Relationship



- A customer is associated with several (possibly 0) loans via borrower
- A loan is associated with several (possibly 0) customers via borrower

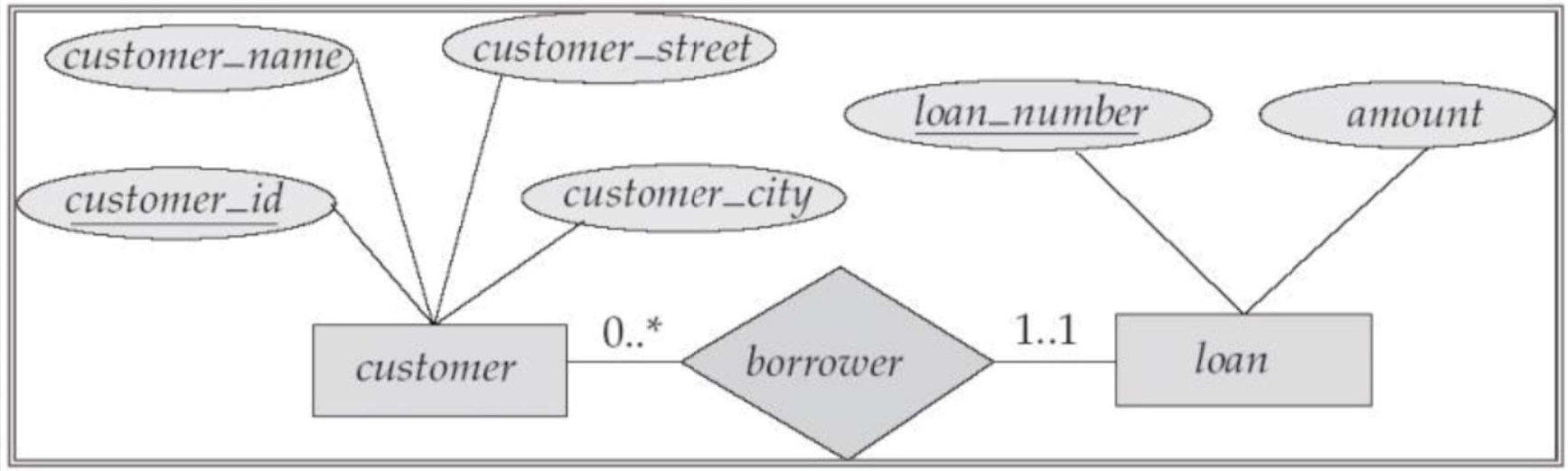
Participating of an Entity Set in a Relationship Set



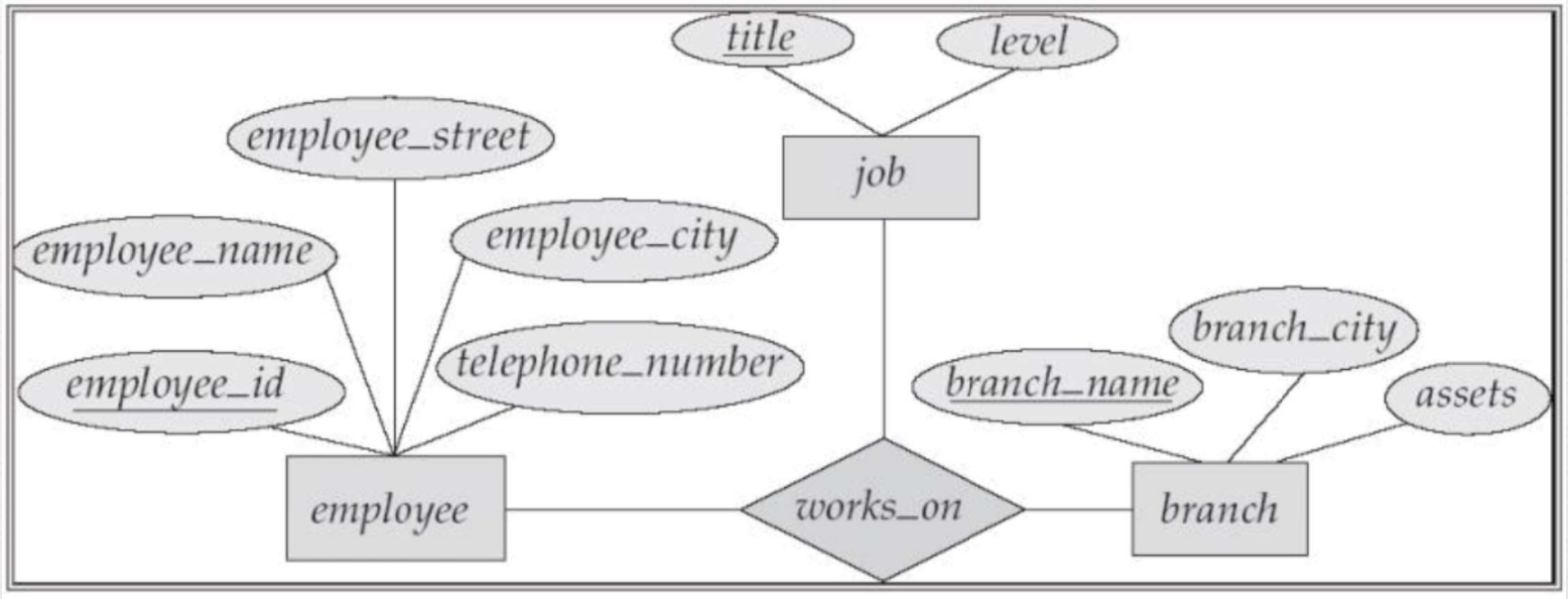
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g. participation of loan in borrower is total
    - every loan must have a customer associated to it via borrower
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of customer in borrower is partial

Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints



E-R Diagram with a Ternary Relationship



Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g. an arrow from works on to job indicates each employee works on at most one job at any branch.
- If there is more than one arrow, there are two ways of defining the meaning.
  - E.g. a ternary relationship R between A, B and C with arrows to B and C could mean
    1. each A entity is associated with a unique entity from B and C or
    2. each pair of entities from (A, B) is associated with a unique C entity, and each pair (A, C) is associated with a unique B
  - Each alternative has been used in different formalisms • To avoid confusion we outlaw more than one arrow

E-R Design Issues

- Use of entity sets vs. attributes

  Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.

- Use of entity sets vs. relationship sets

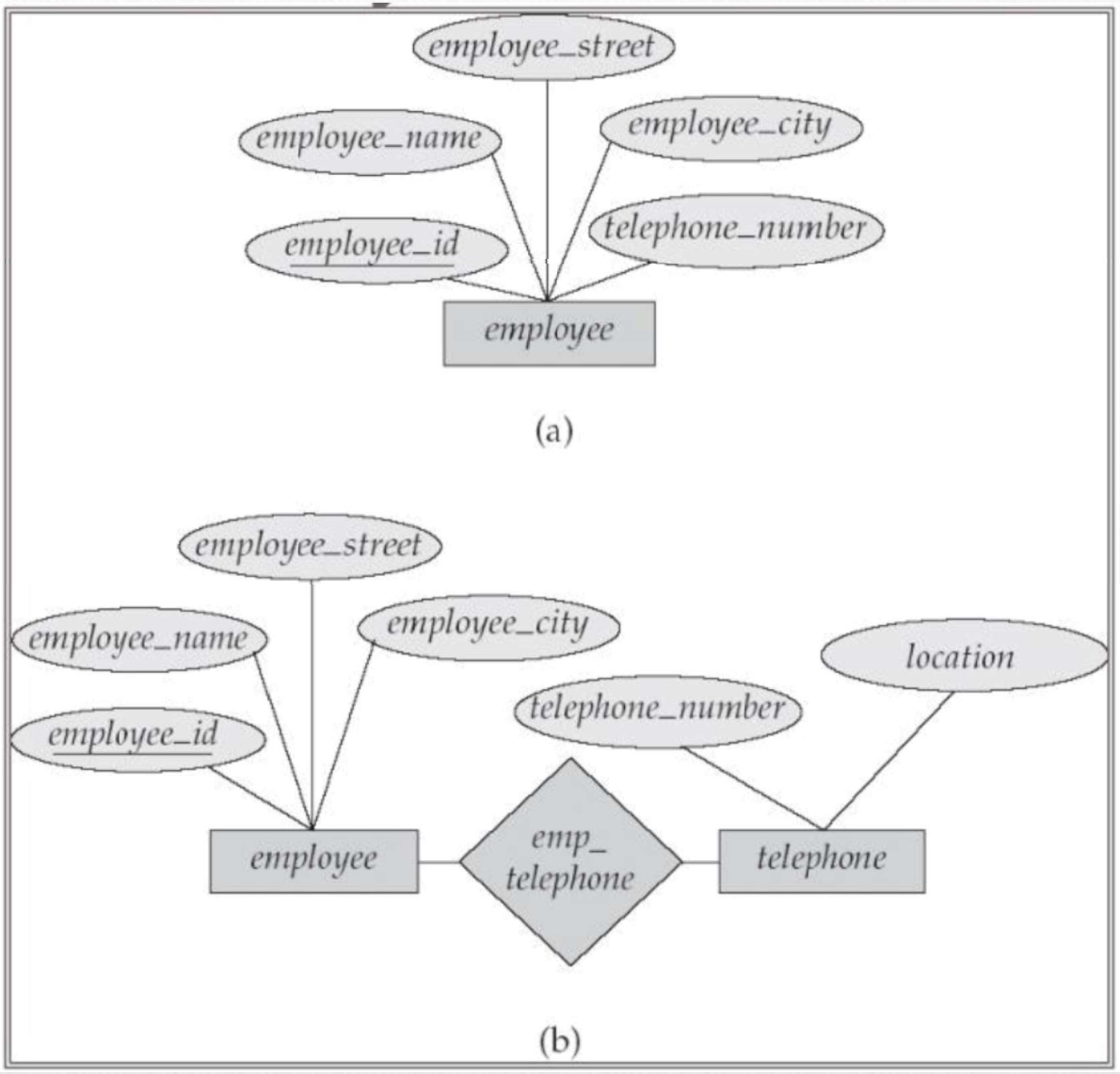
  Possible guideline is to designate a relationship set to describe an action that occurs between entities

- Binary versus n-ary relationship sets

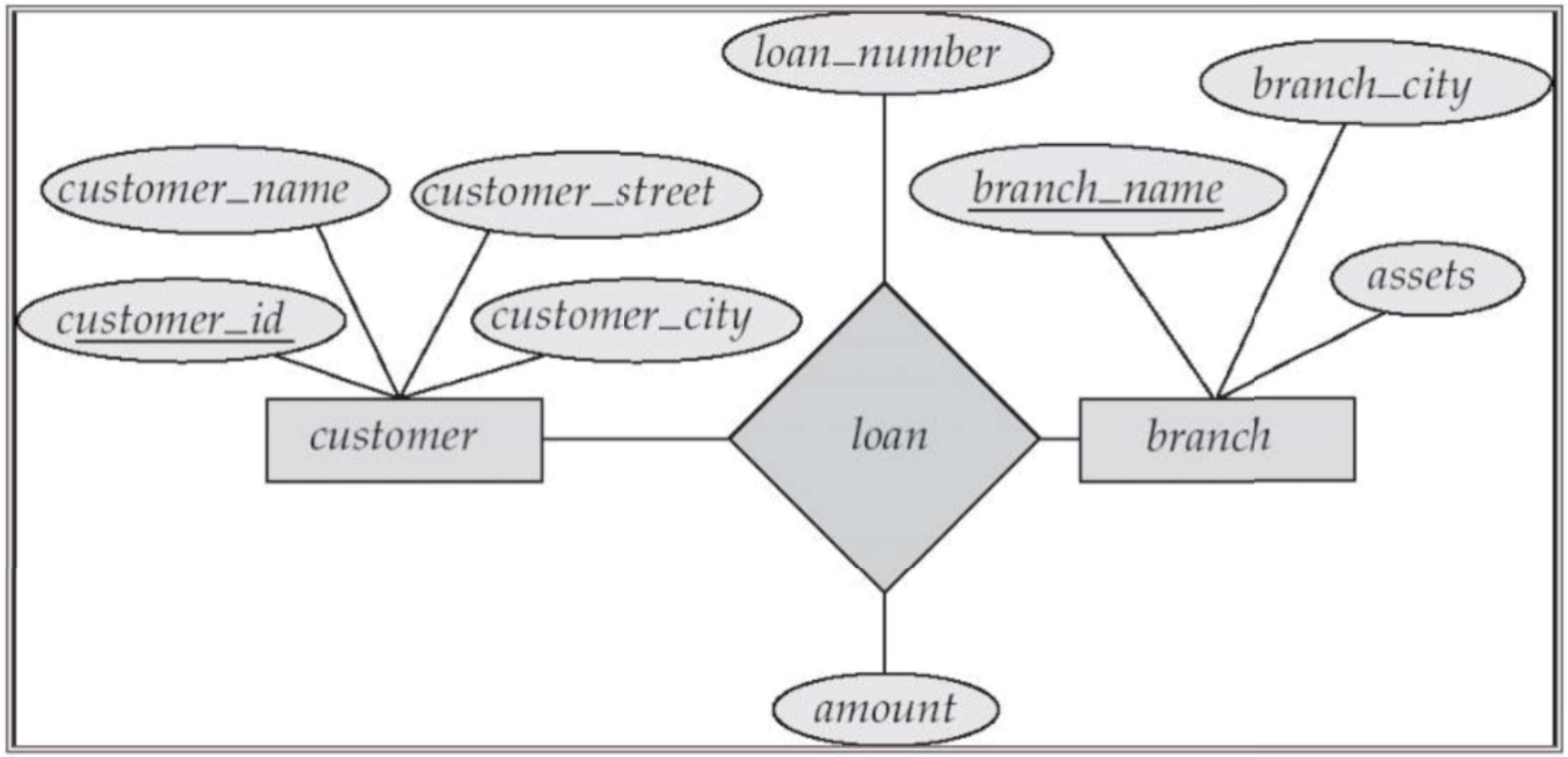
  Although it is possible to replace any nonbinary (n-ary, for n > 2) relationship set by a number of distinct binary relationship sets, a n-ary relationship set shows more clearly that several entities participate in a single relationship.

- Placement of relationship attributes

Use of entity sets vs. attributes



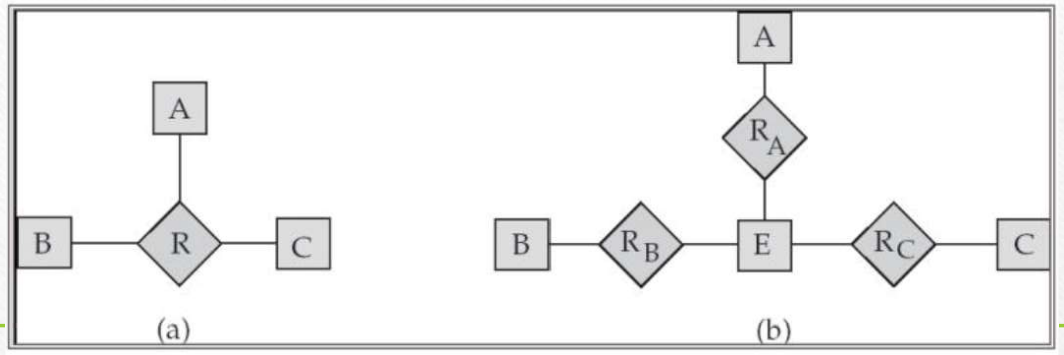
Use of entity sets vs. relationship sets



Binary Vs. Non-Binary Relationship

- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g. A ternary relationship parents, relating a child to his/her father and mother, is best replaced by two binary relationships, father and mother
    - Using two binary relationships allows partial information (e.g. only mother being know)
  - But there are some relationships that are naturally non-binary • Example: works_on

Converting Non-Binary Relationships to Binary Form



- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace R between entity sets A, B and C by an entity set E, and three relationship sets:
    1. R_A, relating E and A
    2. R_B, relating E and B
    3. R_C, relating E and C
  - Create a special identifying attribute for E
  - Add any attributes of R to E
  - For each relationship (a_i , b_i , c_i) in R, create
    1. a new entity e_i in the entity set E
    2. add (e_i , a_i) to R_A
    3. add (e_i , b_i) to R_B
    4. add (e_i , c_i) to R_C

Converting Non-Binary Relationships (Cont.)

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of R
    - Exercise: add constraints to the relationships R_A, R_B and R_C to ensure that a newly created entity corresponds to exactly one entity in each of entity sets A, B and C
  - We can avoid creating an identifying attribute by making E a weak entity set (described shortly) identified by the three relationship sets

Mapping Cardinalities affect ER Design

customer(customer_name)	depositor	account(account_number, access_date)	
Ram		A-16	24-May-19
Hari		A-23	3-Jun-19
Sita		A-30	6-Feb-20
Gita		A-37	17-Jun-19
Krishna		A-44	8-Jan-20
Jones		A-51	8-Jan-20
Salman		A-58	3-Jun-19

- Can make access-date an attribute of account, instead of a relationship attribute, if each account can have only one customer
  - That is, the relationship from account to customer is many to one, or equivalently, customer to account is one to many

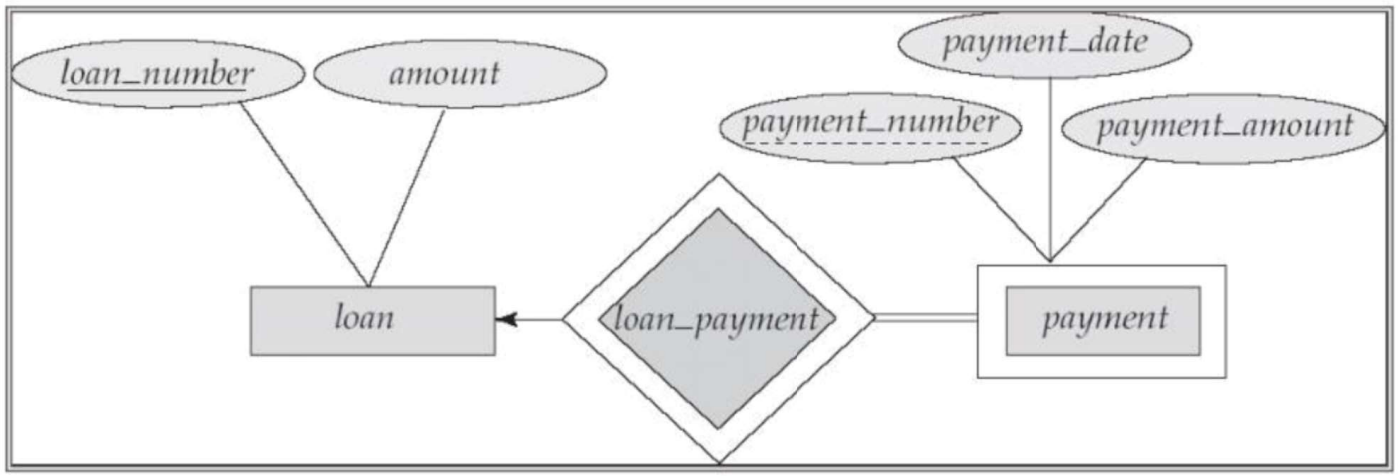
Weak Entity Sets

- An entity set that does not have a primary key is referred to as a weak entity set.
- The existence of a weak entity set depends on the existence of a identifying entity set
  - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - Identifying relationship depicted using a double diamond
- The discriminator(or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

Weak Entity Sets (Cont.)

- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dashed line.

- payment_number – discriminator of the payment entity set
- Primary key for payment – (loan_number, payment_number)

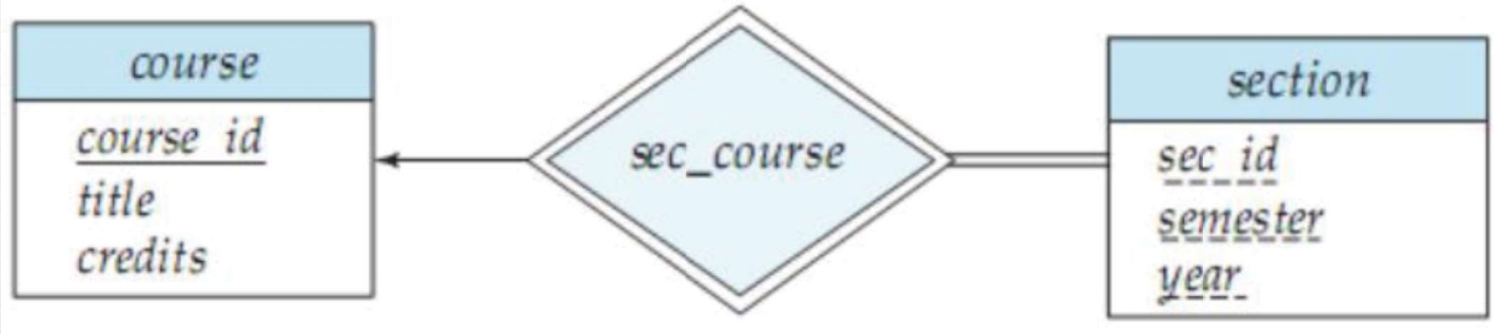


Weak Entity Sets (Cont.)

- Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.
- If loan_number were explicitly stored, payment could be made a strong entity, but then the relationship between payment and loan would be duplicated by an implicit relationship defined by the attribute loan_number common to payment and loan

Weak Entity Sets (Cont.)

- A better example:



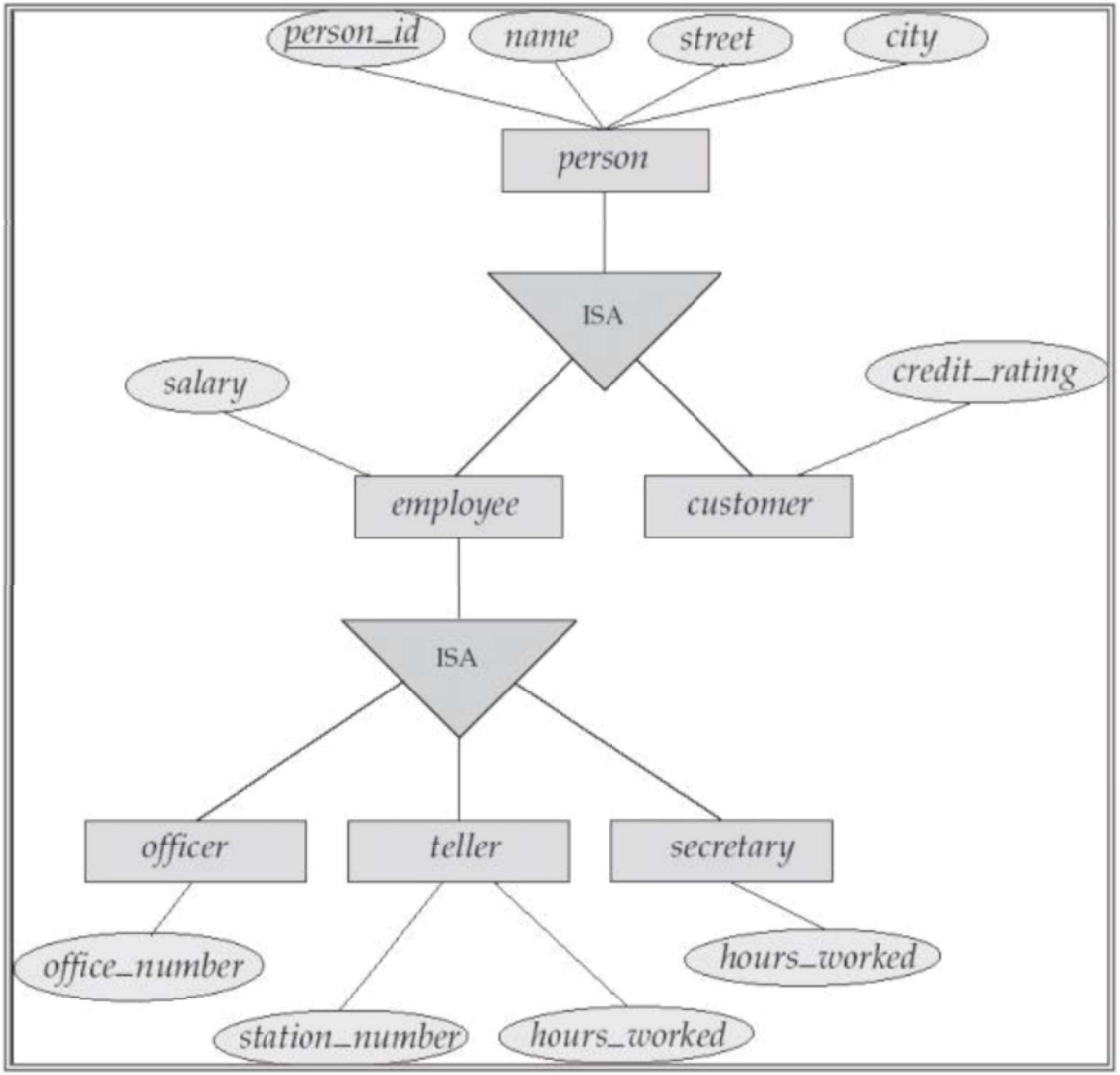
More Weak Entity Set Examples

- In a university, a course is a strong entity and a course_offering can be modeled as a weak entity
- The discriminator of course_offering would be semester (including year) and section_number (if there is more than one section)
- If we model course_offering as a strong entity we would model course_number as an attribute. Then the relationship with course would be implicit in the course_number attribute

Extended E-R Features: Specialization

- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a triangle component labeled ISA (E.g. customer "is a" person).
- Attribute inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

Specialization Example



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 officer vs. secretary vs. teller
- Each particular employee would be
  - a member of one of permanent_employee or temporary_employee,
  - and also a member of one of officer, secretary, or teller
- The ISA relationship also referred to as superclass – subclass relationship

Design Constraints on a Specialization/Generalization

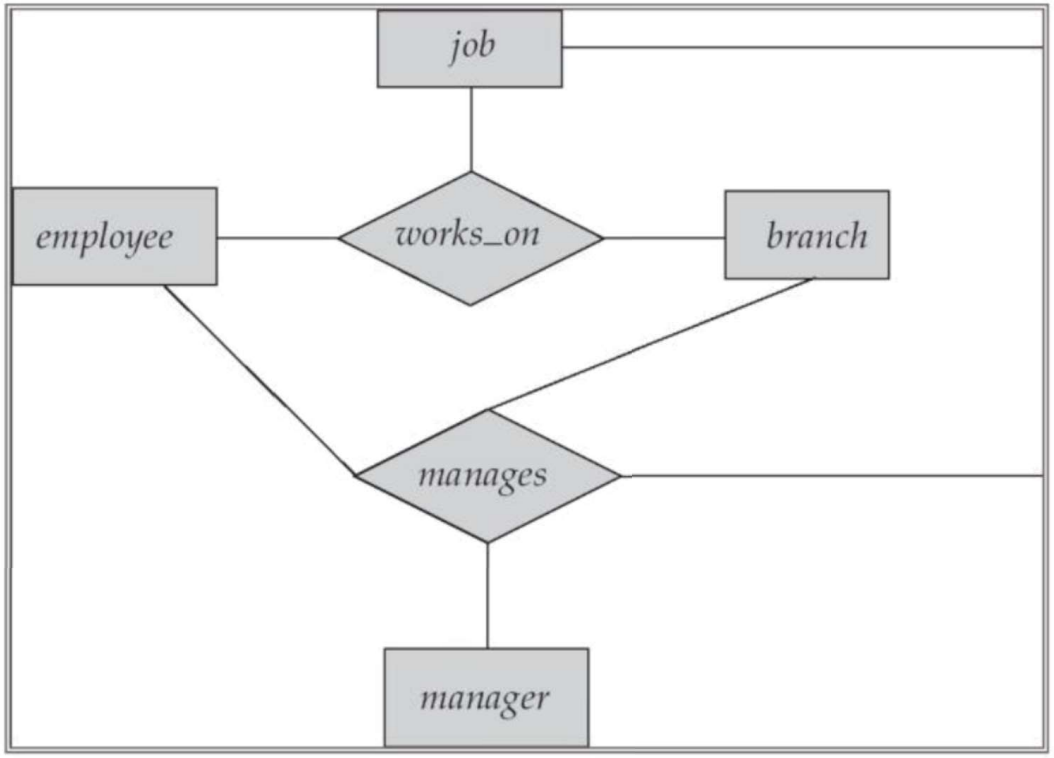
- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person.
  - user-defined
- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
  - Disjoint
    - an entity can belong to only one lower-level entity set
    - Noted in E-R diagram by writing disjoint next to the ISA triangle
  - Overlapping
    - an entity can belong to more than one lower-level entity set

Design Constraints on a Specialization/Generalization

- Completeness constraint - specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - total: an entity must belong to one of the lower-level entity sets
  - partial: an entity need not belong to one of the lower-level entity sets

Aggregation

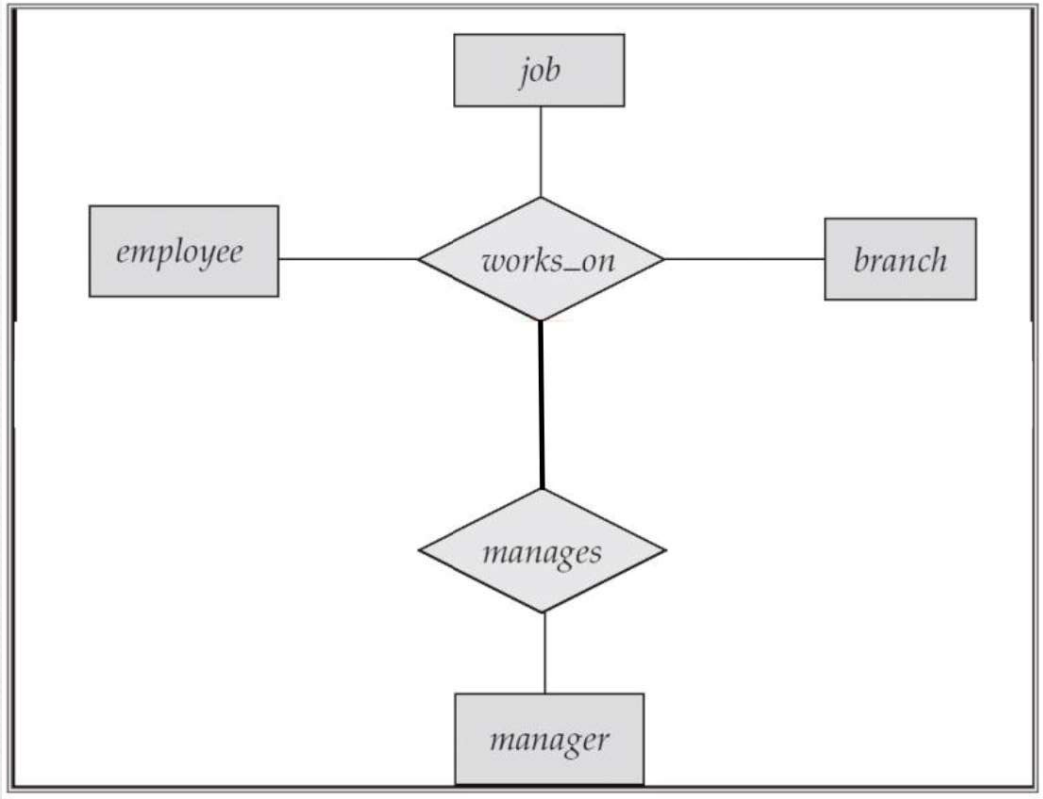
- Consider the ternary relationship works_on, which we saw earlier
- Suppose we want to record managers for tasks performed by an employee at a branch



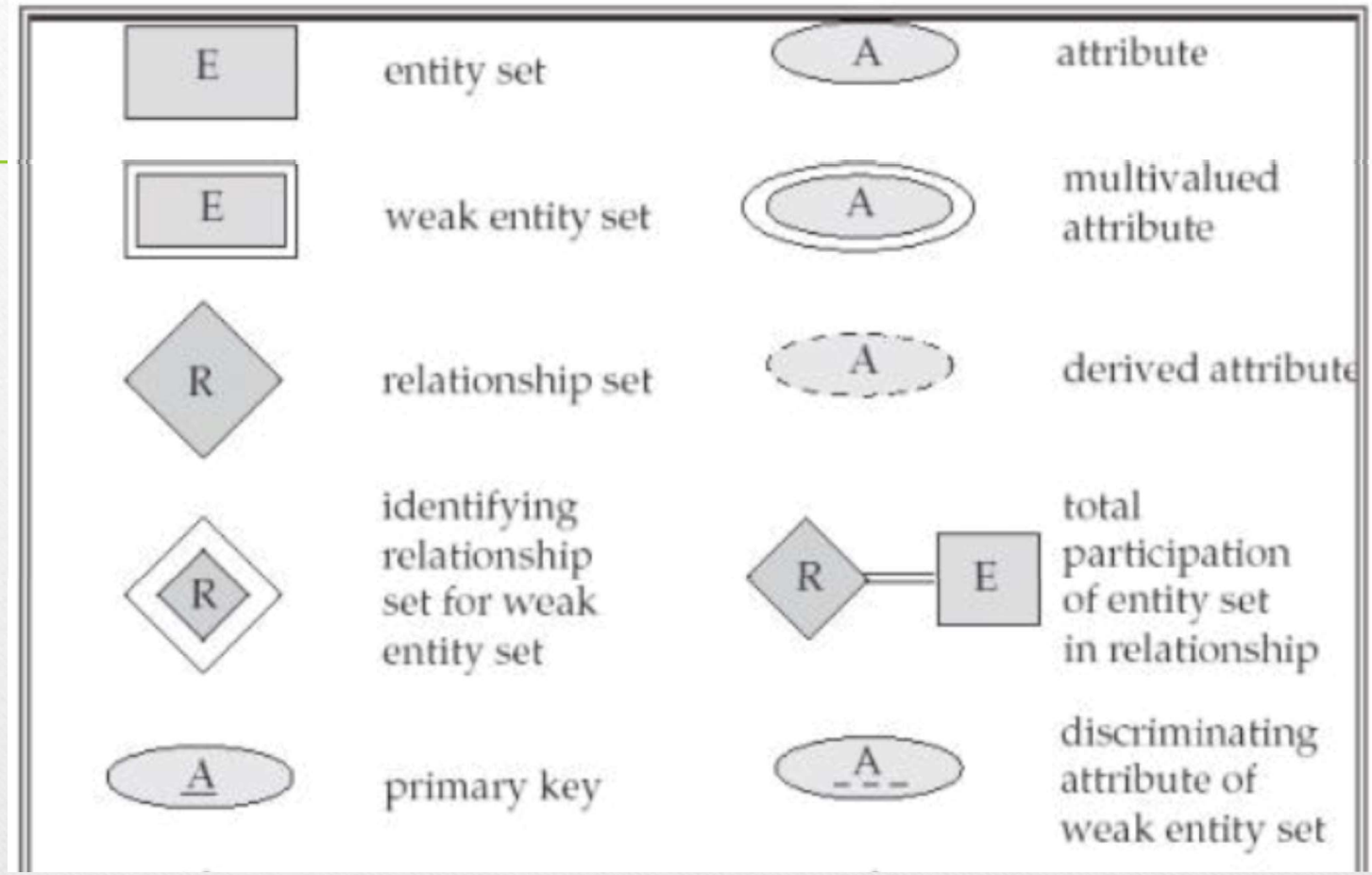
Aggregation (Cont.)

- Relationship sets works_on and manages represent overlapping information
  - Every manages relationship corresponds to a works_on relationship
  - However, some works_on relationships may not correspond to any manages relationships
    - So we can't discard the works_on relationship
- Eliminate this redundancy via aggregation
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity
- Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager

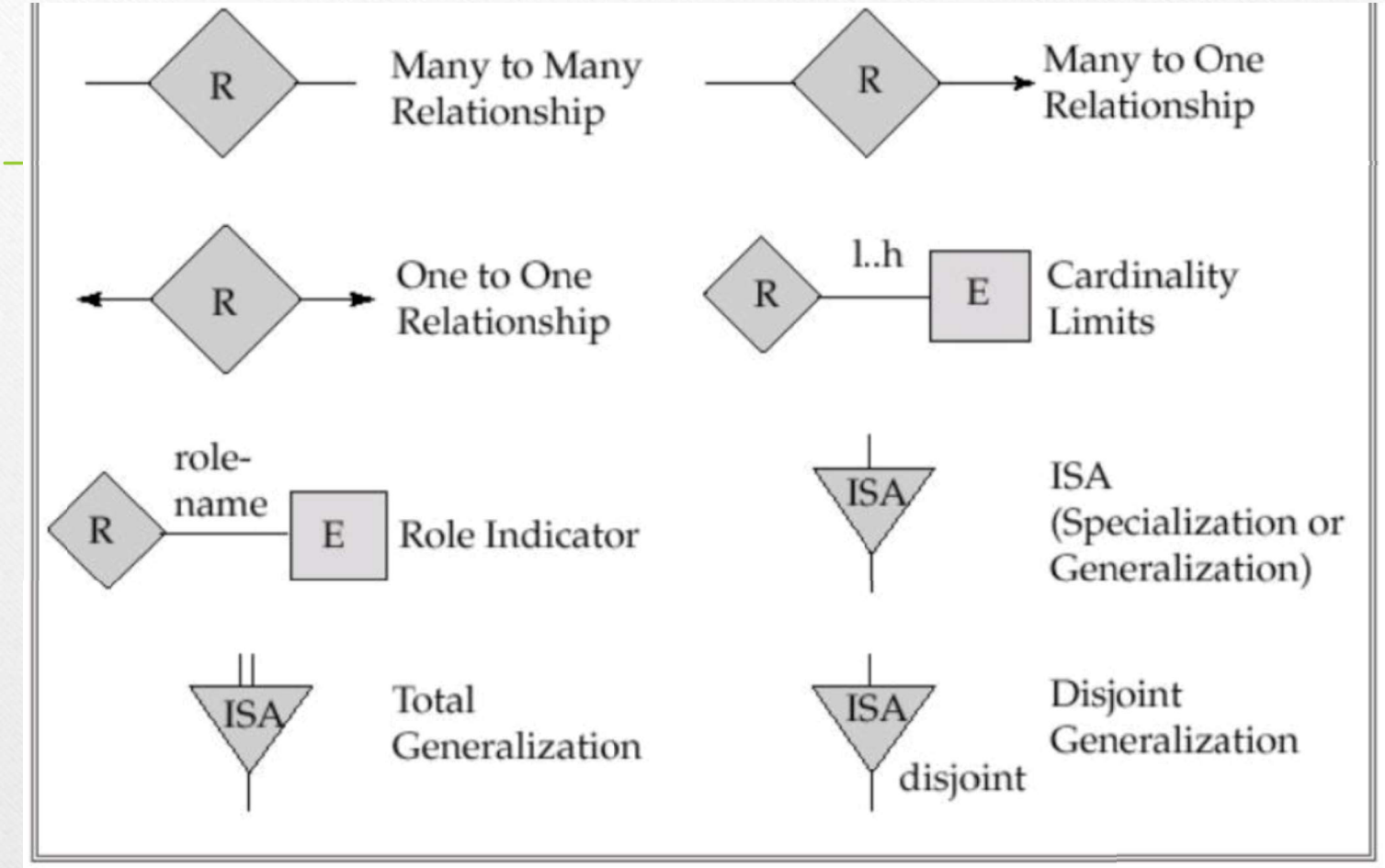
E-R Diagram With Aggregation



Summary of Symbols Used in E-R Notation



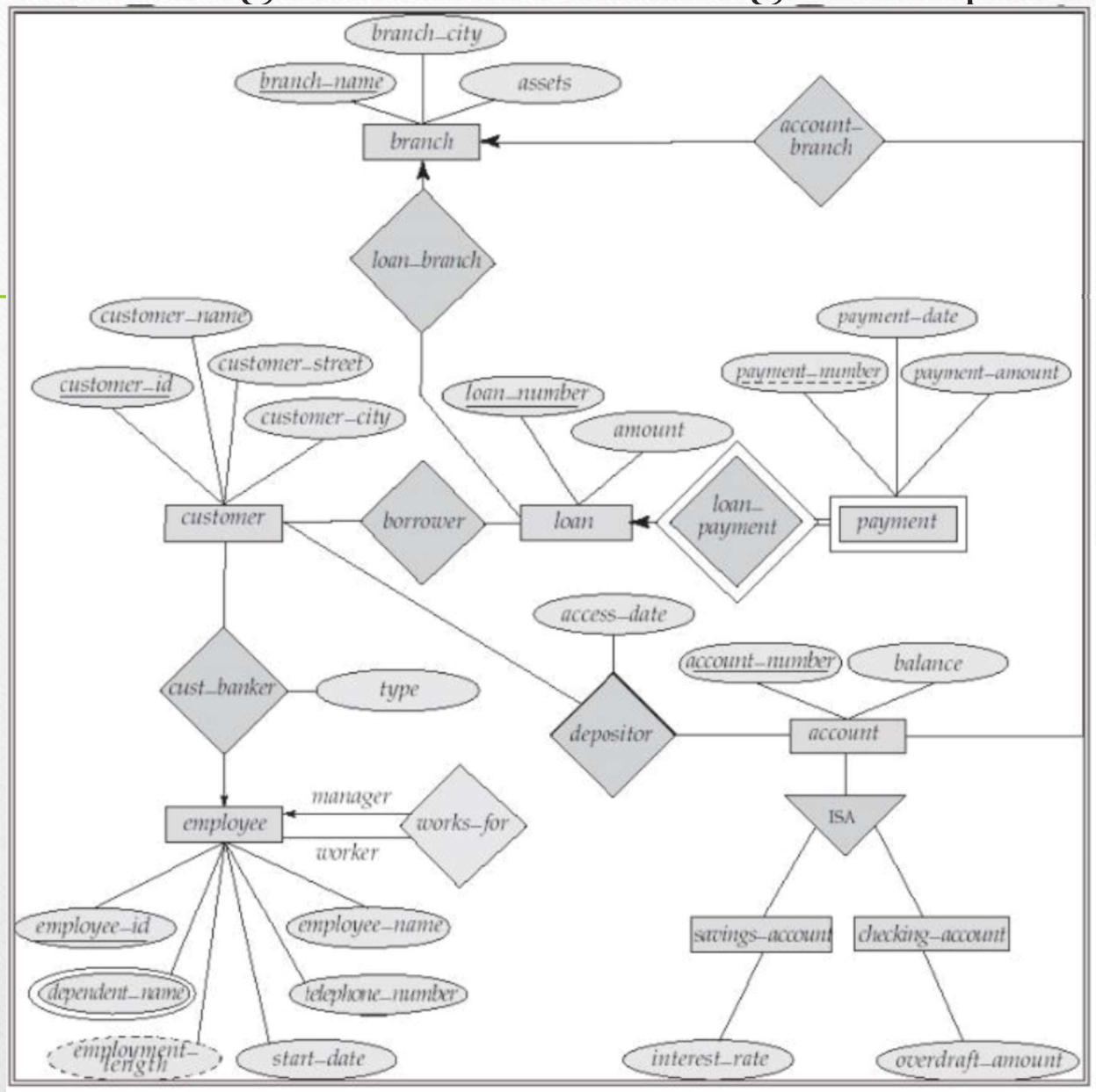
Summary of Symbols (Cont.)



E-R Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

E-R Diagram for a Banking Enterprise



Reduction to Relation Schemas

- Primary keys allow entity sets and relationship sets to be expressed uniformly as relation schemas that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- 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 as Schemas

- A strong entity set reduces to a schema with the same attributes.
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

  payment = (loan_number, payment_number, payment_date, payment_amount)

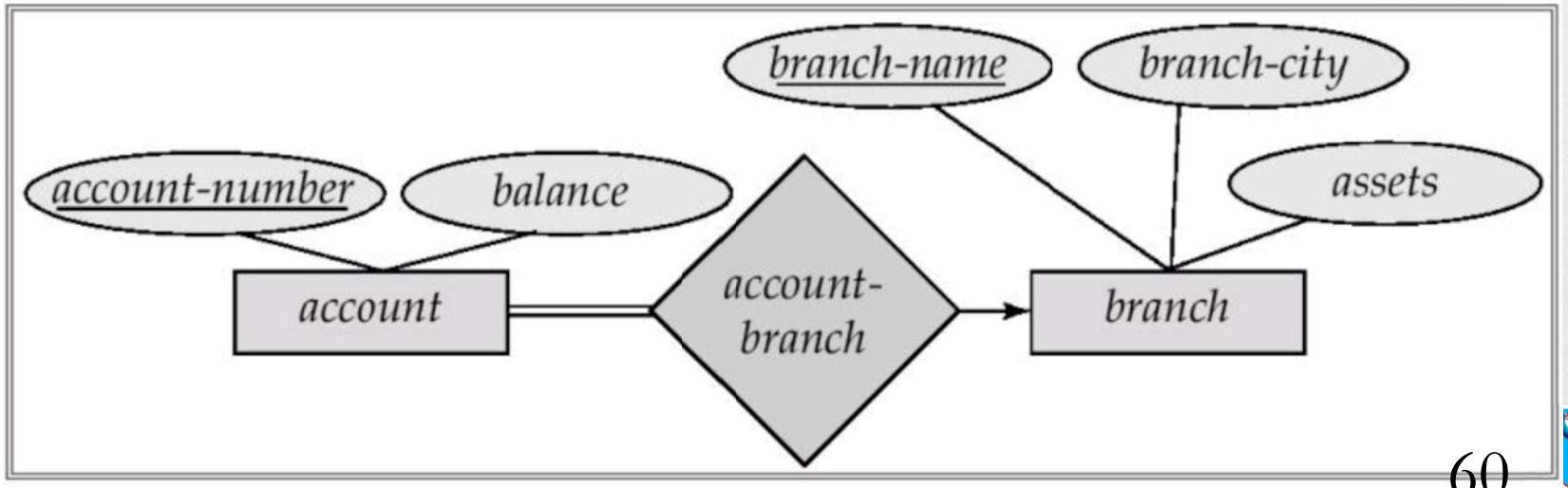
Representing Relationship Sets as Schemas

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set borrower
  borrower = (customer_id, loan_number)

Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side

- Example: Instead of creating a schema for relationship set account_branch, add an attribute branch_name to the schema arising from entity set account



Redundancy of Schemas (Cont.)

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
- That is, extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is partial on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - Example: The payment schema already contains the attributes that would appear in the loan_payment schema (i.e., loan_number and payment_number).

Composite and Multivalued Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set customer with composite attribute name with component attributes first_name and last_name the schema corresponding to the entity set has two attributesname.first_name and name.last_name

Composite and Multivalued Attributes (Cont.)

- 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 dependent_names of employee is represented by a schema:
    employee_dependent_names = (employee_id, dname)
  - Each value of the multivalued attribute maps to a separate tuple of the relation on schema EM
    - For example, an employee entity with primary key 123-45-6789 and dependents Ram and Shyam maps to two tuples:

      (123-45-6789 , Ram) and (123-45-6789 , Shyam)

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

schema	attributes
person	*name, street, city*
customer	*name, credit_rating*
employee	*name, salary*

- 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

Representing Specialization via Schemas (Cont.) (Method 2)

- Form a schema for each entity set with all local and inherited attributes schema attribute

schema	attributes
customer	*name, street, city, credit_rating*
employee	*name, street, city, salary*

- If specialization is total, the schema for the generalized entity set (person) not required to store information
  - Can be defined as a "view" relation containing union of specialization relations
  - But explicit schema may still be needed for foreign key constraints
- Drawback: street and city may be stored redundantly for people who are both customers and employees

Schemas Corresponding to Aggregation

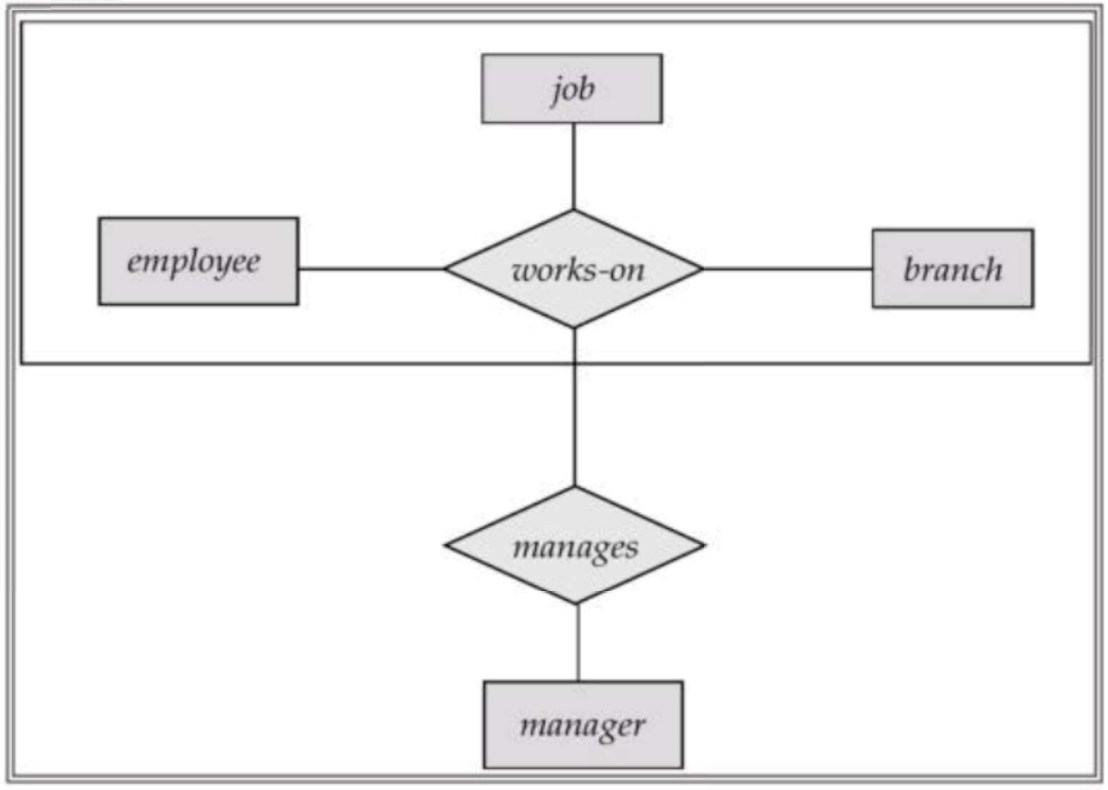
- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - the primary key of the associated entity set
  - any descriptive attributes

Schemas Corresponding to Aggregation (Cont.)

- For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema

  manages (employee_id, branch_name, title, manager_name)

- Schema works_on is redundant provided we are willing to store null values for attribute manager_name in relation on schema manages



The Banking Schema

- branch = (branch_name, branch_city, assets)
- customer = (customer_id, customer_name, customer_street, customer_city)
- loan = (loan_number, amount)
- account = (account_number, balance)
- employee = (employee_id. employee_name, telephone_number, start_date)
- dependent_name = (employee_id, dname)
- account_branch = (account_number, branch_name)
- loan_branch = (loan_number, branch_name)
- borrower = (customer_id, loan_number)
- depositor = (customer_id, account_number)
- cust_banker = (customer_id, employee_id, type)
- works_for = (worker_employee_id, manager_employee_id)
- payment = (loan_number, payment_number, payment_date, payment_amount)
- savings_account = (account_number, interest_rate)
- checking_account = (account_number, overdraft_amount)

The Unified Modeling Language UML

Entity-relationship diagrams help model the data representation component of a software system. Data representation, however, forms only one part of an overall system design.

Other components such as:

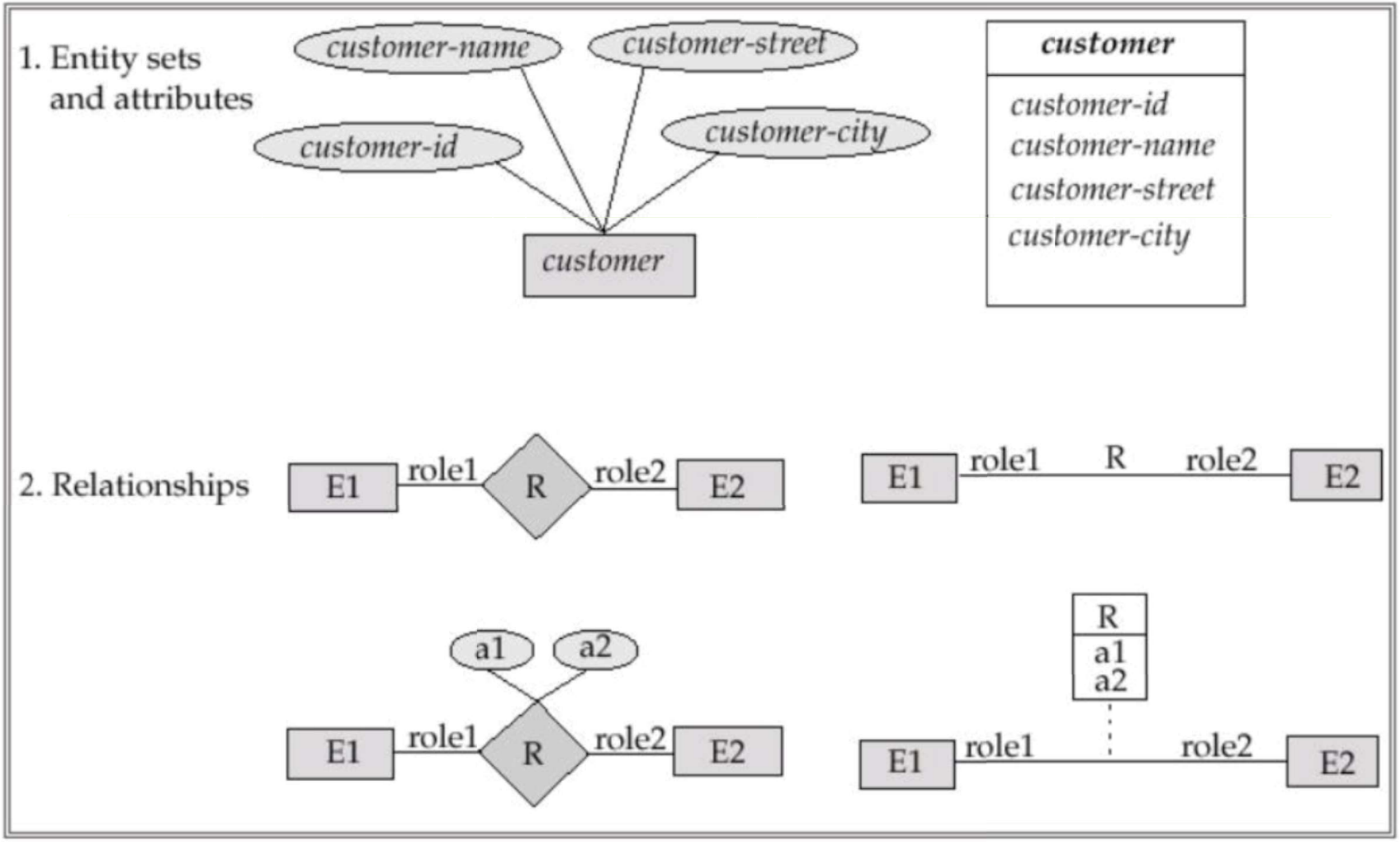
- models of user interactions with the system
- specification of functional modules of the system and their interactions with the system and their interaction.

The Unified Modeling Language (UML) is a standard developed under the auspices of the Object Management Group (OMG) for creating specifications of various components of a software system.

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

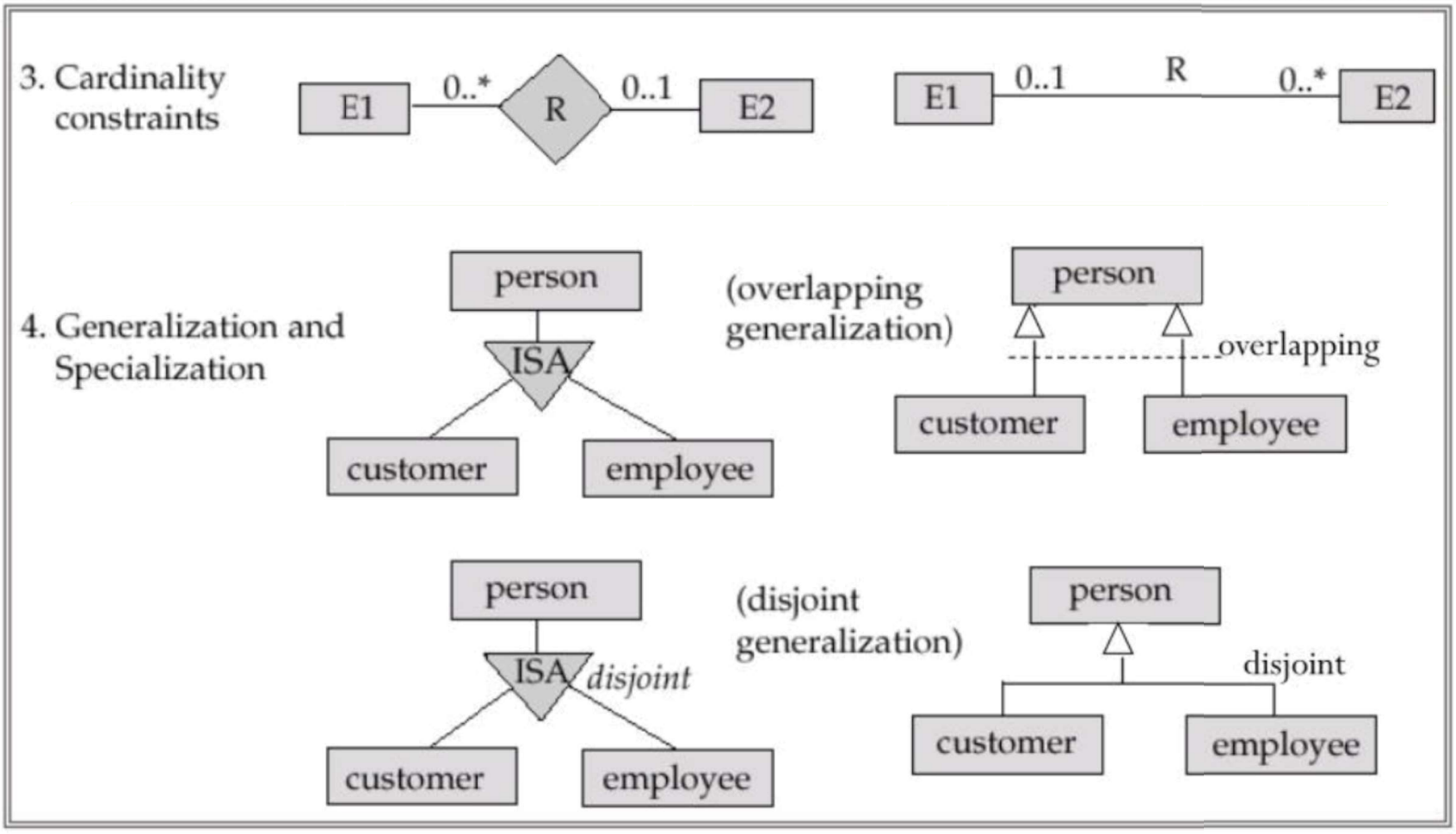
Summary of UML Class Diagram Notation



UML Class Diagrams(Cont.)

- Entity sets are shown as boxes, and attributes are shown within the box, rather than as separate ellipses in E-R diagrams.
- Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.
- The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.
- The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.
- Non-binary relationships drawn using diamonds, just as in ER diagrams

UML Class Diagram Notation (Cont.)



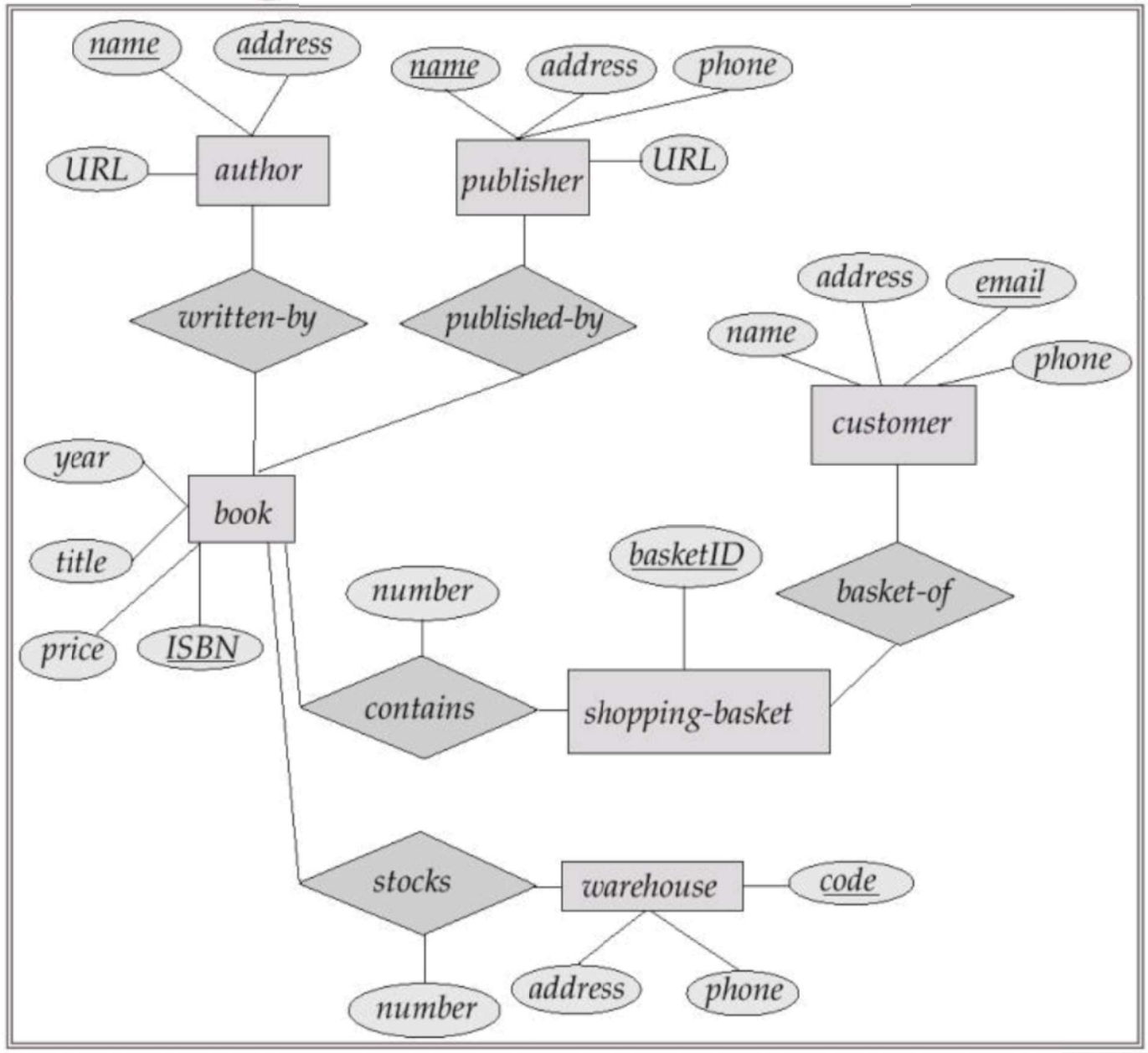
*Note reversal of position in cardinality constraint depiction

*Generalization can use merged or separate arrows independent of disjoint/overlapping

UML Class Diagrams (Cont.)

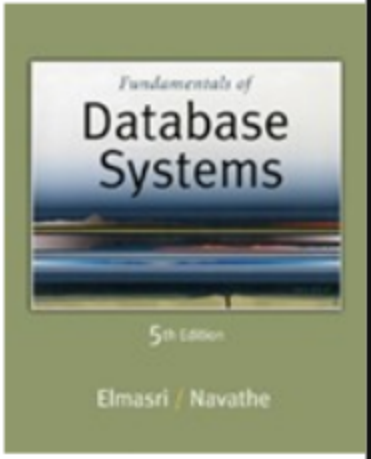
- Cardinality constraints are specified in the form l..h, where l denotes the minimum and h the maximum number of relationships an entity can participate in.
- Beware: the positioning of the constraints is exactly the reverse of the positioning of constraints in E-R diagrams.
- The constraint 0..* on the E2side and 0..1 on the E1 side means that each E2 entity can participate in at most one relationship, whereas each E1 entity can participate in many relationships; in other words, the relationship is many to one from E2 to E1.
- Single values, such as 1 or * may be written on edges; The single value 1 on an edge is treated as equivalent to 1..1, while * is equivalent to 0..*.

E-R Diagram for Bookstore



Chapter 5

The Relational Data Model and Relational Database Constraints



PEARSON
Addison
Wesley

Relational Model Concepts

- The relational model represents the database as a collection of *relations.*
- A Relation is a mathematical concept based on the ideas of sets
- The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper:
  - "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970

Why Relations?

- Very simple model.
- *Often* matches how we think about data.
- Abstract model that underlies SQL, the most important database language today.

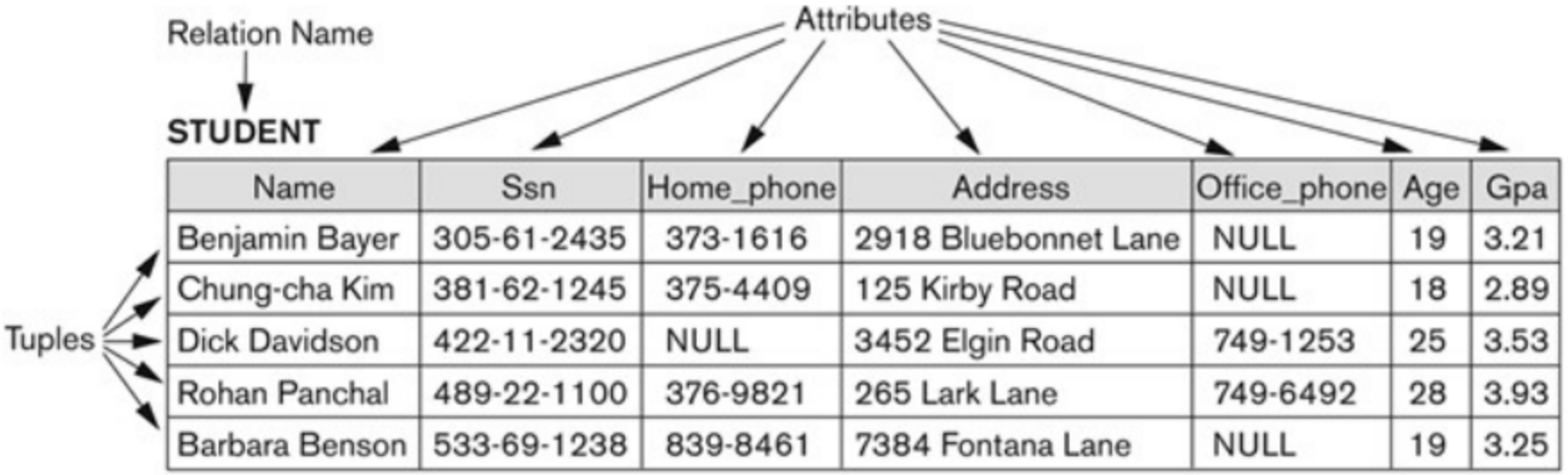
From E/R Diagrams to Relations

- Entity set -> relation.
  - Attributes -> attributes.
- Relationships -> relations whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself.

Informal Definitions

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows** .
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Example of a Relation



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Informal Definitions

- Key of a Relation:
  - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
    - Called the *key*
  - In the STUDENT table, SSN is the key
  - Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
    - Called *artificial key* or *surrogate key*

Formal Definitions - Schema

- The **Schema** (or description) of a Relation:
  - Denoted by R(A1, A2, ...An)
  - R is the **name** of the relation
  - The **attributes** of the relation are A1, A2, ..., An
- Example:
  CUSTOMER (Cust-id, Cust-name, Address, Phone#)
  - CUSTOMER is the relation name
  - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values.
- **Domain** is the set of allowable values for one or more attributes.
  - For example, the domain of Cust-id is 6 digit numbers.

Comparative Terms

Formal	Oracle
Relation schema	Table description
Relation	Table
Tuple	Row
Attribute	Column
Domain	Value set

- Notation
  Course (courseno, subject, equipment)
  Student(studno,name,hons)
  Enrol(studno,courseno,labmark)

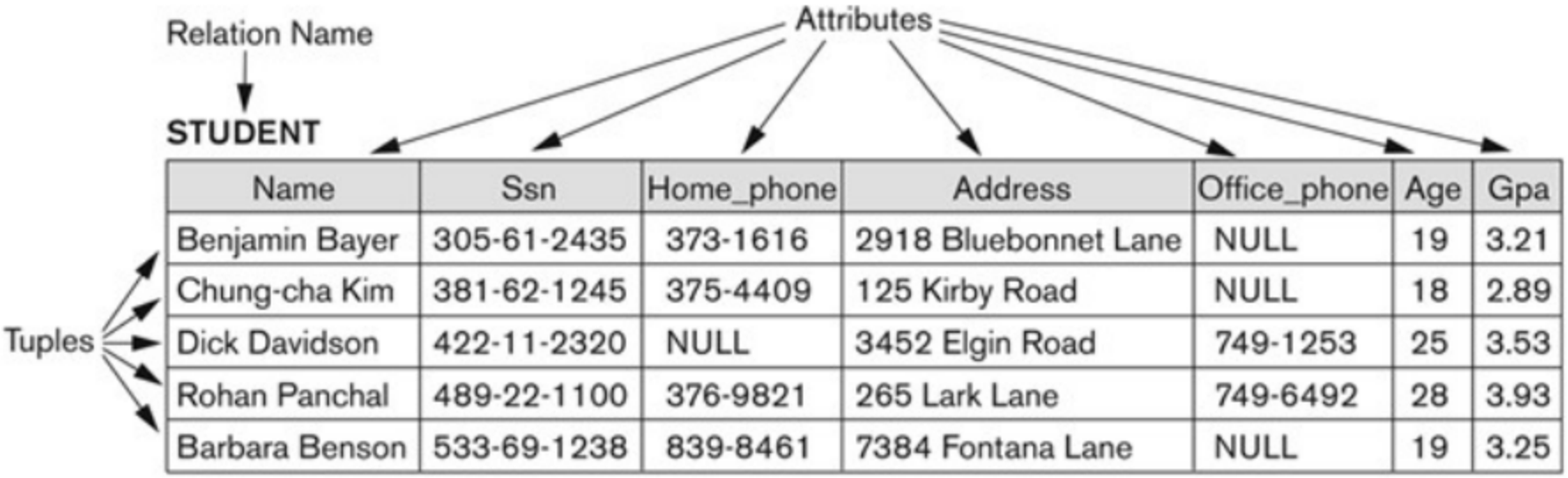
Relational Model Terminology

- **Tuple** is a row of a relation.
  - A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:
    - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
- **Degree** is the number of attributes in a relation.
- **Cardinality** is the number of tuples in a relation.
- **Relational Database** is a collection of normalized relations with distinct relation names.

Definition Summary

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Example – A relation STUDENT



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Properties of Relations

- Relation name is distinct from all other relation names in relational schema.
- Each cell of relation contains exactly one atomic (single) value.
- Each attribute has a distinct name.
- Values of an attribute are all from the same domain.
- Each tuple is distinct; there are no duplicate tuples.

Properties of Relations

- Order of attributes has no significance.
- Order of tuples has no significance, theoretically.
- A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.
  - Represents the absence of a value and is not the same as zero or spaces, which are values.

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of constraints in the relational model:
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints
- Another implicit constraint is the **domain** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null** , if allowed for that attribute)