BKM_DATS: Databázové systémy

6. Entity-Relationship Model

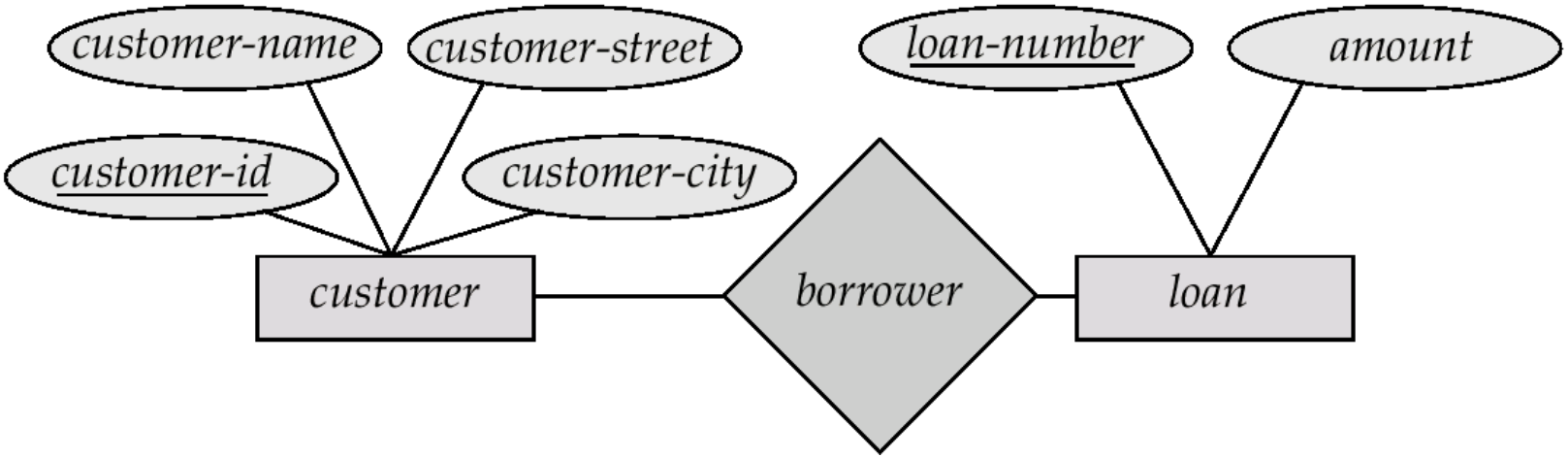
Vlastislav Dohnal

Entity-Relationship Model

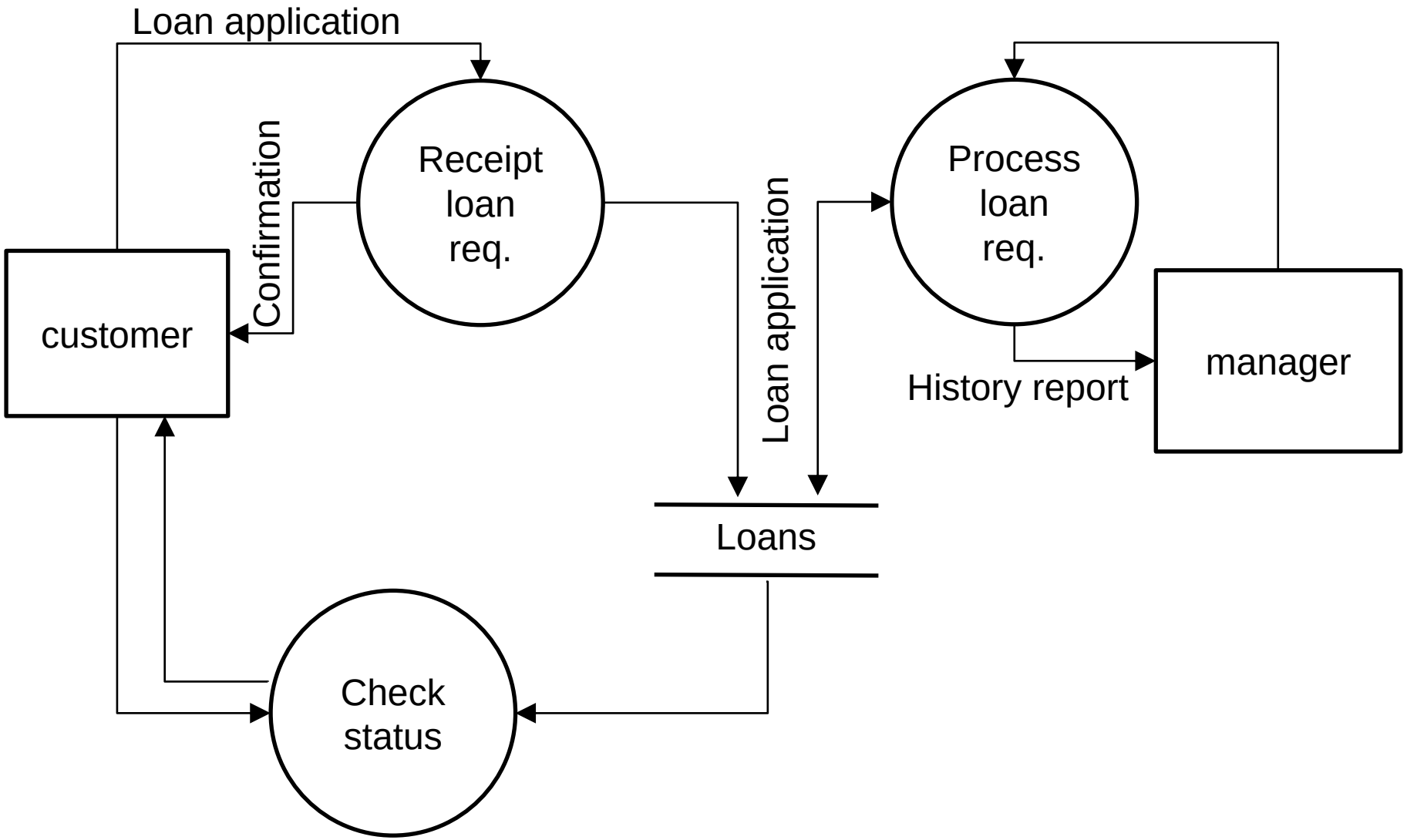
- Modeling
- E-R Diagram
  - Entity Sets and Relationships
  - Weak Entity Sets
  - Extended E-R Features
  - Design of the Bank Database
- UML

Entitně-relační model

- Konceptuální model používaný při vývoji IS
  - Během analýzy požadavků
  - Modeluje *informace* ukládané v DB
- Snadný pro porozumění
  - Zákazník mu „rozumí“



DFD – půjčka v bance



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: person has a *name* and *address*
- 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; *borrower* — relationship set; A-102 — *loan* entity

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

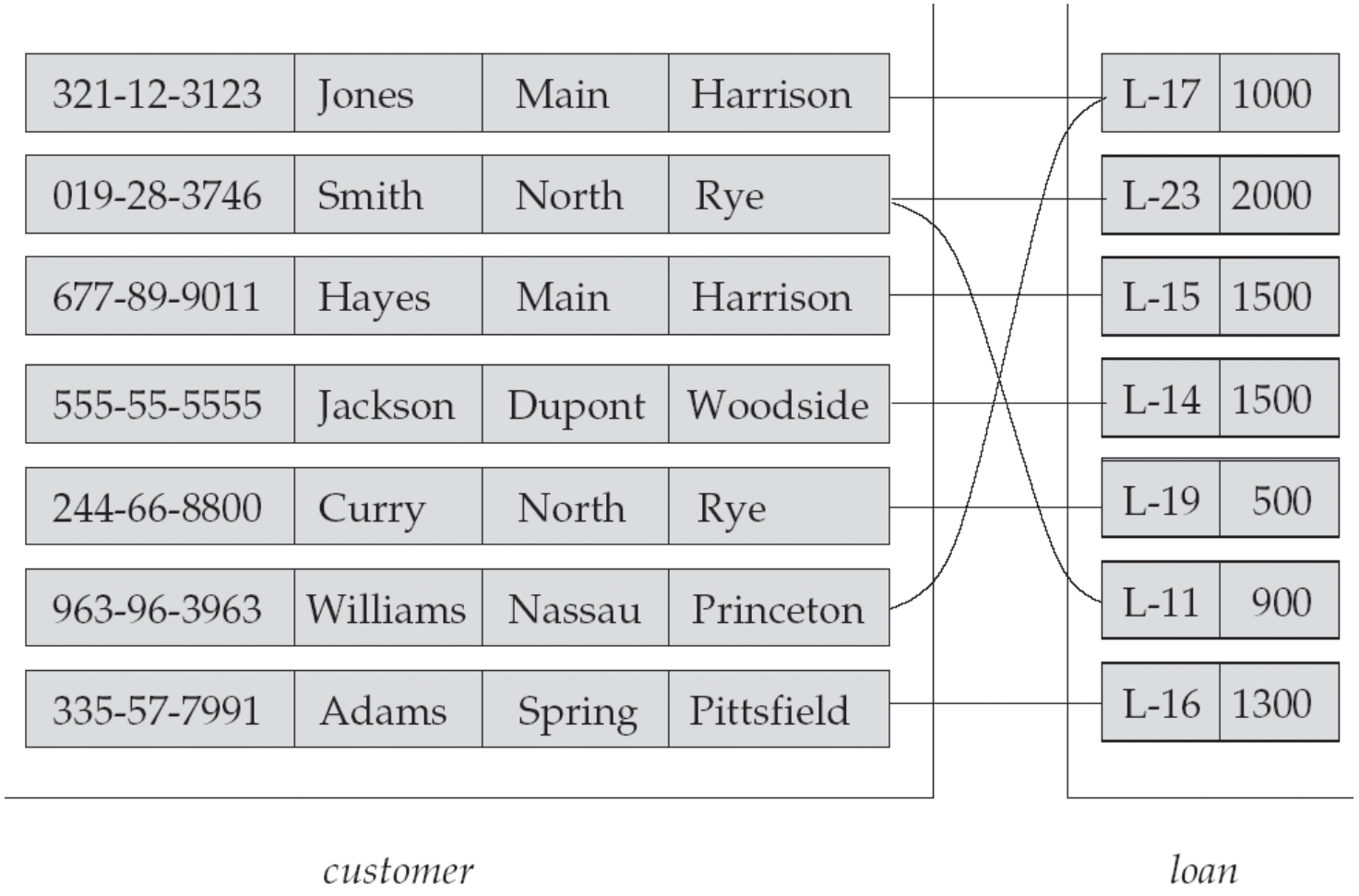
$$R = \{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

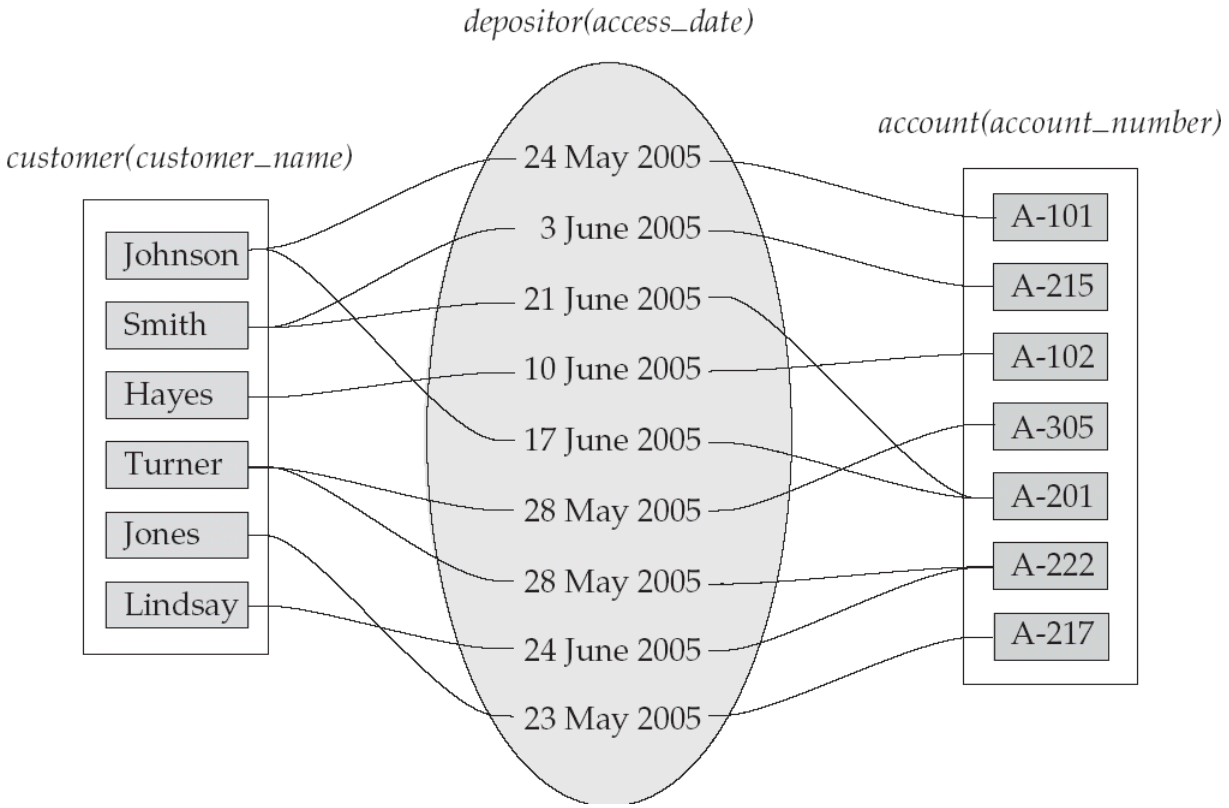
    (Hayes, A-102) $\in$ *borrower*

Relationship Set *borrower*



Relationship Sets (Cont.)

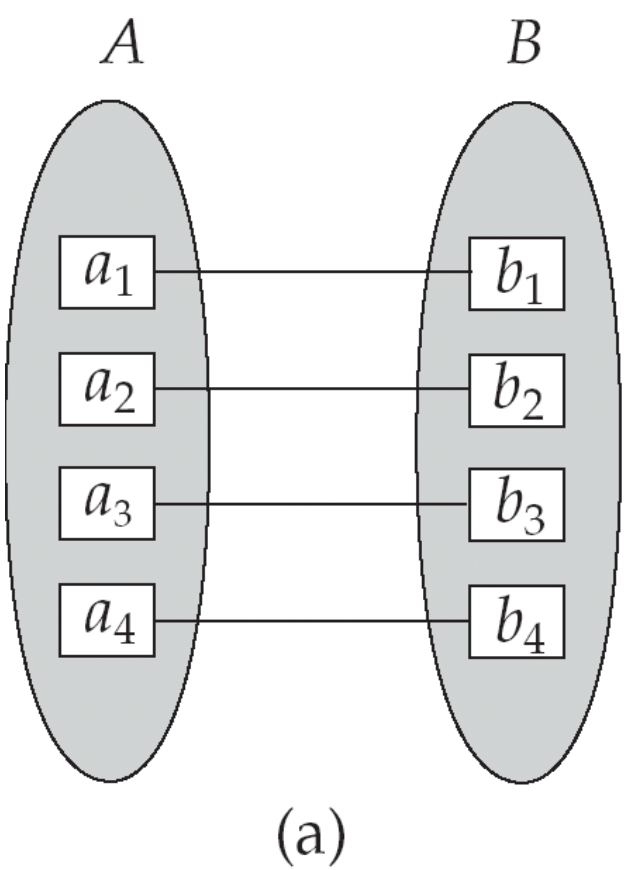
- An **attribute** can also be property of a relationship set.
- For instance, the *depositor* relationship set between entity sets *customer* and *account* may have the attribute *access_date*



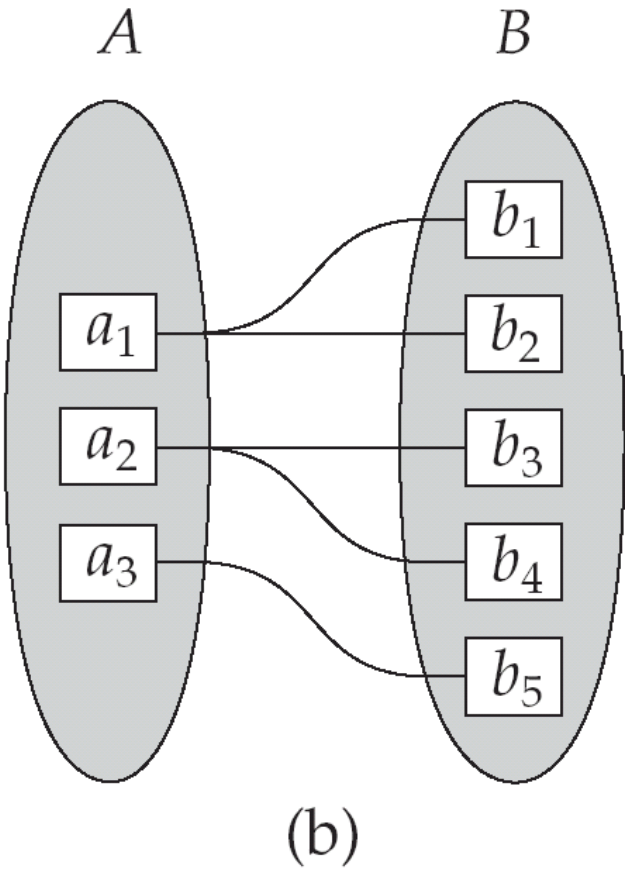
Mapping Cardinality Constraints

- 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



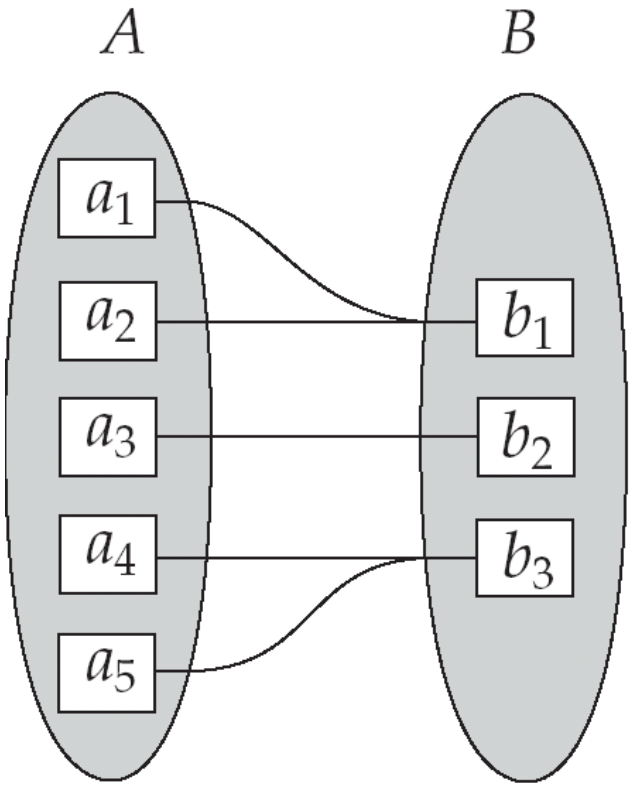
One to one



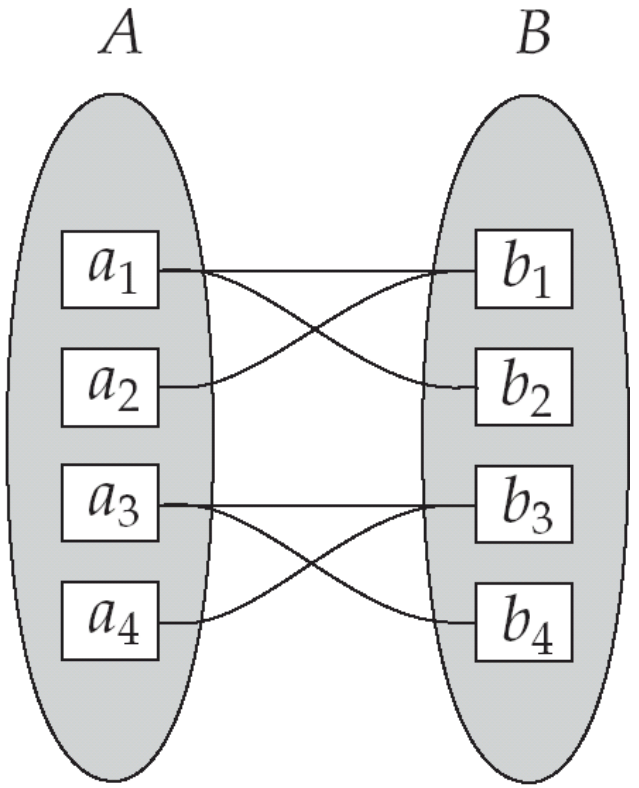
One to many

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

321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye

Attributes

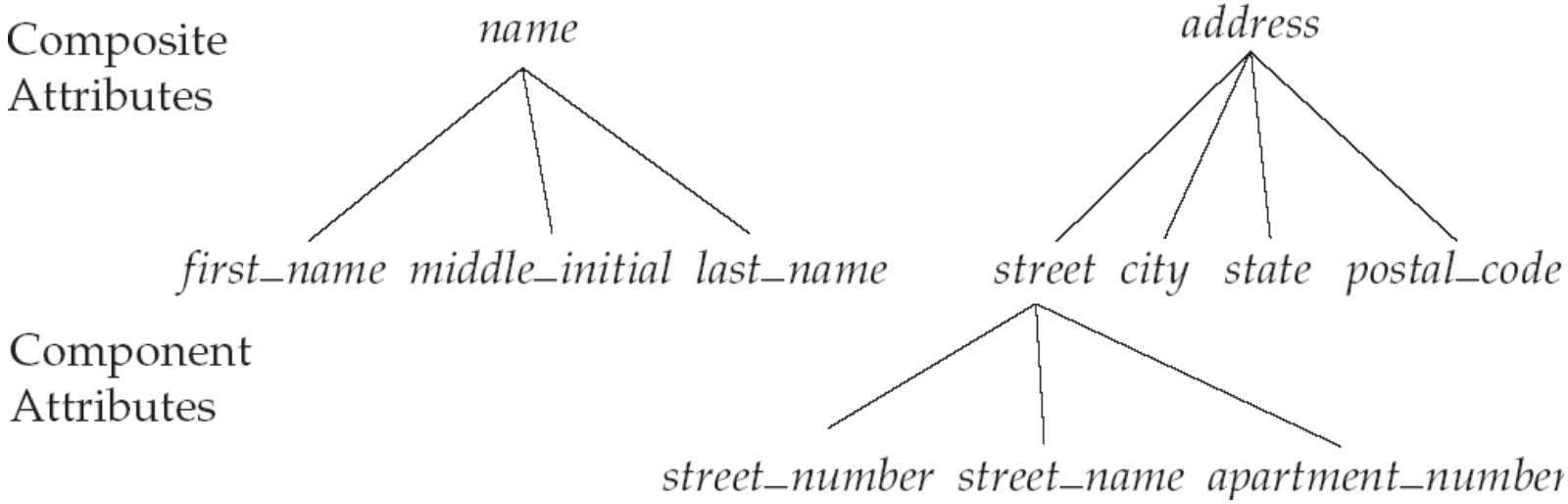
- An entity is represented by a set of attributes
  - = descriptive properties possessed by all members of an entity set.

Example:

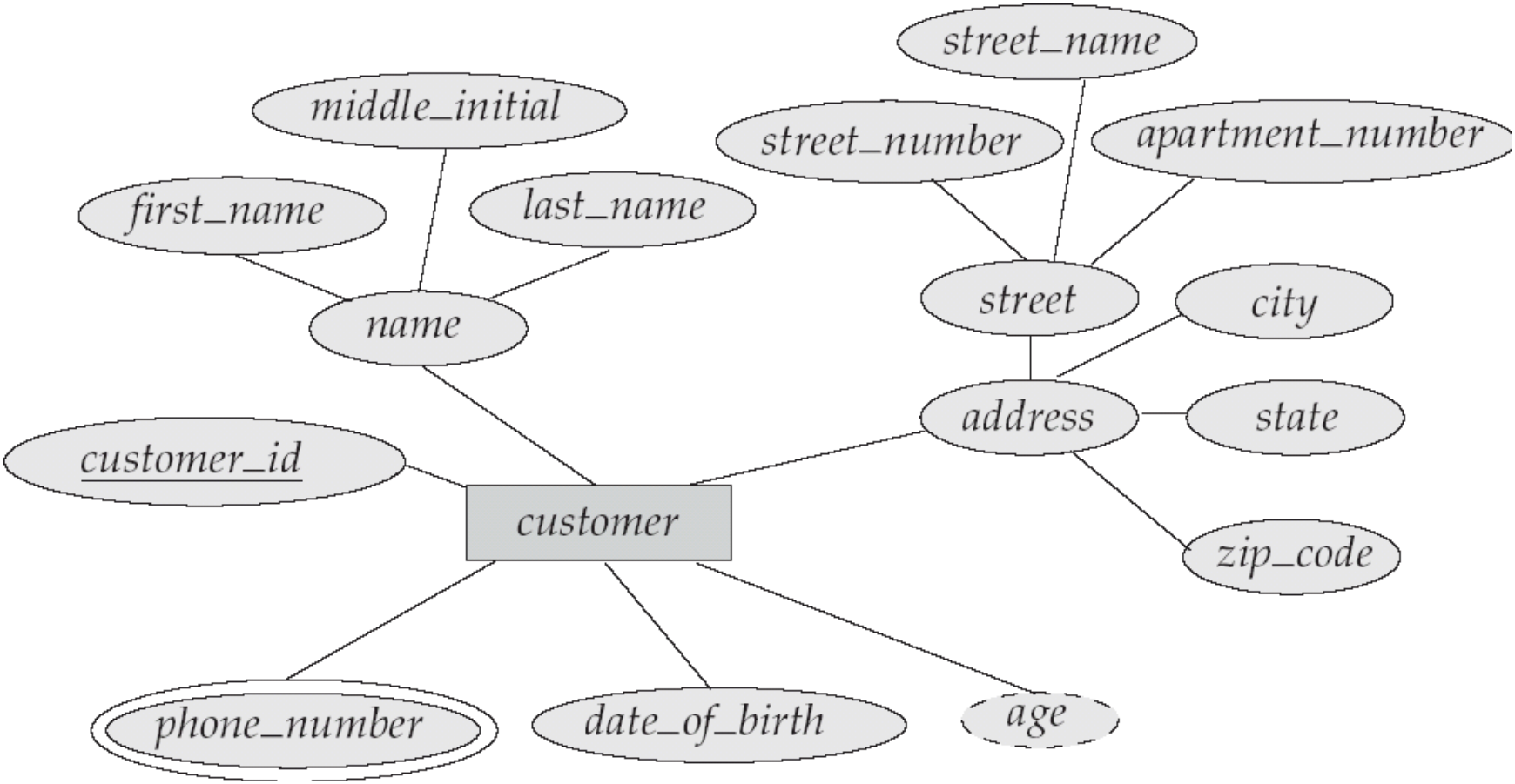
customer = (*customer_id, customer_name, customer_street, customer_city*)
loan = (*loan_number, amount*)

- **Name** – each attribute has its name unique within an entity
- **Domain** – the set of permitted values for each attribute
- **Attribute type**
  - *Simple* attribute – single value
  - *Composite* attribute – single value but structured
  - *Multi-valued* attribute – multiple values, can repeat
    - Example: *phone_numbers*
  - *Derived* attribute
    - Can be computed from other entity's attributes
    - Example: *age*, given *date_of_birth*

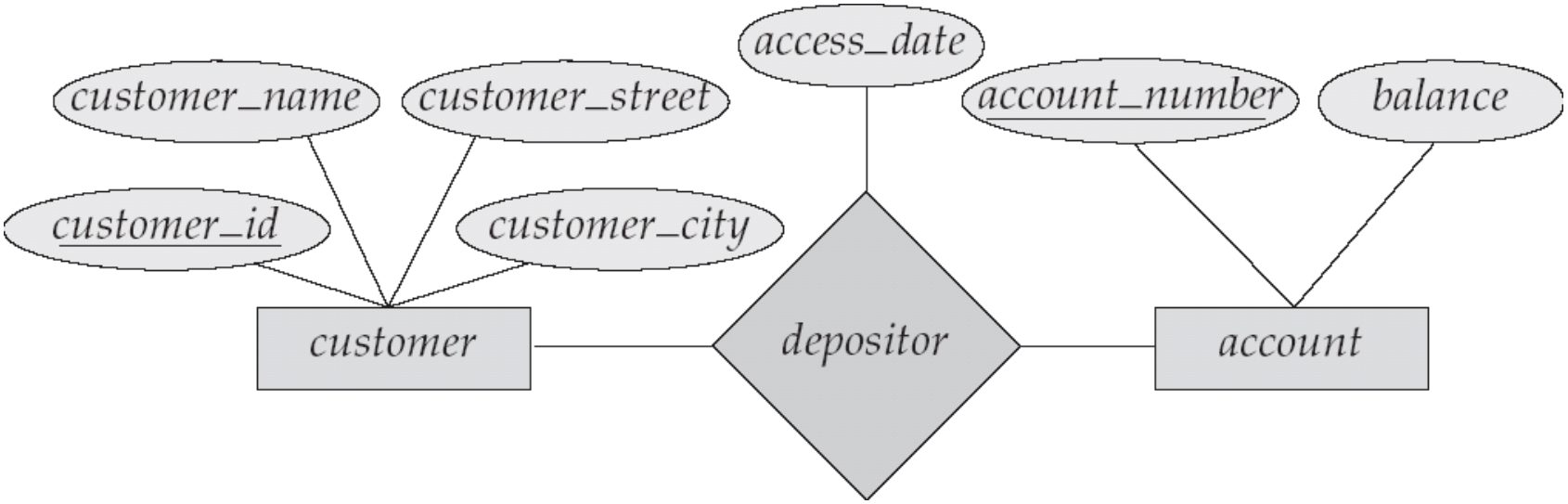
Composite Attributes



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



Relationship Sets with Attributes

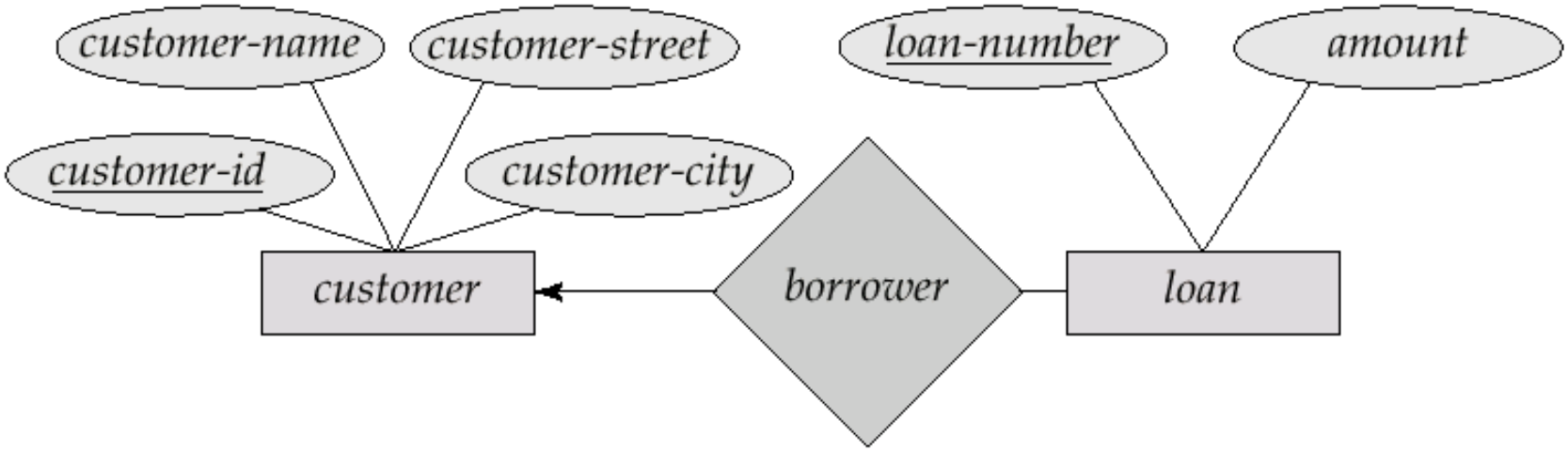


Mapping Cardinality Constraints

- 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*

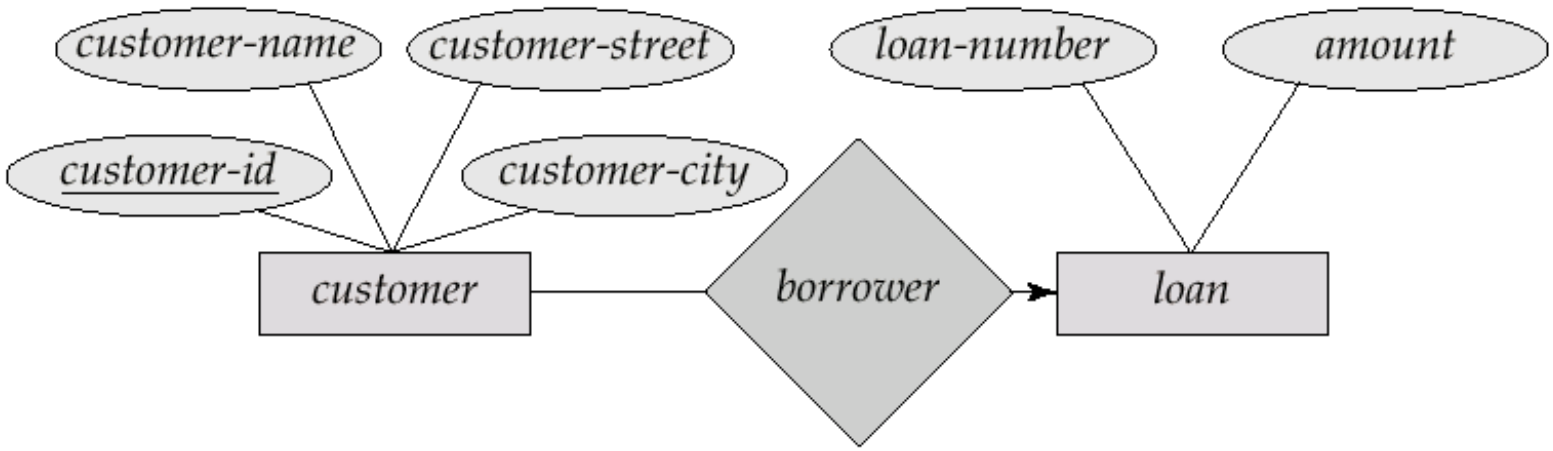
One-To-Many Relationship

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



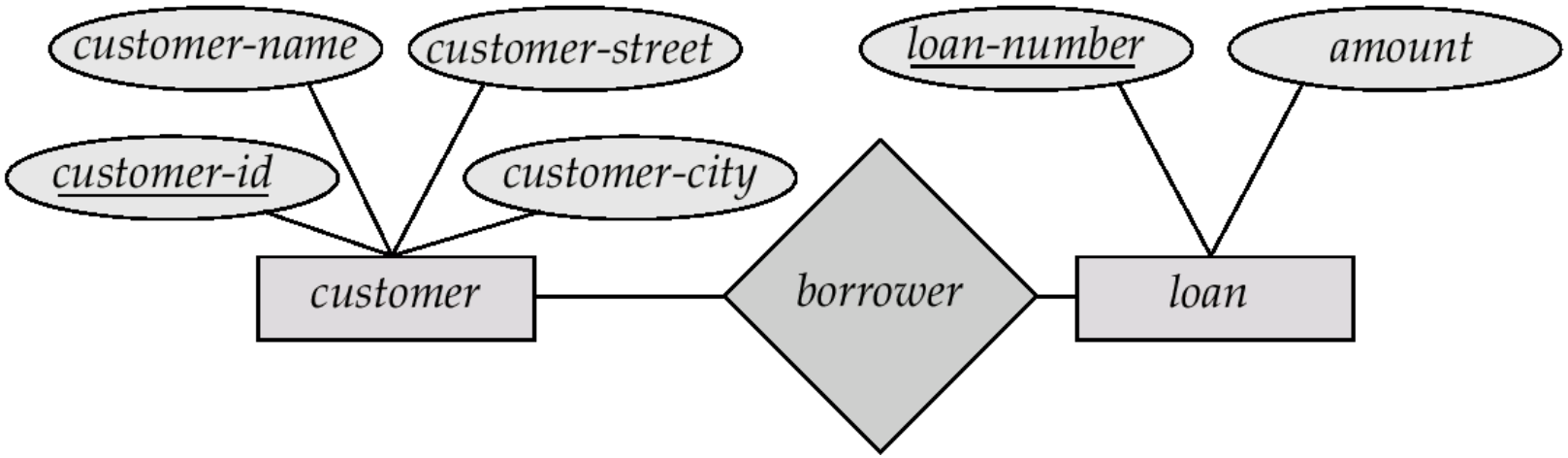
Many-To-One Relationships

- In a many-to-one relationship, a loan is associated with several (including zero) 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 zero) loans via borrower
- A loan is associated with several (possibly zero) customers via borrower



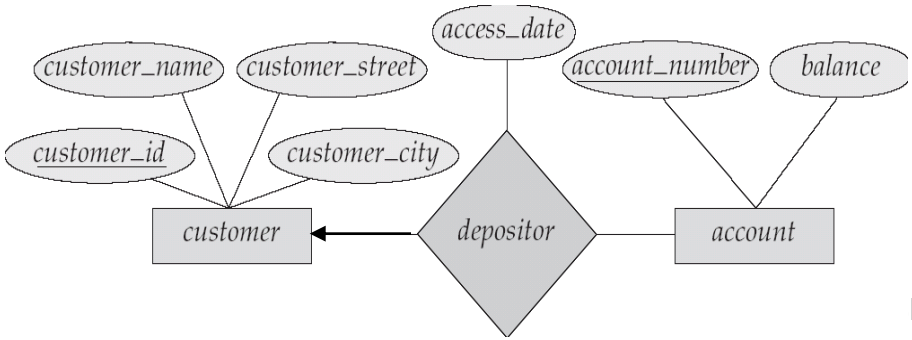
Keys

- Key = a subset of attributes of “special” interest
  - Search key
  - “Database / identification / unique” key
  - Referencing an entity

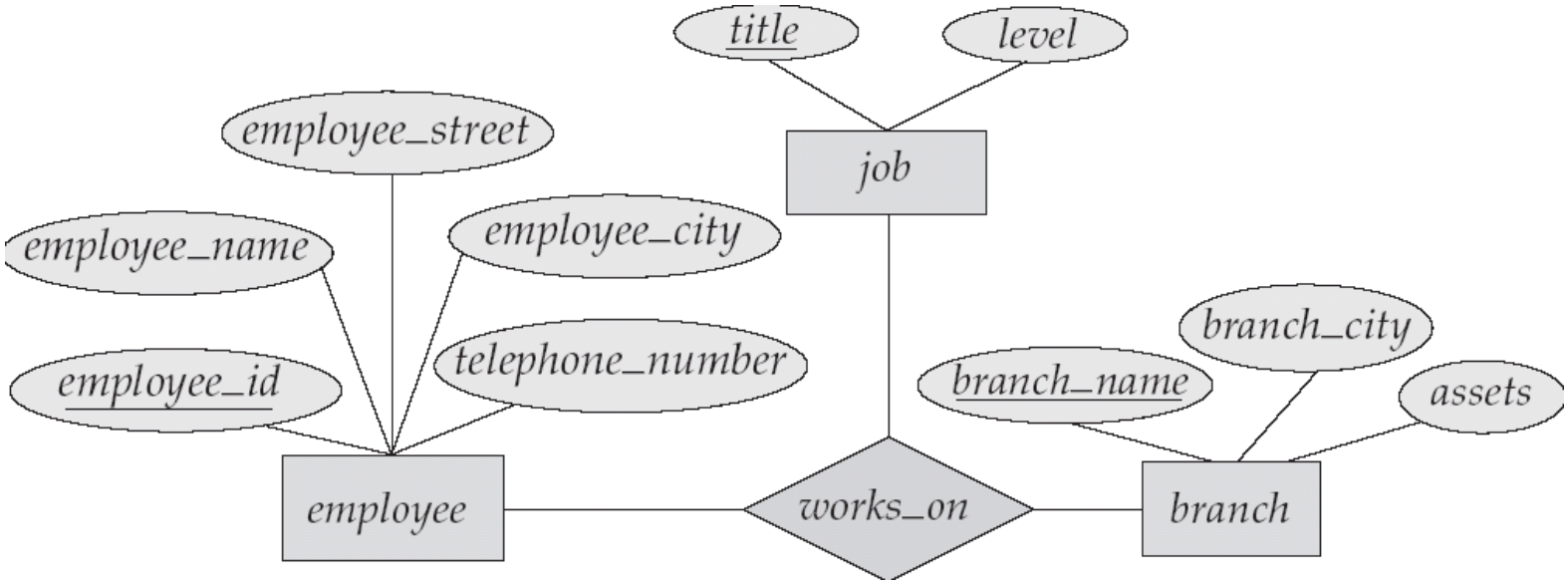
- “Database key” (primary key constraint)
  - Defined for unique identification of each entity and/or relationship
- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A **candidate key** of an entity set is a minimal super key
  - *customer_id* is a candidate key of *customer*
  - *account_number* is a 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 entities 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 may use a multivalued attribute.
- Must consider the mapping cardinality of the relationship set when deciding what the candidate keys are
- Need to consider semantics of relationship set in selecting the *primary key* in case of more than one candidate key



E-R Diagram with a Ternary Relationship

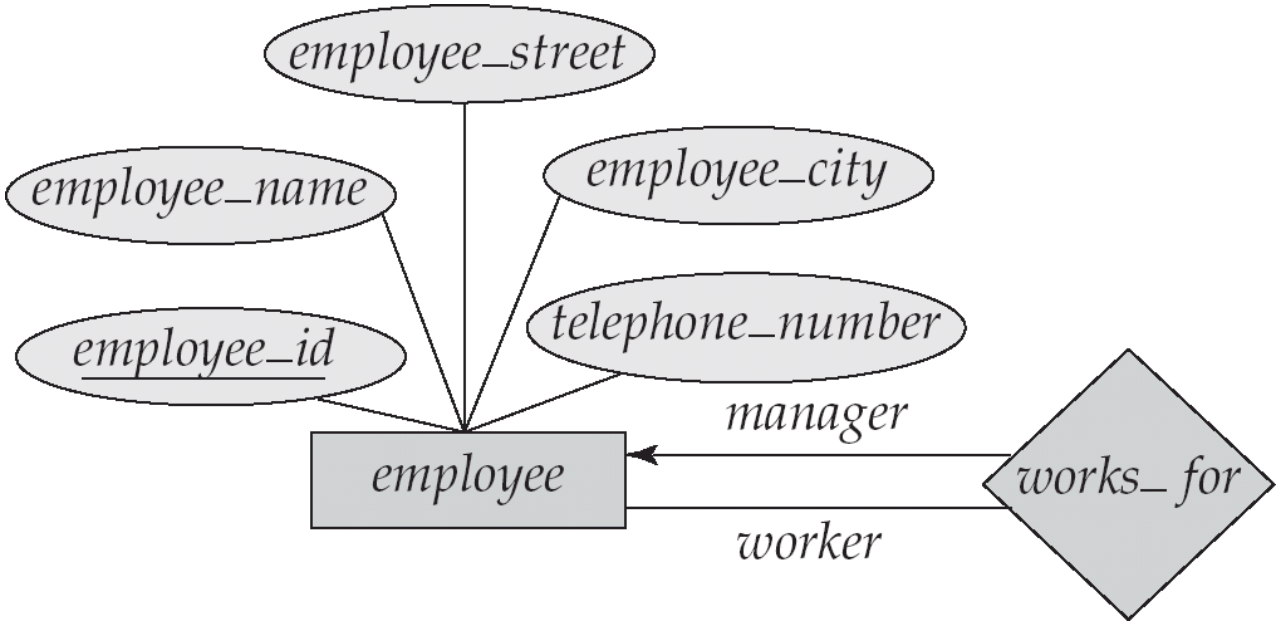


Cardinality Constraints on Ternary Relationship

- 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 an employee works at a branch on at most one job.
- 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

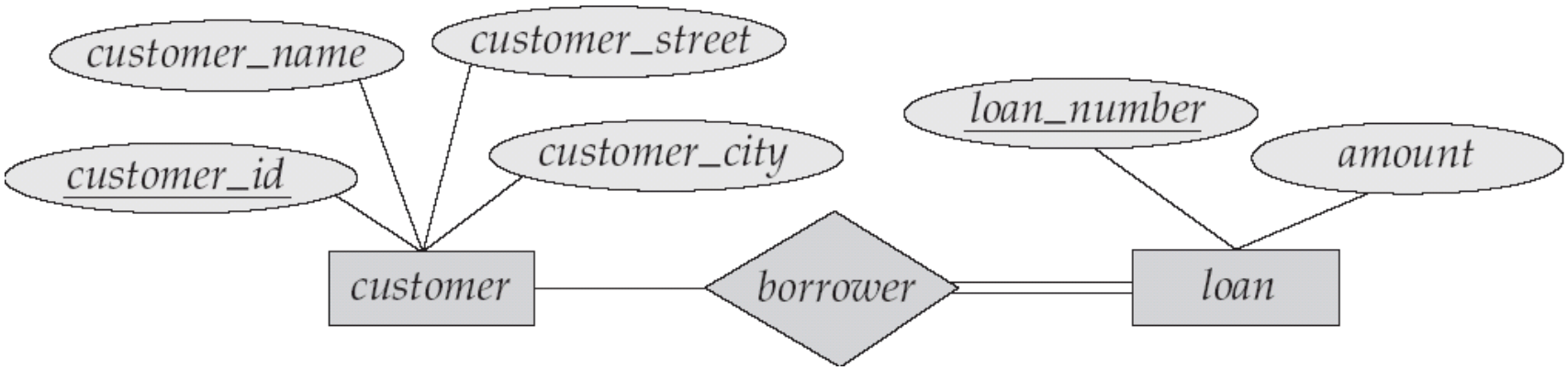
Roles

- 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



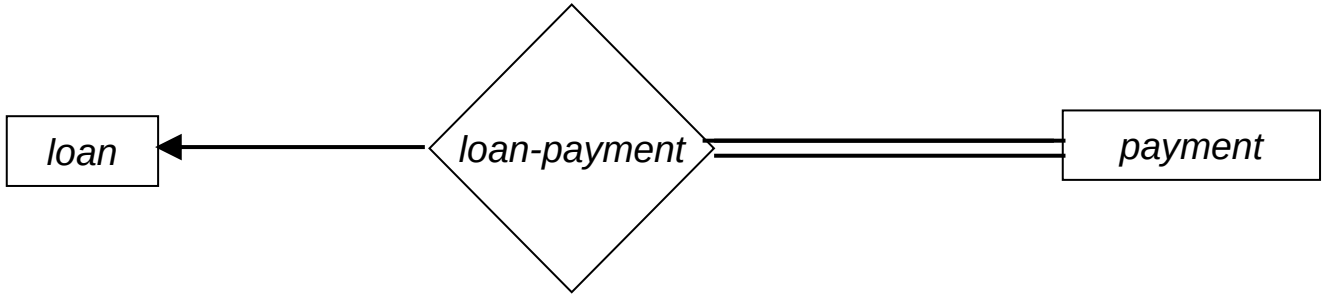
Participation of an Entity Set in a Relationship Set

- 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 (default)
  - some entities may not participate in any relationship in the relationship set
  - Example: participation of customer in borrower is partial



Existence Dependencies

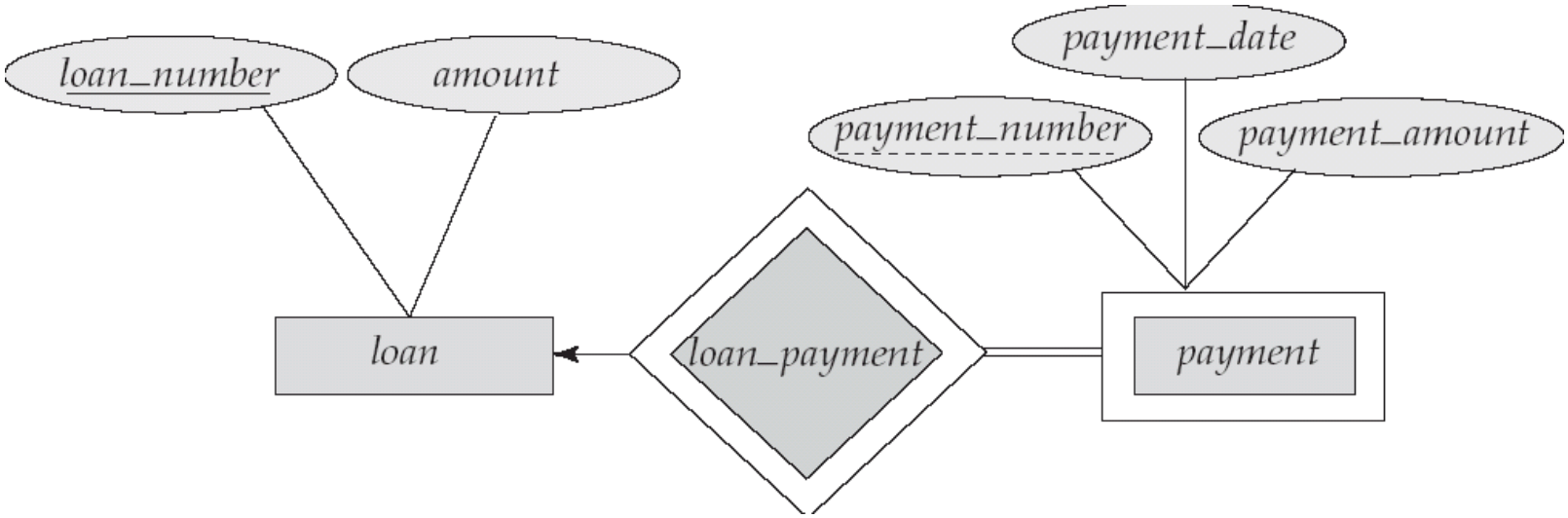
- If the existence of entity *x* depends on the existence of entity *y*, then *x* is said to be *existence dependent* on *y*.
  - *y* is a *dominant entity* (in example below, *loan*)
  - *x* is a *subordinate entity* (in example below, *payment*)



- If a *loan* entity is deleted, then all its associated *payment* entities must also be deleted.

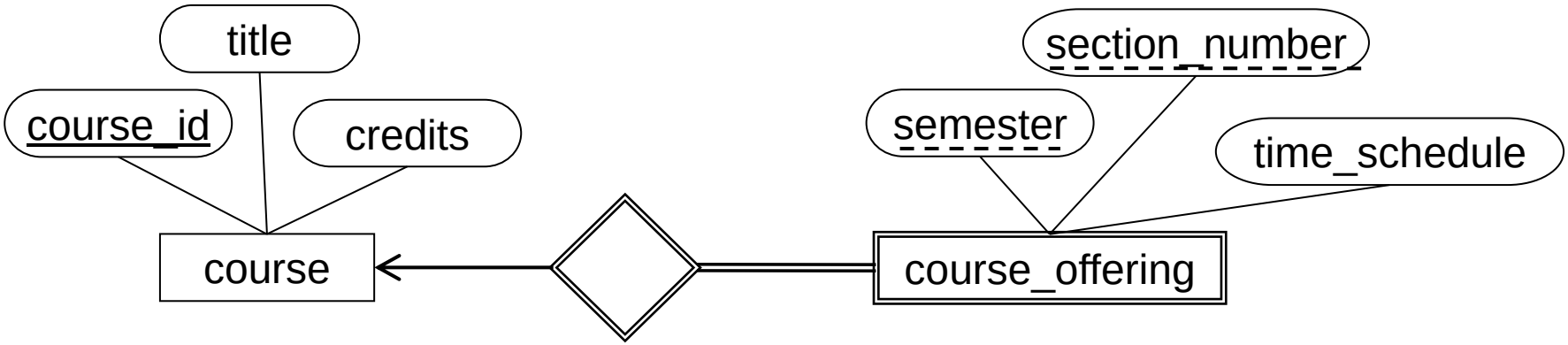
Weak Entity Sets (Cont.)

- 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
  - So, it can represent the order of individual payments of a loan.
- Primary key for *payment* is (*loan_number, payment_number*)



More Weak Entity Set Examples

- In a university, a *course* is a regular entity set and a *course_offering* can be modeled as a weak entity set.
  - The discriminator of *course_offering* is *semester* (including year) and *section_number* (if there is more than one section)



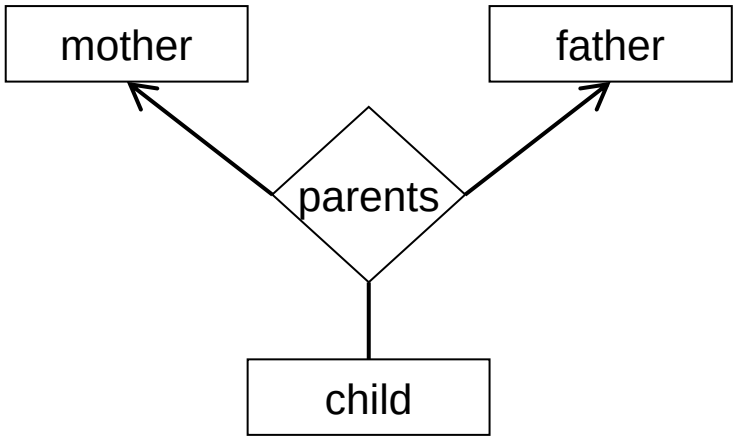
- If we modeled *course_offering* as a regular entity, we would add *course_id* as an attribute.
  - Then the relationship to *course* would be also implicit in the *course_offering.course_id* attribute.

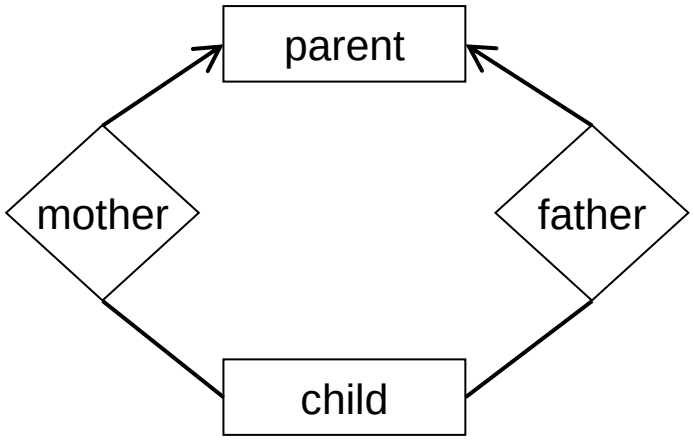
Design Issues

- **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, an *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*

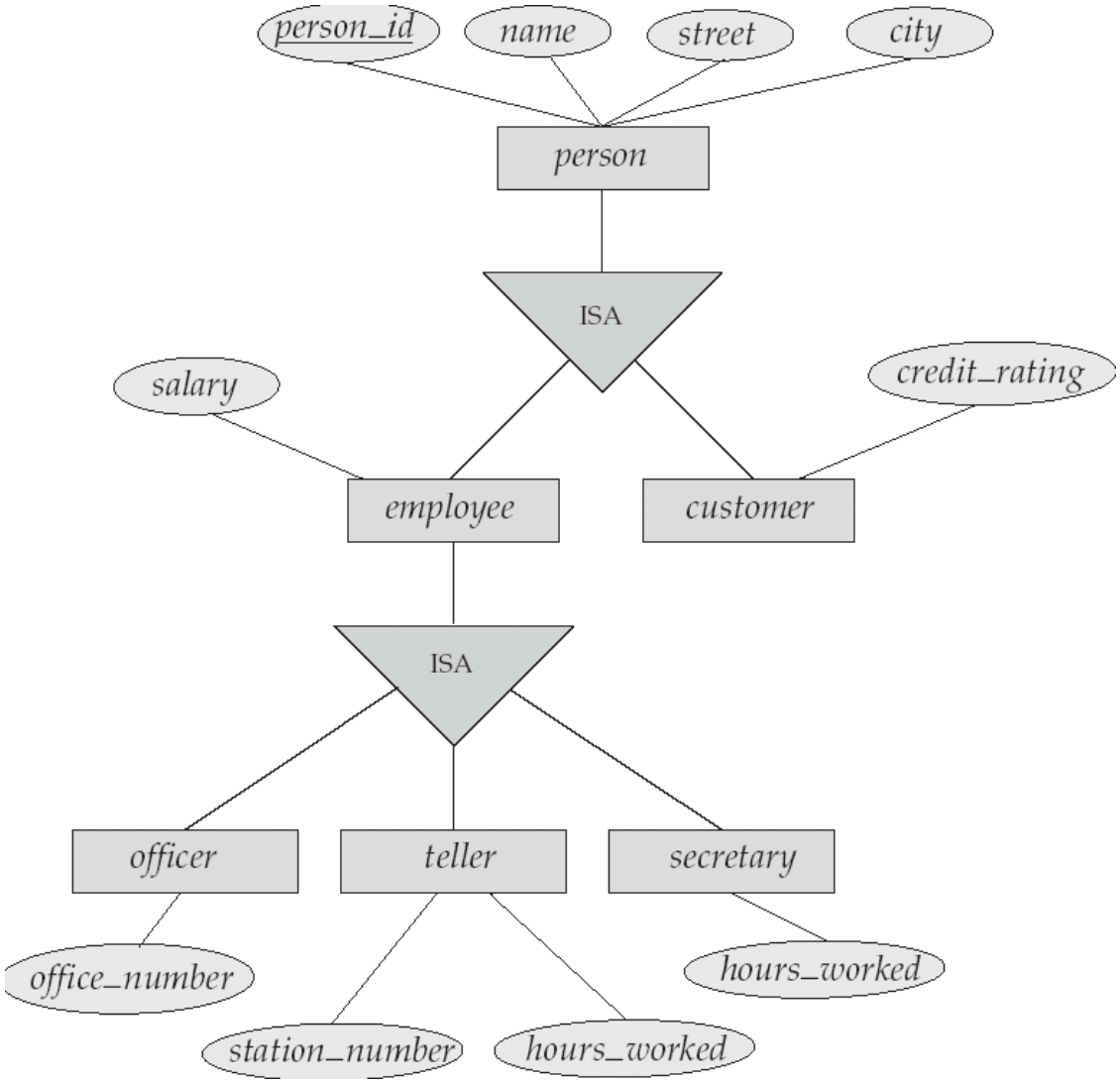




Extended E-R Features: Specialization

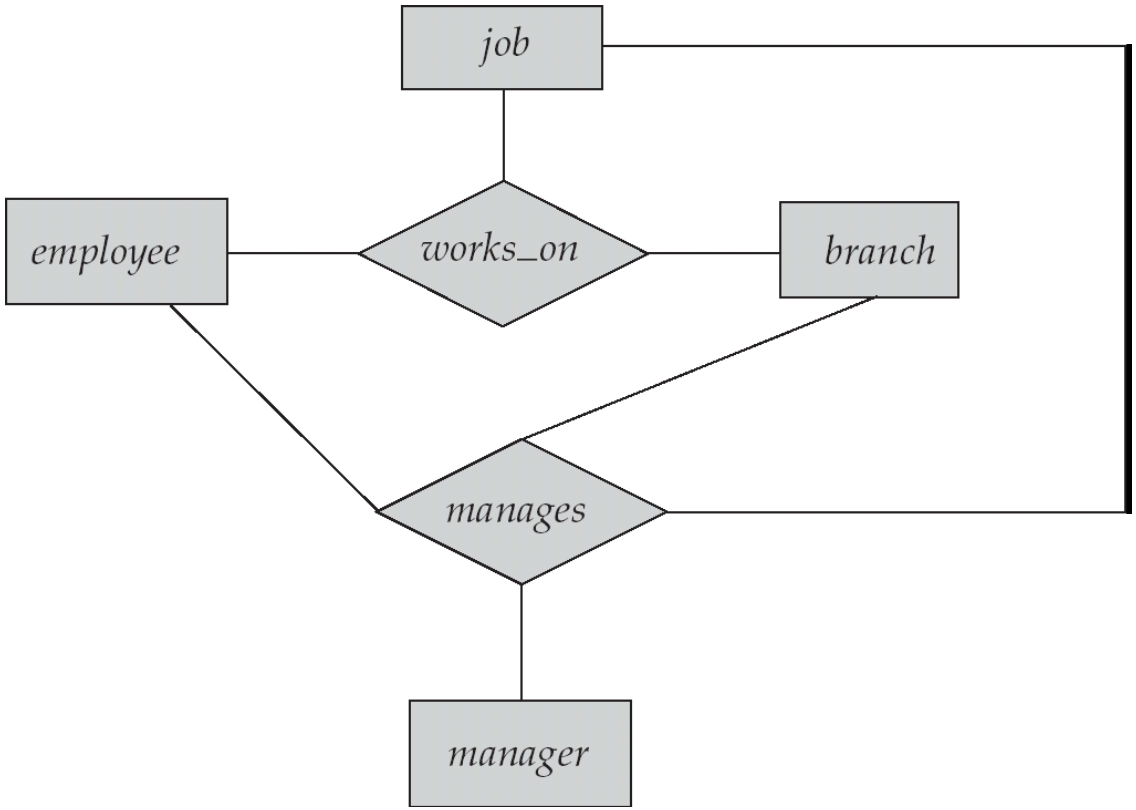
- **A top-down design process**
  - We designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets
  - can have attributes or participate in relationships
  - but do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA
  - E.g., *customer* “is a” *person*.
- **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



Aggregation

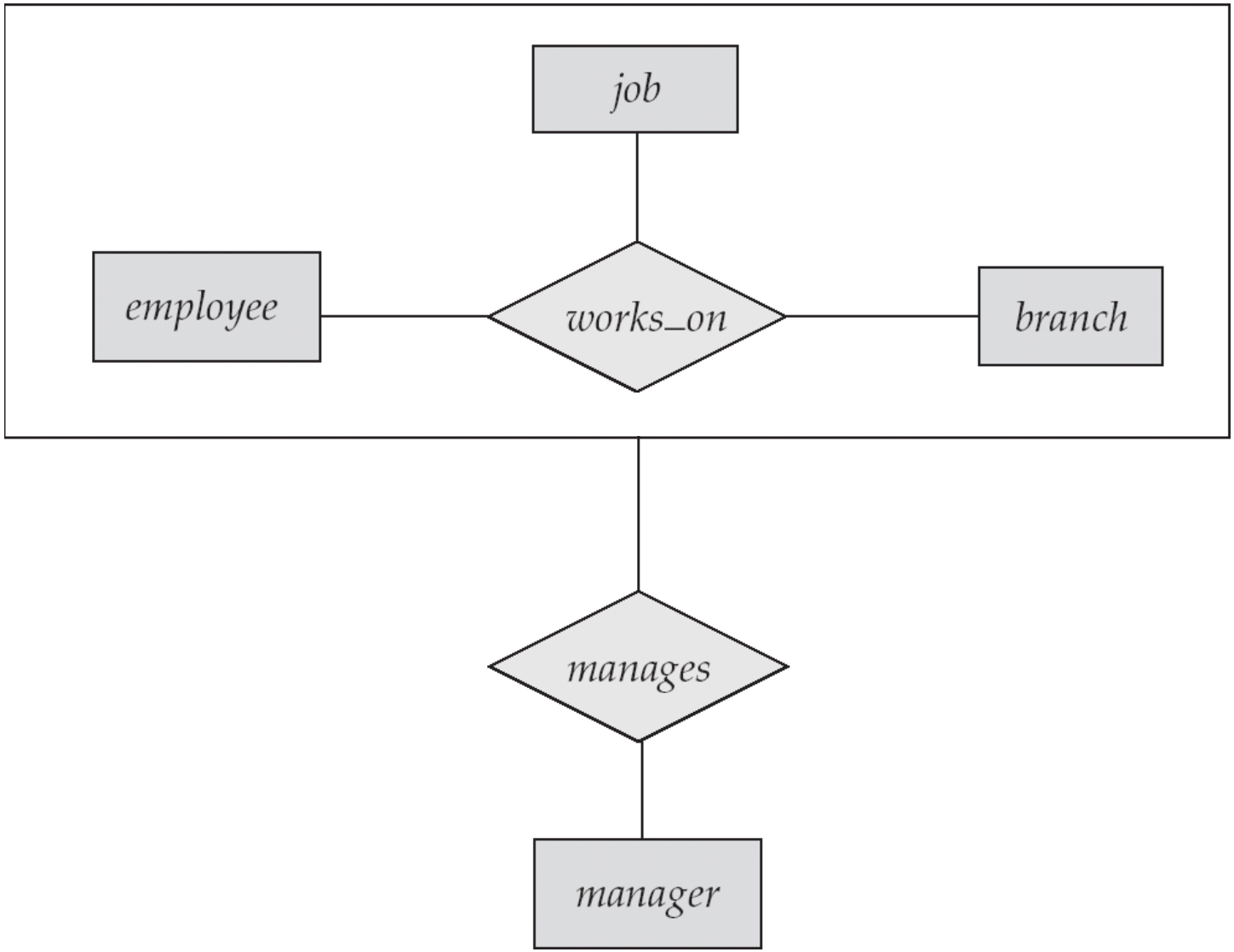
- Consider the ternary relationship *works_on*, which we saw earlier
- Suppose we want to record managers for some tasks performed by an employee at a branch



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 a 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

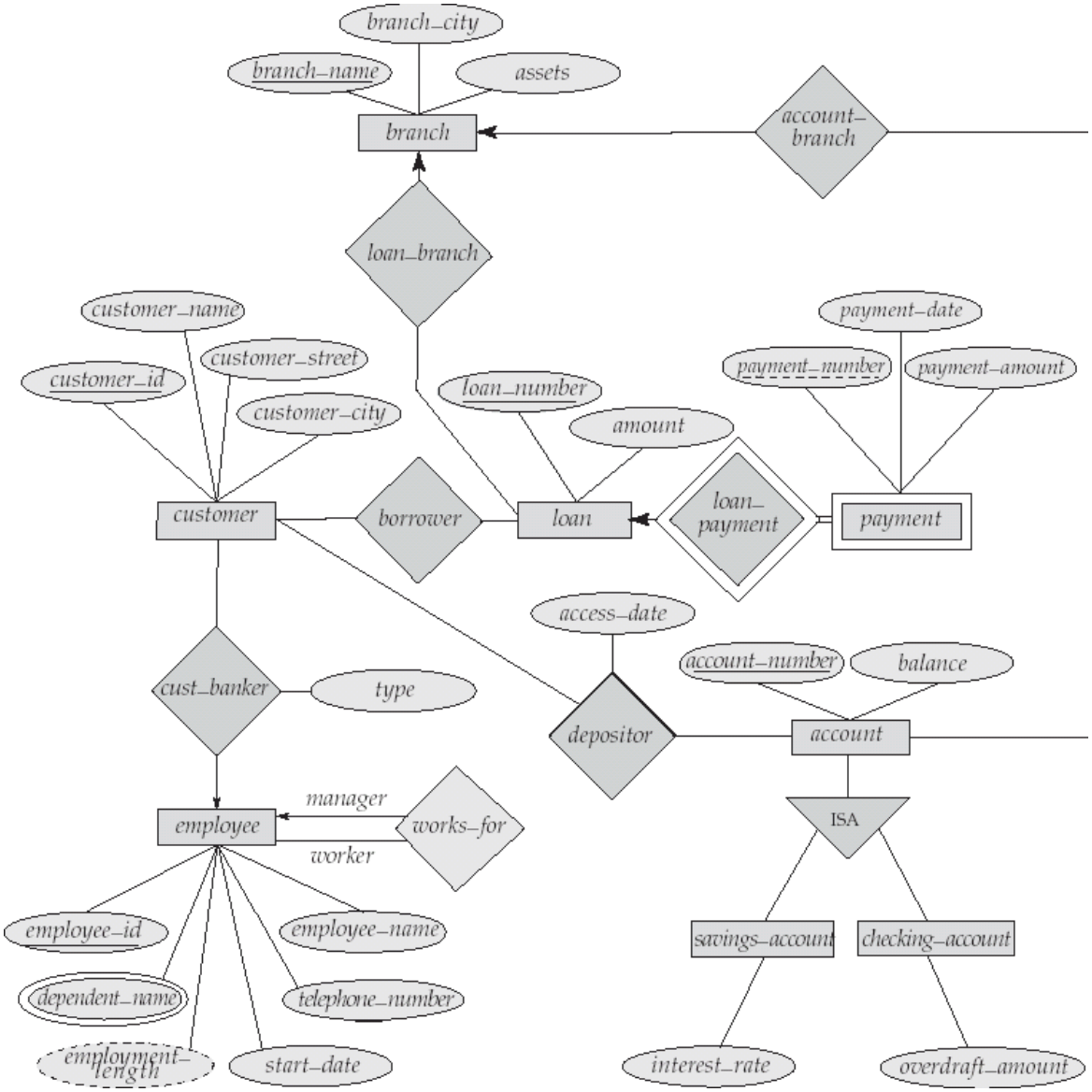
E-R Diagram With Aggregation



E-R Design Decisions

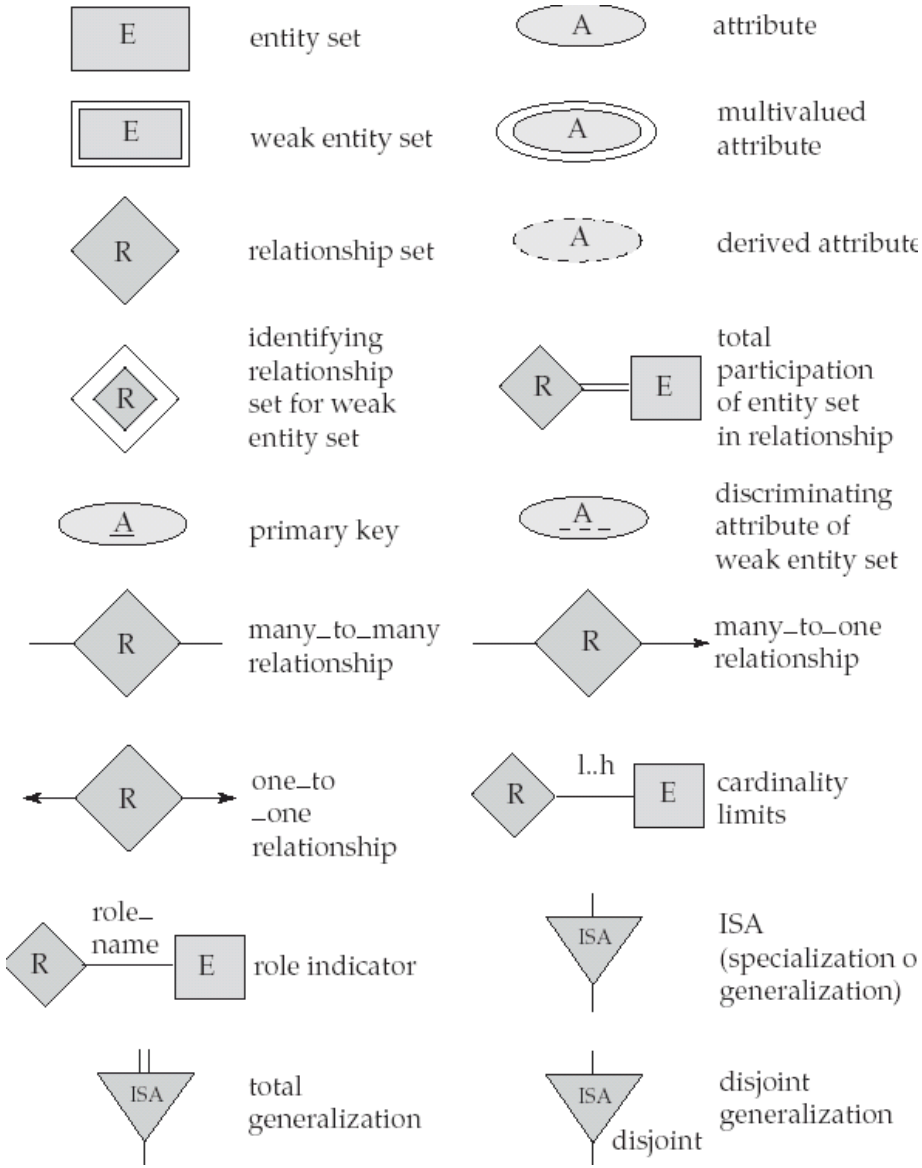
- Already discussed:
  - The use of an attribute or entity set to represent an object.
  - Whether a real-world concept is best expressed by an entity set or a relationship set.
  - The use of a ternary relationship versus a set of binary relationships.
- The use of a regular (strong) or weak entity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

E-R Diagram for a Banking Enterprise



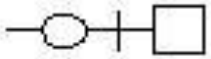
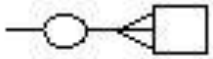
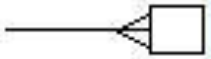
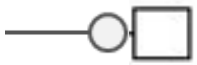
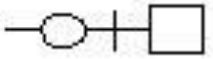
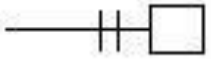
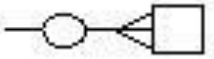
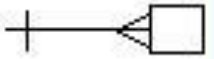
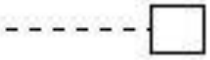
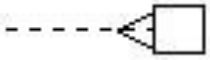
Summary of Symbols Used in E-R Notation

Chen E-R Notation



Alternative E-R Notations

OTHER RELATIONSHIP CARDINALITY NOTATION

Notation	Zero or One Relation-ship	One and Only One	Zero or Many Relation-ship	One or Many Relation-ship
Crow's Foot Notation				
Arrow Notation				
Bachman Notation				
ADW				
Oracle Case				

https://www.softwareideas.net/erd-relation-arrows

UML

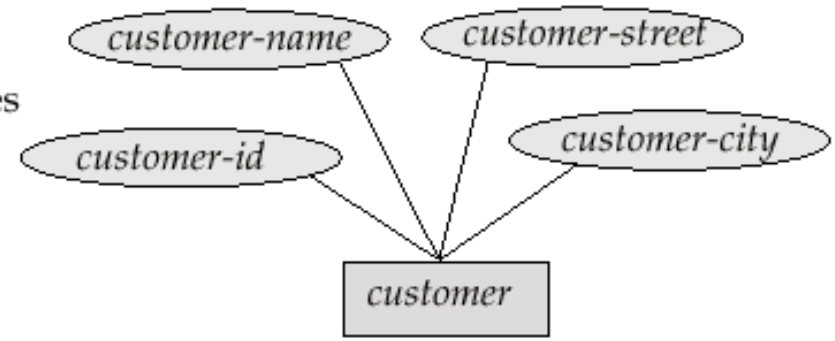
- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- Supported techniques
  - data modeling (entity relationship diagrams)
  - business modeling (workflows)
  - object modeling
  - component modeling
- UML Class Diagrams correspond to E-R Diagram
  - but there are several differences.

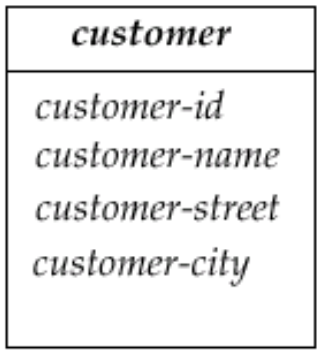
Summary of UML Class Diagram Notation

Chen notation

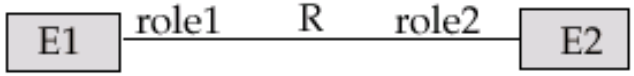
UML notation

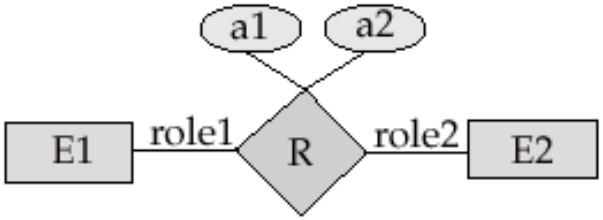
1. Entity sets and attributes

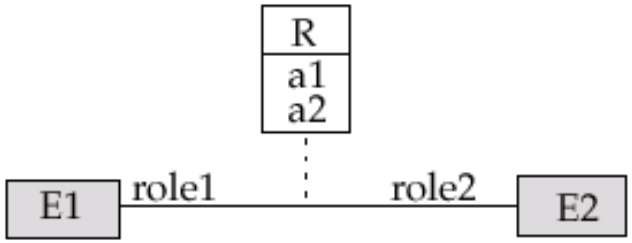




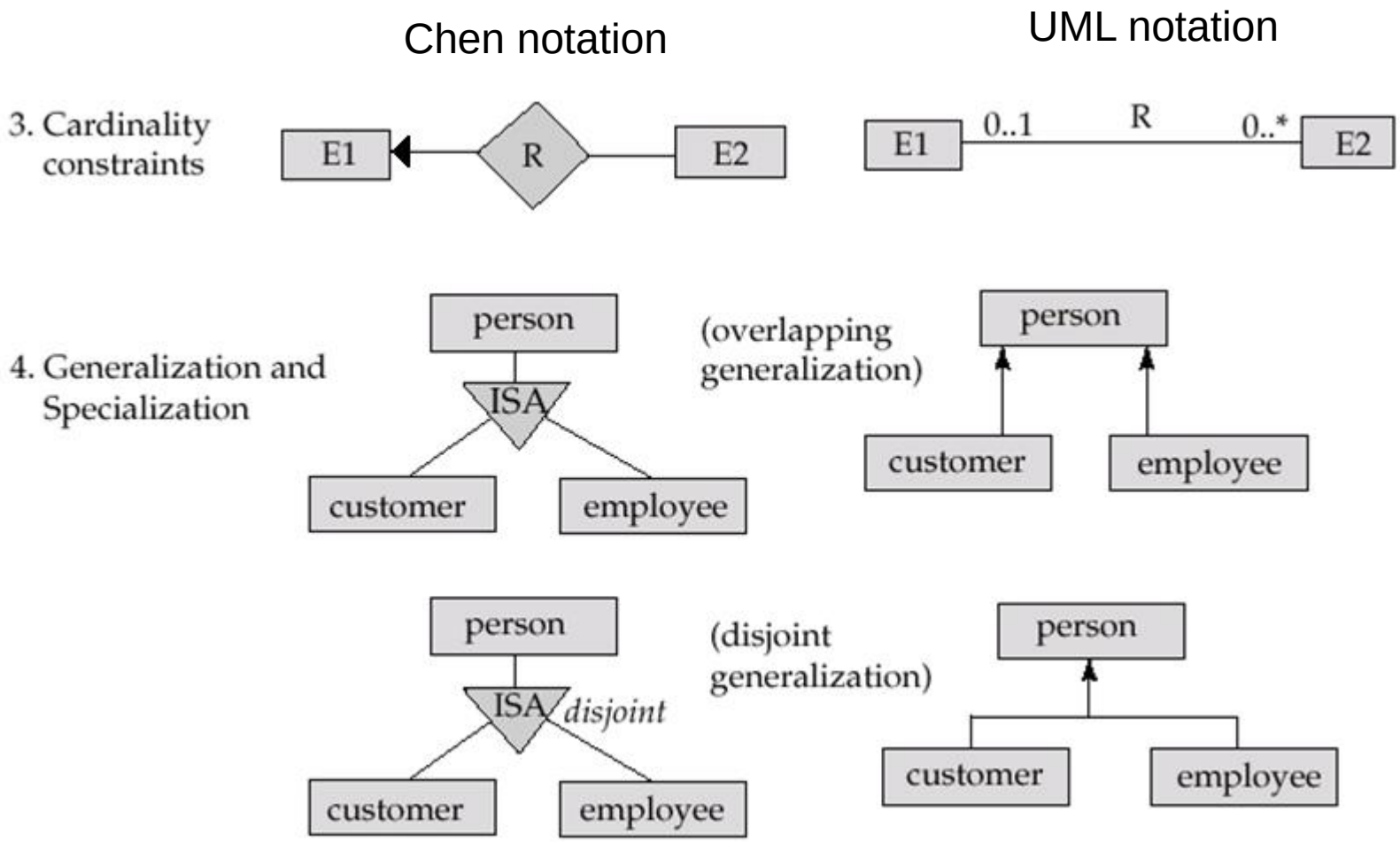
2. Relationships







UML Class Diagram Notation (Cont.)



* Note the reversal notation of numeric relationship cardinality constraints in UML
* Generalization can use merged or separate arrows independent of disjoint/overlapping