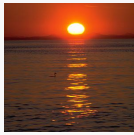
Entity-Relationship Model

崇志宏

http://cse.seu.edu.cn/people/zhchong/index.htm

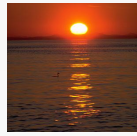
13814066974

WebDB & P2P Open Group, Southeast University



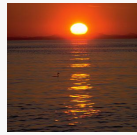
Chapter 6: Entity-Relationship Model

- 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
- UML



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	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

loan_number	amount
L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

loan

Relationship Sets

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

  Example:

  Hayes — *customer* entity; *depositor* — relationship set; A-102 — *account* entity

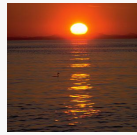
- A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

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

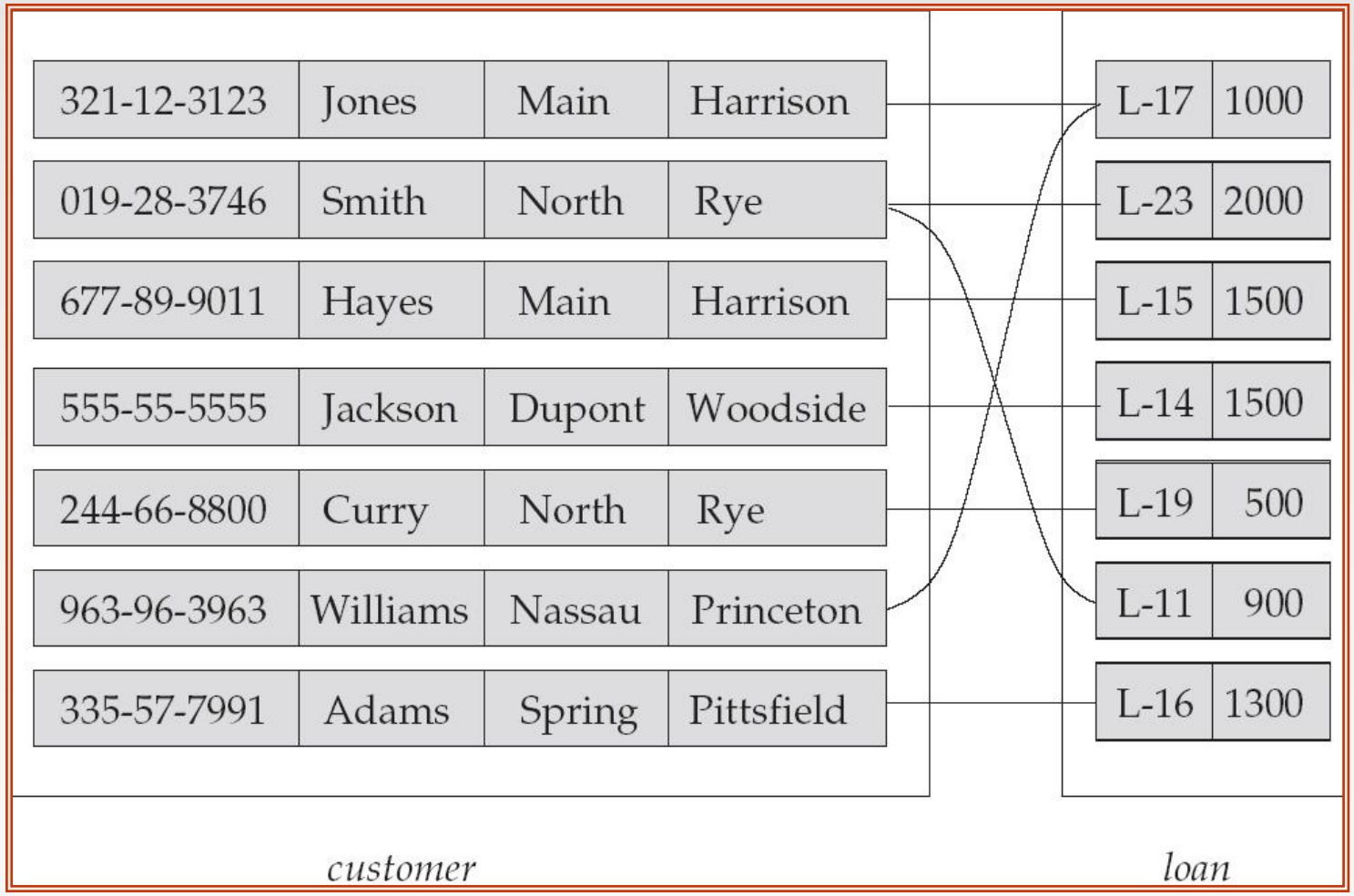
  where $(e_1, e_2, \ldots, e_n)$ is a relationship

  - Example:

    (Hayes, A-102) $\in$ *depositor*

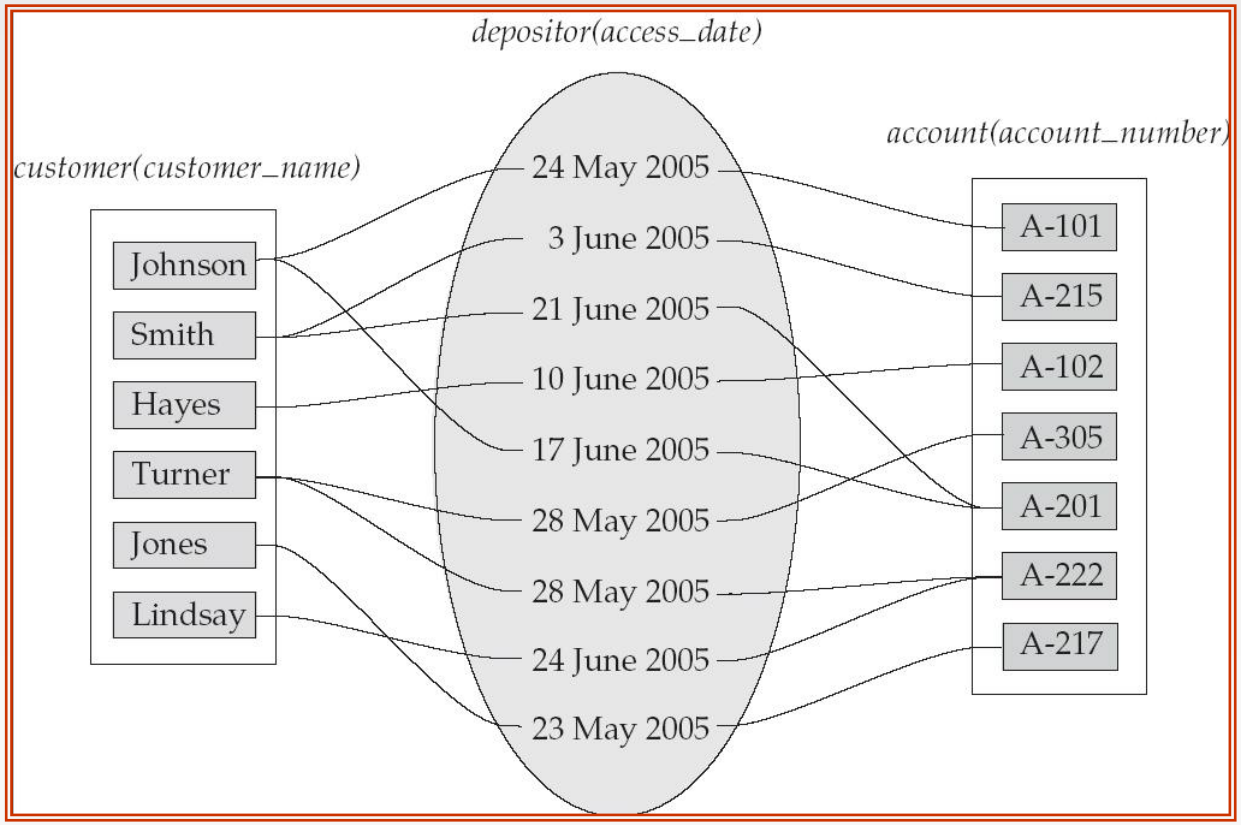


Relationship Set *borrower*



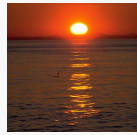
Relationship Sets (Cont.)

- 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.)

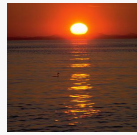


Attributes

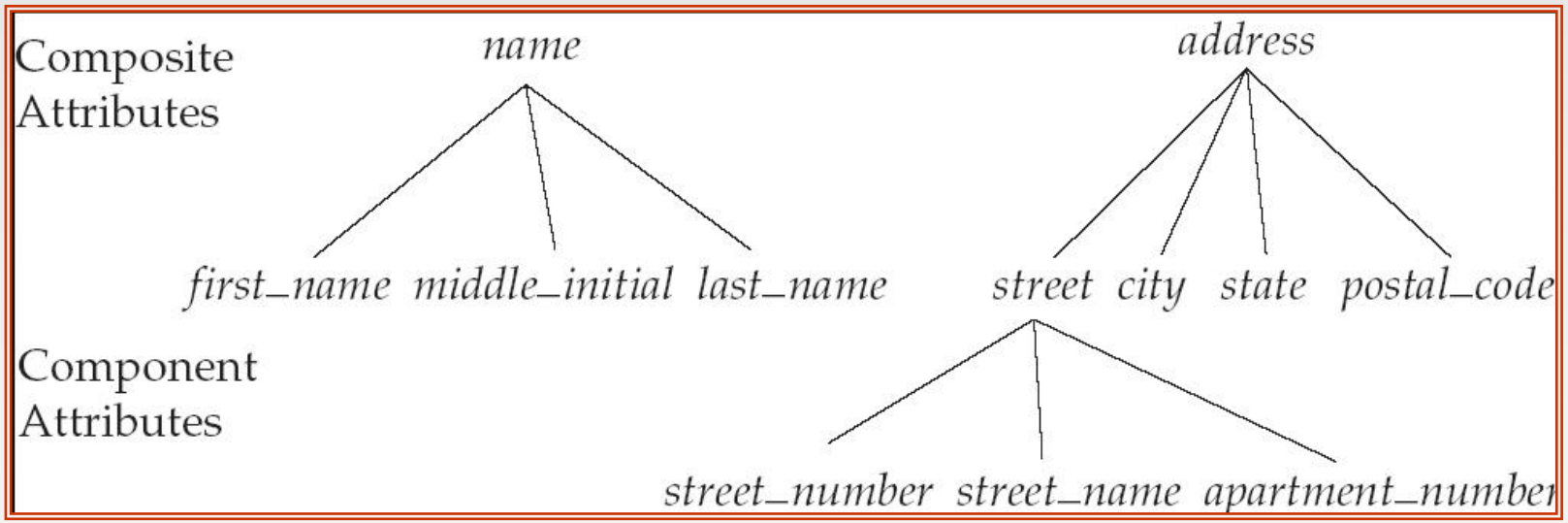
- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  Example:

  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

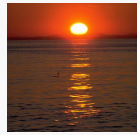


Composite Attributes

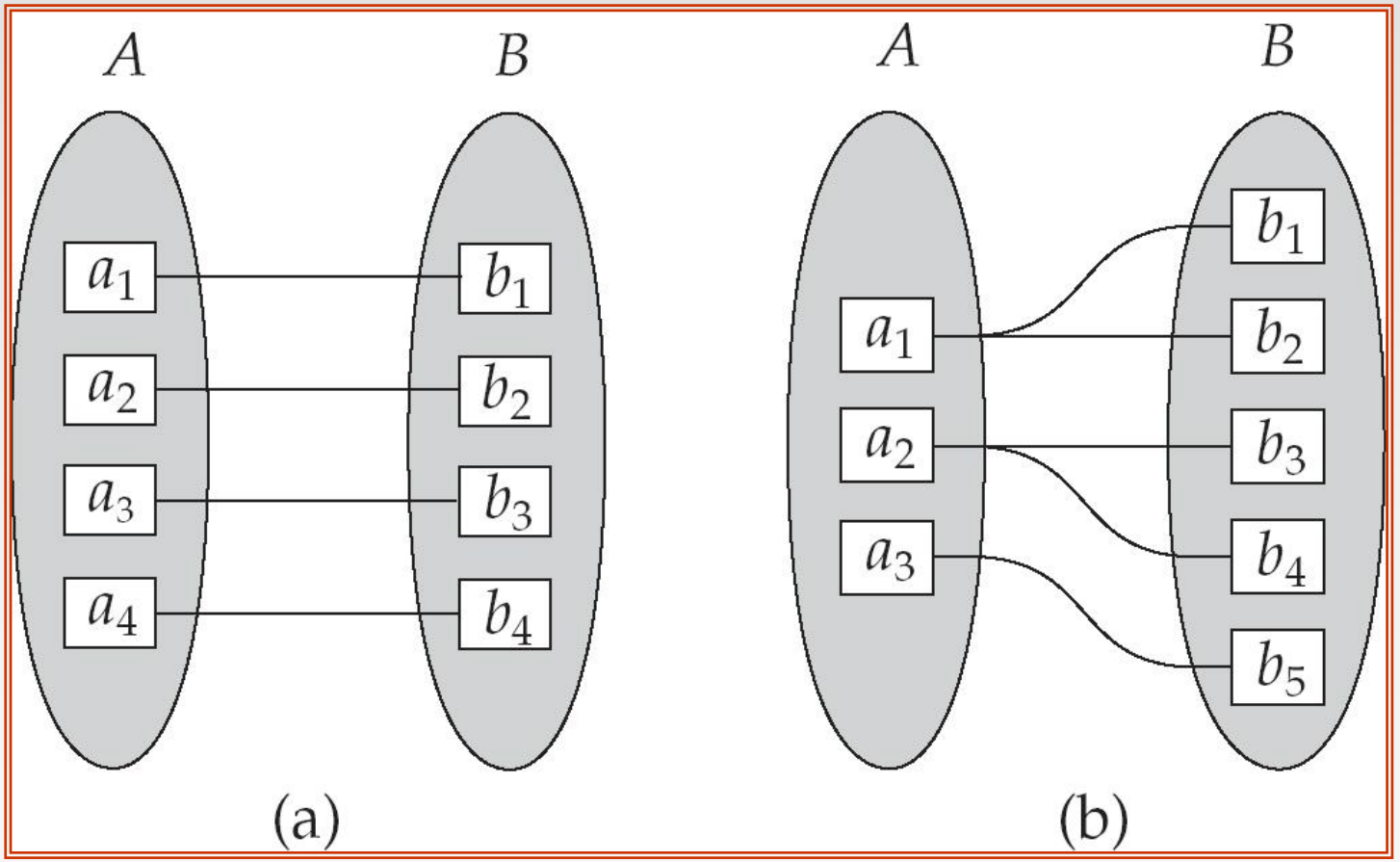


Mapping Cardinality Constraints

- 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

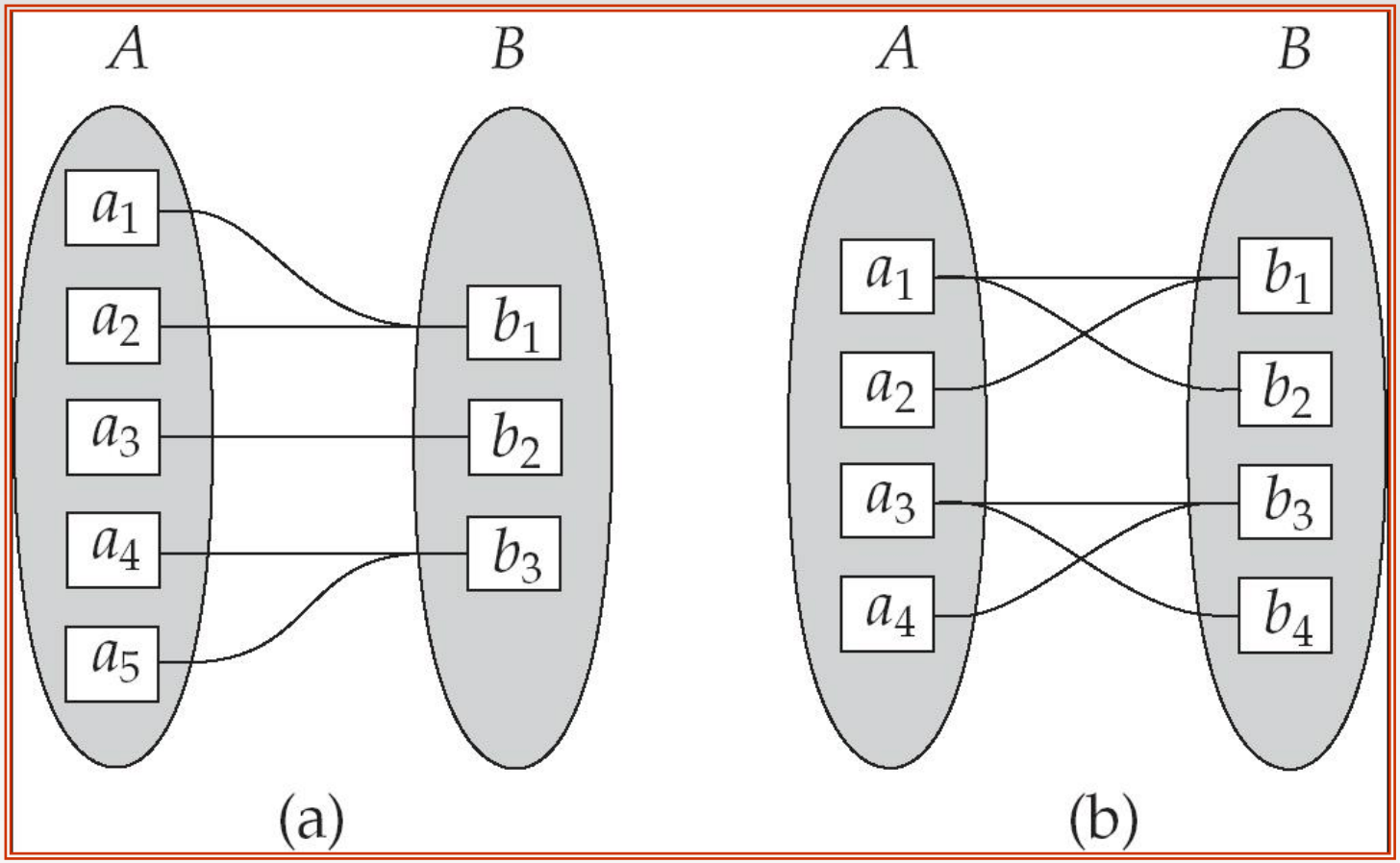


One to one

One to many

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

Mapping Cardinalities



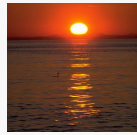
Many to one

Many to many

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

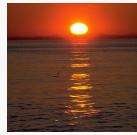
Keys

- 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**.

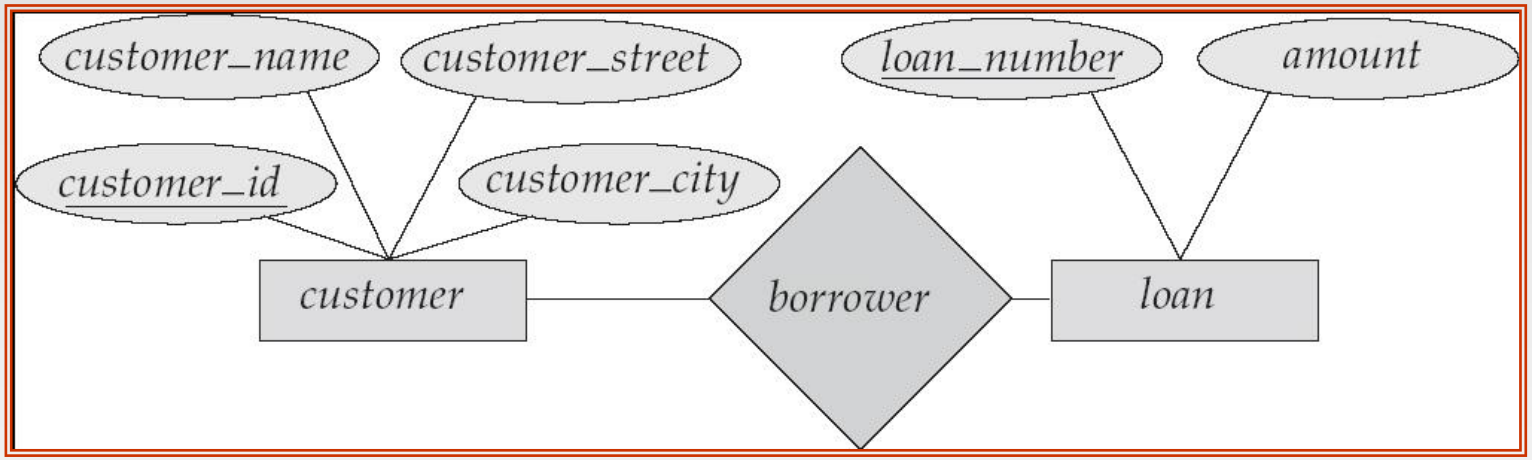


Keys for Relationship Sets

- 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

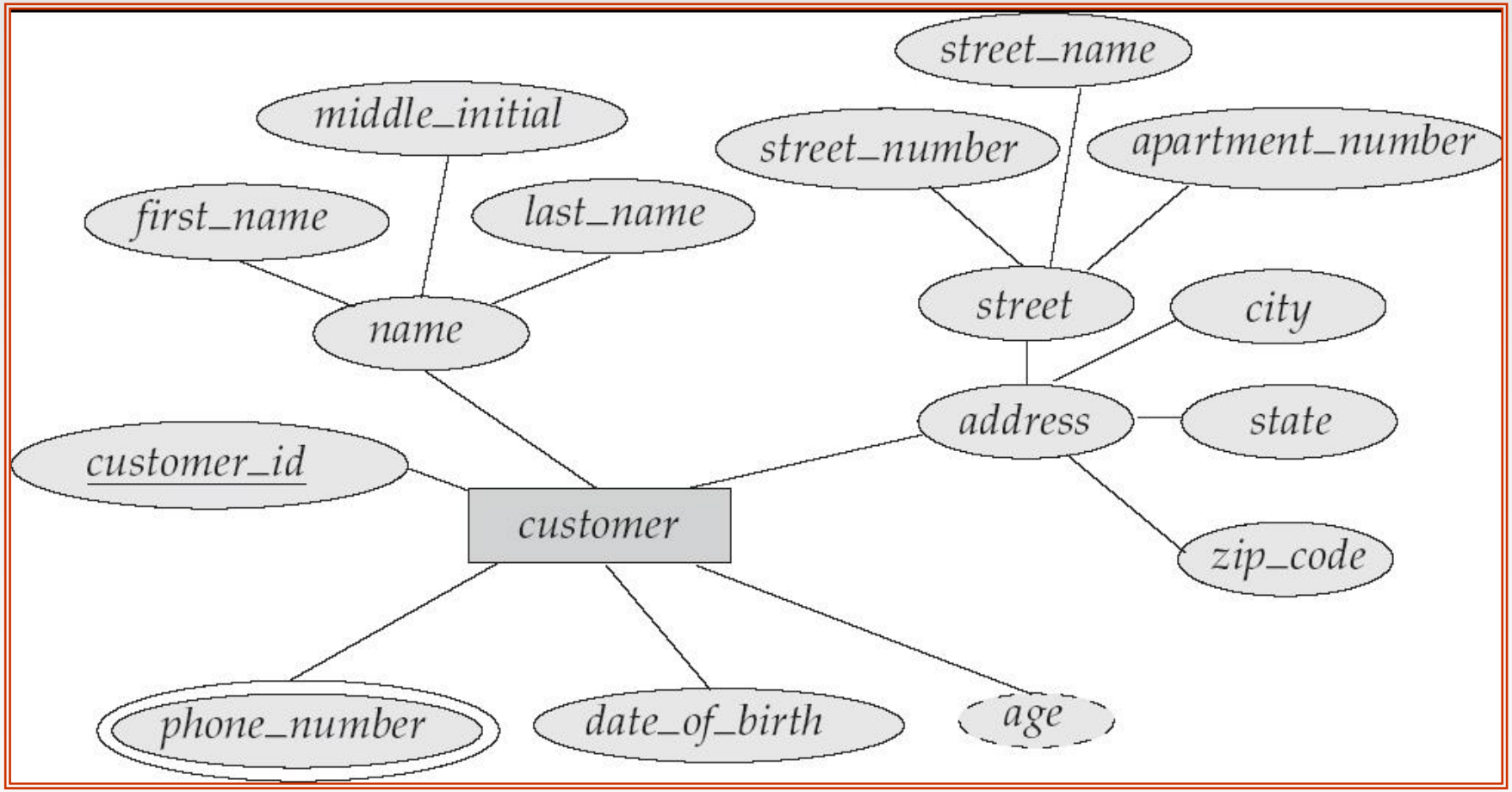


E-R Diagrams

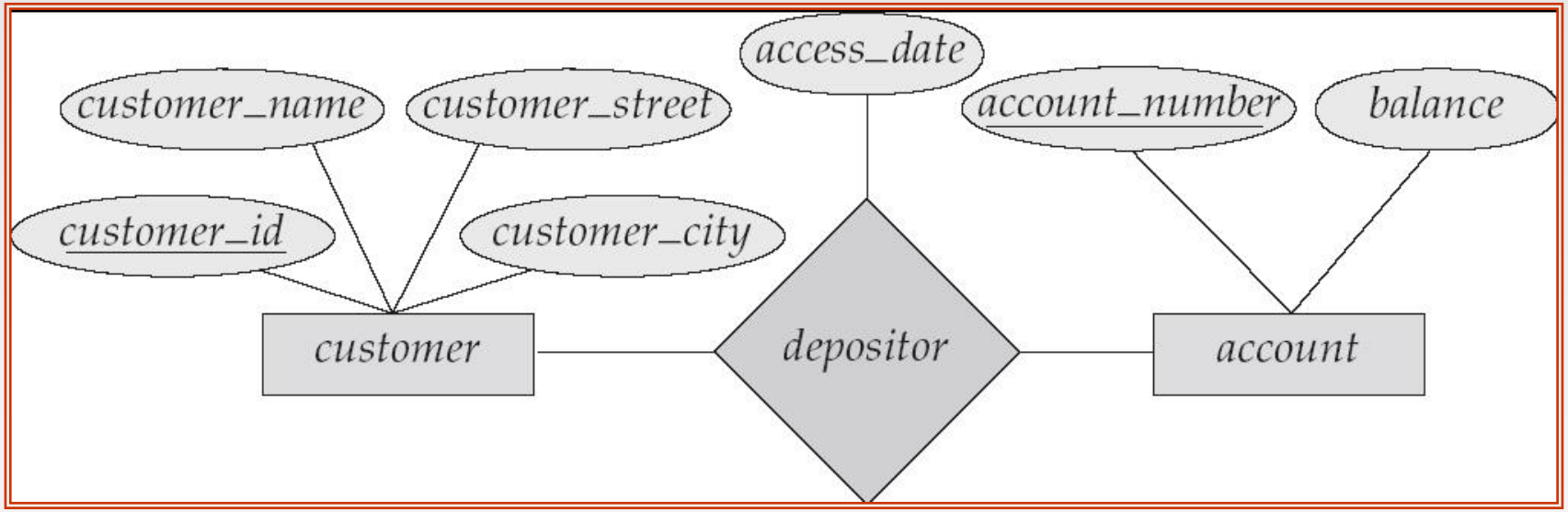


- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Ellipses represent attributes
  - Double ellipses represent multivalued attributes.
  - Dashed ellipses denote derived attributes.
- Underline indicates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

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

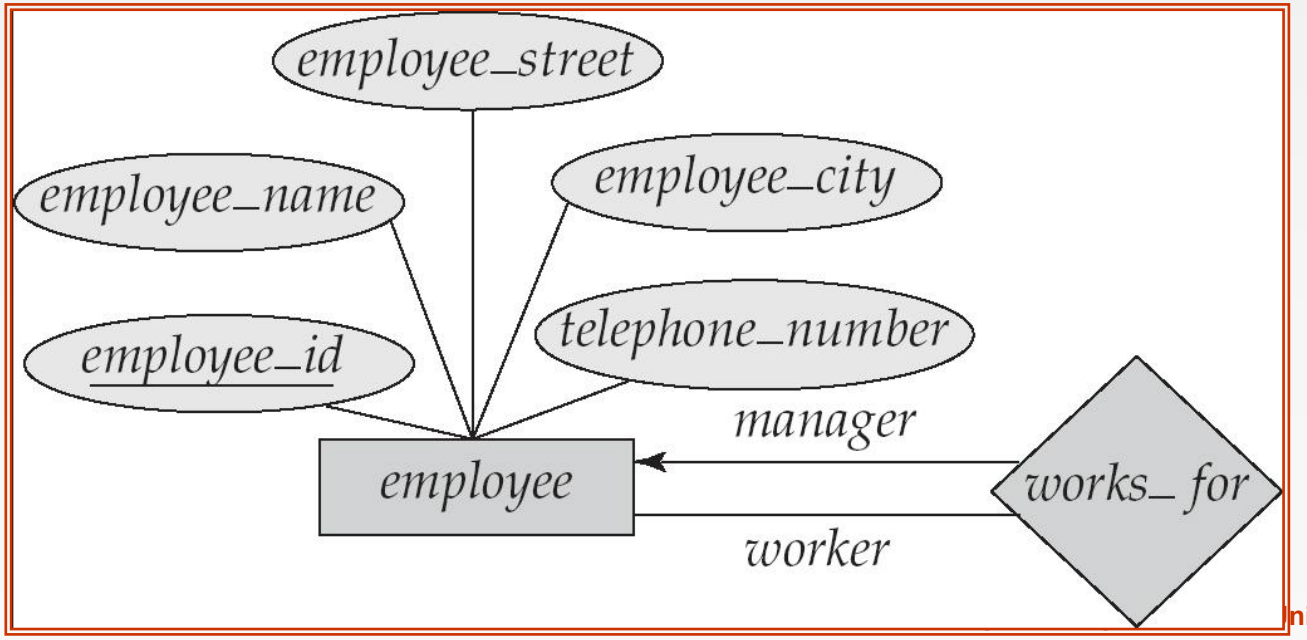


Relationship Sets with Attributes



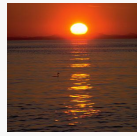
Roles

- 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



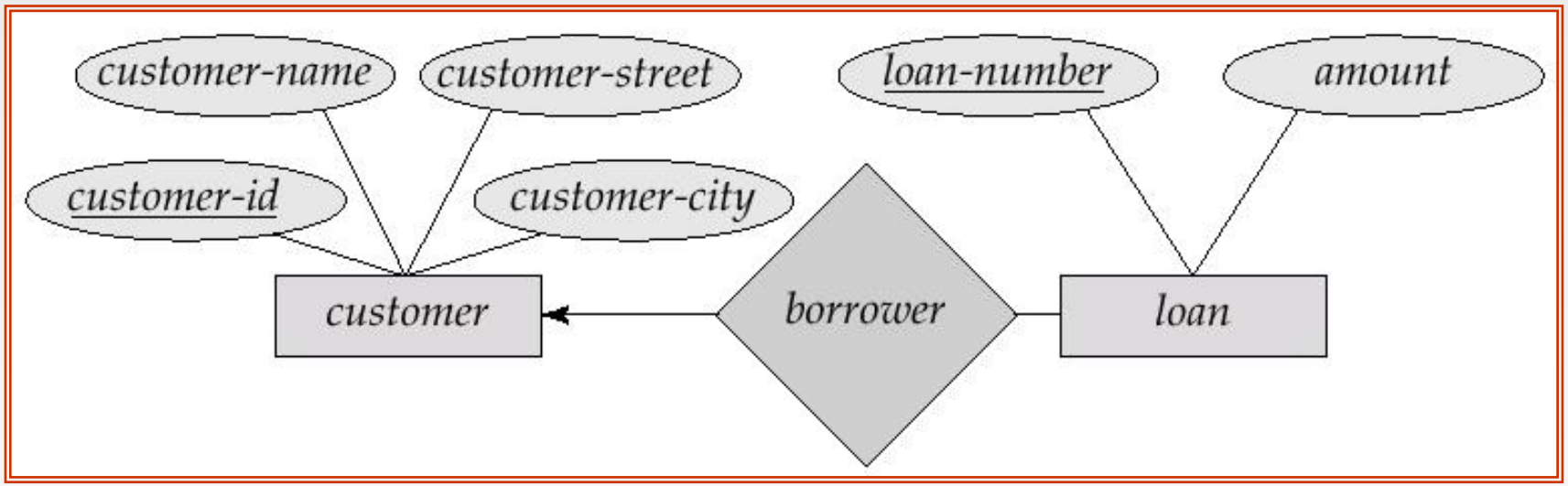
Cardinality Constraints

- We express cardinality constraints by drawing either a directed line ($\rightarrow$), 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*



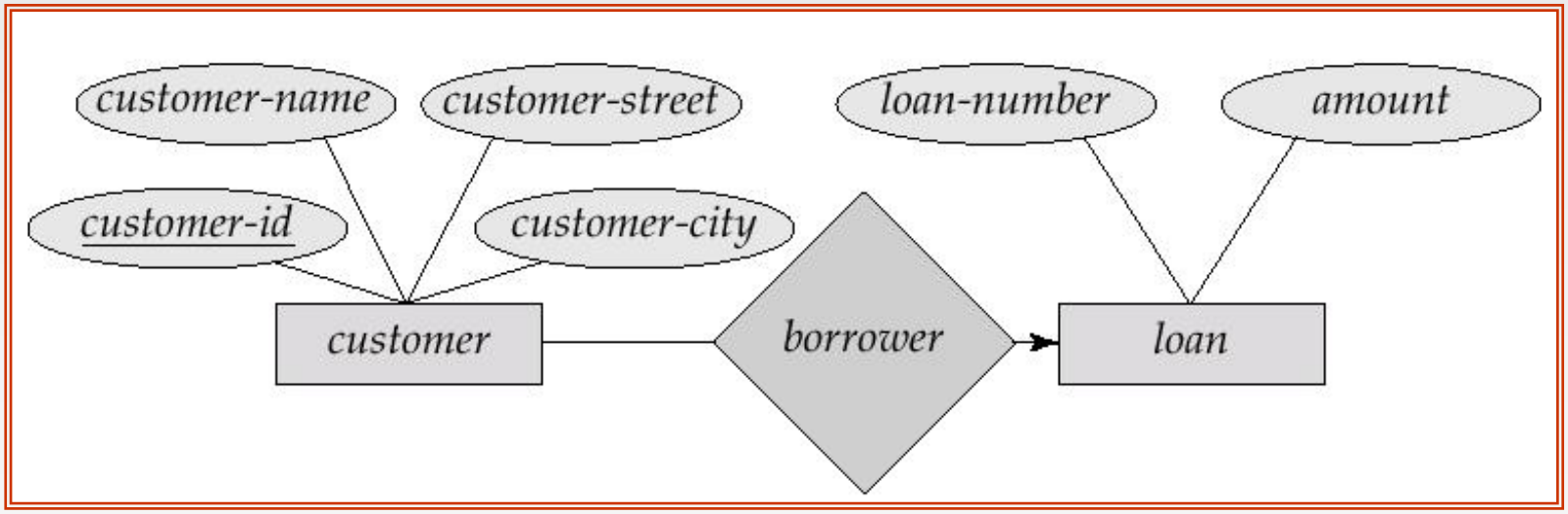
One-To-Many Relationship

- 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*



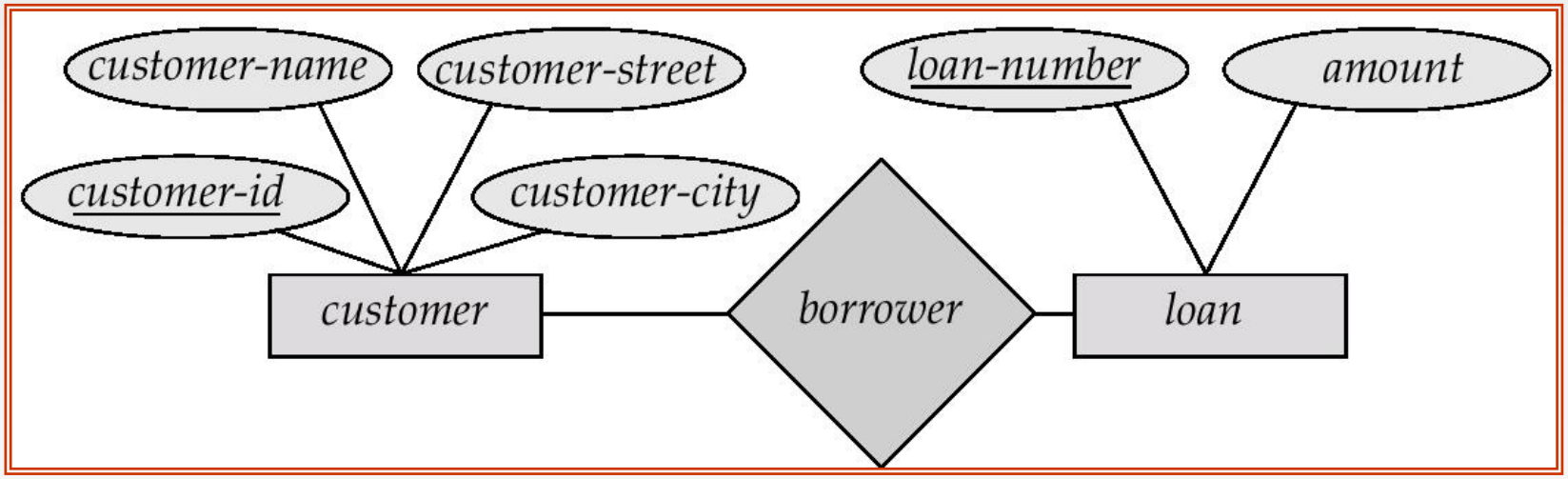
Many-To-One Relationships

- 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*



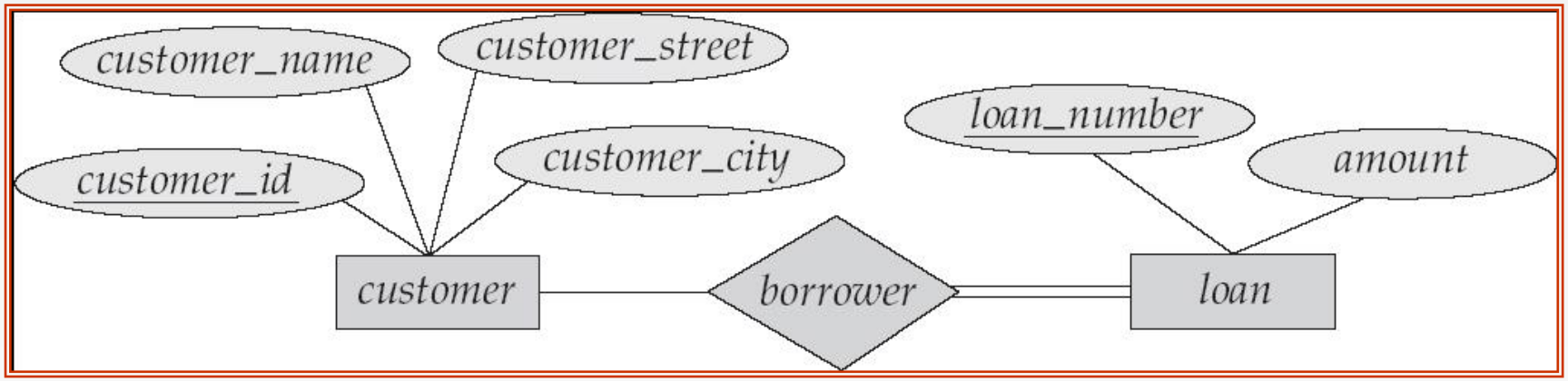
Many-To-Many Relationship

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



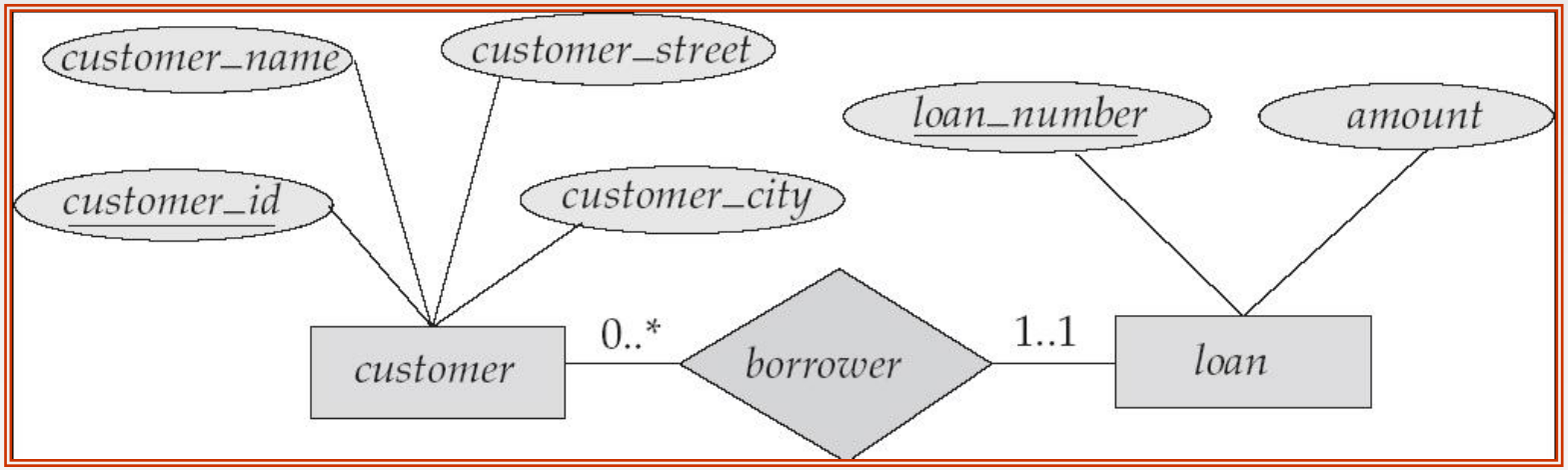
Participation of an Entity Set in a Relationship Set

- 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

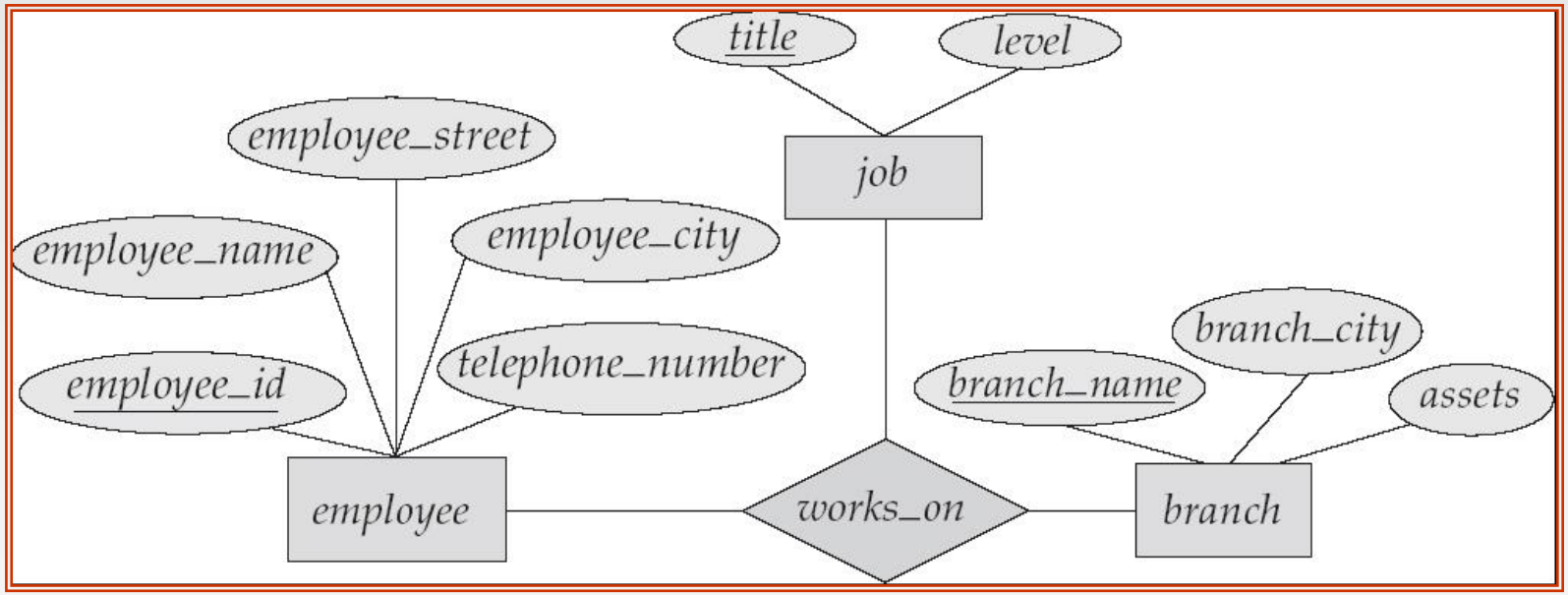


Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints

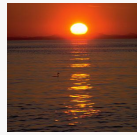


E-R Diagram with a Ternary Relationship



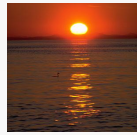
Design Issues

- **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**



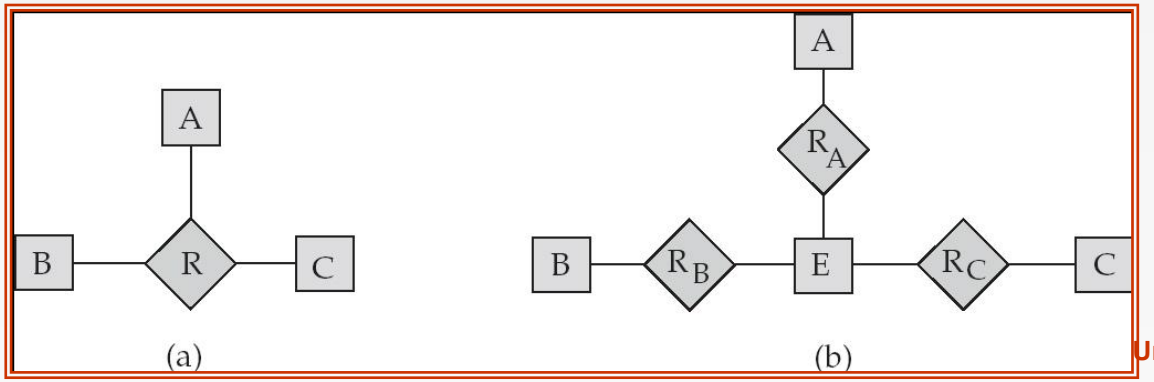
Binary Vs. Non-Binary Relationships

- 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*



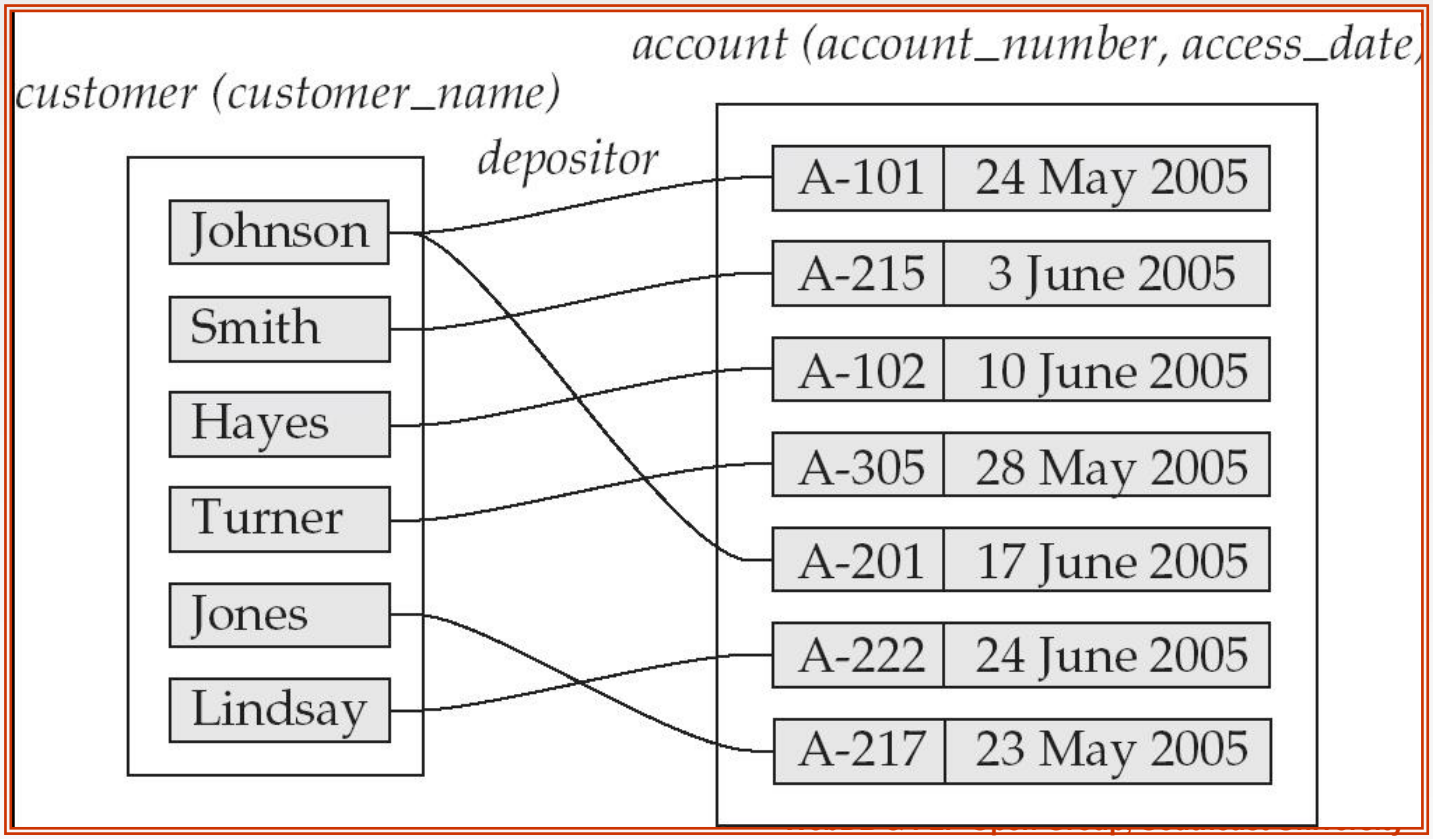
Converting Non-Binary Relationships to Binary Form

- 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

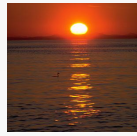


Mapping Cardinalities affect ER Design

- 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

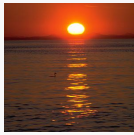


How about doing an ER design interactively on the board? Suggest an application to be modeled.

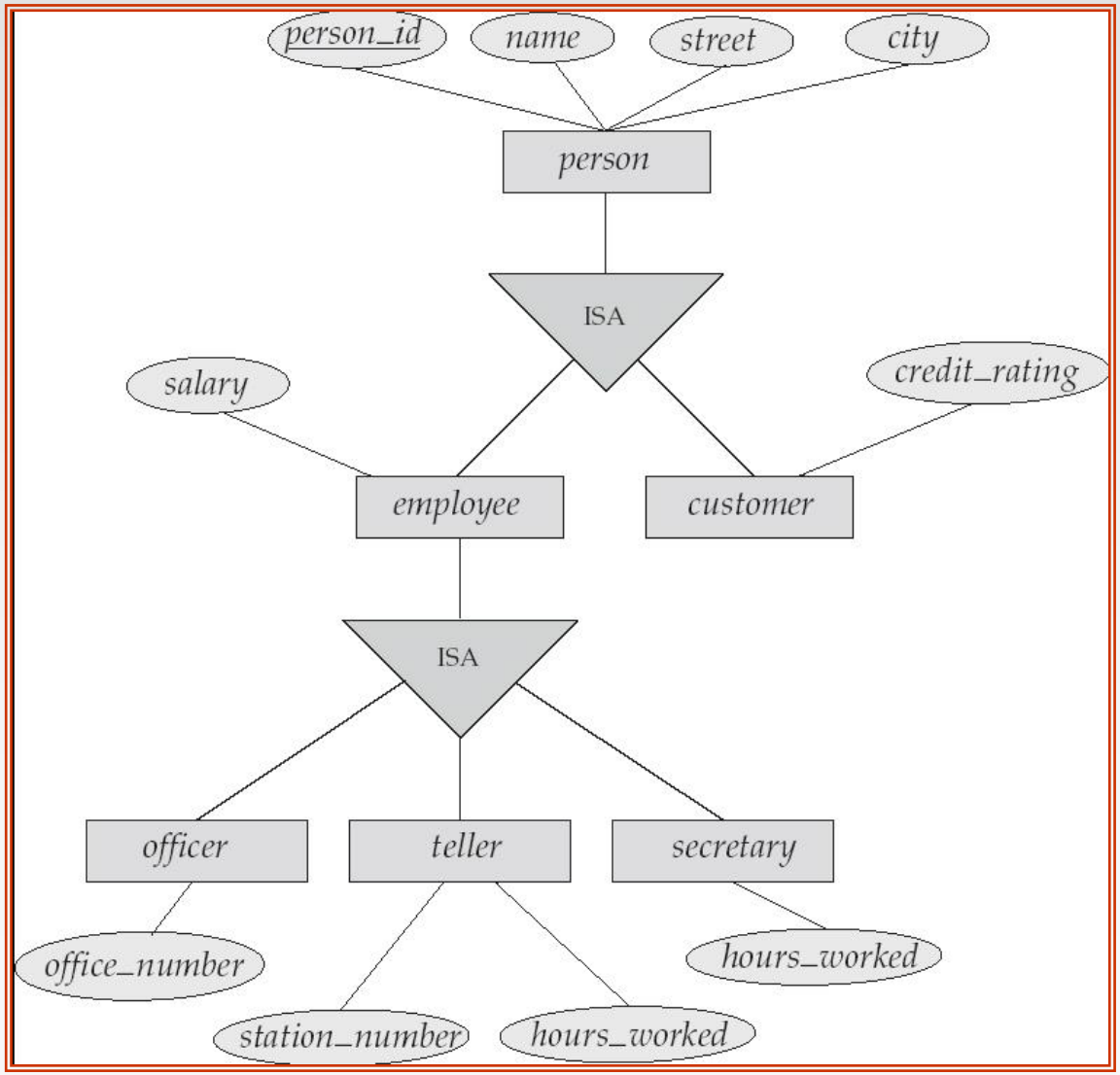


Extended E-R Features: Specialization

- 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.

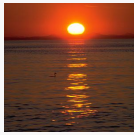


Specialization Example



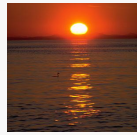
Extended ER Features: Generalization

- **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.



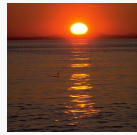
Specialization and Generalization (Cont.)

- 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



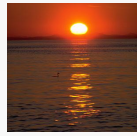
Design Constraints on a Specialization/Generalization

- 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

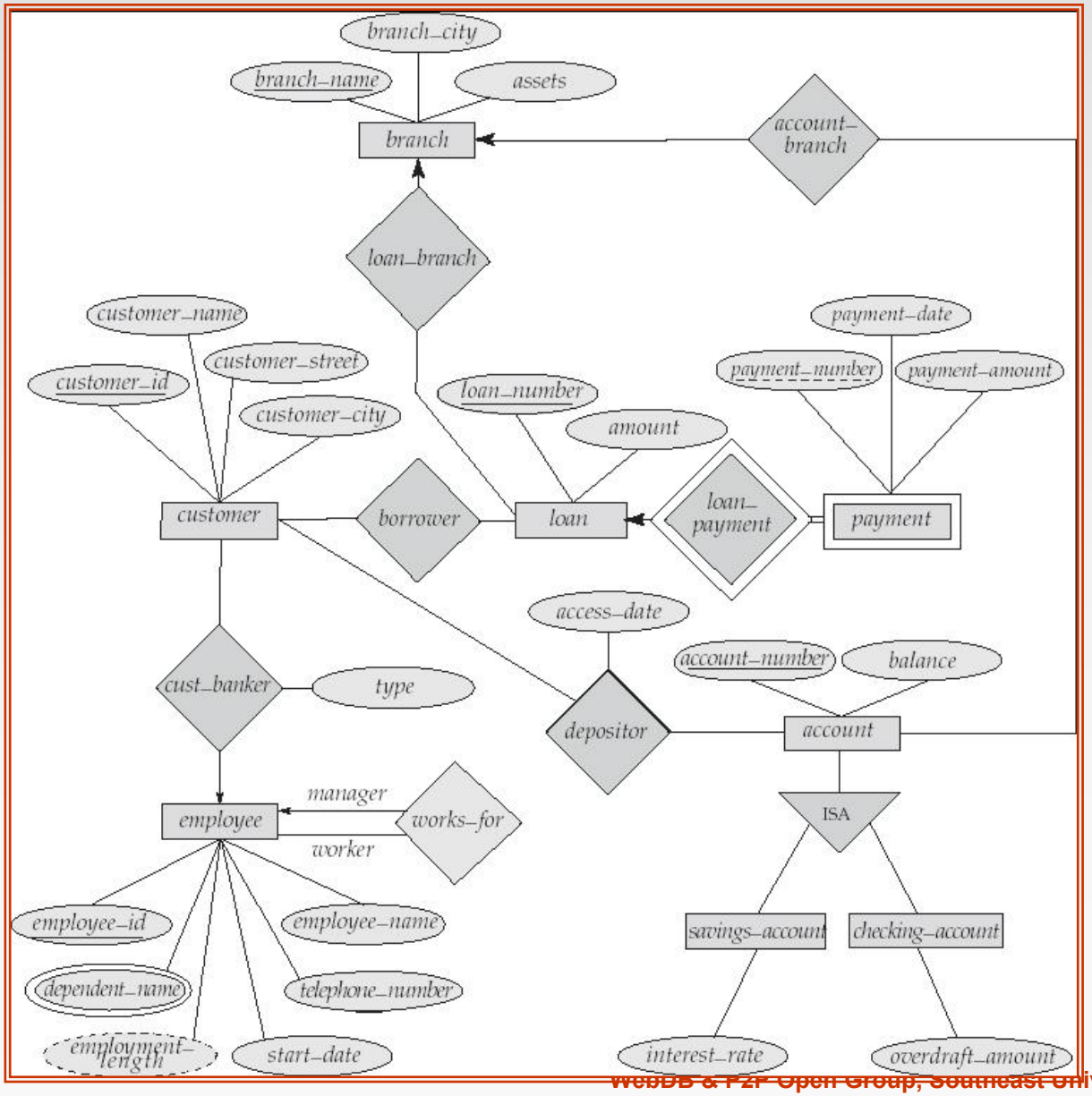


Design Constraints on a Specialization/Generalization (Cont.)

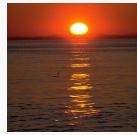
- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total** : an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong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



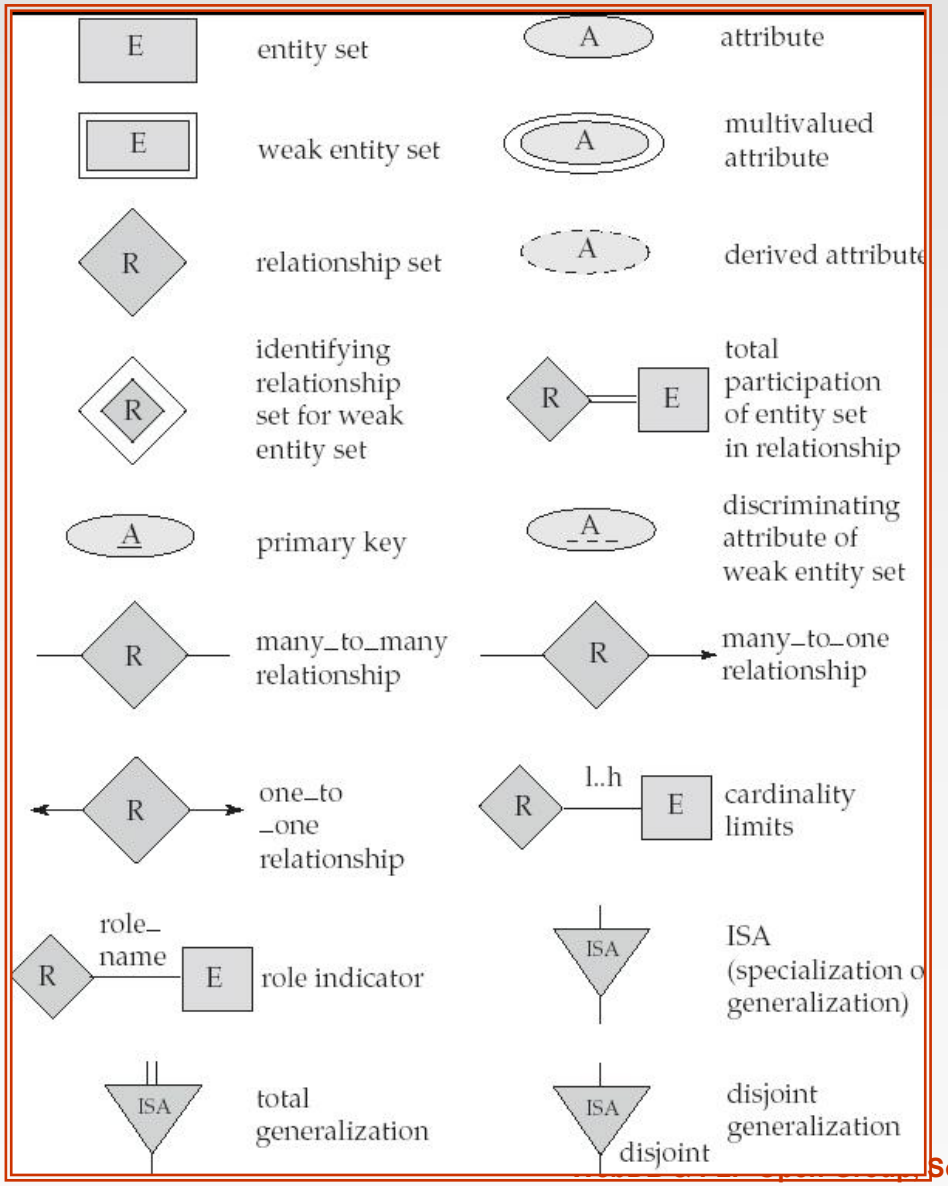
E-R Diagram for a Banking Enterprise



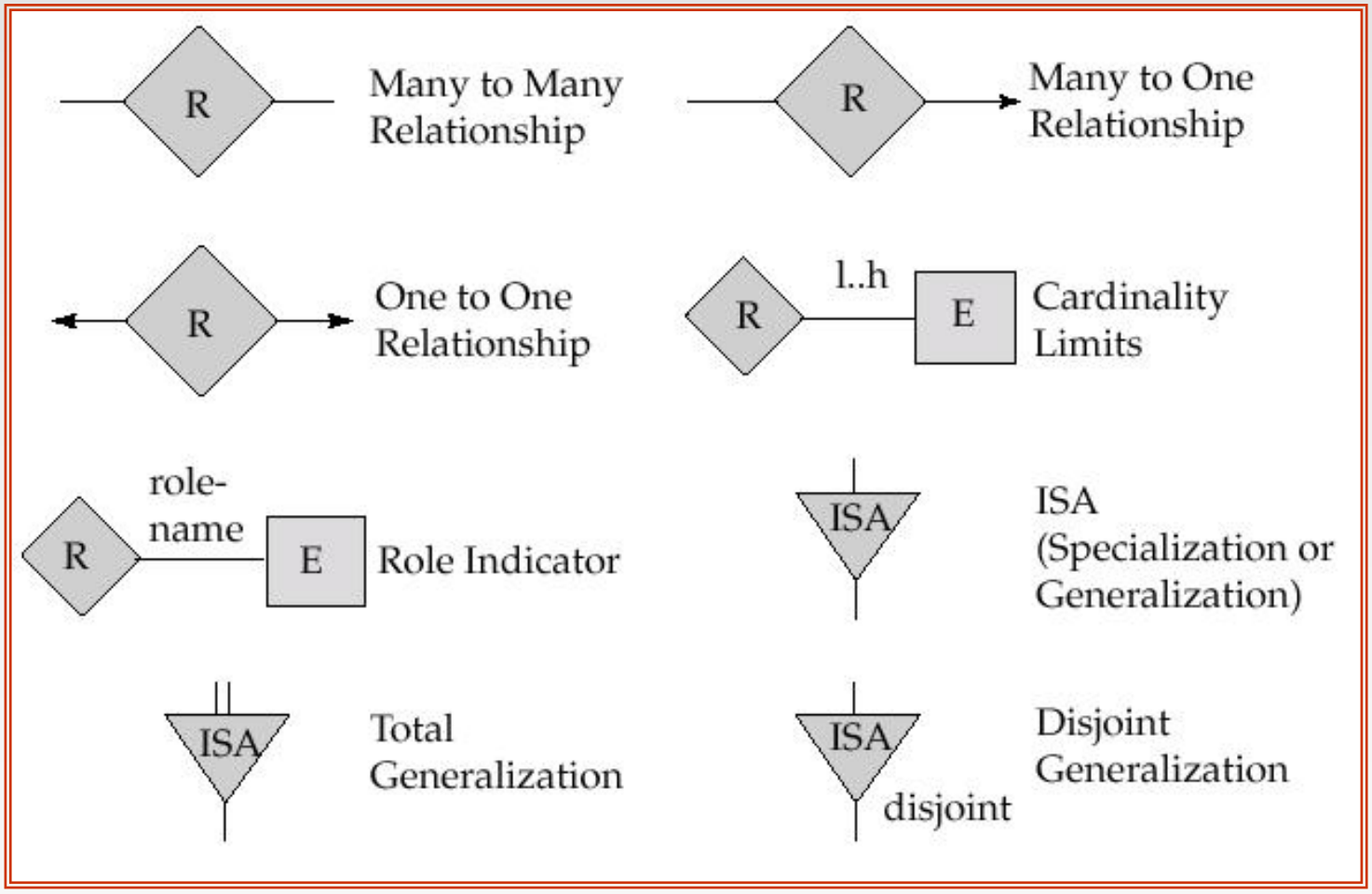
How about doing another ER design interactively on the board?



Summary of Symbols Used in E-R Notation

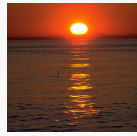


Summary of Symbols (Cont.)



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.

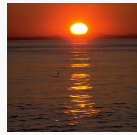


Representing Entity Sets as Schemas

- 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 =

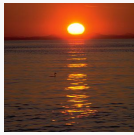
  (*loan_number*, *payment_number*, *payment_date*, *payment_amount*)



Representing Relationship Sets as Schemas

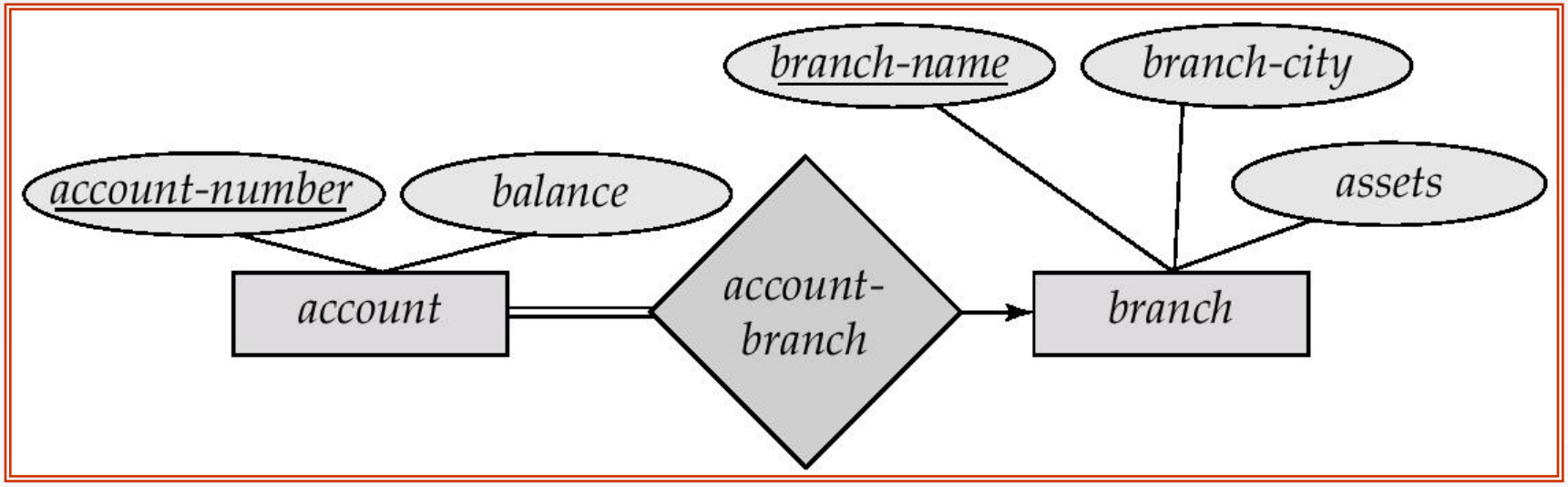
- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set borrower

  borrower = (*customer_id, loan_number*)



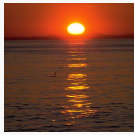
Redundancy of Schemas

- 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*



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 attributes
    name.first_name and *name.last_name*
- 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 Jack and Jane maps to two tuples:
      (123-45-6789 , Jack) and (123-45-6789 , Jane)

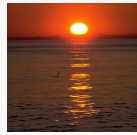


Representing Specialization via Schemas

- 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

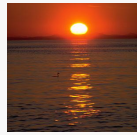


Representing Specialization as Schemas (Cont.)

- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

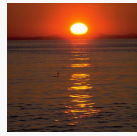
schema	attributes
person	*name, street, city*
customer	*name, street, city, credit_rating*
employee	*name, street, city, salary*

  - 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



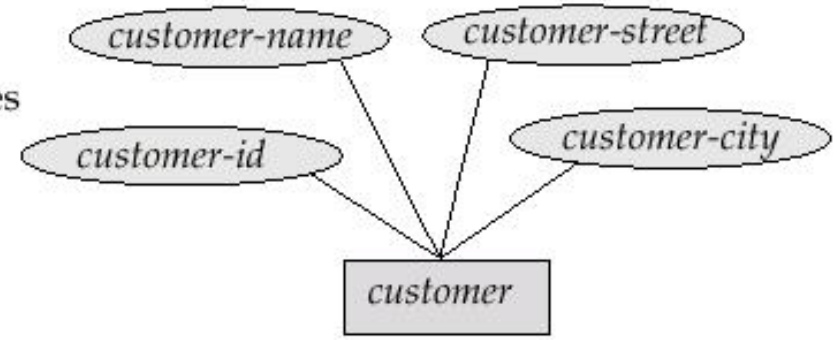
UML

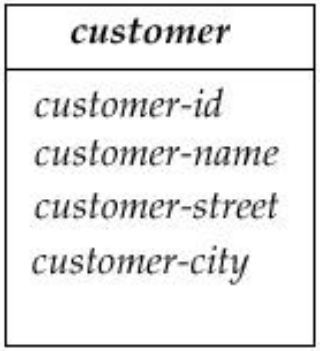
- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.



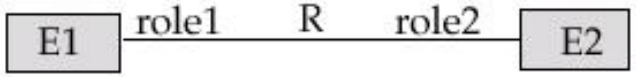
Summary of UML Class Diagram Notation

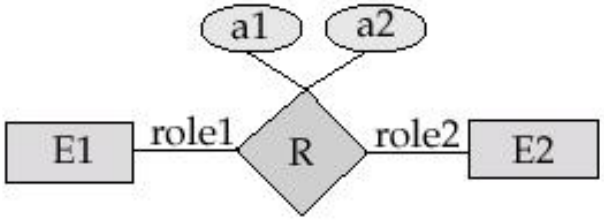
1. Entity sets and attributes

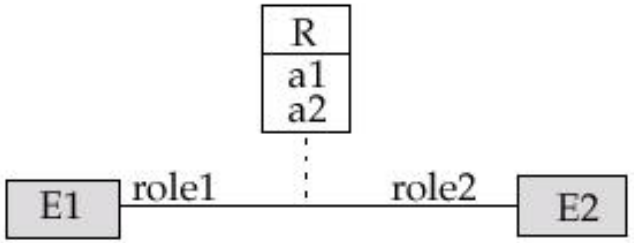




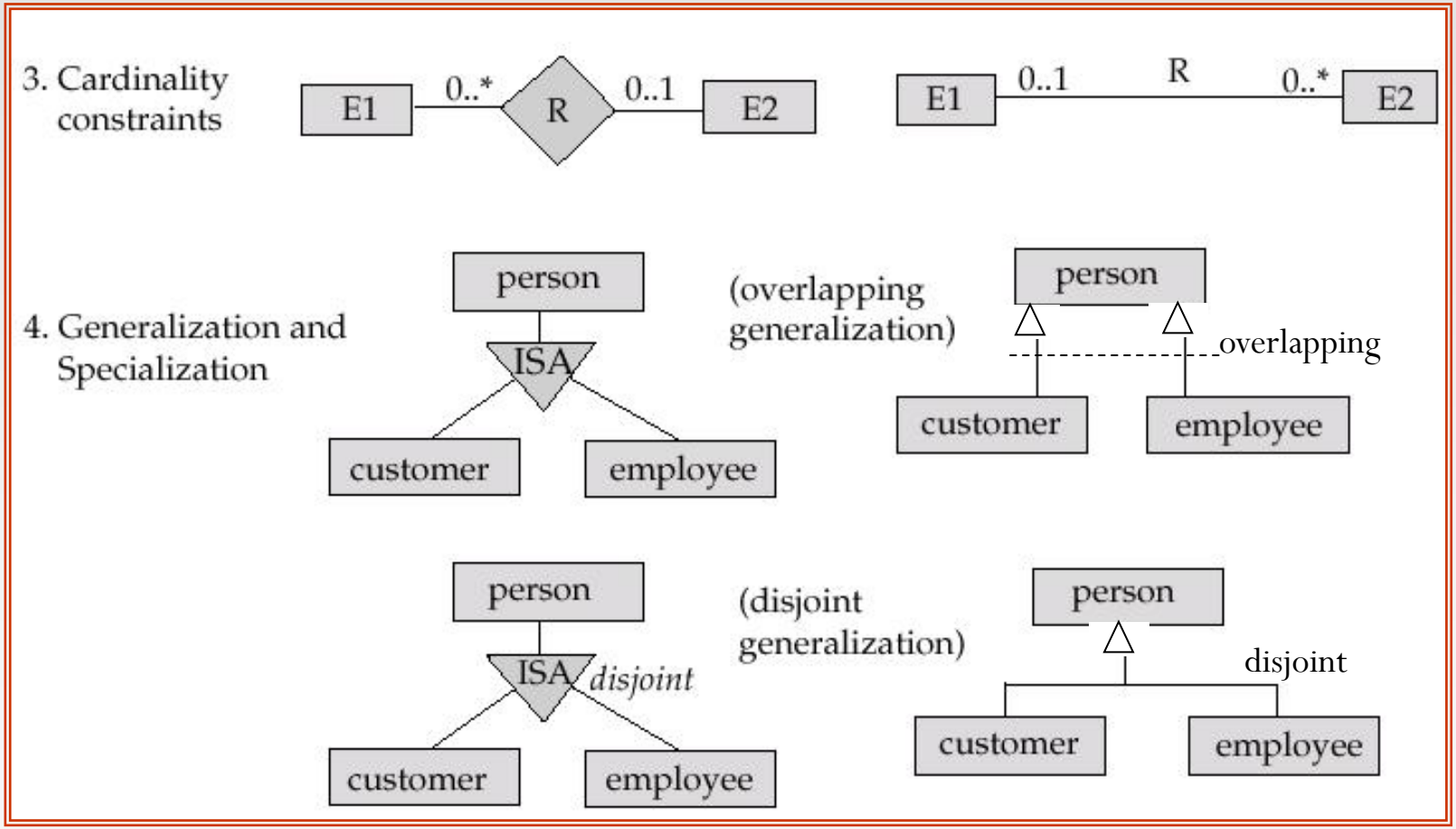
2. Relationships







UML Class Diagram Notation (Cont.)



*Note reversal of position in cardinality constraint depiction

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

End of Chapter 2