Entity Relationship model

Dr. M. Brindha
Assistant Professor
Department of CSE
NIT, Trichy-15

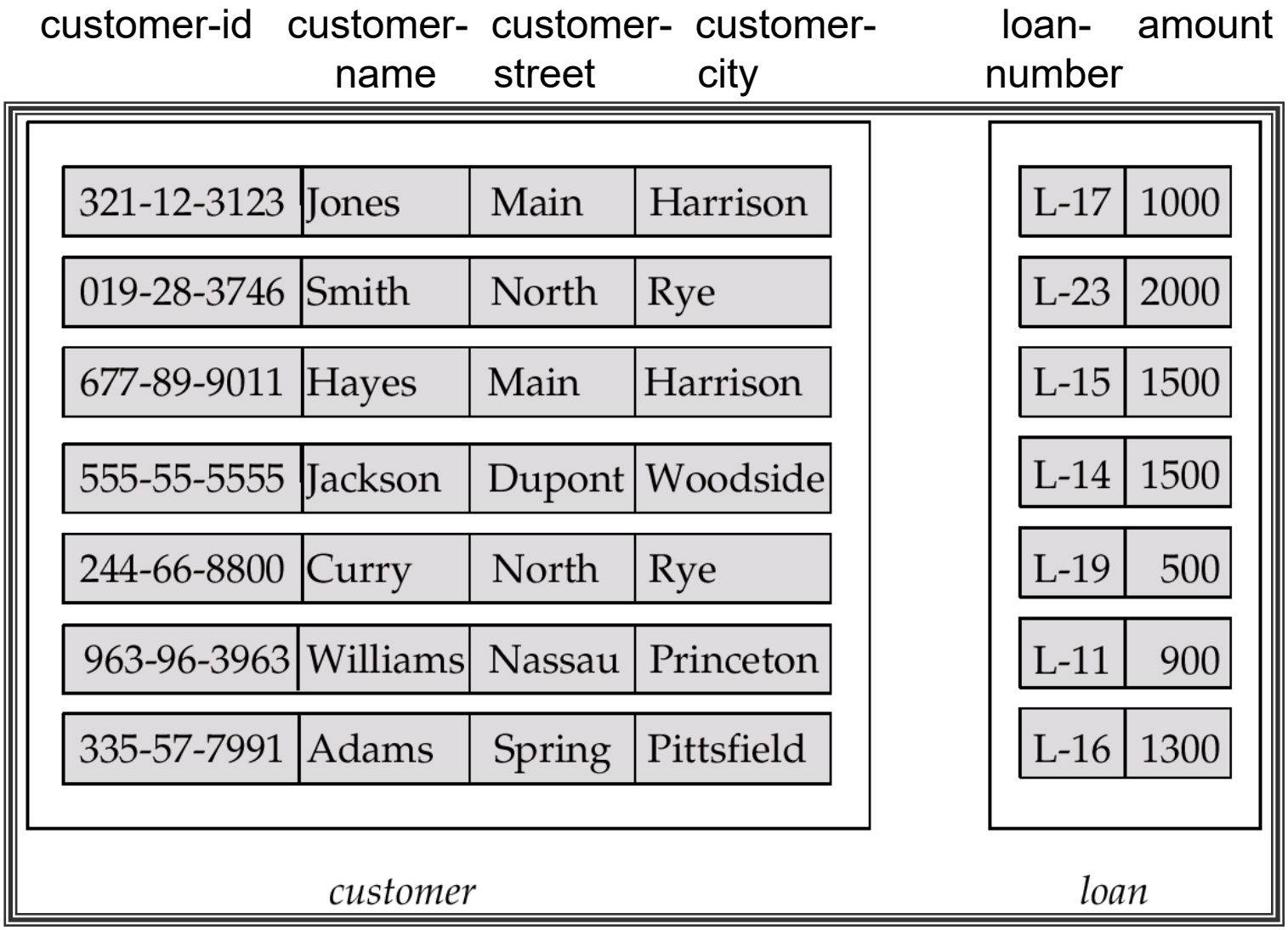
Entity-Relationship Model

- Entity Sets
- Relationship Sets
- Design Issues
- Mapping Constraints
- Keys
- E-R Diagram
- Extended E-R Features
- Design of an E-R Database Schema
- Reduction of an E-R Schema to Tables

Entity Sets

- 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

Attributes

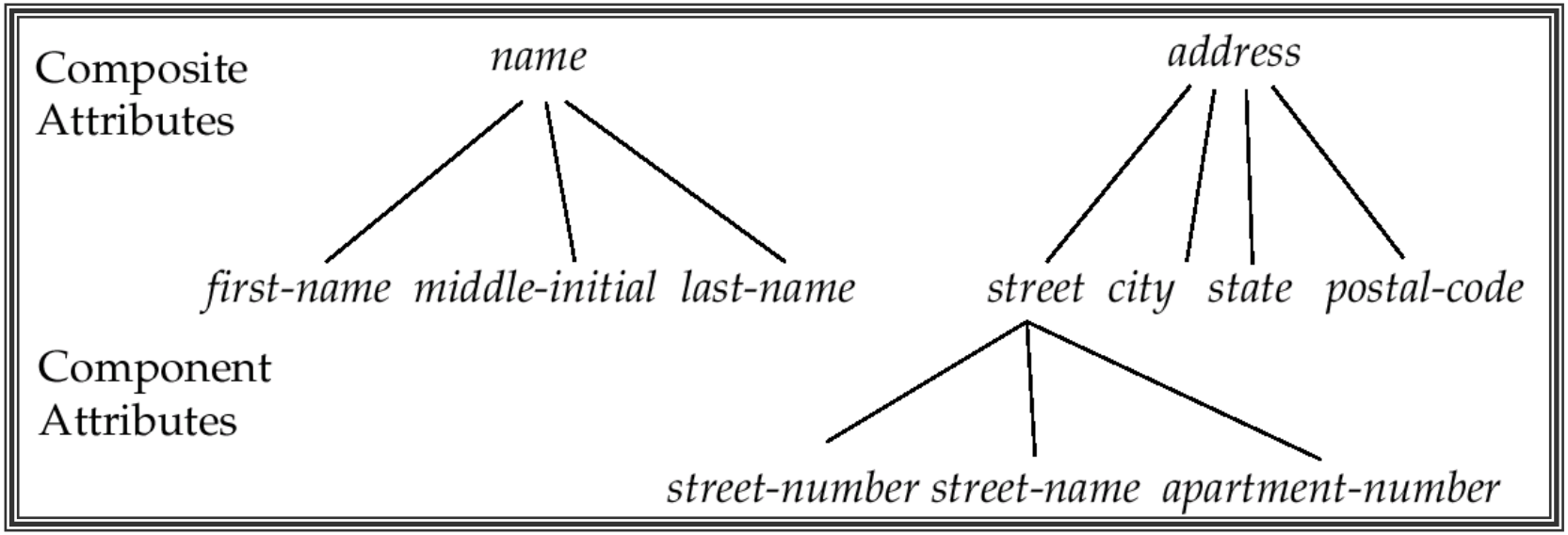
- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

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
    - E.g. multivalued attribute: *phone-numbers*
  - *Derived* attributes
    - Can be computed from other attributes
    - E.g. *age*, given date of birth

Composite Attributes



Relationship Sets

- A relationship is an association among several entities

Example:

Hayes *depositor* A-102

customer entity relationship set *account* entity

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

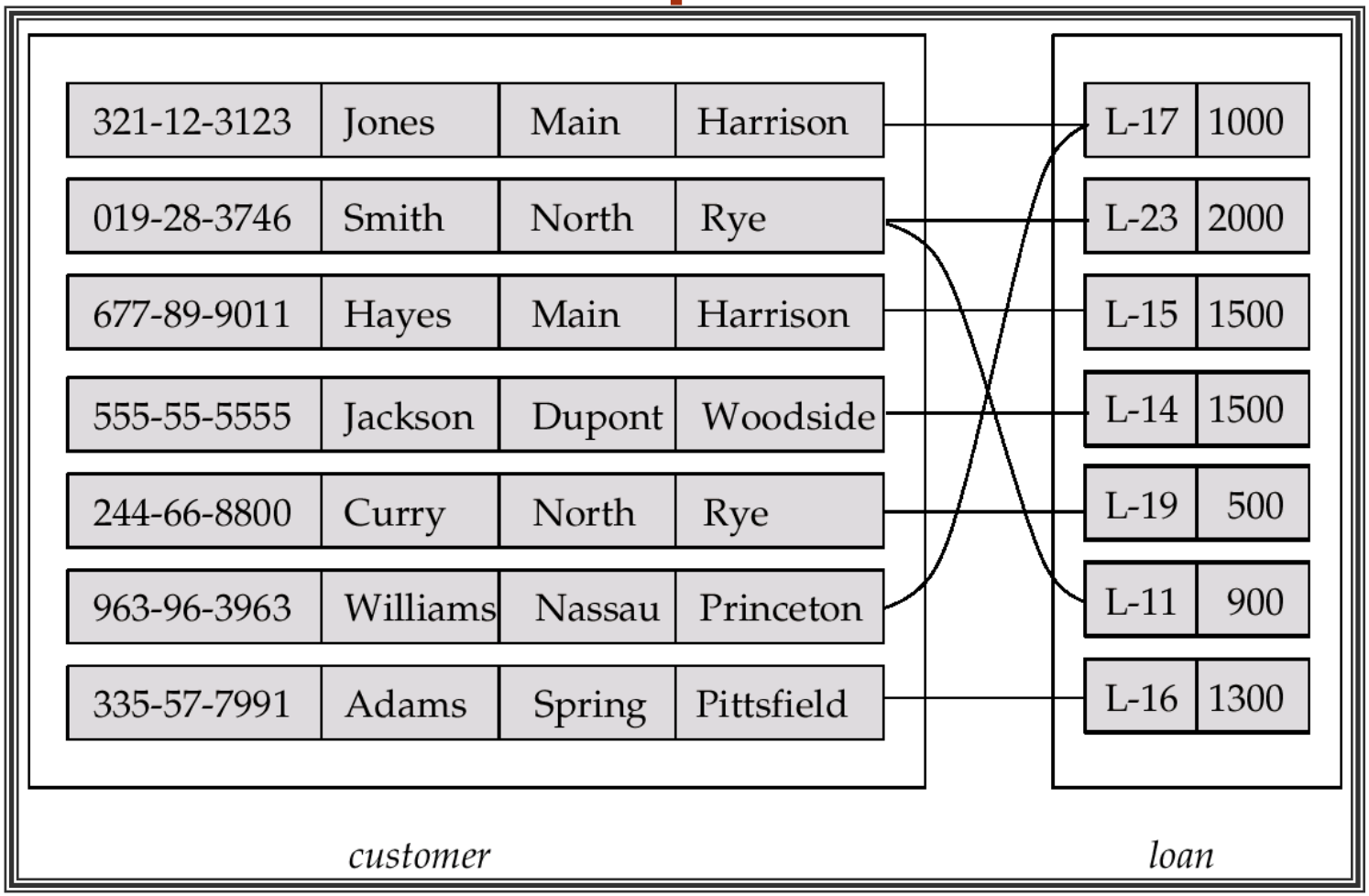
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

- Example:

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

Relationship Set *borrower*



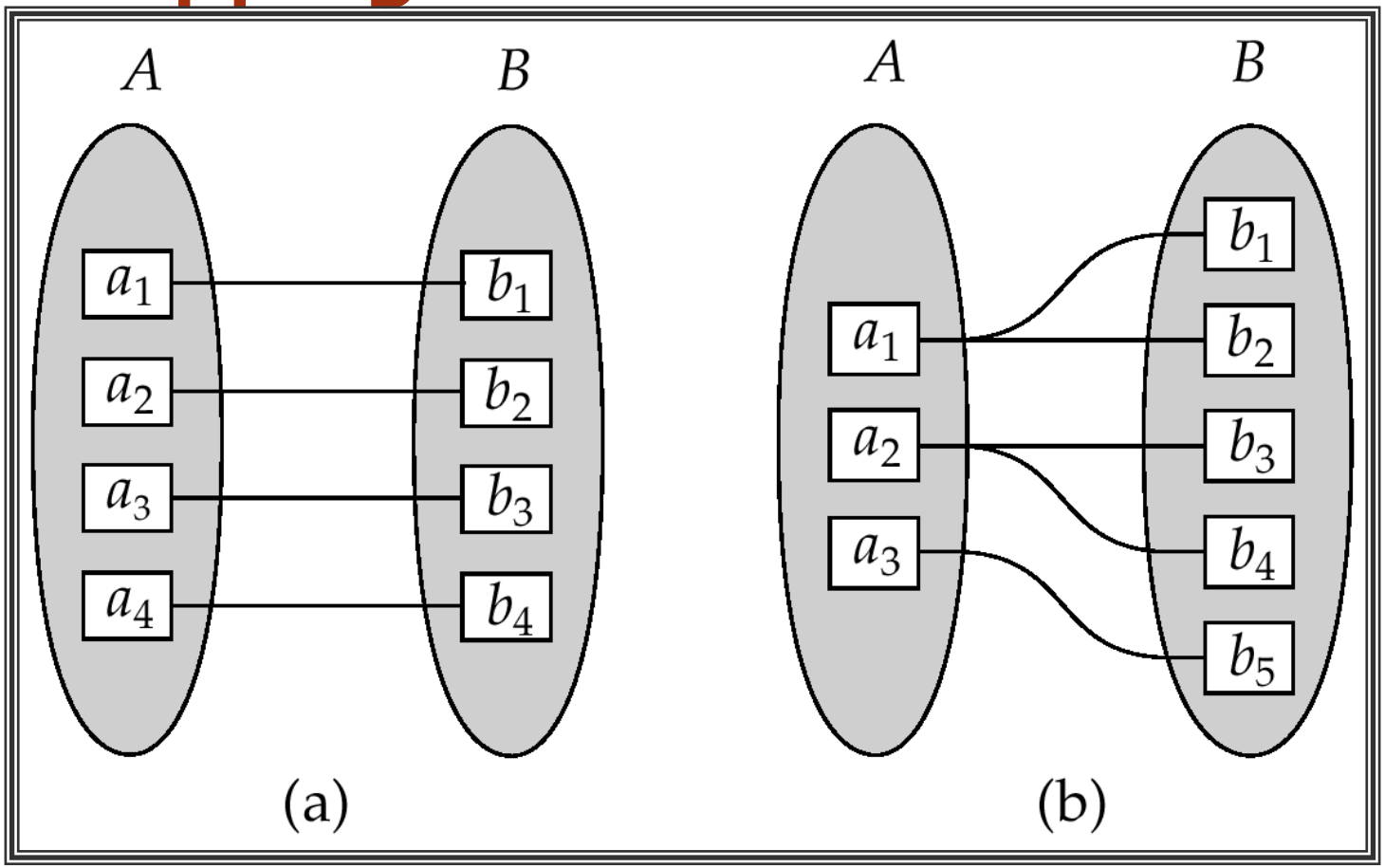
Degree of a Relationship Set

- 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.
  - E.g. 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.

Mapping Cardinalities

- 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

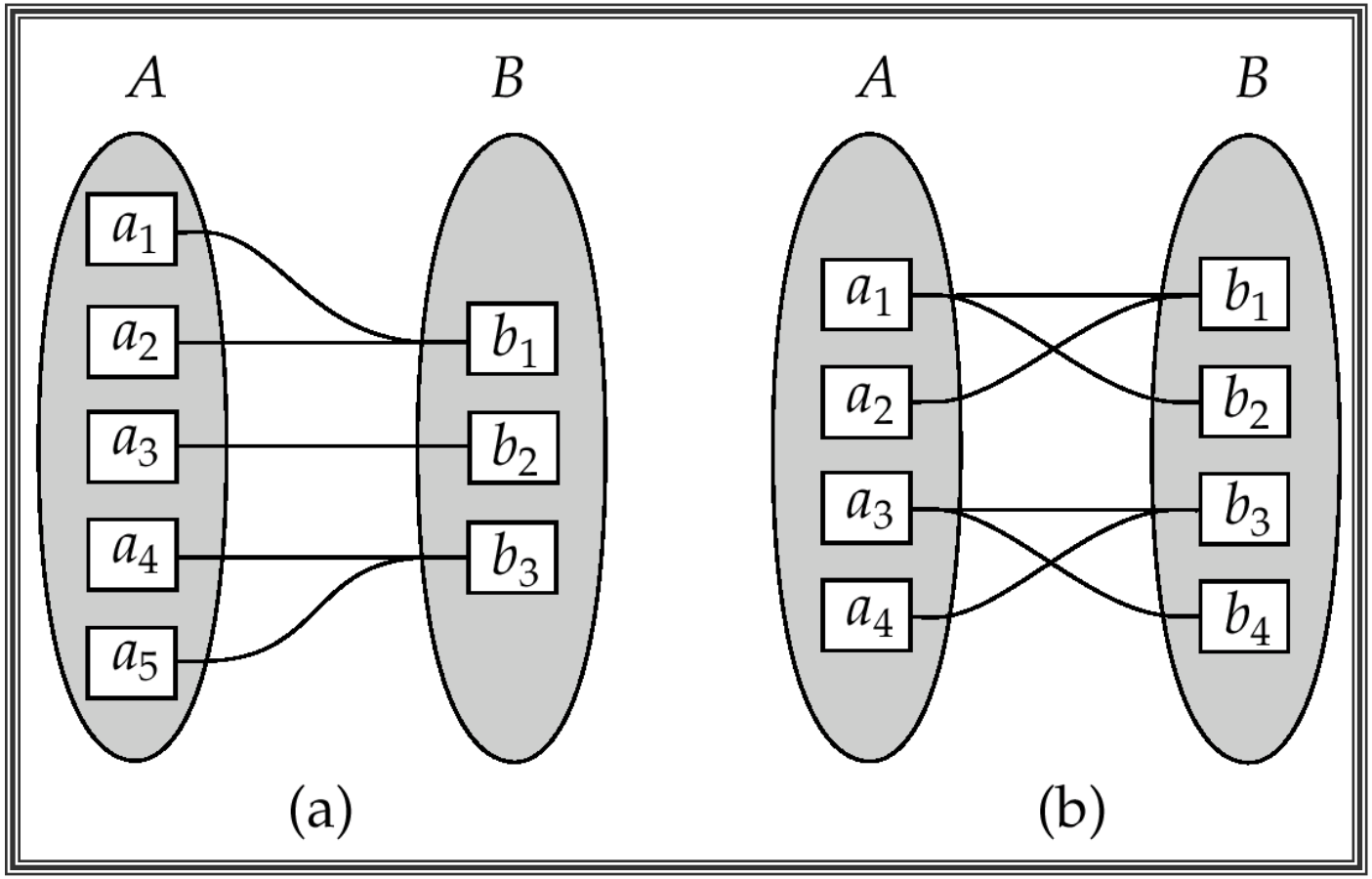


One to one

One to many

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

Mapping Cardinalities

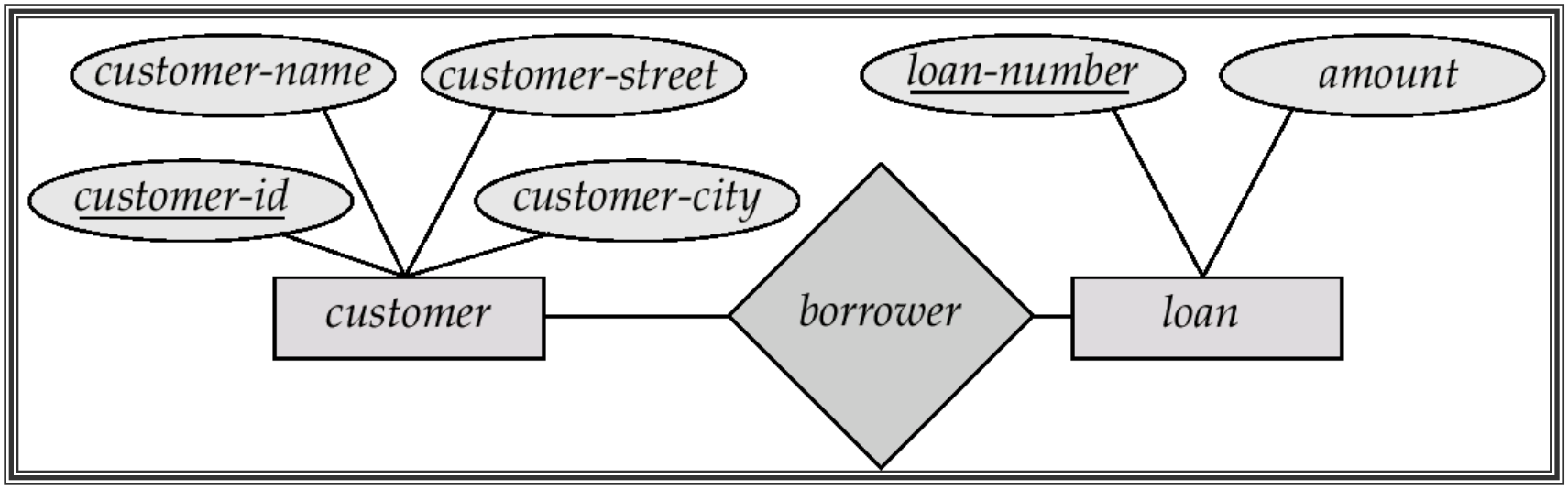


Many to one

Many to many

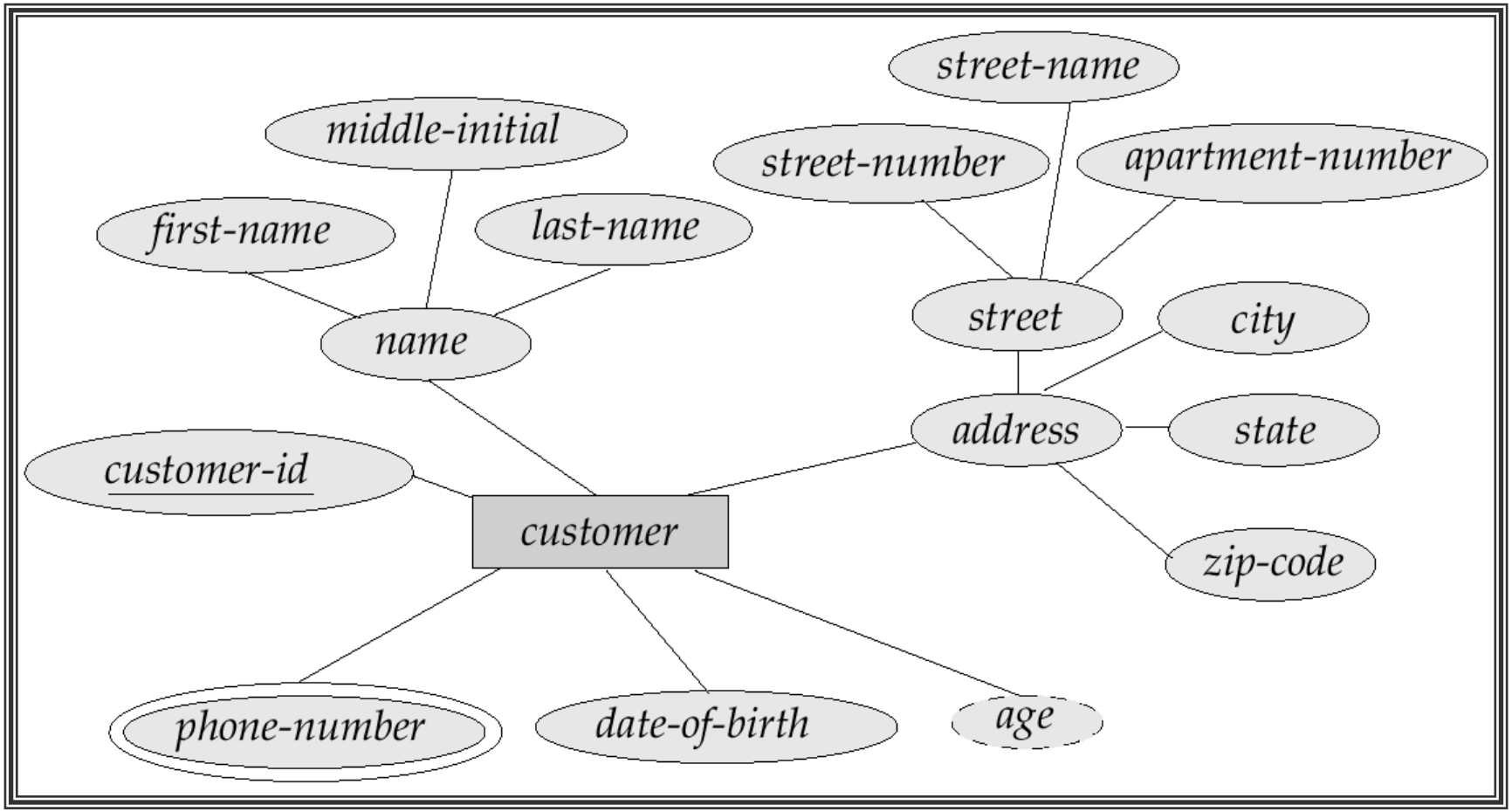
Note: Some elements in A and B may not be mapped to any elements in the other set

E-R Diagrams

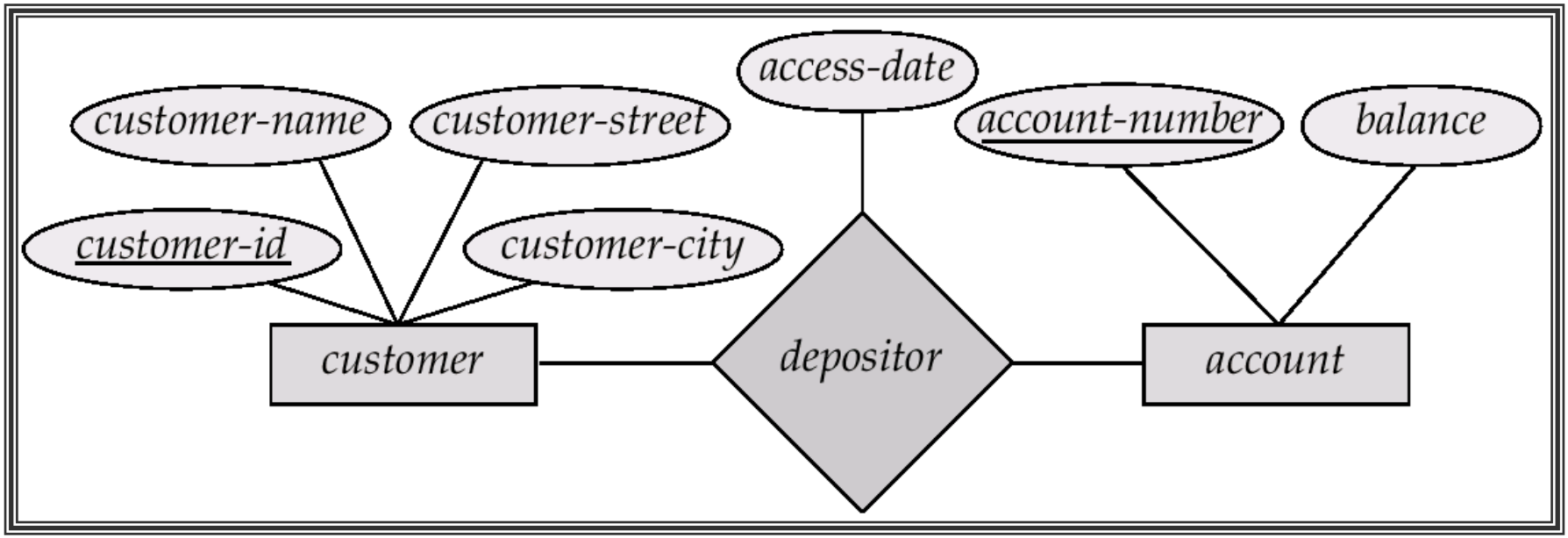


- **Rectangles** represent entity sets.
- **Diamonds** represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- **Ellipses** represent attributes
  - Double ellipses represent multivalued attributes.
  - Dashed ellipses denote derived attributes.
- Underline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

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

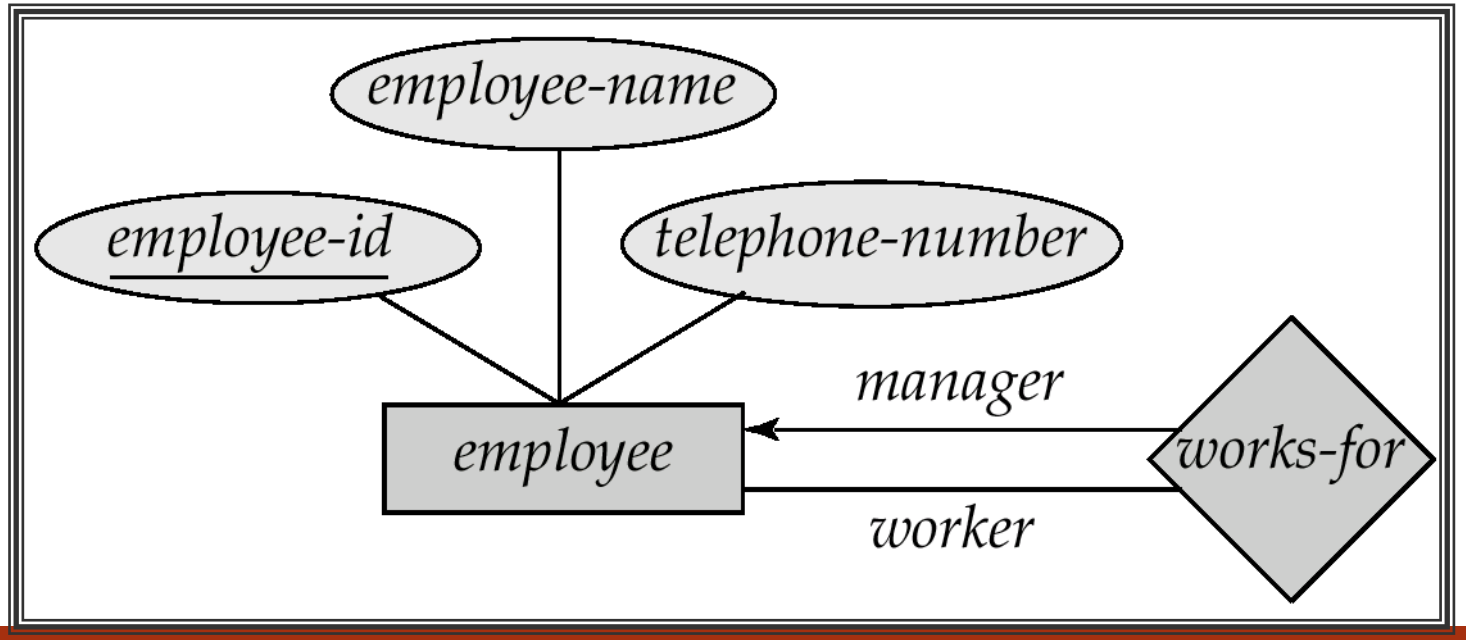


Relationship Sets with Attributes



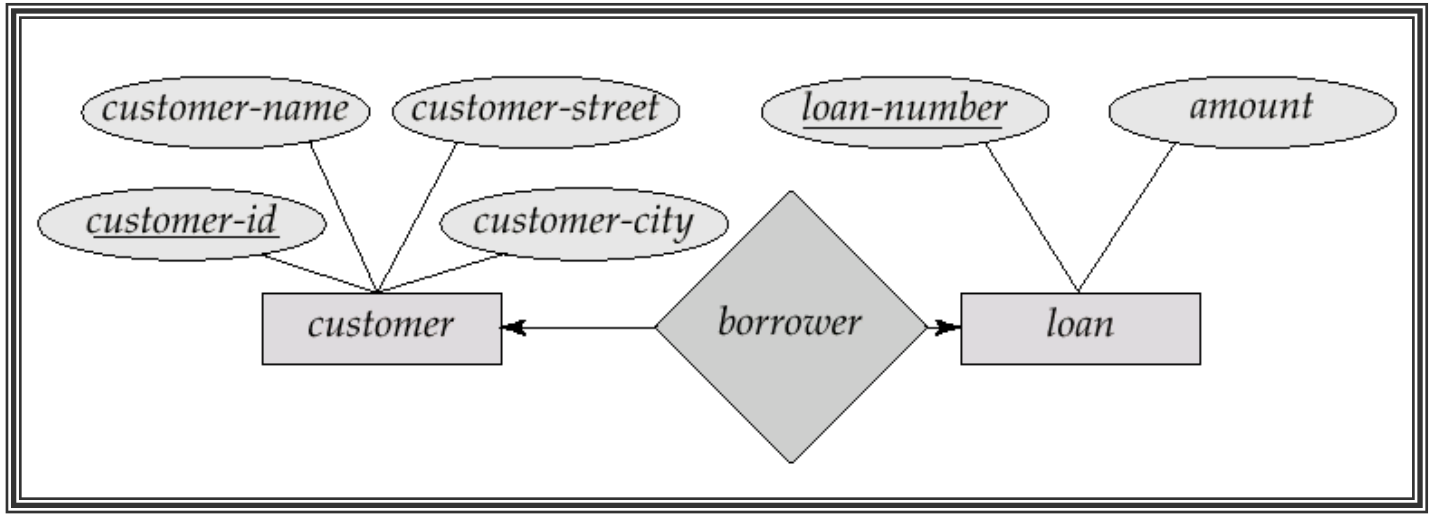
Roles

- 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



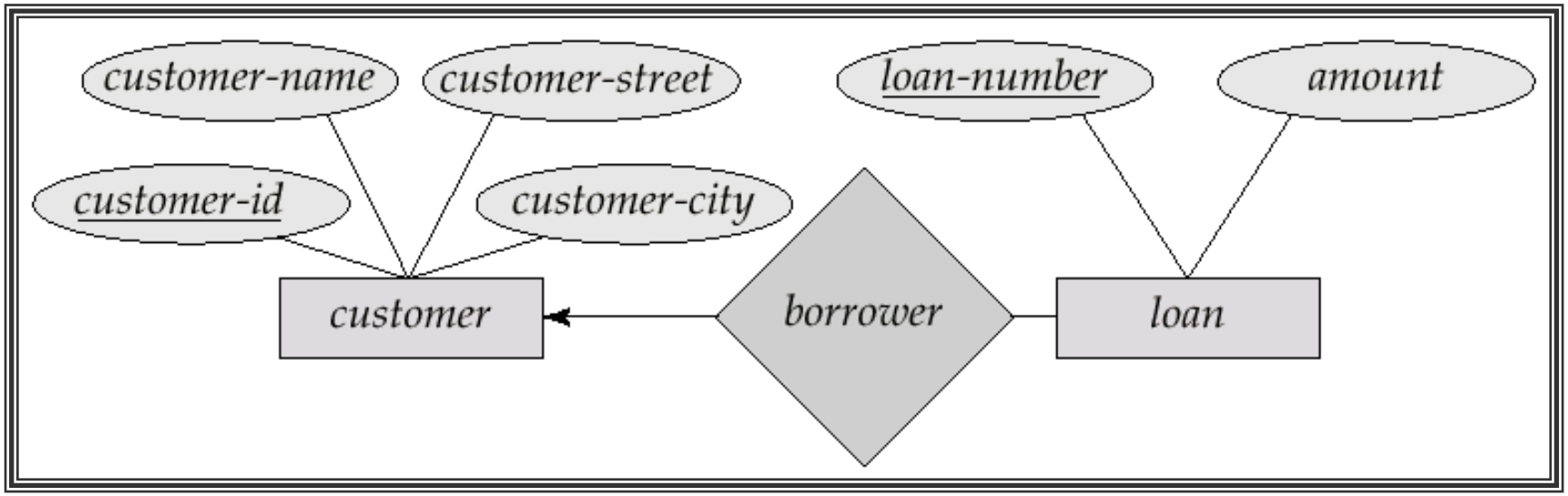
Cardinality Constraints

- 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.
- E.g.: 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*



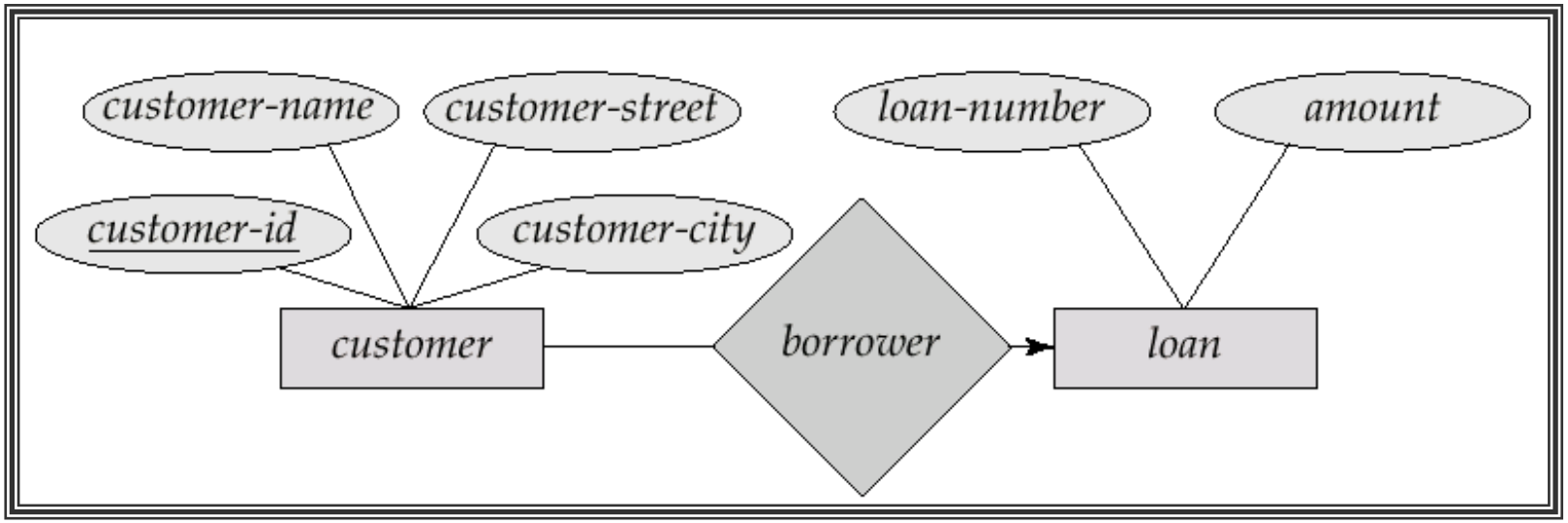
One-To-Many Relationship

- 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*

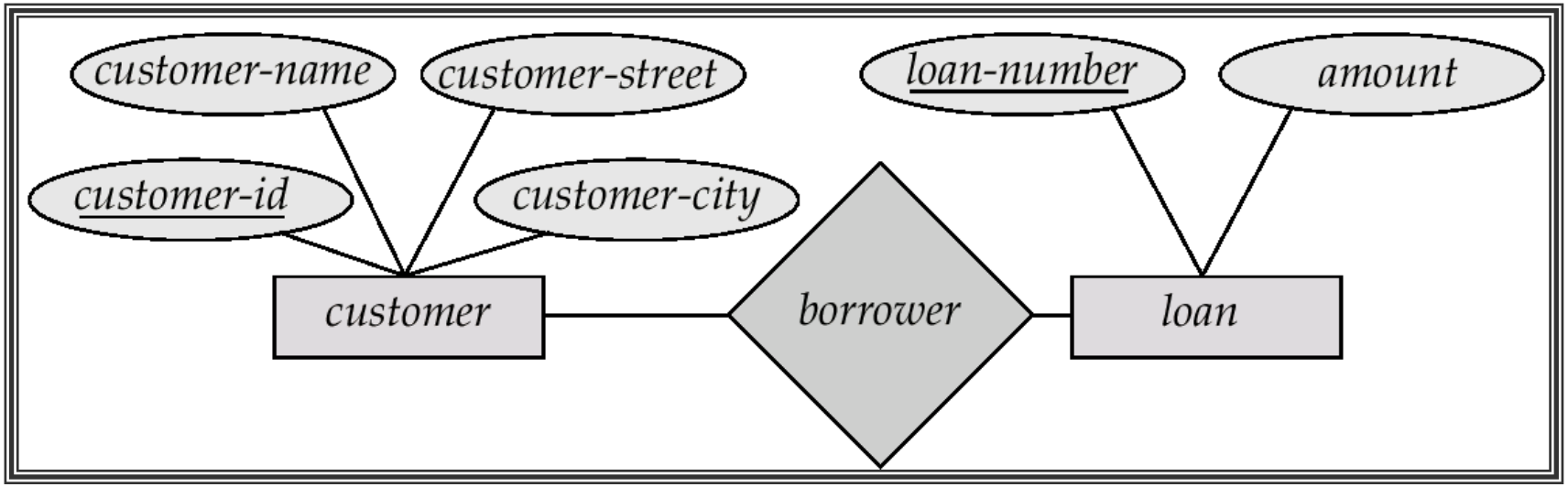


Many-To-One Relationships

- 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*



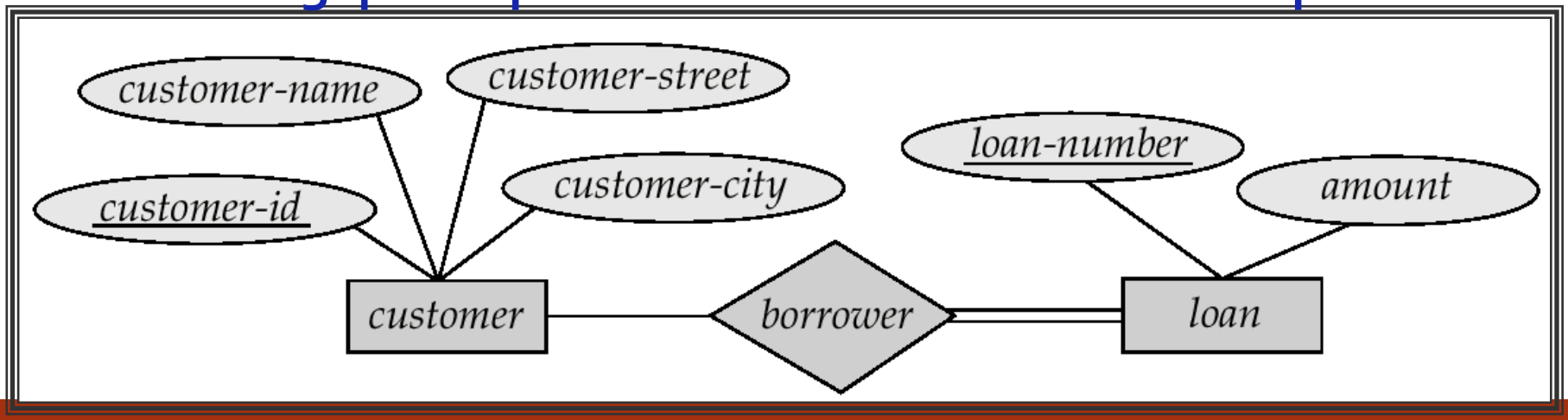
Many-To-Many Relationship



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

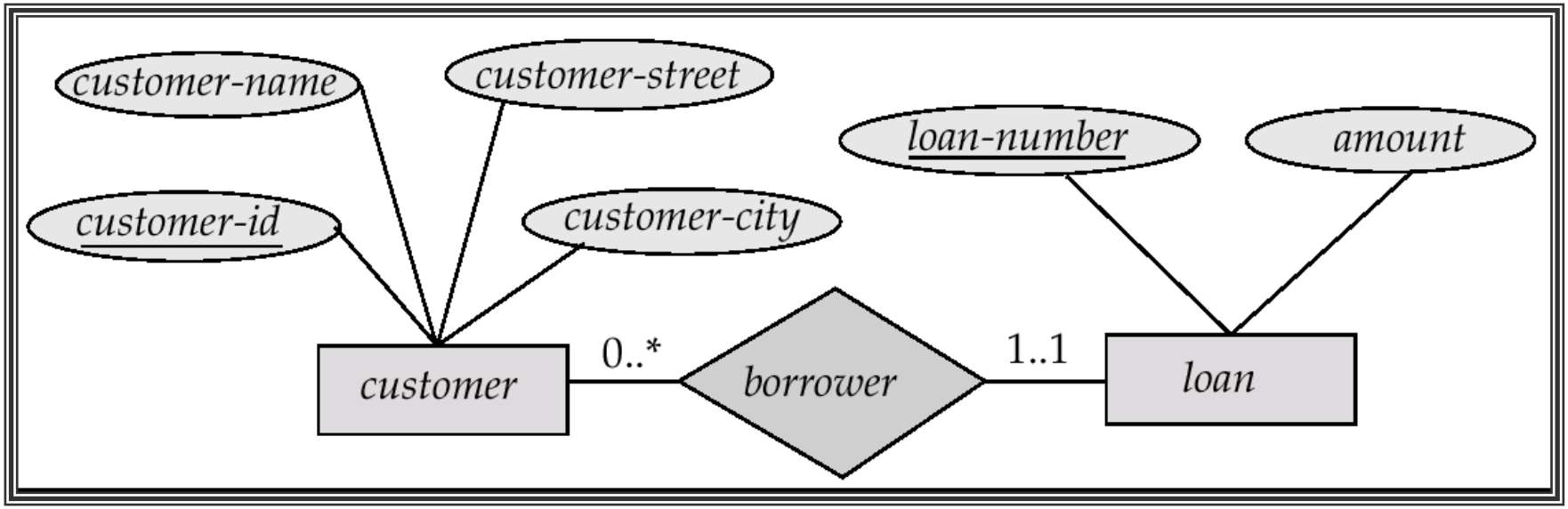
Participation of an Entity Set in a Relationship Set

- **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
  - E.g. participation of *customer* in *borrower* is partial



Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints



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

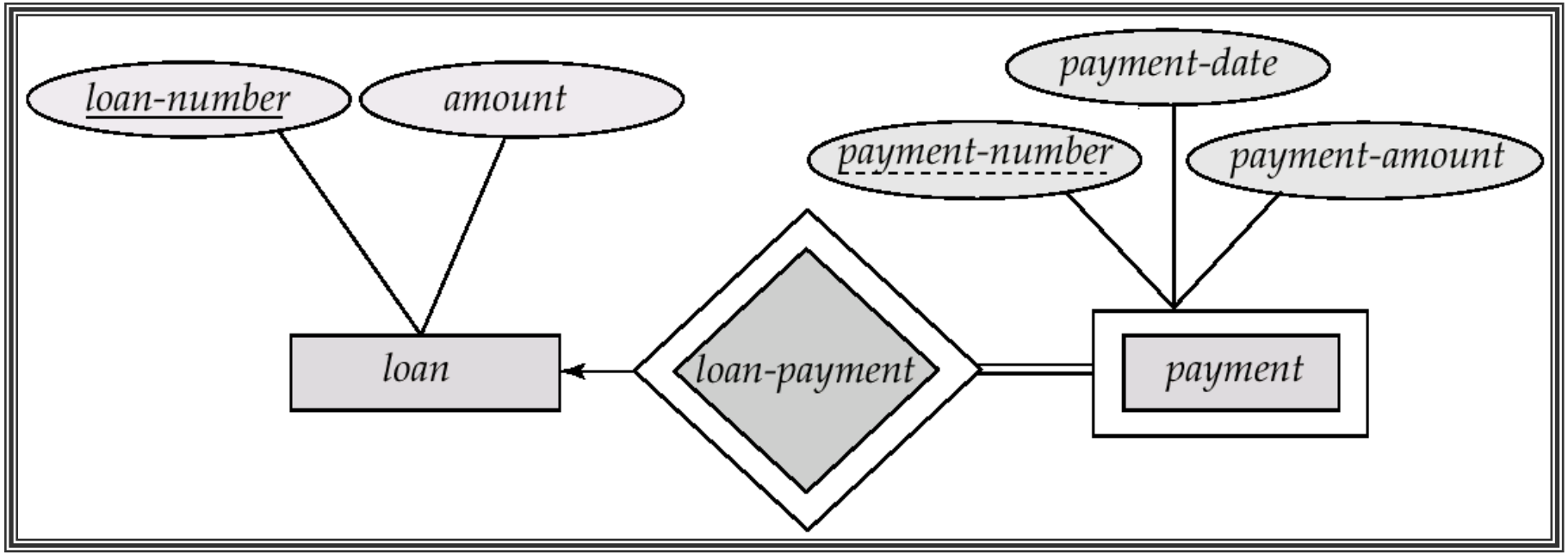
- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (*customer-id*, *account-number*) is the super key of *depositor*
  - *NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.*
- Must consider the mapping cardinality of the relationship set when deciding the 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

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



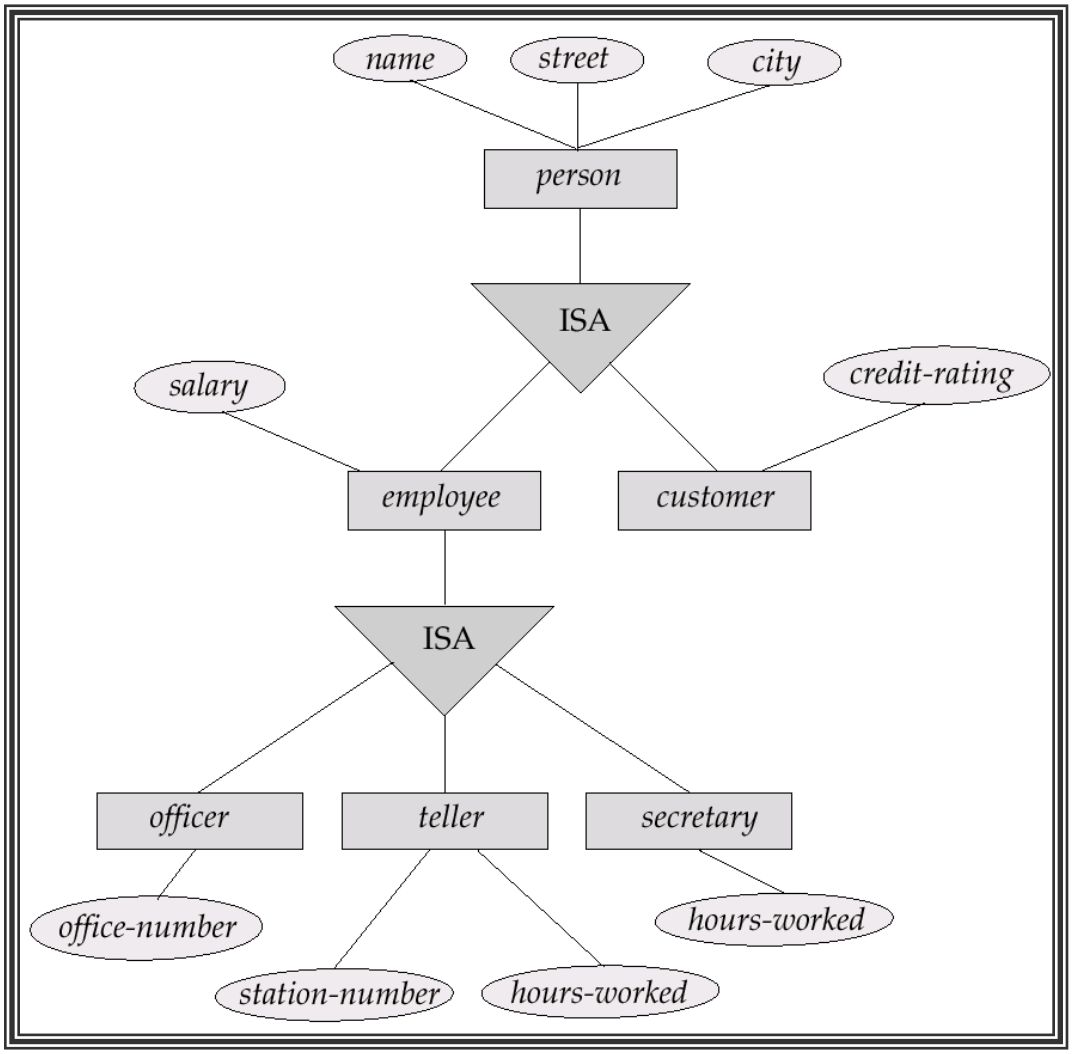
Weak Entity Sets (Cont.)

- **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***

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



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 (Contd.)

- 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

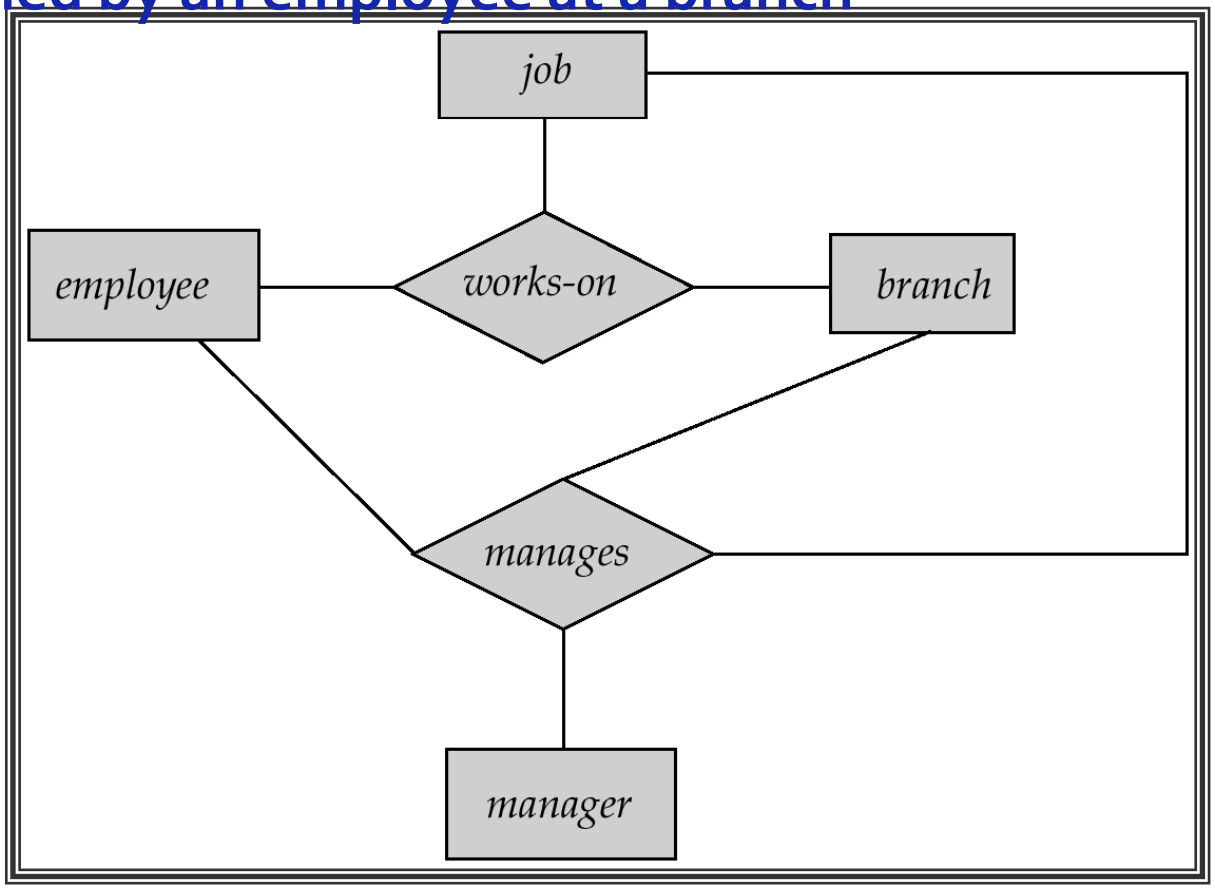
- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - E.g. all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
  - user-defined
- Constraint on whether or not entities may belong to more than one 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 (Contd.)

- 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

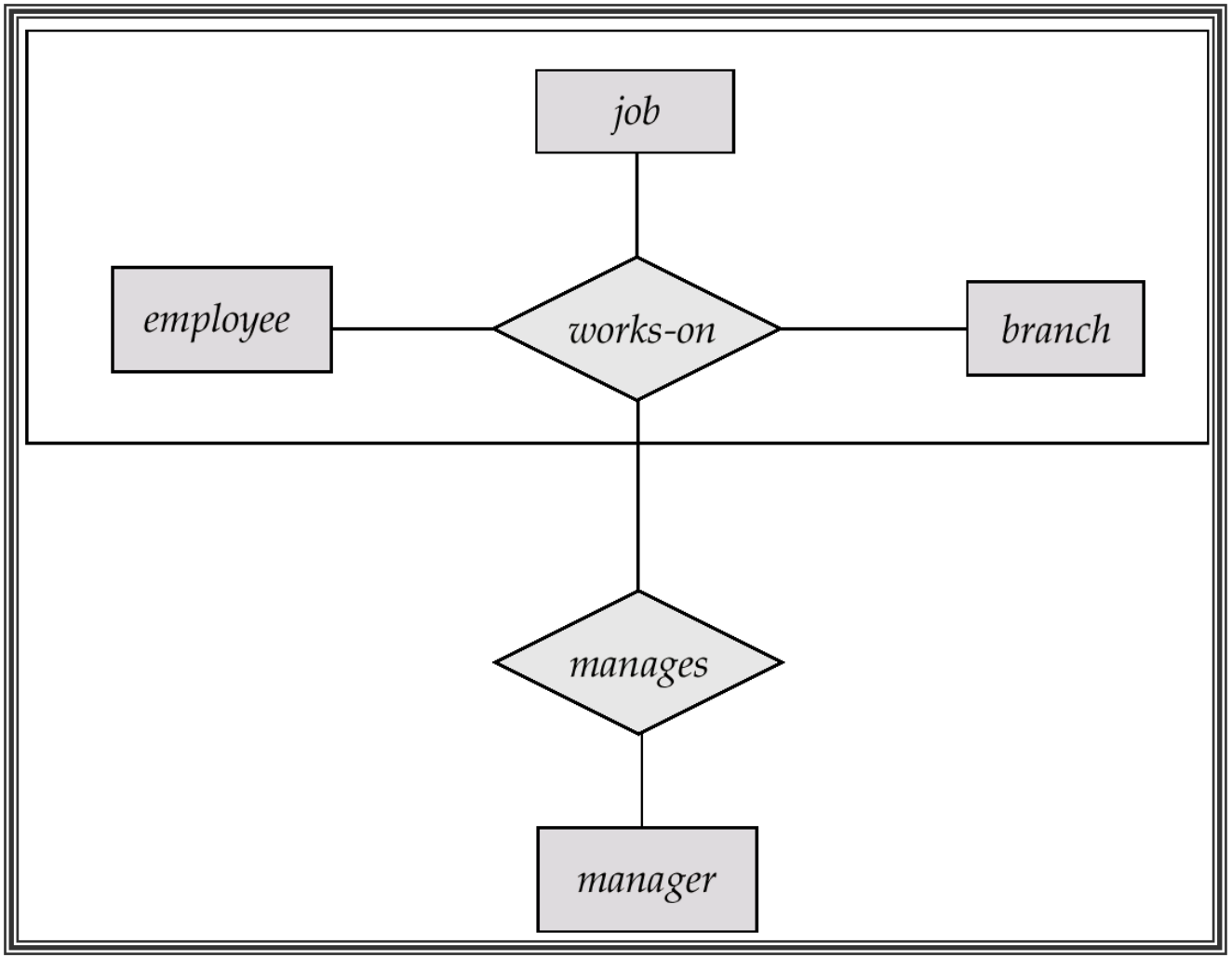
- Consider the ternary relationship *works-on*, which we saw earlier
- Suppose we want to record managers for tasks performed by an employee at a branch



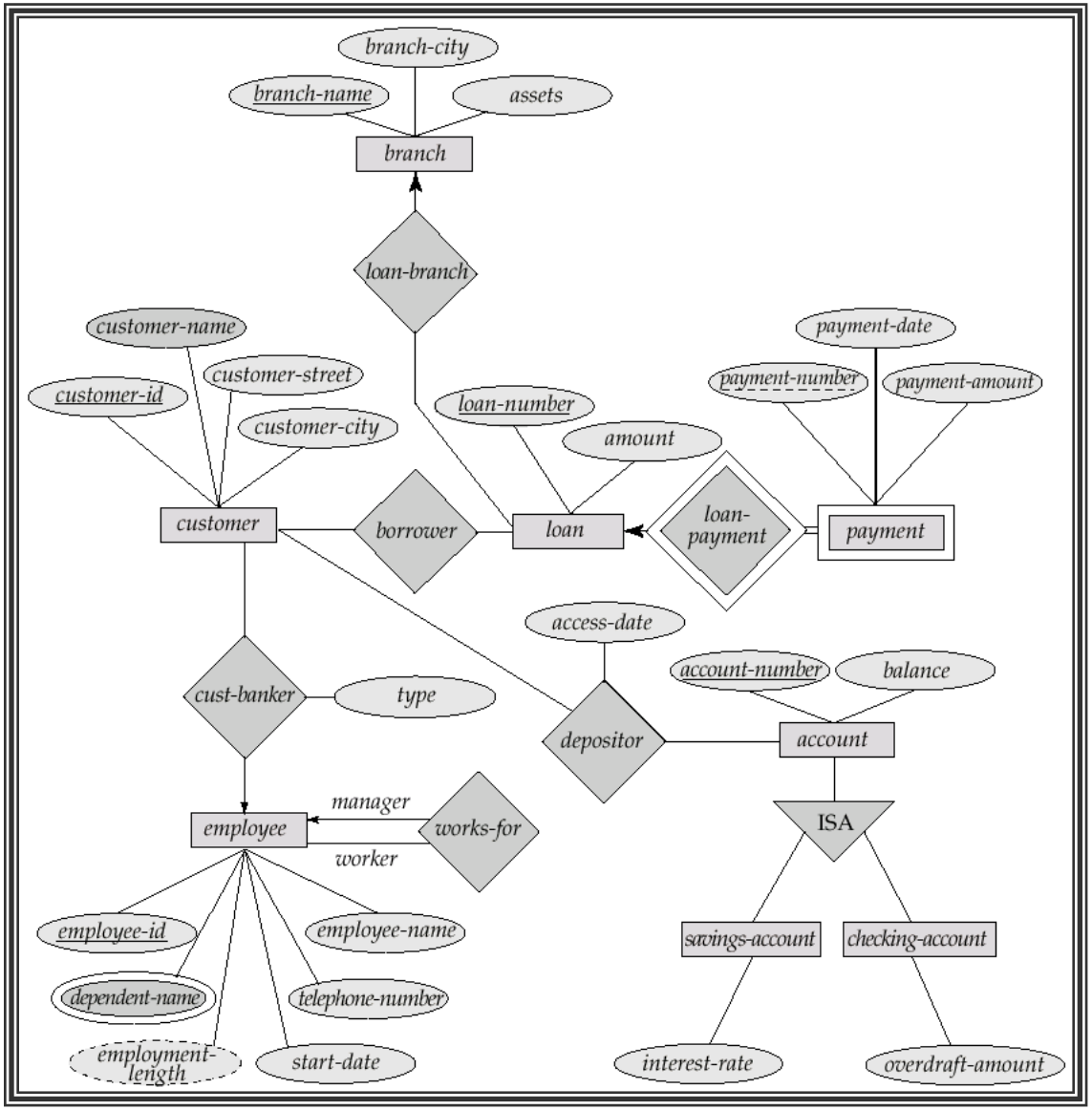
Aggregation (Cont.)

- 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

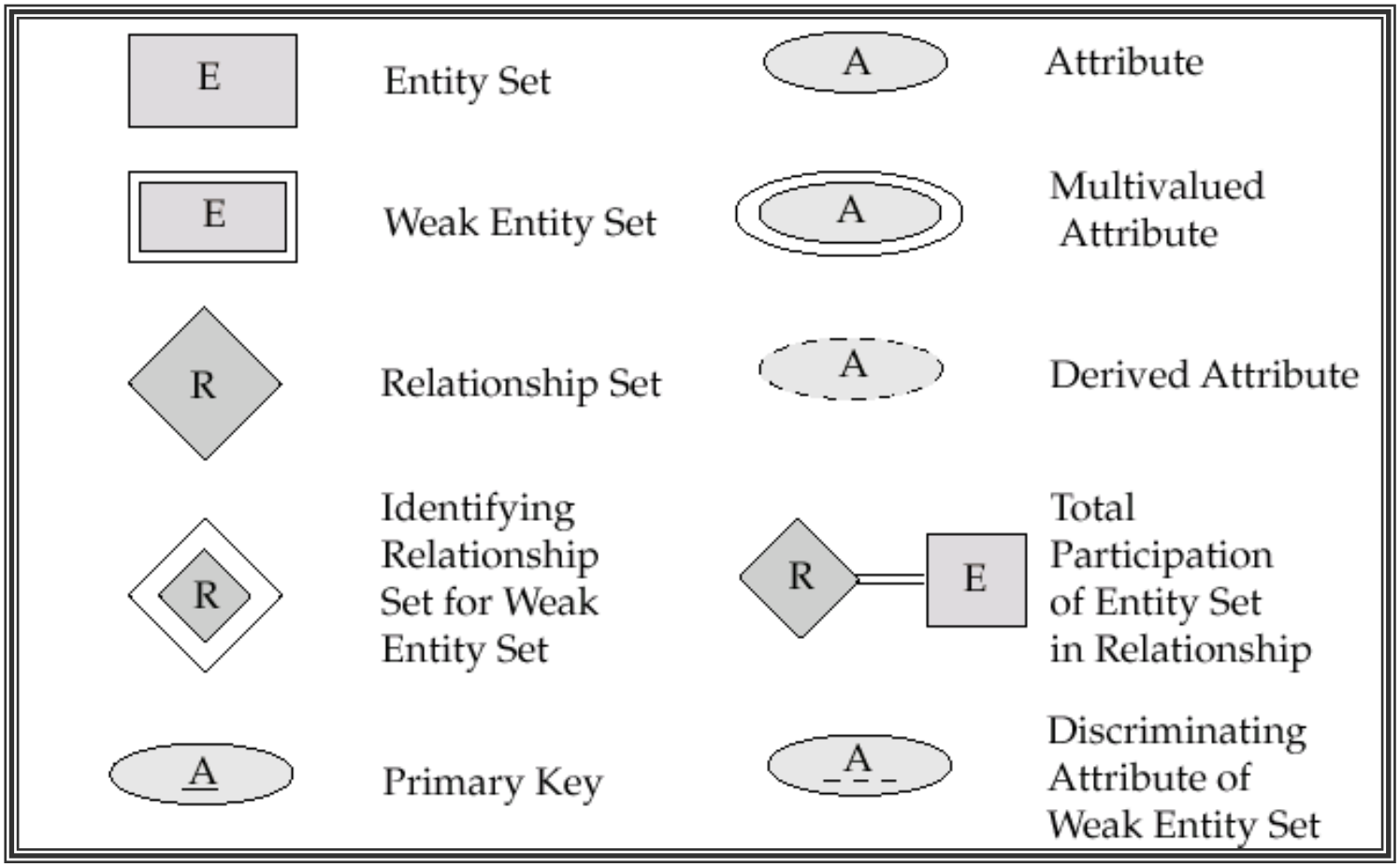
E-R Diagram With Aggregation



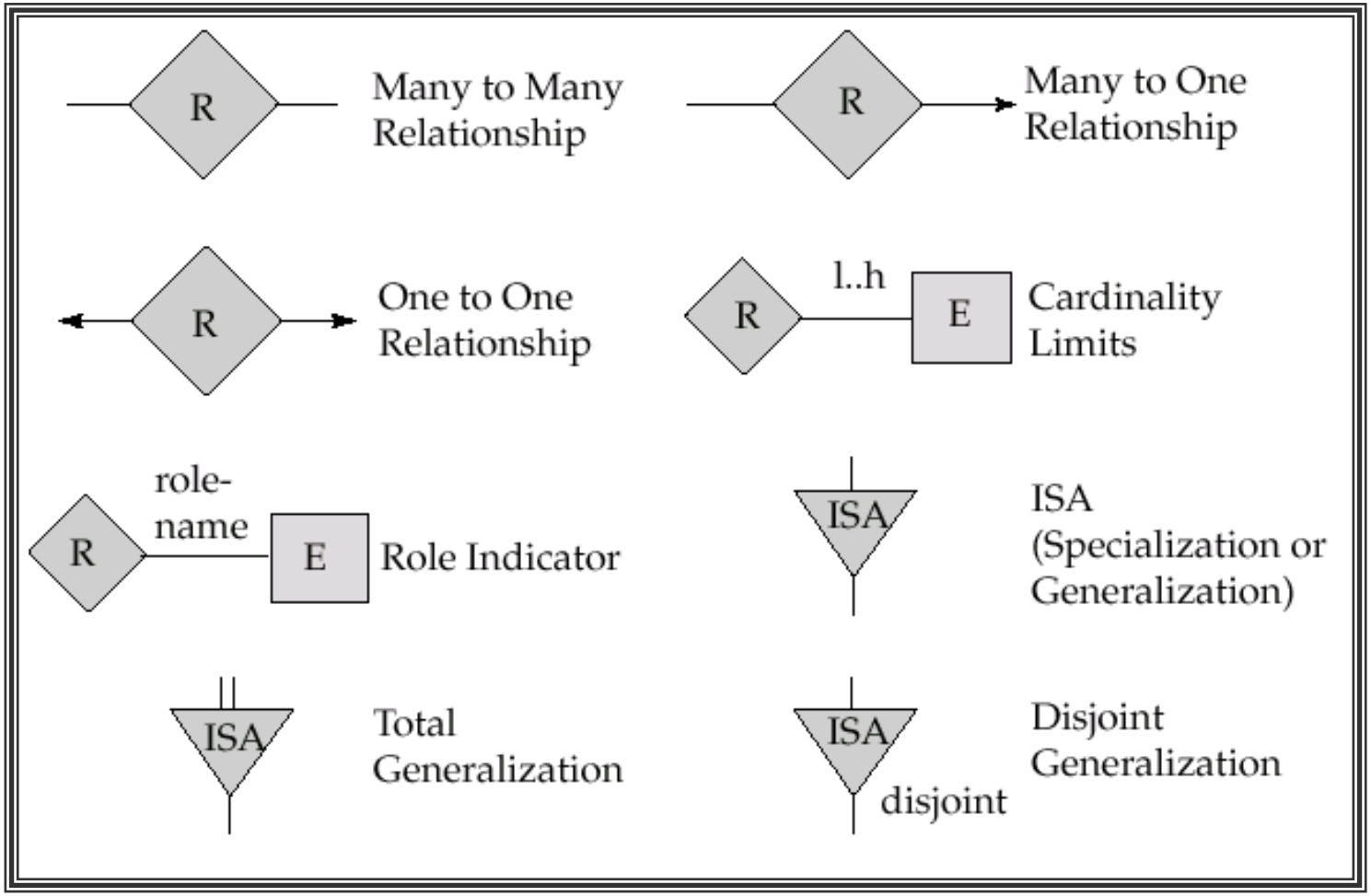
E-R Diagram for a Banking Enterprise



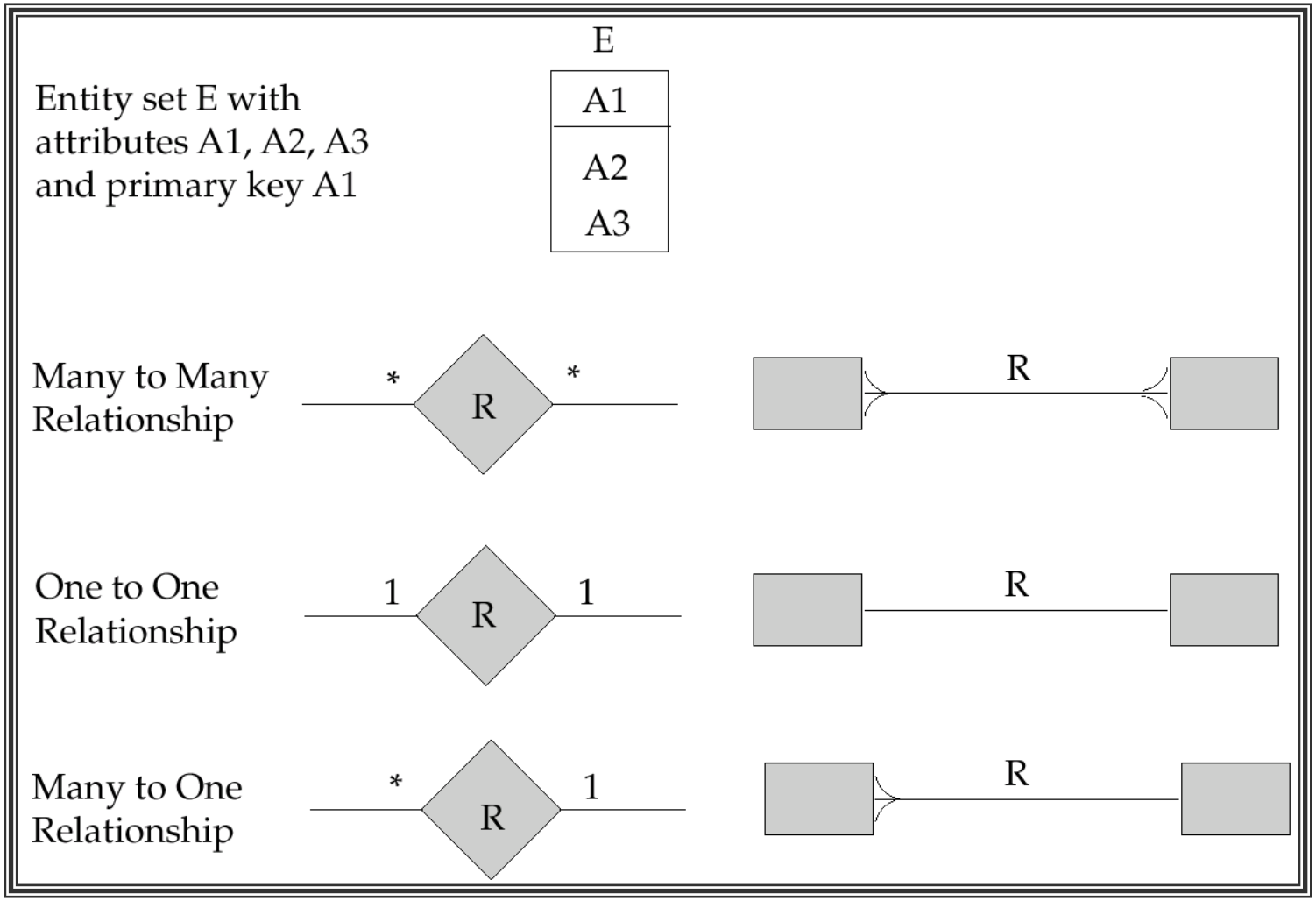
Summary of Symbols Used in E-R Notation



Summary of Symbols (Cont.)



Alternative E-R Notations



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.

Design Issues

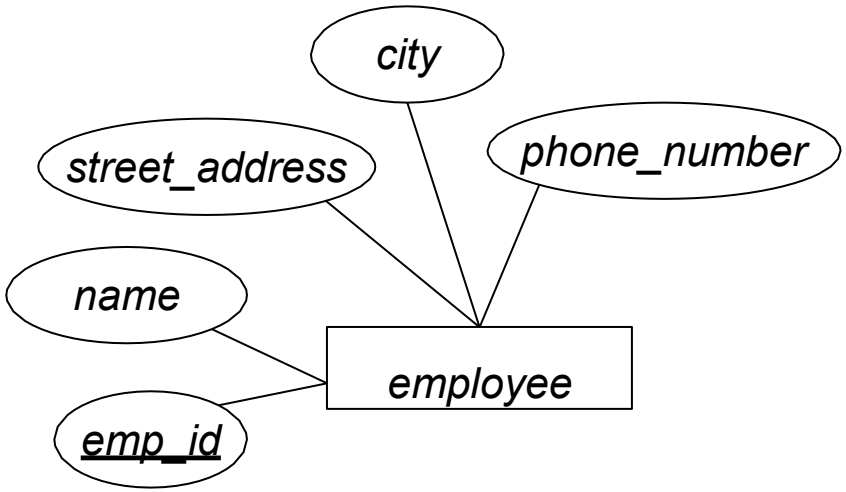
- **Use of entity sets vs. attributes**

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**

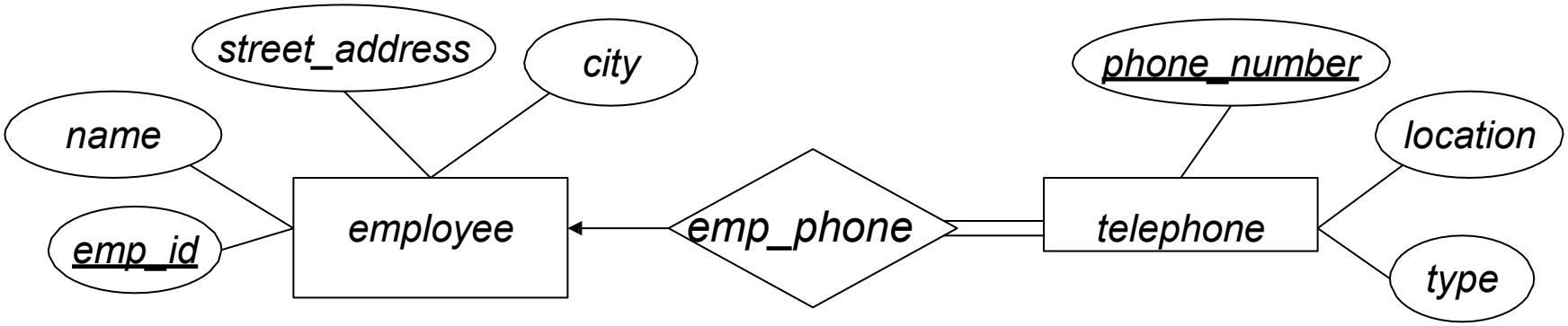
1. Use of entity sets vs. attributes

- When to represent a value as an attribute?
- When to represent a value as a separate entity-set?
- Representing as a separate entity-set allows details to be added later
- Example:
  - Phone number is a good candidate for representing as a separate entity-set
  - Employee name definitely should stay an attribute



Attributes vs. Entity-Sets (2)

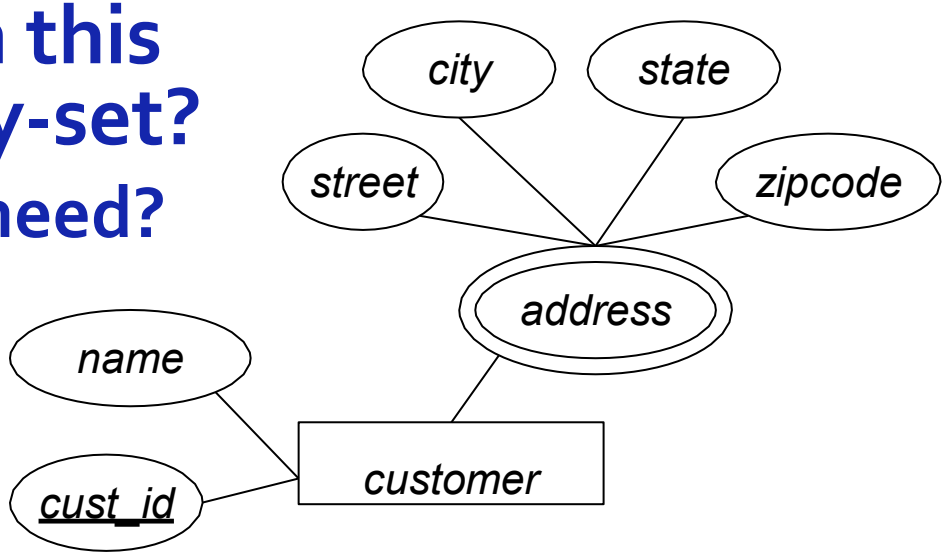
- **Could move *phone_number* to another entity-set**
  - **Add *location*, *type* attributes as well**



- **Differences:**
  - **Attribute was single-valued before…**
  - **Now employees can have multiple phone numbers**
  - **Need to choose participation, cardinality constraints!**
  - **Could also just use a multi-valued composite attribute**

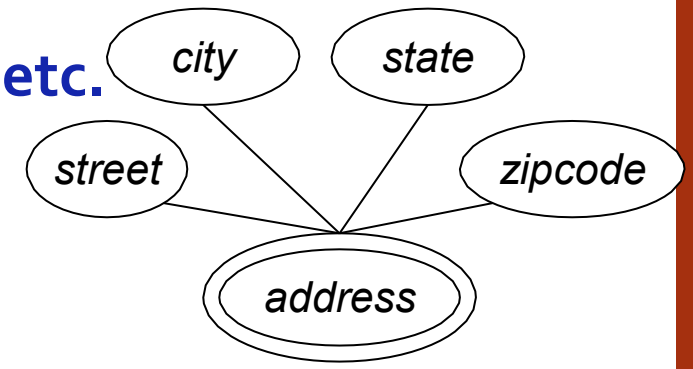
Composite Attributes

- Composite attributes have a big overlap with entity-sets
- Example:
  - Customers can have multiple addresses
  - Make address a composite attribute
- Differences between this and a separate entity-set?
  - What do entity-sets need?



Composite Attributes (2)

- Entity-sets need a primary key, or at least a partial key
- Does composite attribute have a key?
  - If so, should probably migrate to a separate entity-set
  - With this example, entire address is composite key. Not ideal for a separate entity-set.
- If no key attributes, could use a weak entity-set
  - Need to specify a discriminator
  - Set up an identifying relationship, etc.

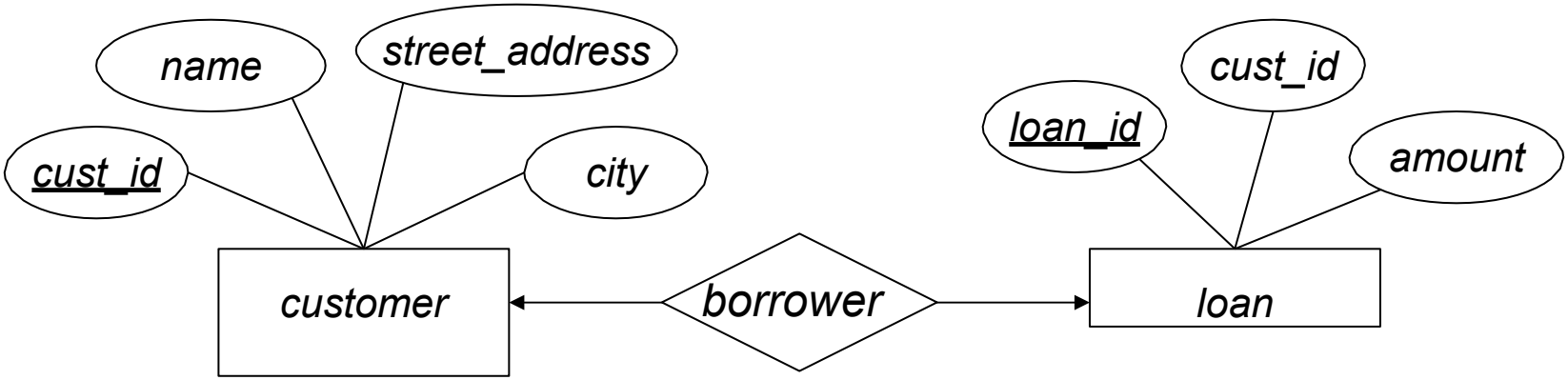


Composite Attributes (3)

- Mapping multi-valued composite attributes to relational schema very similar to mapping for weak entity-sets
- Recommendation:
  - Prefer weak entity-sets over multi-valued composite attributes
  - Most databases don't support user-defined composite types
  - Using weak entity-sets will more closely model implementation schema

Common Attribute Mistakes

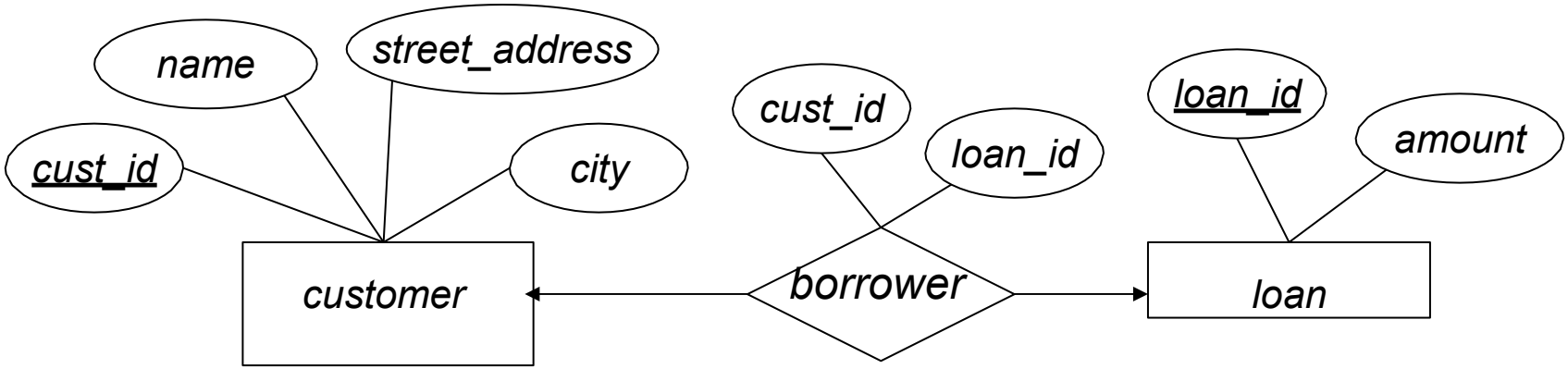
- Don't include entity-set primary key attributes on other entity-sets
  - e.g. customers andloans



- Even if loans are owned by only one customer, this is still wrong
  - Association is contained by the relationship, so specifying foreign key attributes is redundant

Common Attribute Mistakes (2)

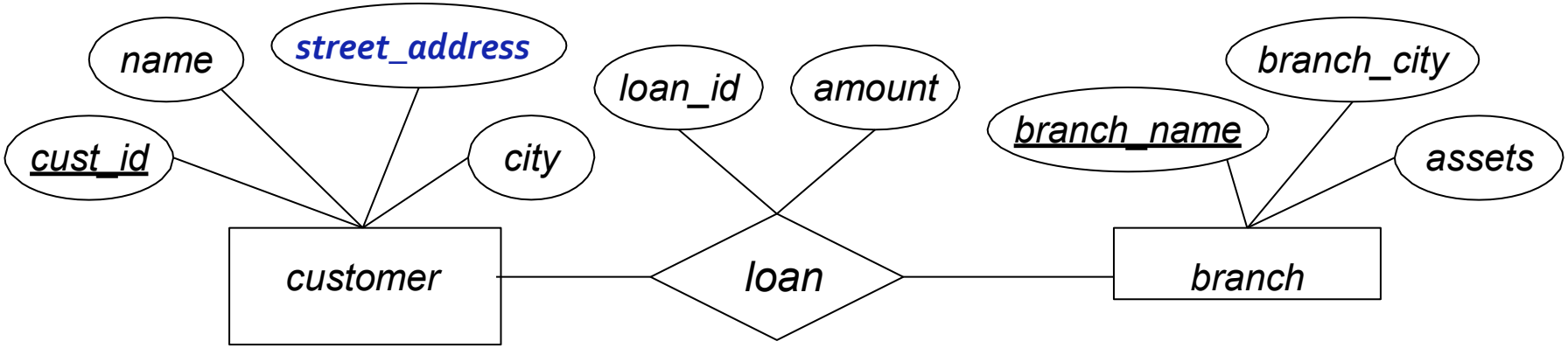
- Don't include primary key attributes as descriptive attributes on relationship-set
- Example: *customer* and *loan* relations again
  - IDs used as descriptive attributes on *borrower*



- Again, this is implicit in the relationship

2. Use of entity sets vs. relationship sets

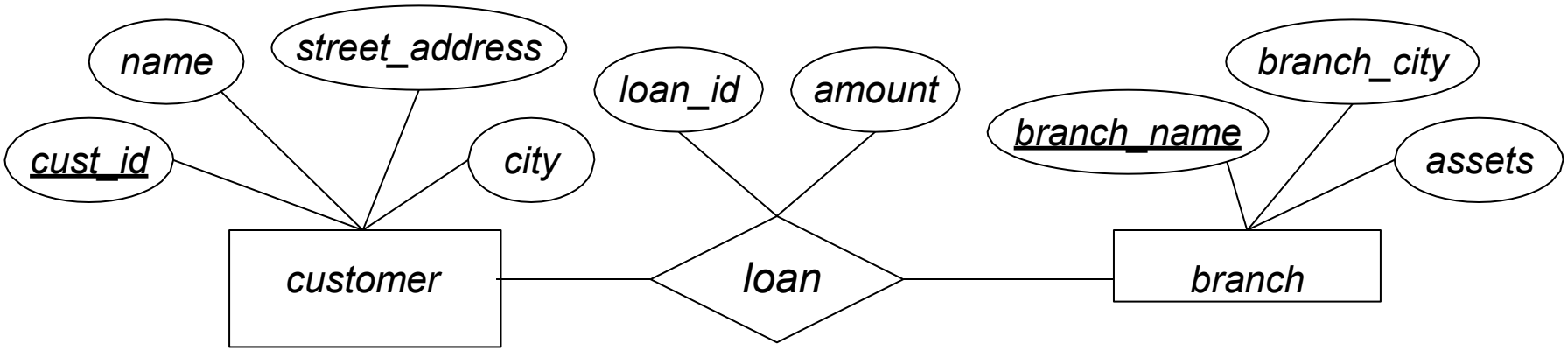
- Want to represent loans given to exactly one customer
  - Each loan is given at a particular bank branch
- What about this:



  - Represent loans as relationships between customers and branches
  - Loan number and amount are descriptive attributes

Entities vs. Relationships (2)

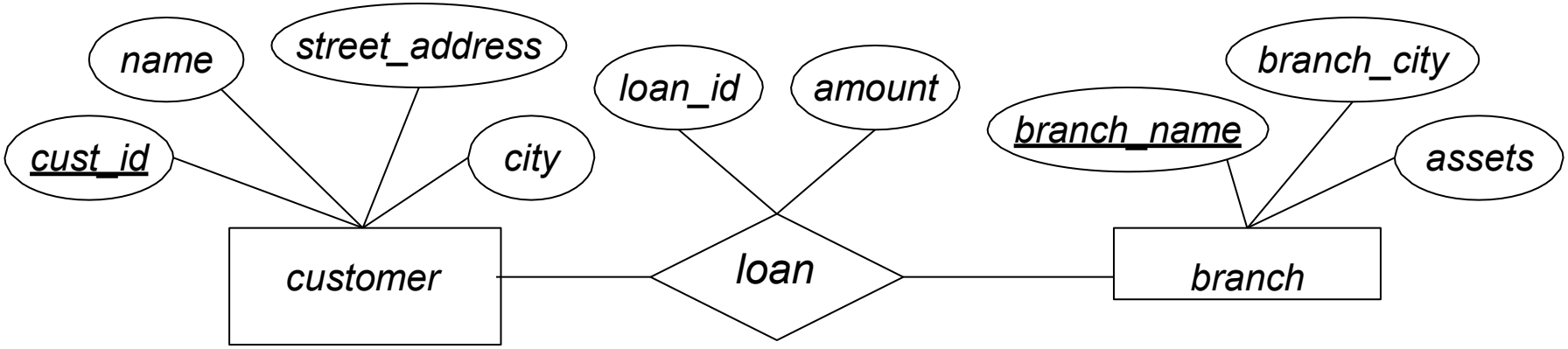
- Would definitely work…



- Problem:
  - How to enforce uniqueness of loan IDs?
  - Can't make keys from descriptive attributes on relationships
  - Using this approach, can't constrain values. Not good

Entities vs. Relationships (3)

- Also constrains us to an unrealistic scenario
  - Normally want to allow joint ownership of loans
- *Could* we model joint ownership with this?

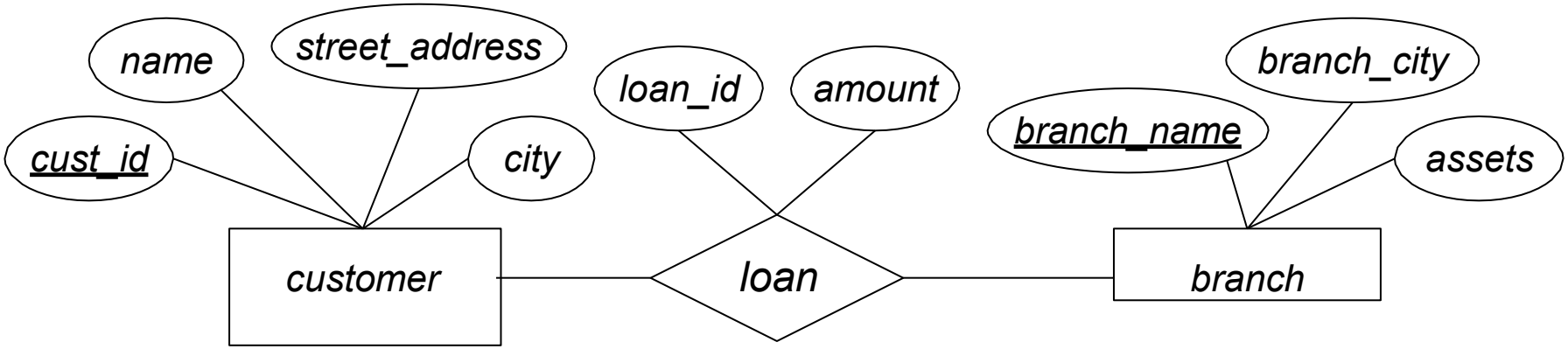


  - *Could* have multiple relationships with same loan ID…

ng"
n "action" involving entities
ate schema against

necessary redundancy, or
istency issues

Entities vs. Relationships (4)



- Modeling multiple-owner loans with this schema causes big problems
  - Same loan number and amount must appear in multiple relationships
  - Wastes storage space
  - Presents serious consistency issues!
- Want to avoid necessary redundancy!

Entities vs. Relationships (5)

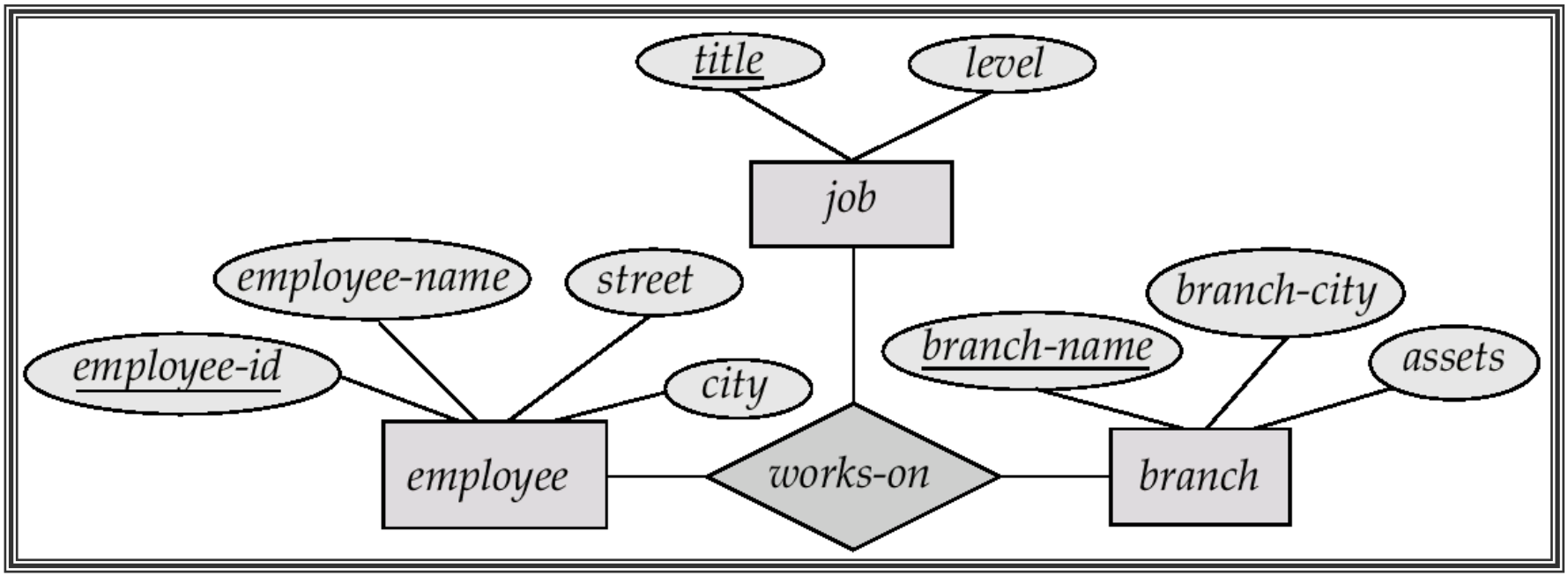
- Simple guideline for choosing entities or relationships
  - An entity is a "thing"
  - A relationship is an "action" involving entities
- In general, evaluate schema against various scenarios
  - Watch out for unnecessary redundancy, or potential for consistency issues

4. Binary Vs. Non-Binary Relationships

- Some relationships that appear to be non-binary may be better represented using binary relationships

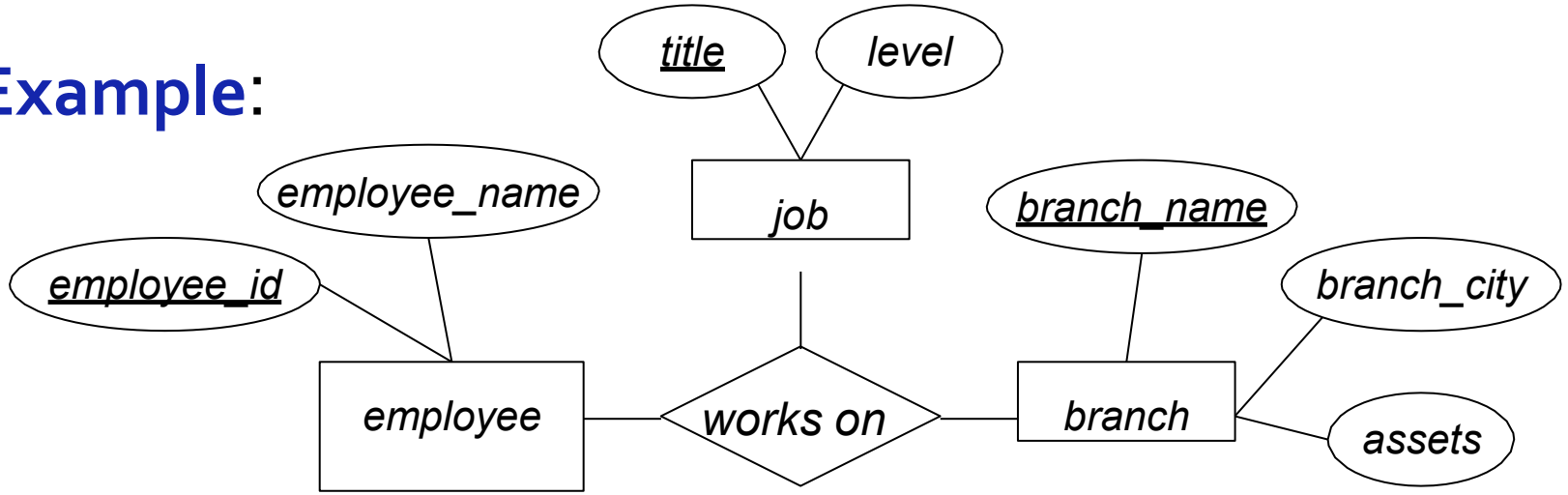
  But there are some relationships that are naturally non-binary

  - E.g. *works-on*



N-ary Relationships

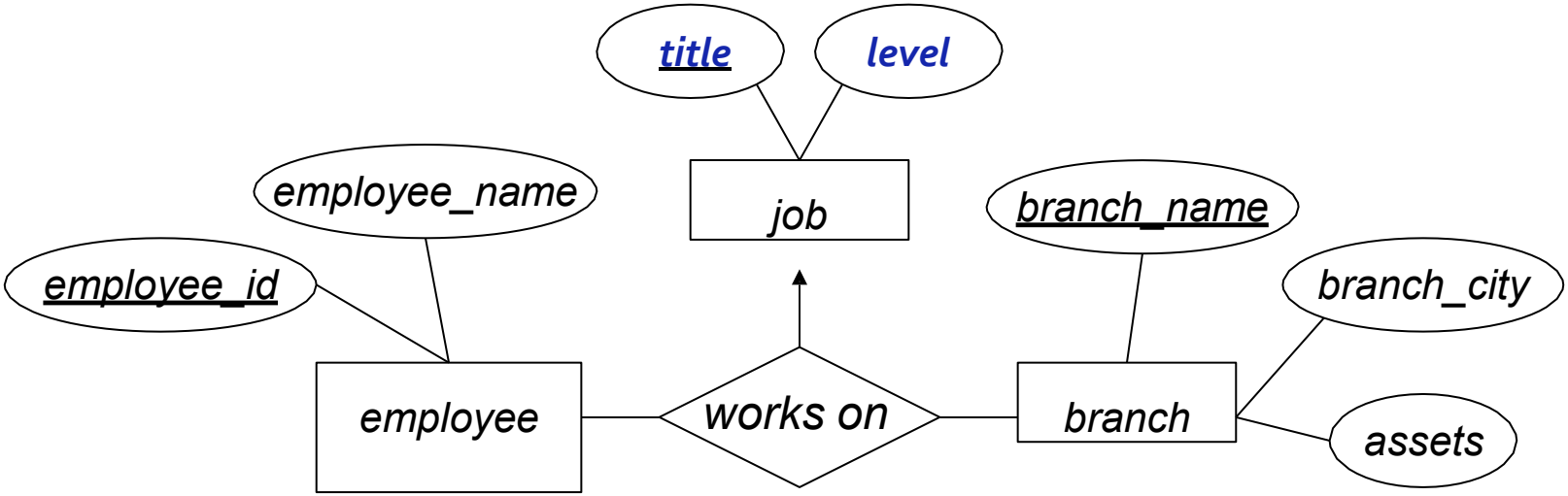
- Can specify relationships of degree > 2 in E-R model
- Example:



  - Employees are assigned to jobs at various branches
  - Many-to-many mapping: any combination of employee, job, and branch is allowed
  - An employee can have several jobs at one branch

N-ary Mapping Cardinalities

- Can specify *some* mapping cardinalities on relationships with degree > 2
- Constrain employees to one job per branch:



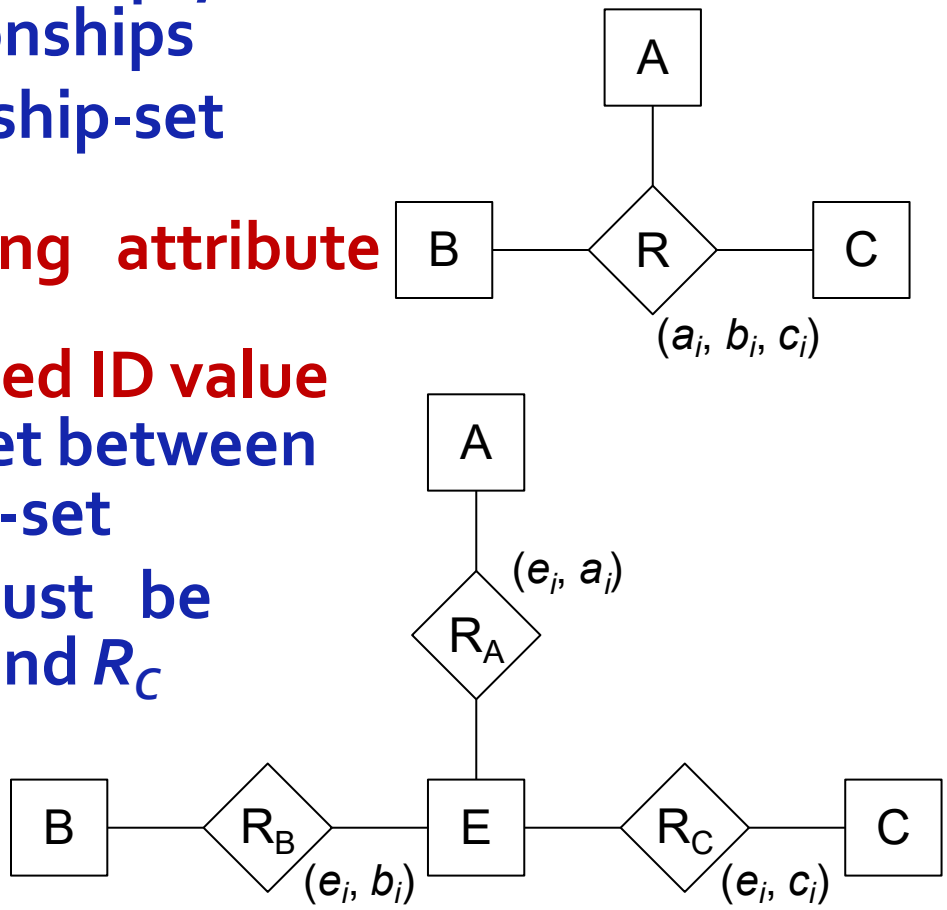
  - Each combination of employee and branch can only be associated with one job

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

Binary vs. N-ary Relationships

- Normally have only binary relationships in database schemas
- For degree > 2 relationships, can replace with binary relationships
  - Replace N-ary relationship-set with a new entity-set *E*
    - Create an identifying attribute for *E*
    - e.g. an auto-generated ID value
  - Create a relationship-set between *E* and each other entity-set
  - Relationships in *R* must be represented in R_A, R_B, and R_C

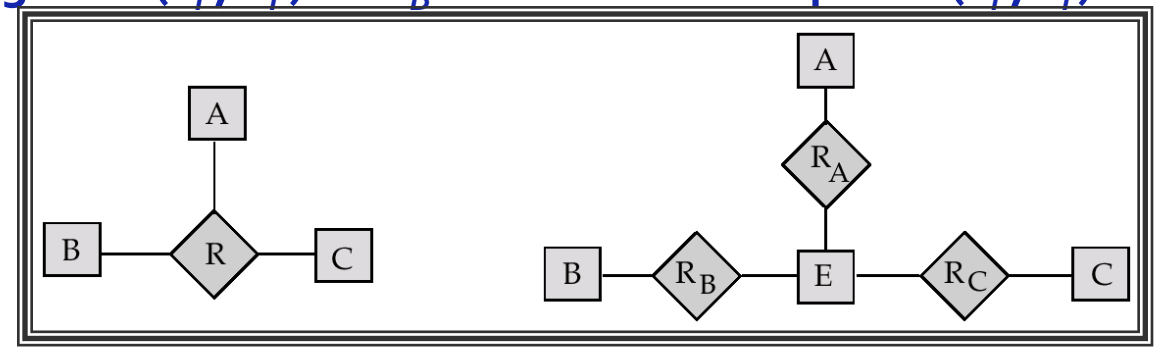


Converting Non-Binary Relationships to Binary Form

- 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:

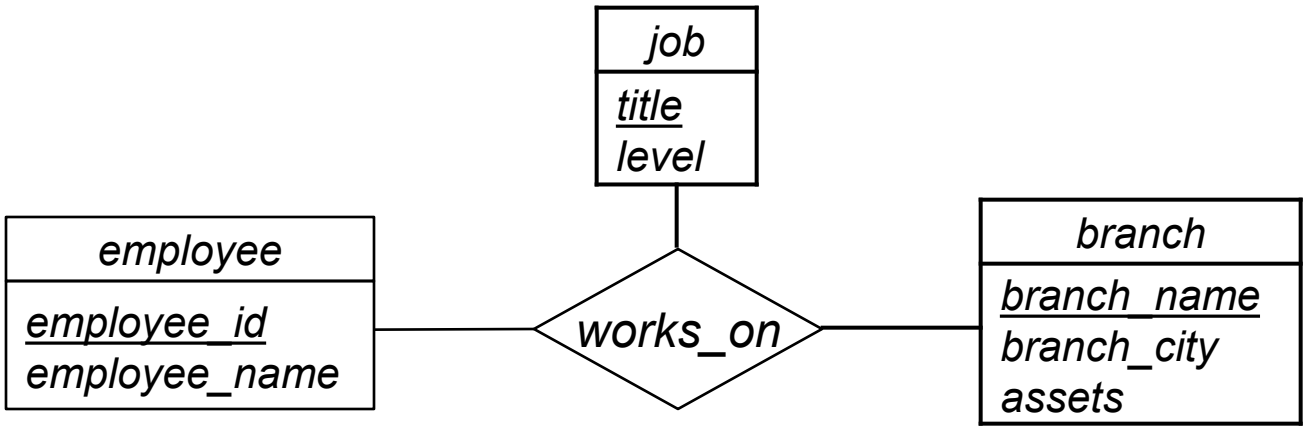
    1. R_A, relating *E* and *A*
    2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*
  - Create a special identifying attribute for *E*
  - Add any attributes of *R* to *E*
  - For each relationship (a_i , b_i , c_i) in *R*, create

    1. a new entity e_a. 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



N-ary Relationships

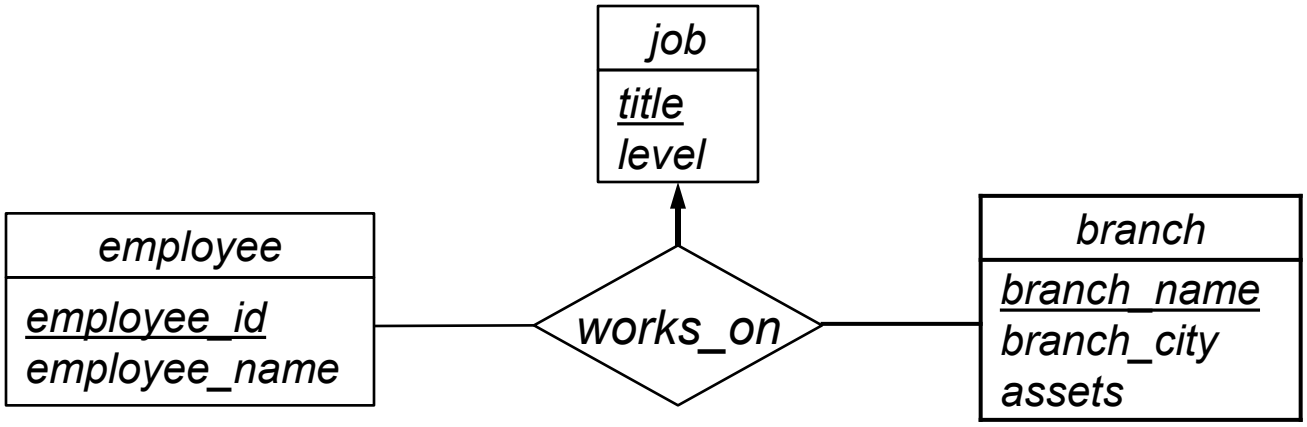
- Can specify relationships of degree > 2 in E-R model
- Example:



  - Employees are assigned to jobs at various branches
  - Many-to-many mapping: any combination of employee, job, and branch is allowed
  - An employee can have several jobs at one branch

N-ary Mapping Cardinalities

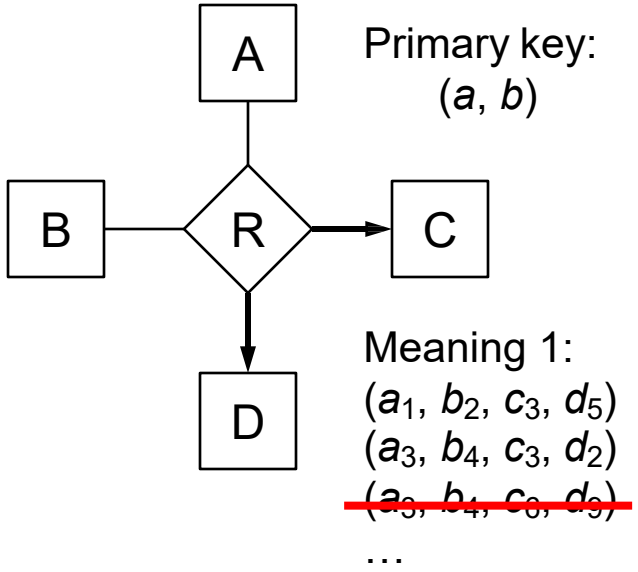
- Can specify *some* mapping cardinalities on relationships with degree > 2
- Each combination of employee and branch can only be associated with <u>one</u> job:



  - Each employee can have only one job at each branch

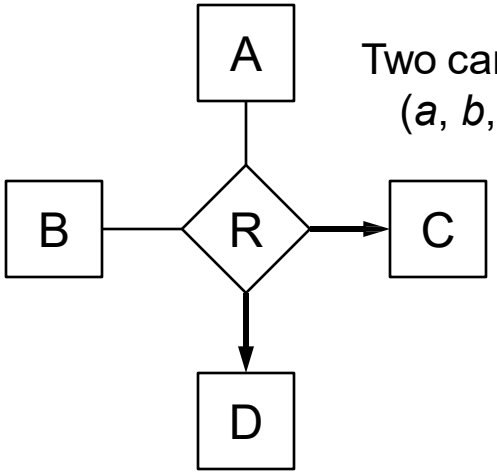
N-ary Mapping Cardinalities (2)

- For degree > 2 relationships, we only allow at most one edge with an arrow
- Reason: multiple arrows on N-ary relationship-set is ambiguous
  - (several meanings have been defined for this in the past)
- Relationship-set R associating entity-sets $A_1, A_2, \ldots, A_n$
  - No arrows on edges $A_1, \ldots, A_i$
  - Arrows are on edges to $A_{i+1}, \ldots, A_n$
- Meaning 1 (the simpler one):
  - A particular combination of entities in $A_1, \ldots, A_i$ can be associated with at most one set of entities in $A_{i+1}, \ldots, A_n$
  - Primary key of R is union of primary keys from set $\{ A_1, A_2, \ldots, A_i \}$



N-ary Mapping Cardinalities (3)

- Relationship-set R associating entity-sets $A_1, A_2, \ldots, A_n$
  - No arrows on edges $A_1, \ldots, A_i$; arrows on edges to $A_{i+1}, \ldots, A_n$
- Meaning 2 (the insane one):
  - For each entity-set A_k ($i < k \leq n$), a particular combination of entities from *all other* entity-sets can be associated with at most one entity in A_k
  - R has a candidate key for each arrow in N-ary relationship-set
  - For each k ($i < k \leq n$), another candidate key of R is union of primary keys from entity-sets $\{ A_1, A_2, \ldots, A_{k-1}, A_{k+1}, \ldots, A_n \}$



Two candidate keys:
(a, b, c), (a, b, d)

Meaning 2:
(a_1, b_2, c_3, d_5)
(a_3, b_4, c_3, d_2)
(a_1, b_2, c_1, d_4)
(a_3, b_4, c_5, d_7)
~~(a_1, b_2, c_3, d_6)~~
~~(a_3, b_4, c_8, d_2)~~
…

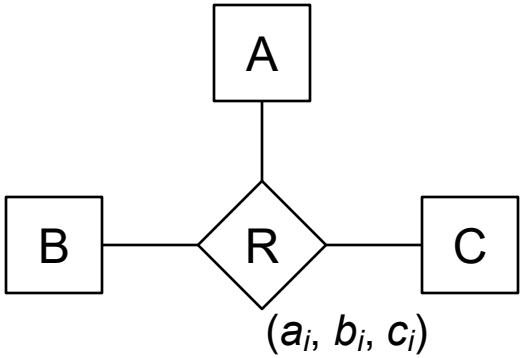
All disallowed by meaning 1!

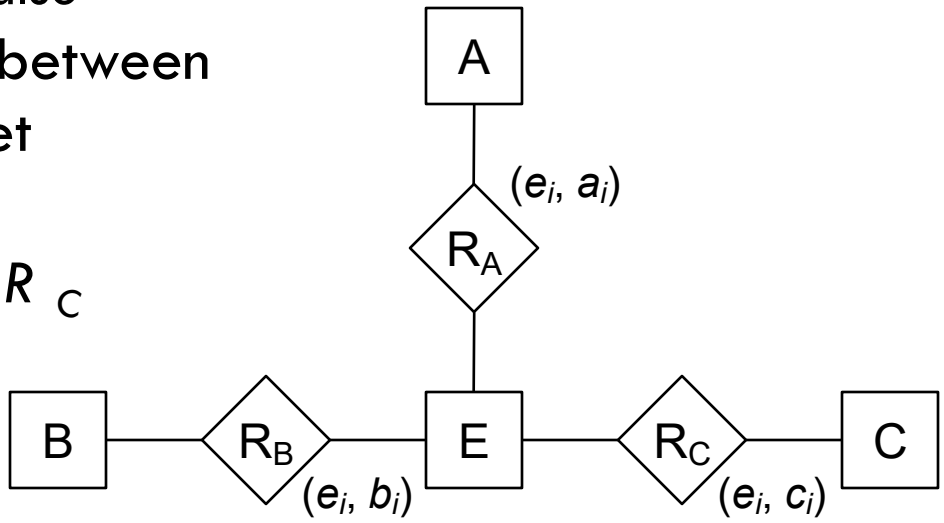
N-ary Mapping Cardinalities (4)

- Both interpretations of multiple arrows have been used in books and papers…
- If we only allow one edge to have an arrow, both definitions are equivalent
  - The ambiguity disappears

Binary vs. N-ary Relationships

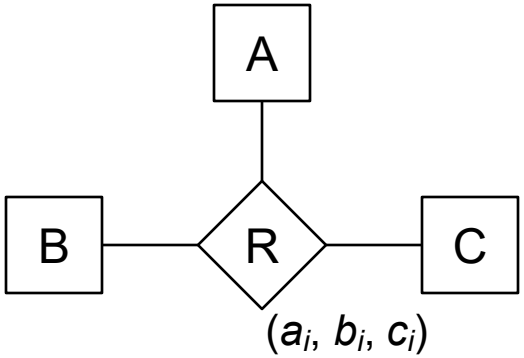
- Often have only binary relationships in DB schemas
- For degree > 2 relationships, *could* replace with binary relationships
  - Replace N-ary relationship-set with a new entity-set *E*
    - Create an identifying attribute for *E*
    - e.g. an auto-generated ID value
  - Create a relationship-set between *E* and each other entity-set
  - Relationships in *R* must be represented in R_A, R_B and R_C

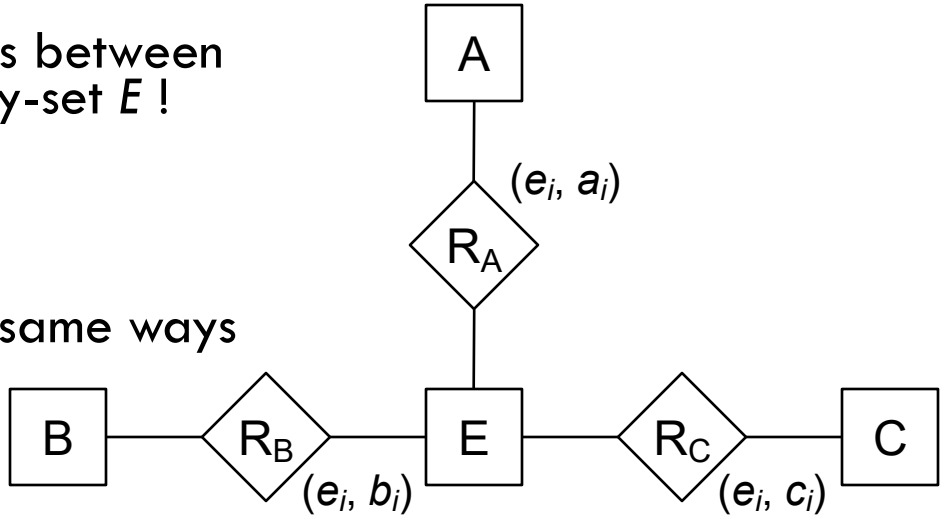




Binary vs. N-ary Relationships (2)

- **Are these representations identical?**
- Example: Want to represent a relationship between entities a_5, b_1 and c_2
  - How many relationships can we actually have between these three entities?
- Ternary relationship set:
  - Can only store one relationship between a_5, b_1 and c_2, due to primary key of *R*
- Alternate approach:
  - Can create <u>many</u> relationships between these entities, due to the entity-set *E* !
    - (a_5, e_1), (b_1, e_1), (c_2, e_1)
    - (a_5, e_2), (b_1, e_2), (c_2, e_2)
    - …
  - Can't constrain in exactly the same ways



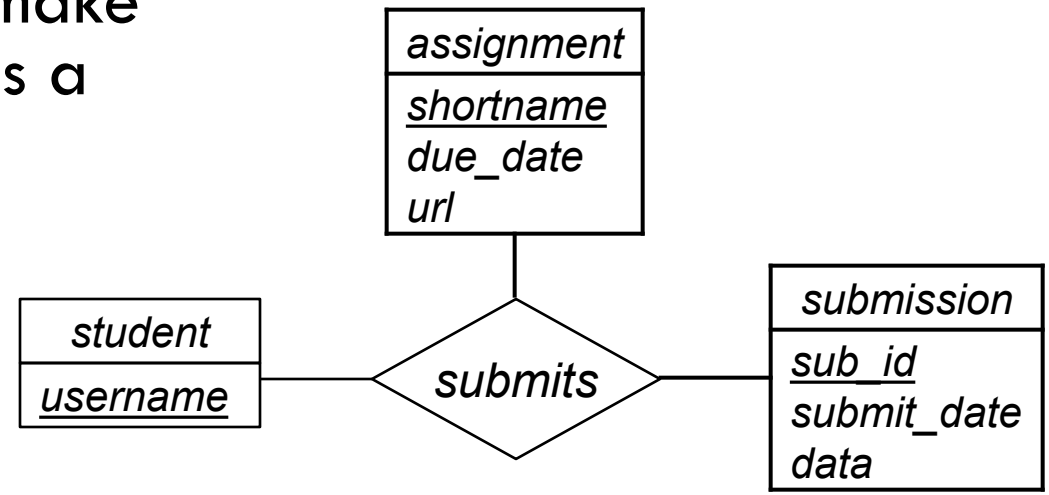


Binary vs. N-ary Relationships (3)

- Using binary relationships is sometimes more intuitive for particular designs
- Example: office-equipment inventory database
  - Ternary relationship-set *inventory*, associating *department*, *machine*, and *vendor* entity-sets
- What if vendor info is unknown for some machines?
  - For ternary relationship, must use *null* values to represent missing vendor details
  - With binary relationships, can simply not have a relationship between *machine* and *vendor*
- For cases like these, use binary relationships
  - If it makes sense to model as separate binary relationships, do it that way!

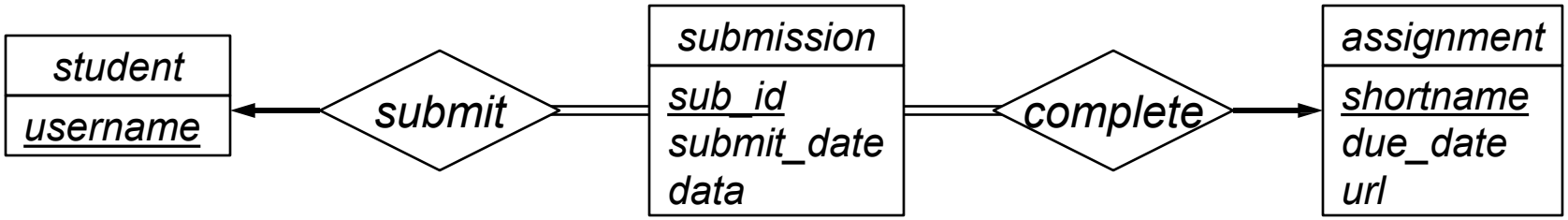
Course Database Example

- What about this case:
  - Ternary relationship between *student*, *assignment*, and *submission*
  - Need to allow multiple submissions for a particular assignment, from a particular student
- In this case, it could make sense to represent as a ternary relationship
  - Doesn't make sense to have only two of these three entities in a relationship



Course Database Example (2)

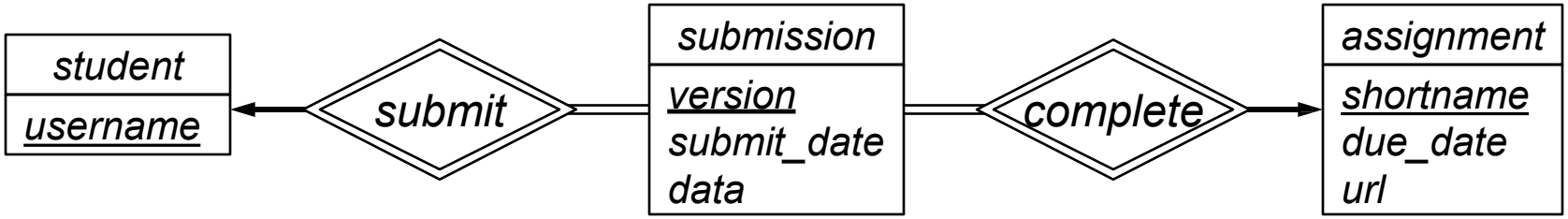
- Other ways to represent students, assignments and submissions?
- Can also represent as two binary relationships



- Note the total participation constraints!
  - Required to ensure that every *submission* has an associated *student*, and an associated *assignment*
  - Also, two one-to-many constraints

Course Database Example (3)

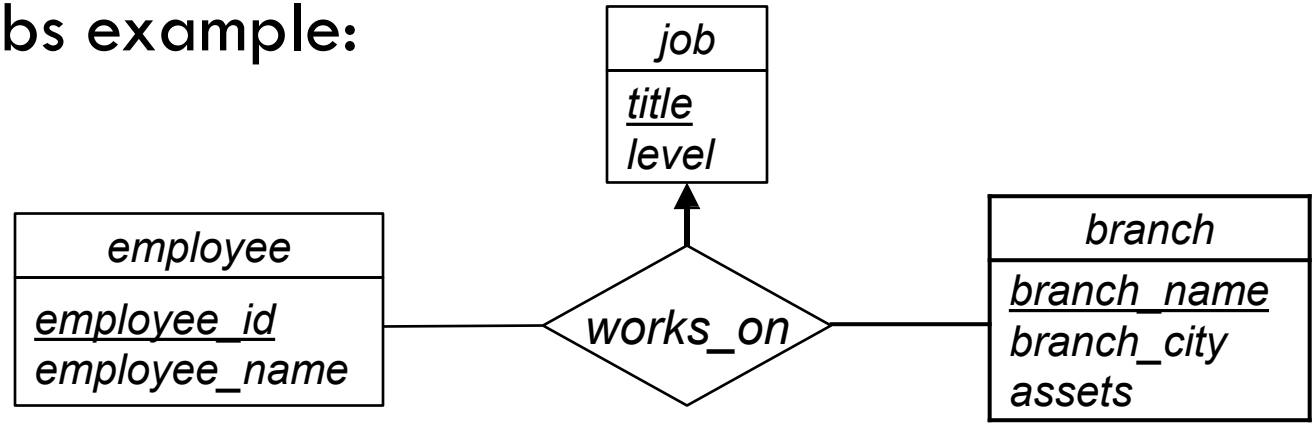
- Could even make *submission* a weak entity-set
  - Both *student* and *assignment* are identifying entities!



- Discriminator for *submission* is version number
- Primary key for *submission* ?
  - Union of primary keys from all owner entity-sets, plus discriminator
  - (*username*, *shortname*, *version*)

Binary vs. N-ary Relationships

- Sometimes ternary relationships are best
  - Clearly indicates all entities involved in relationship
  - Only way to represent certain constraints!
- Bank jobs example:



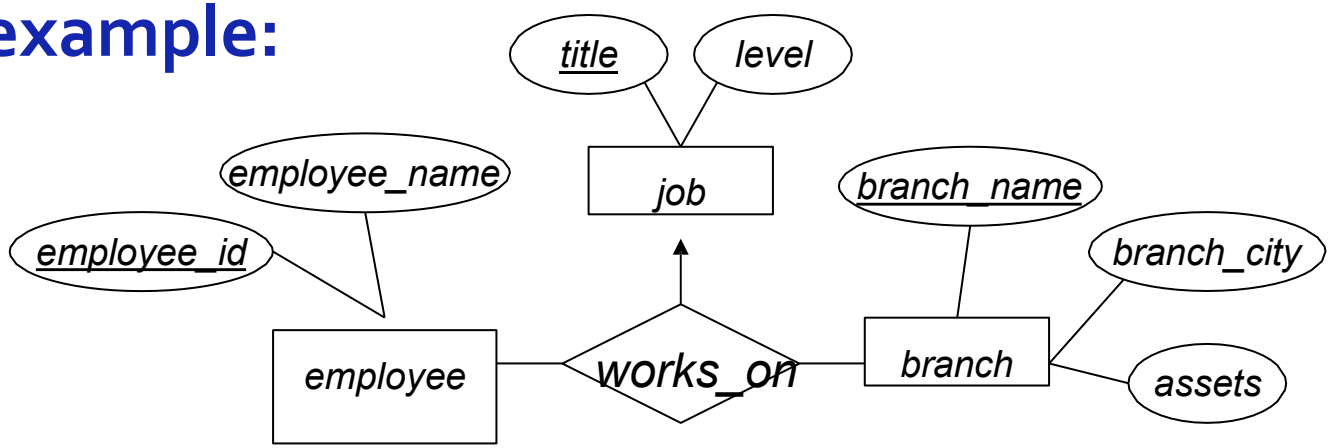
  - Each (*employee*, *branch*) pair can have only one job
  - Simply cannot construct the same constraint using only binary relationships
    - *(Reason is related to issue identified on slide 8)*

Binary vs. N-ary Relationships (2)

- Using binary relationships is sometimes more intuitive for particular designs
- Example: office-equipment inventory database
  - Ternary relationship-set *inventory*, associating *department*, *machine*, and *vendor* entity-sets
  - For ternary relationship, must use *null* values to represent missing vendor details
  - With binary relationships, can simply not have a relationship between *machine* and *vendor*
- For cases like these, use binary relationships
  - If it makes sense to model as separate binary relationships, do it that way!

Binary vs. N-ary Relationships

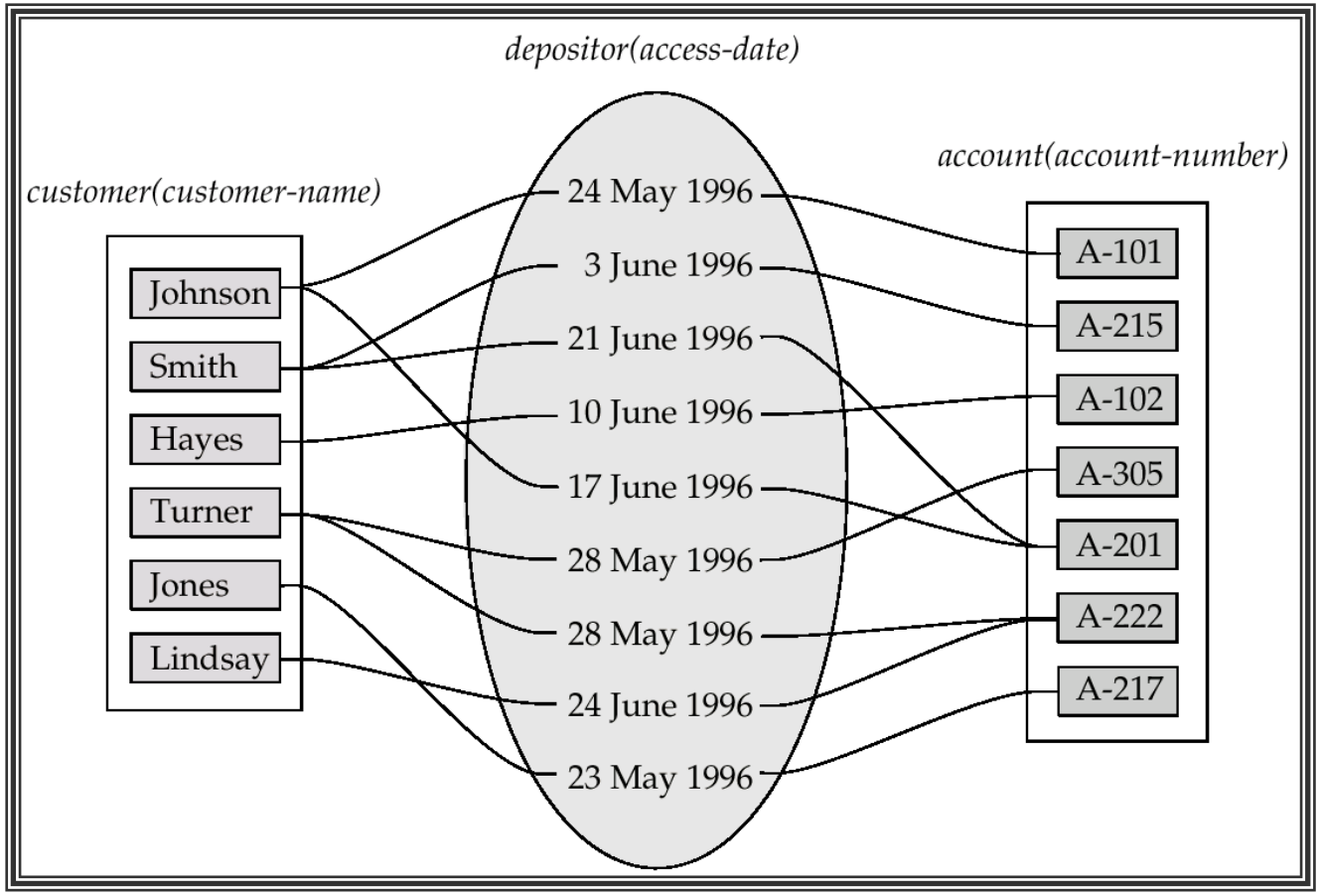
- Sometimes ternary relationships are best
  - Clearly indicates all entities involved in relationship
  - Only way to represent certain constraints!
- Bank jobs example:



  - Each (*employee*, *branch*) pair can have only one job
  - Simply cannot construct same constraint using only binary relationships

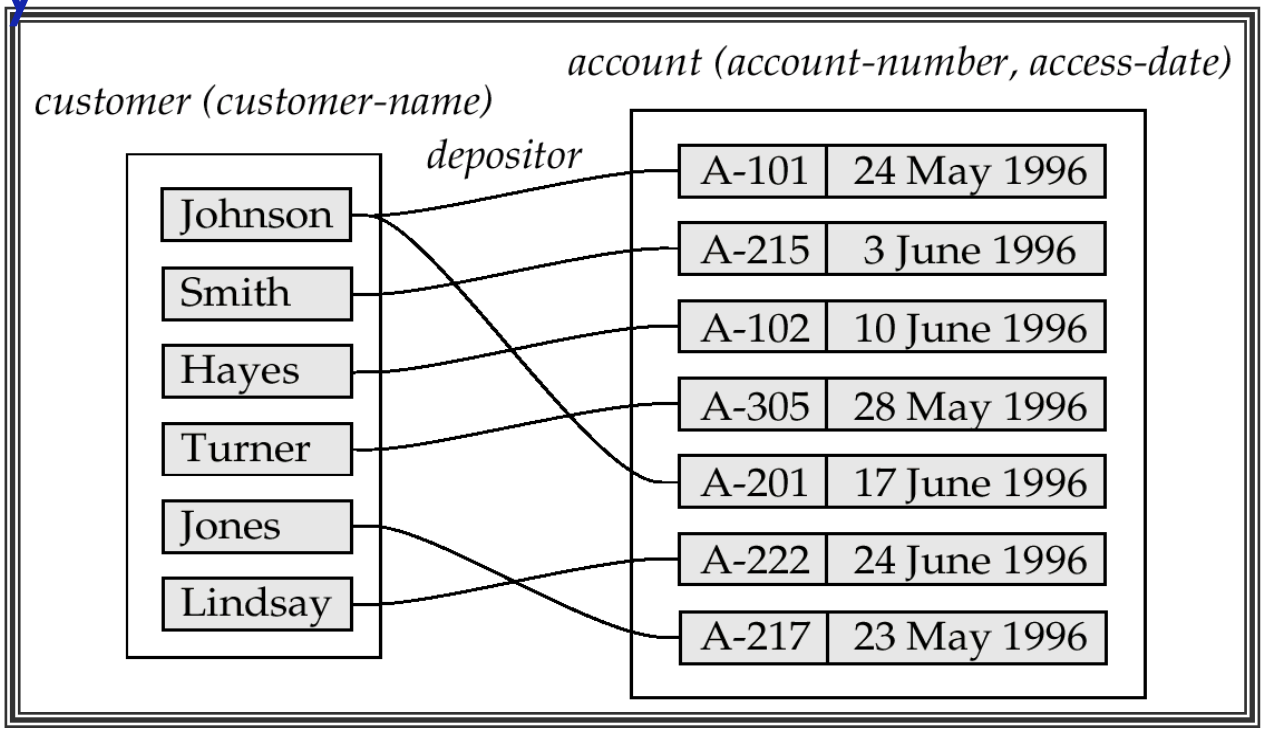
4. Placement of relationship attributes

- 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*



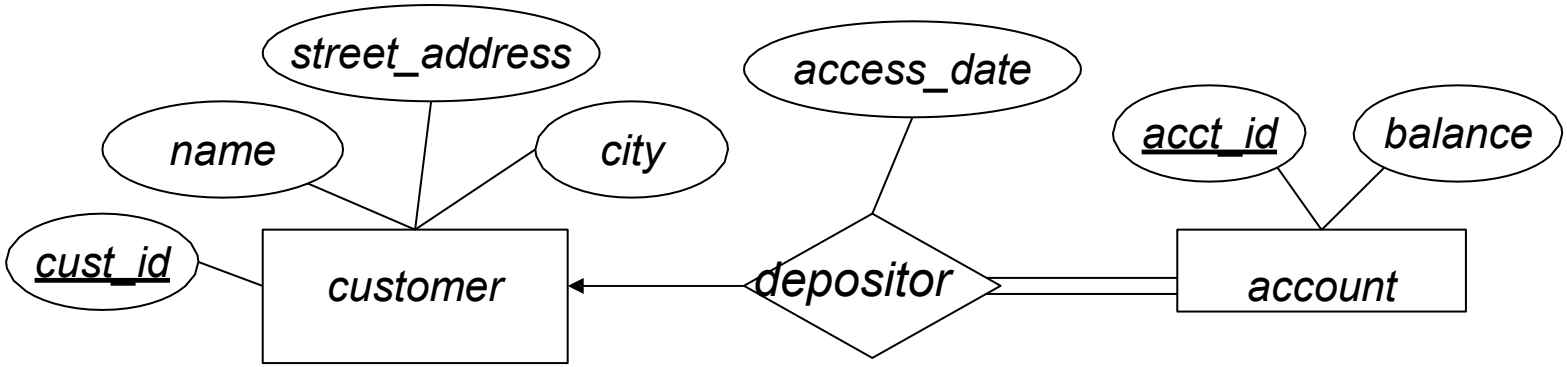
Mapping Cardinalities affect ER Design

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



Placement of Attributes

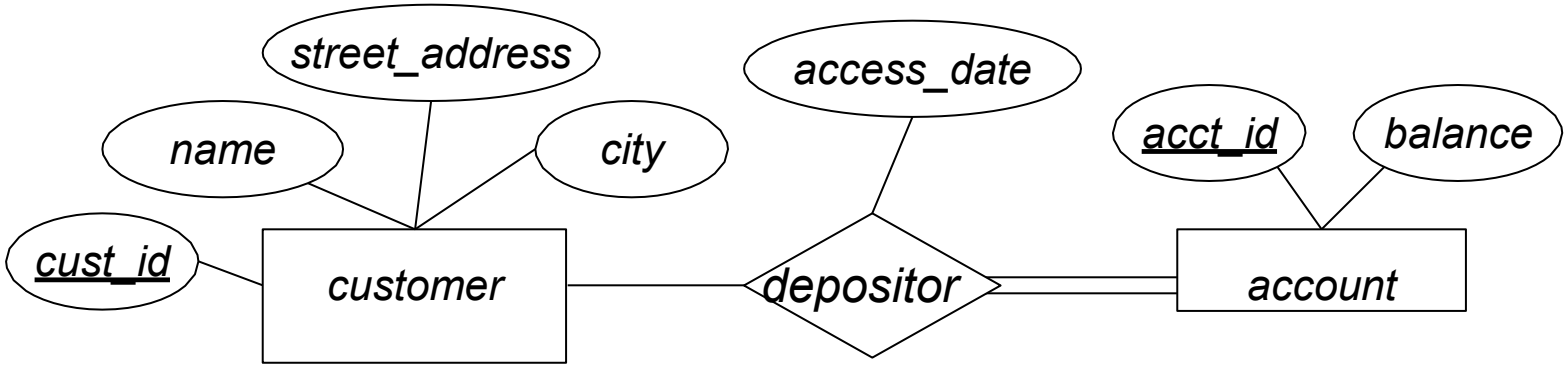
- Relationship-sets can have descriptive attributes
- For one-to-one or one-to-many relationship-sets:
  - Can associate descriptive attributes with one of the participating entity-sets, instead of the relationship-set
- Example: *customer*, *account*, and *depositor*



  - Would be identical to have *access_date* on *account*, since each account associated with just one customer

Placement of Attributes (3)

- If relationship-set is many-to-many, different placements have different implications!
- With banking example again:



  - Now an account can have multipleowners
- What if *access_date* were on *account* entity-set?
  - Couldn't tell *which customer* made last access!
  - Actually affects what schema can represent.

Placement of Attributes (4)

- For one-to-one and one-to-many relationships:
  - Flexibility in where descriptive attributes are place
  - Can place descriptive attribute on "many" side of relationship-set
  - Choose option that best models the enterprise
- For many-to-many relationships:
  - Different options have different implications
  - If attribute value should be associated with the combination of entities, must appear on relationship
  - If attribute value relates only to a particular entity, move it to that entity-set

Thank You!!!