MDA104 Introduction to Databases

2. Entity-Relationship Model

Vlastislav Dohnal

Credits

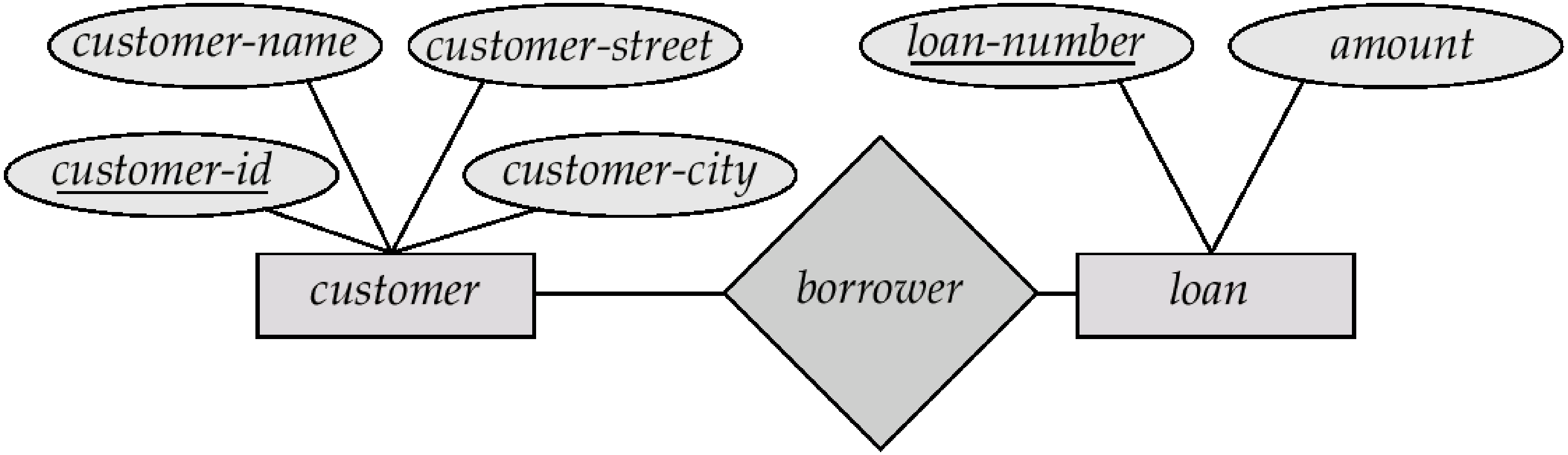
- Slides are part of the database bible:
  - Database System Concepts, Seventh Edition. Avi Silberschatz, Henry F. Korth, S. Sudarshan.
  - https://db-book.com/
- Experience from courses of Faculty of Informatics, Masaryk University
  - PB168 - Fundamentals of Database and Information Systems
  - PB154 - Fundamentals of Database Systems

Entity-Relationship Model

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

Entity-Relationship model

- Conceptual model used in the development of IS
  - During requirements analysis
  - Models *information* stored in the DB
- Easy to understand
  - The customer "understands" it

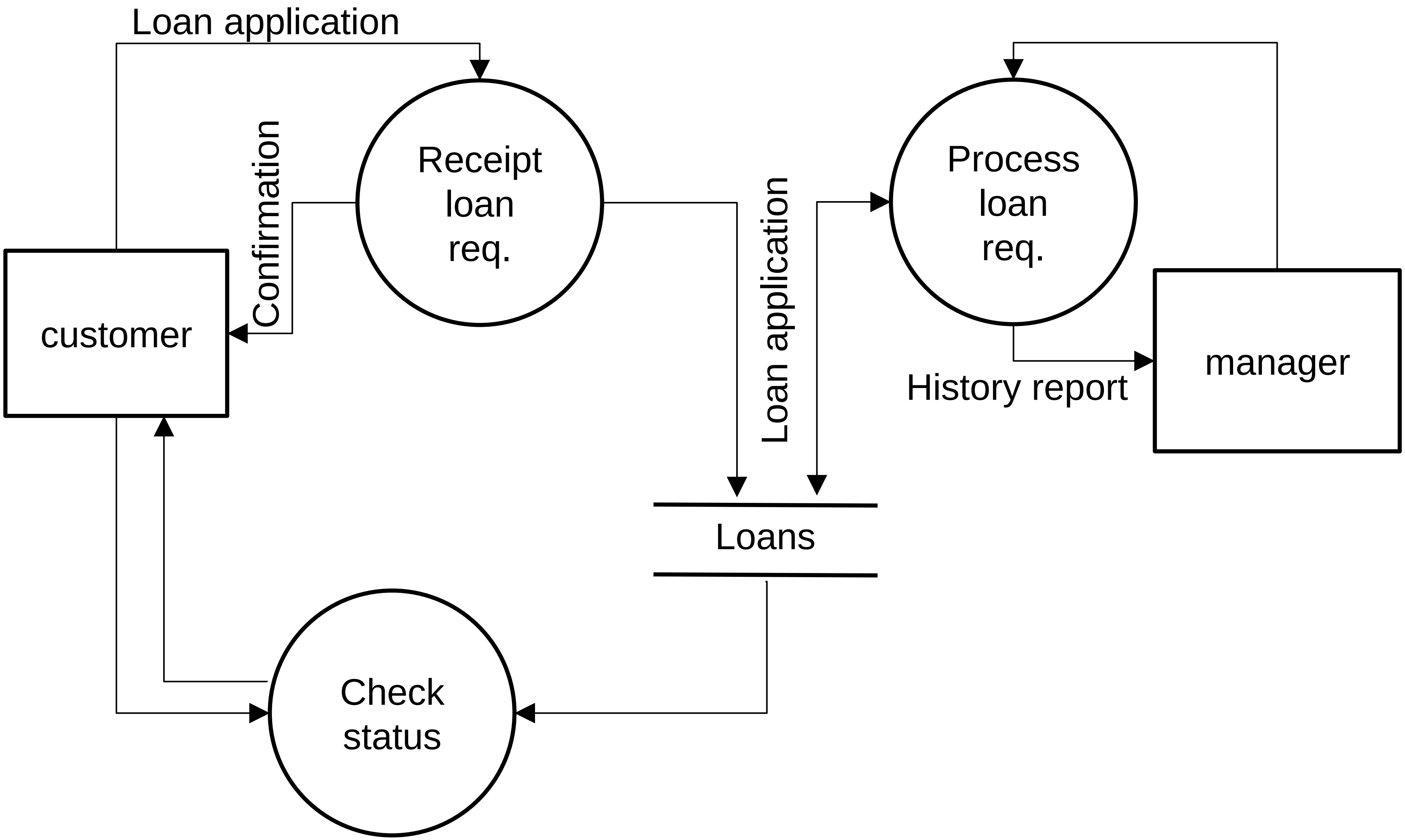


Example – Loan in a bank

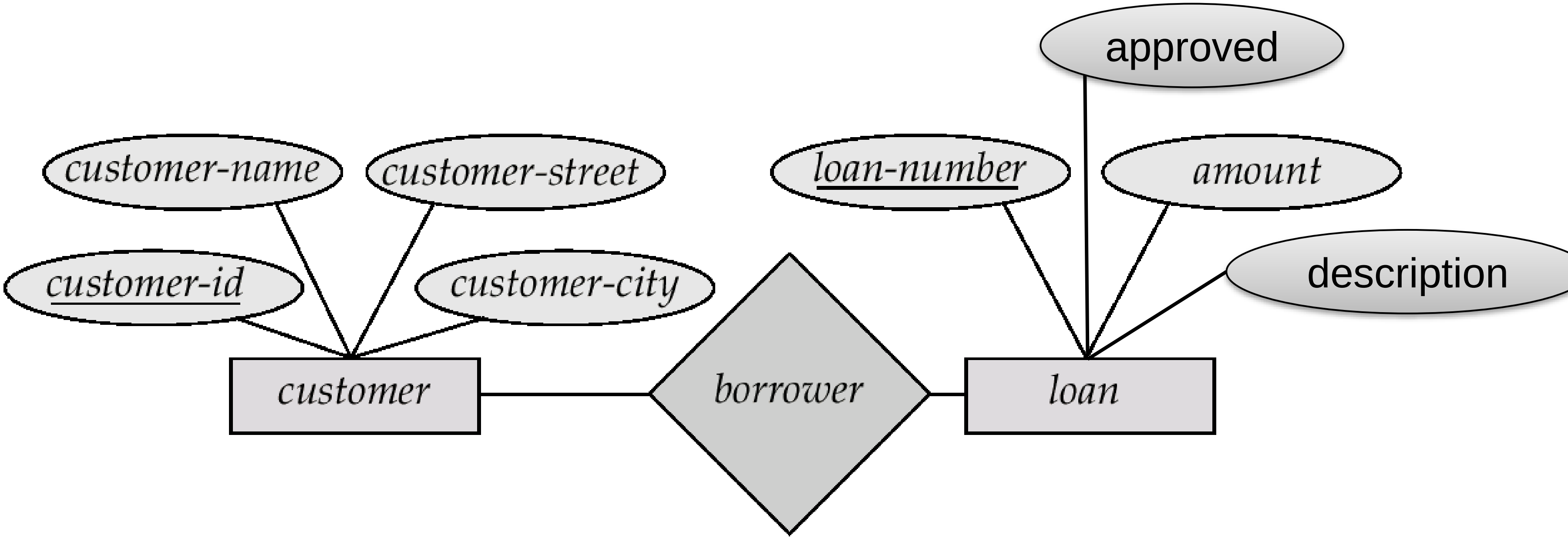
- Requirements
  - A client applies for a loan
    - purpose, how much, information about the client
  - The bank approves the loan
- Decision
  - What is data and what are processes?

- ERD for data
- DFD (Data-flow diagram) for processes

DFD – Loan in a bank



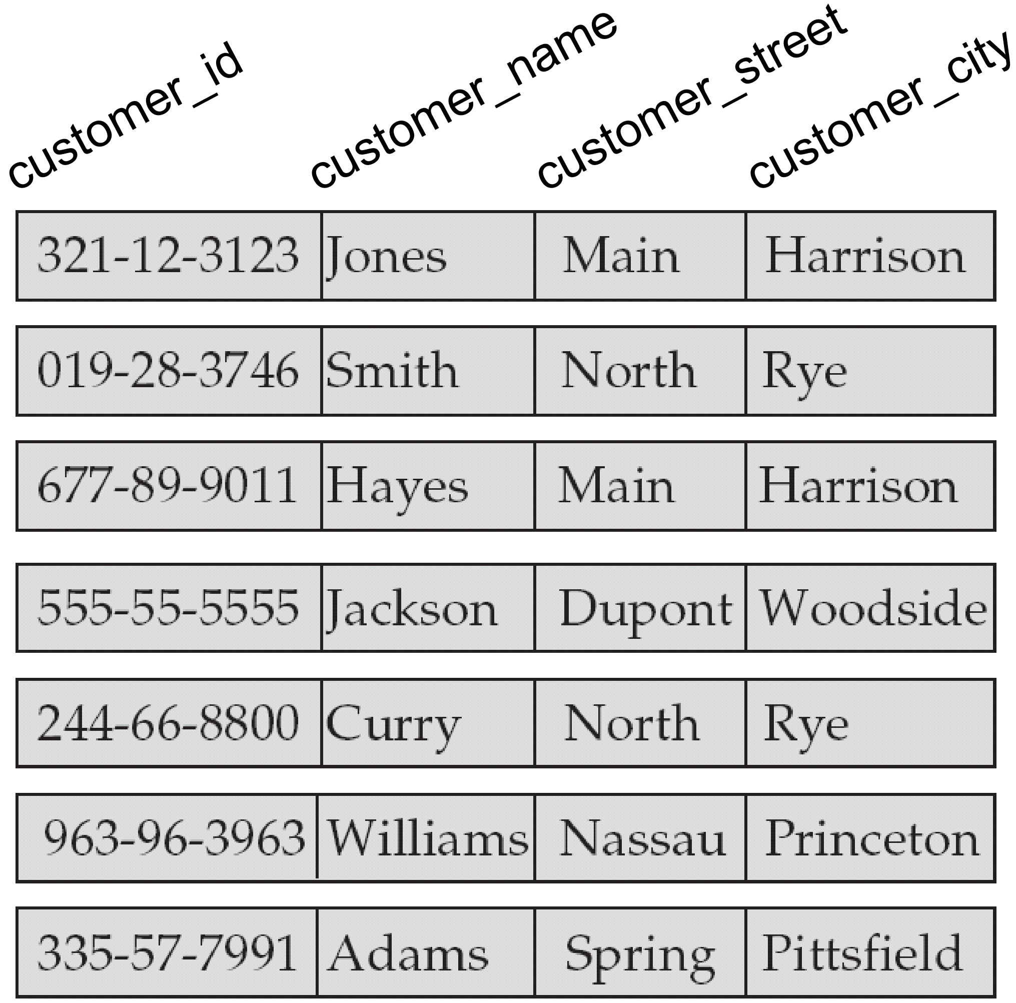
ERD – Loan in a bank



Modeling

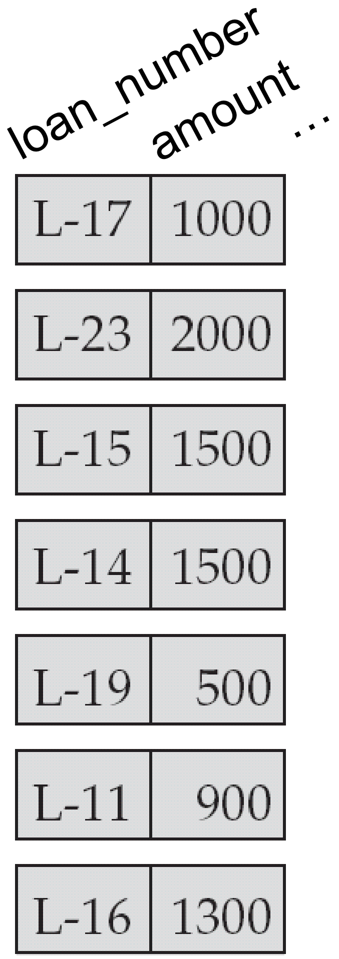
- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - Also, an entity must be "remembered"
  - Example: specific person, company, plant, event, product, invoice
- Entities 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	*borrower*	A-102
customer entity	relationship set	*loan* entity

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

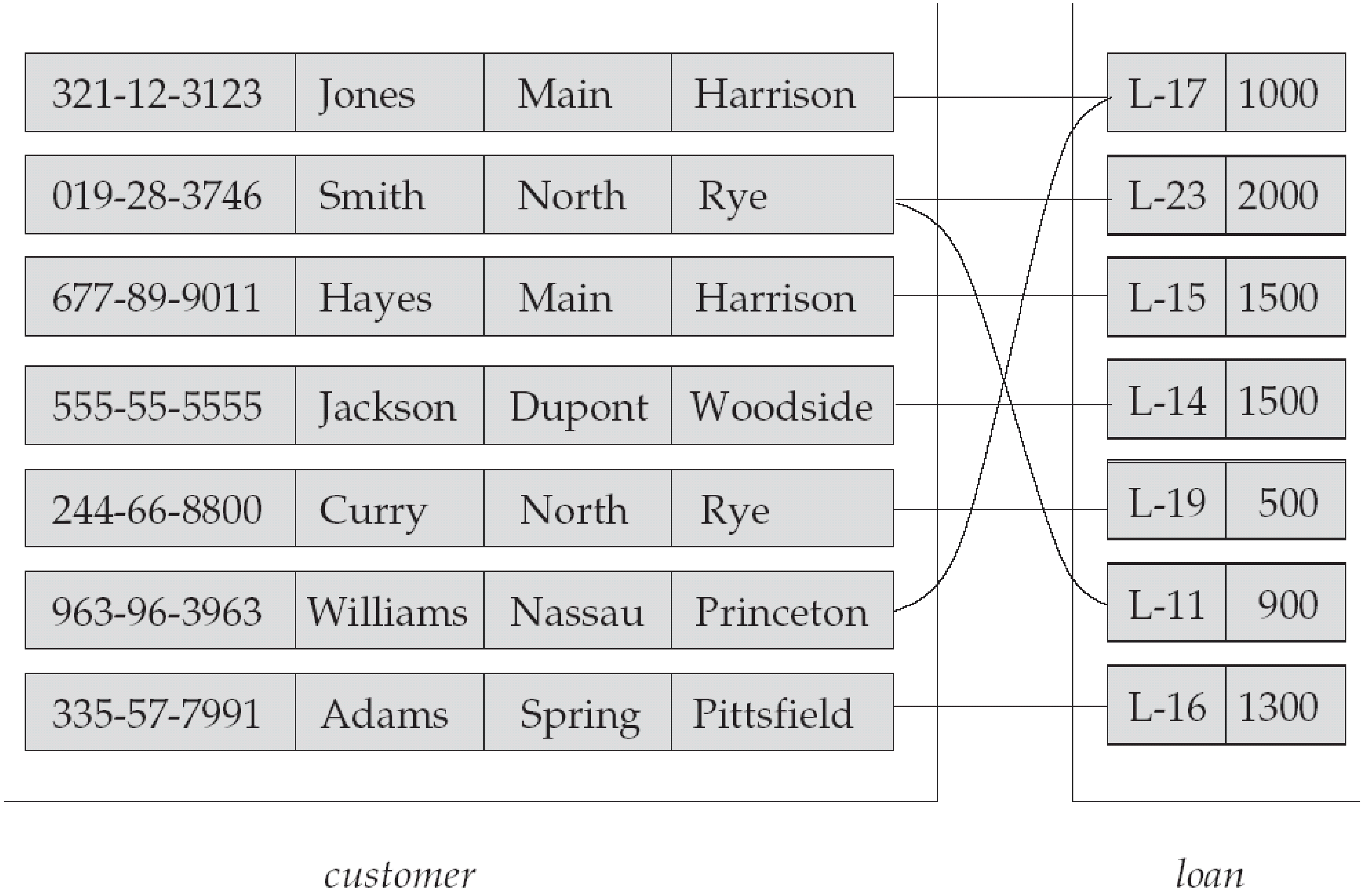
  $$\mathrm{R} = \{(e_1, e_2, \ldots\, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

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

Relationship Set *borrower*



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

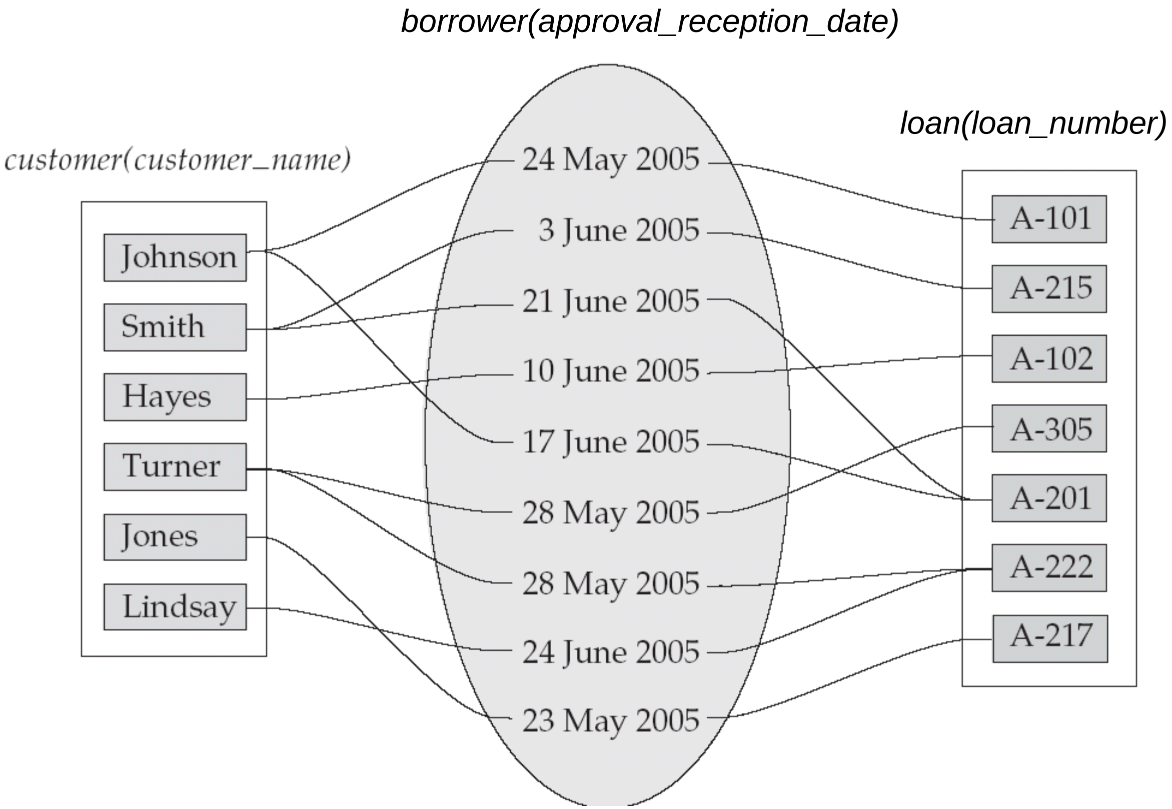
customer

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

loan

Relationship Sets (Cont.)

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



Degree of a Relationship Set

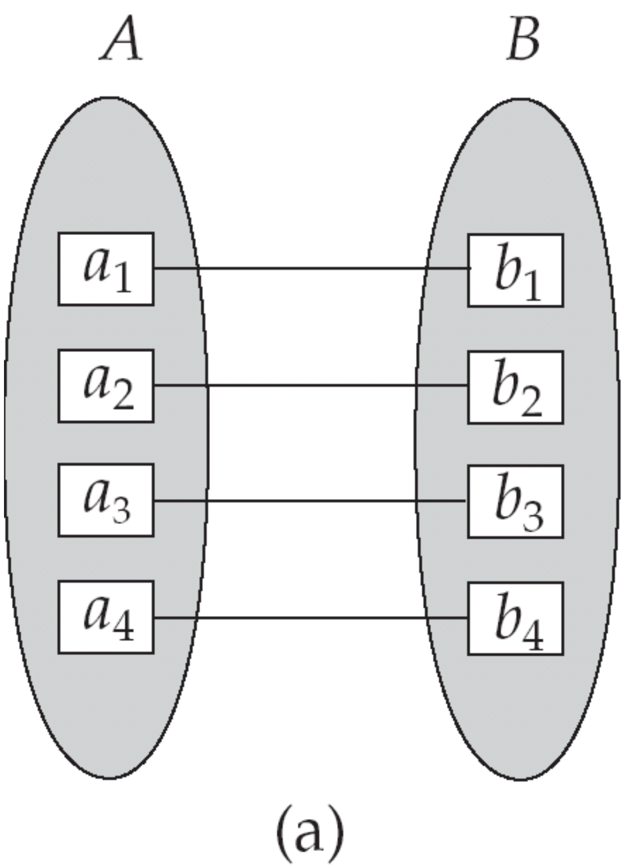
- Refers to the number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are **binary**
  - Degree = two
  - Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets.
  - Example:
    - 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.
  - Again, most relationships are binary. (More on this later.)

Mapping Cardinality Constraints

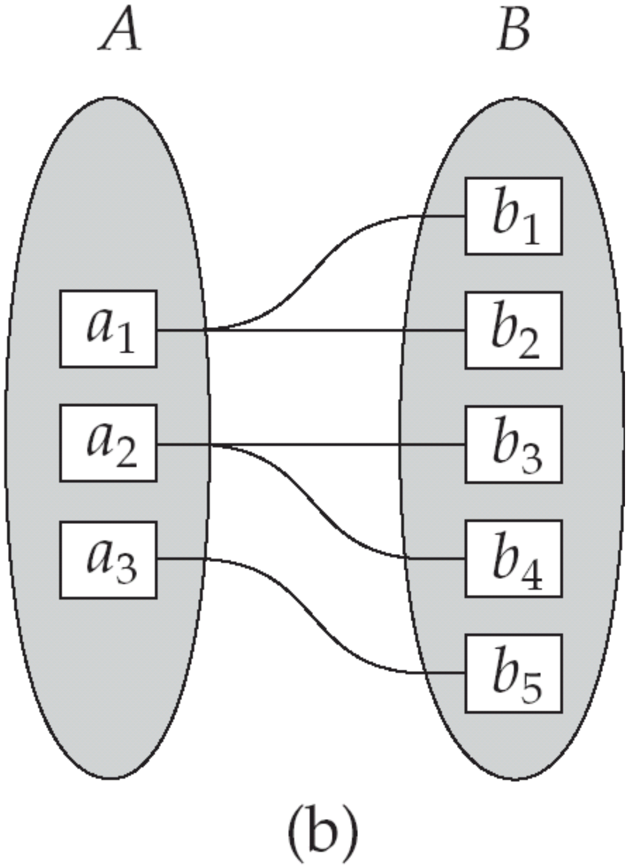
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many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

- Mind that cardinality itself does not enforce the existence of a “mapping”, i.e., a customer may not have any loan.

Mapping Cardinalities



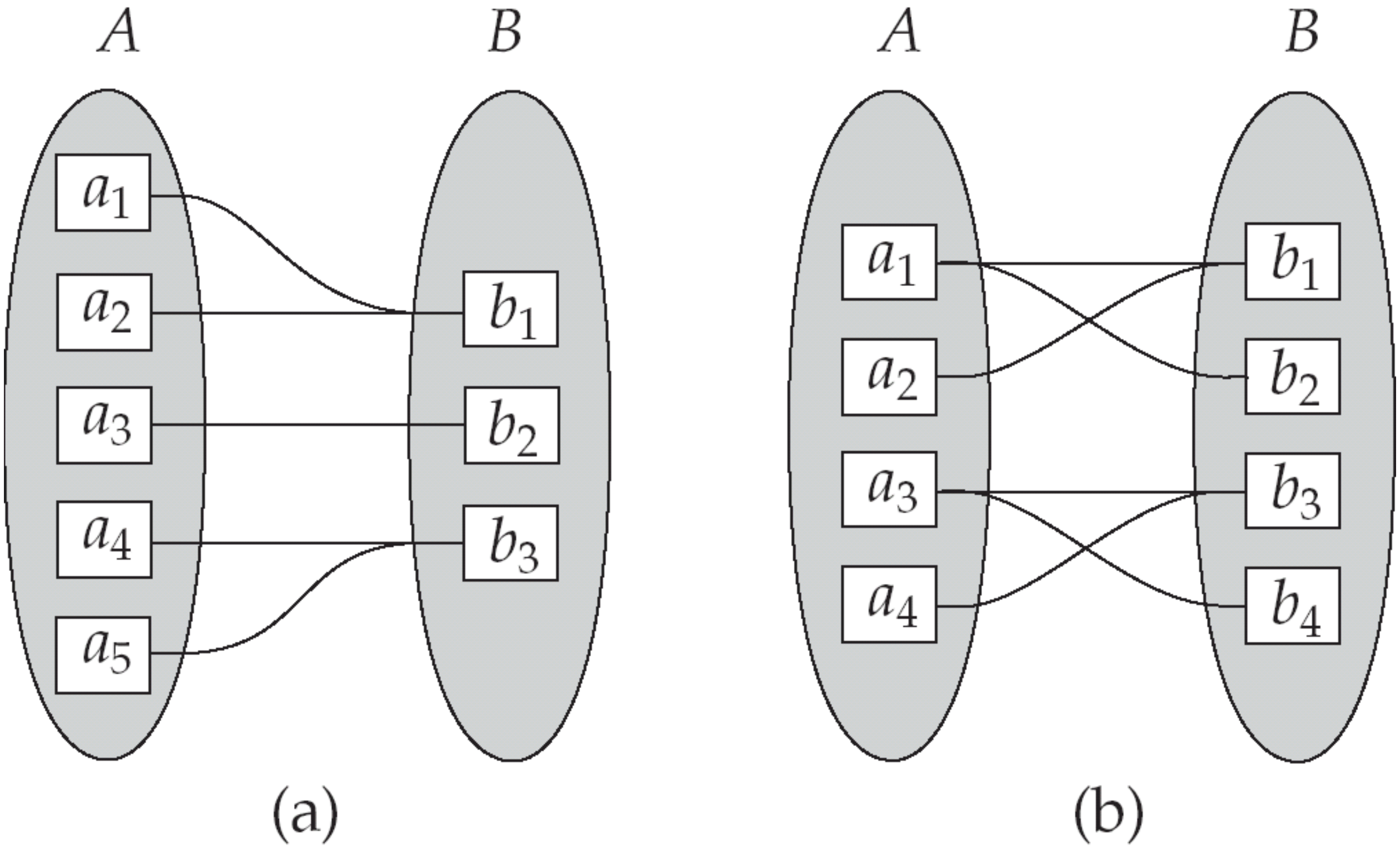
One to one



One to many

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

Mapping Cardinalities

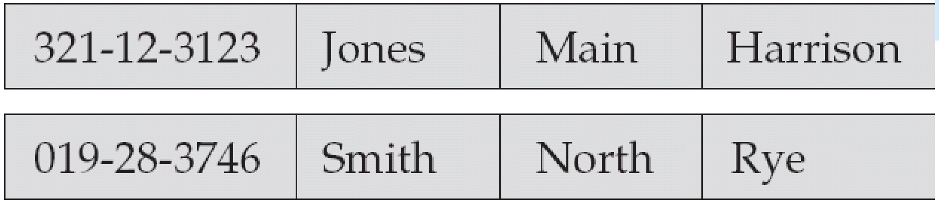


Many to one

Many to many

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

Attributes



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

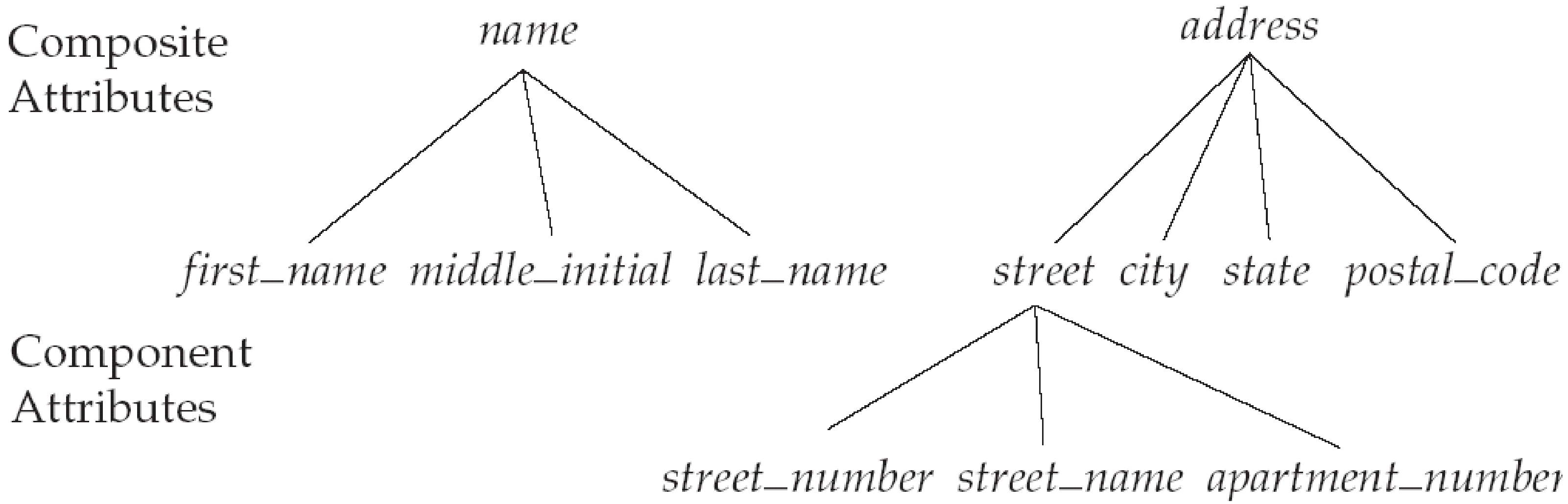
- An entity is represented by a set of attributes
  - = descriptive properties possessed by all members of an entity set.

Example:

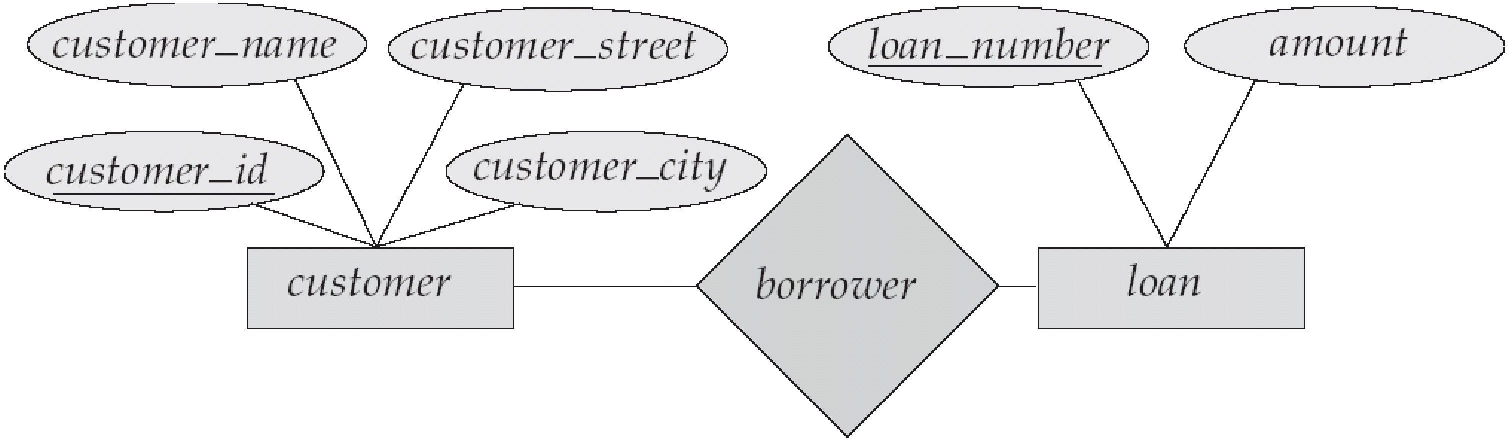
customer = (*customer_id, customer_name, customer_street, customer_city*)
loan = (*loan_number, amount*)

- **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

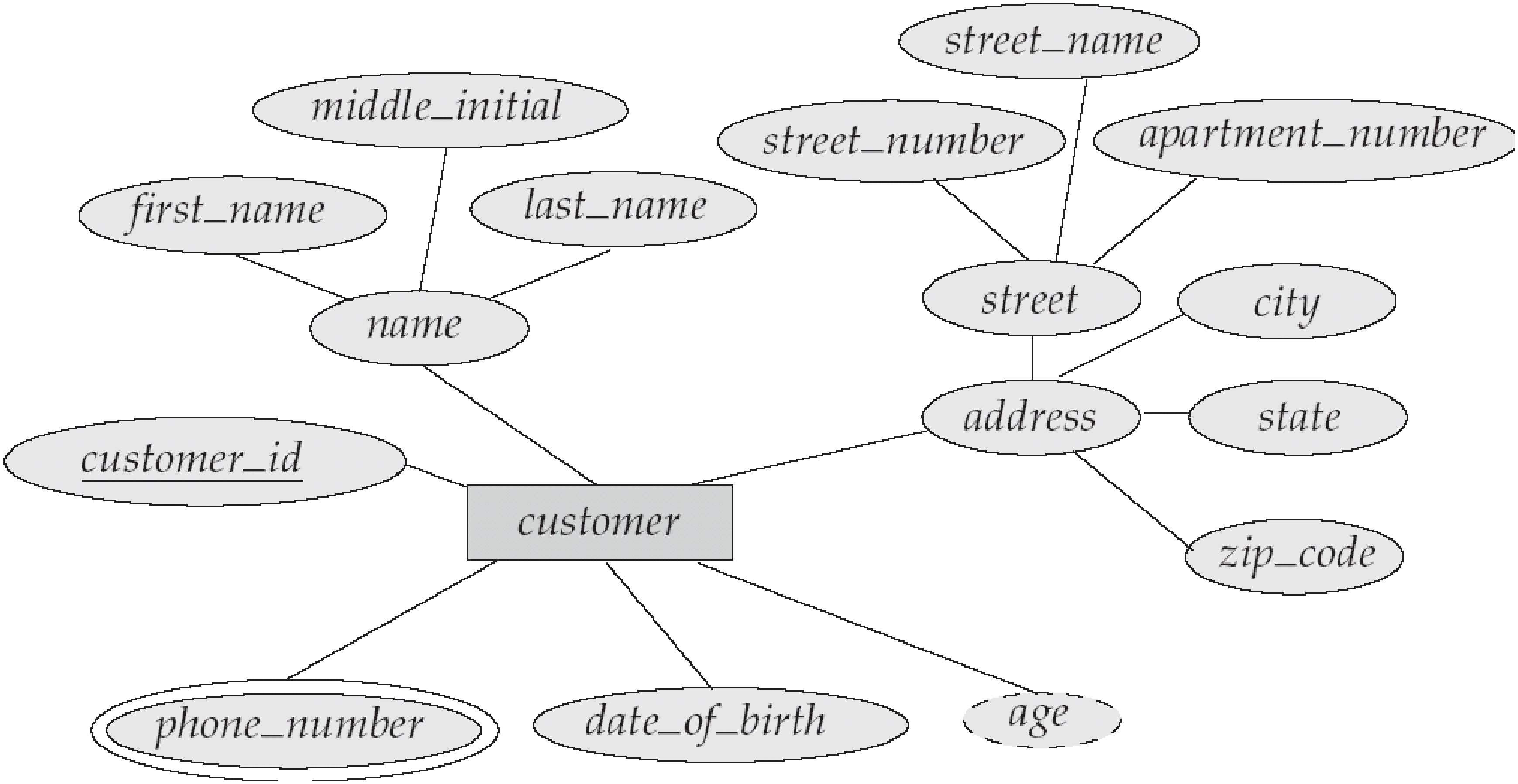


E-R Diagrams

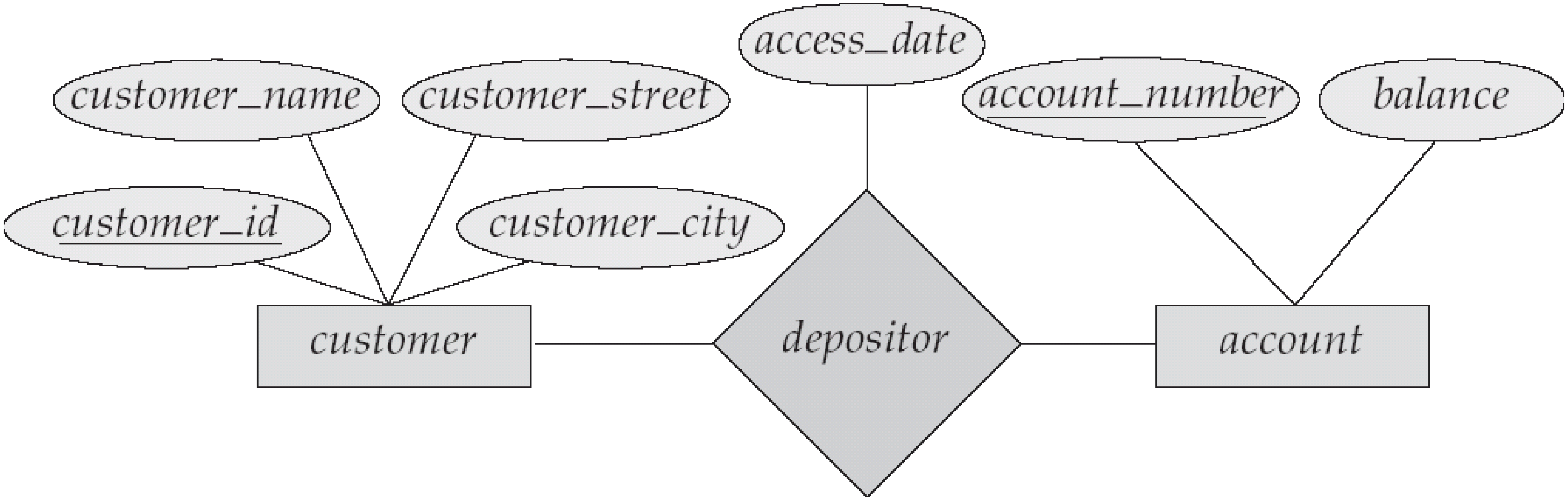


- *Rectangles* represent entity sets.
- *Diamonds* represent relationship sets.
- *Ellipses* represent attributes
- *Lines* link attributes to entity sets and entity sets to relationship sets.
- Attributes:
  - *Double ellipses* represent multivalued attributes.
  - *Dashed ellipses* denote derived attributes.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

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



Relationship Sets with Attributes

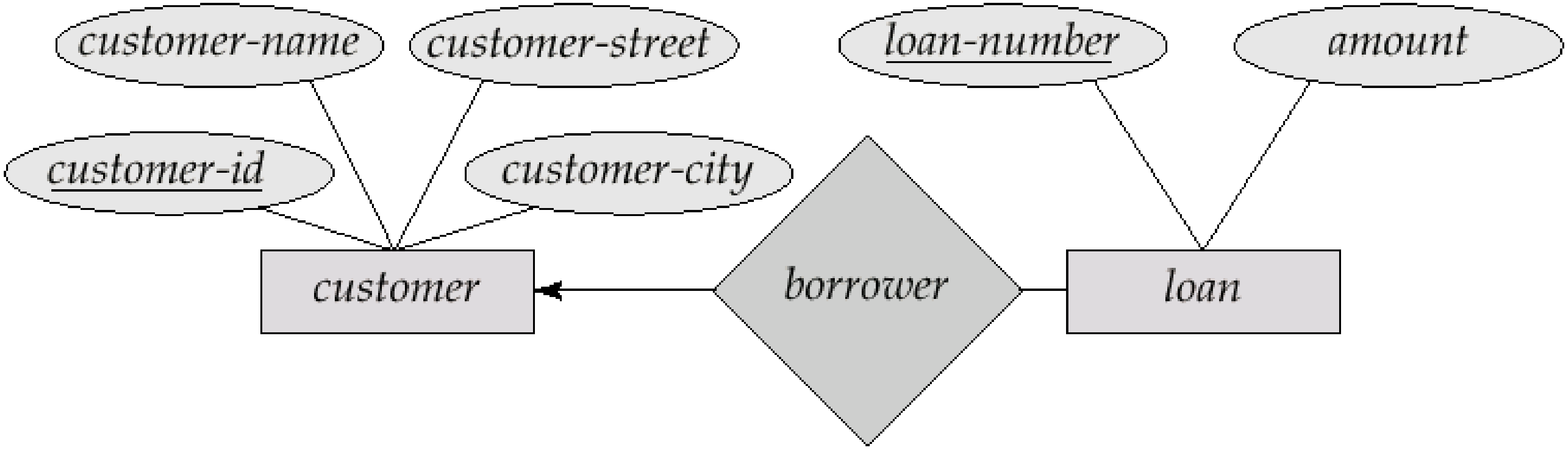


Mapping Cardinality Constraints

- 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*

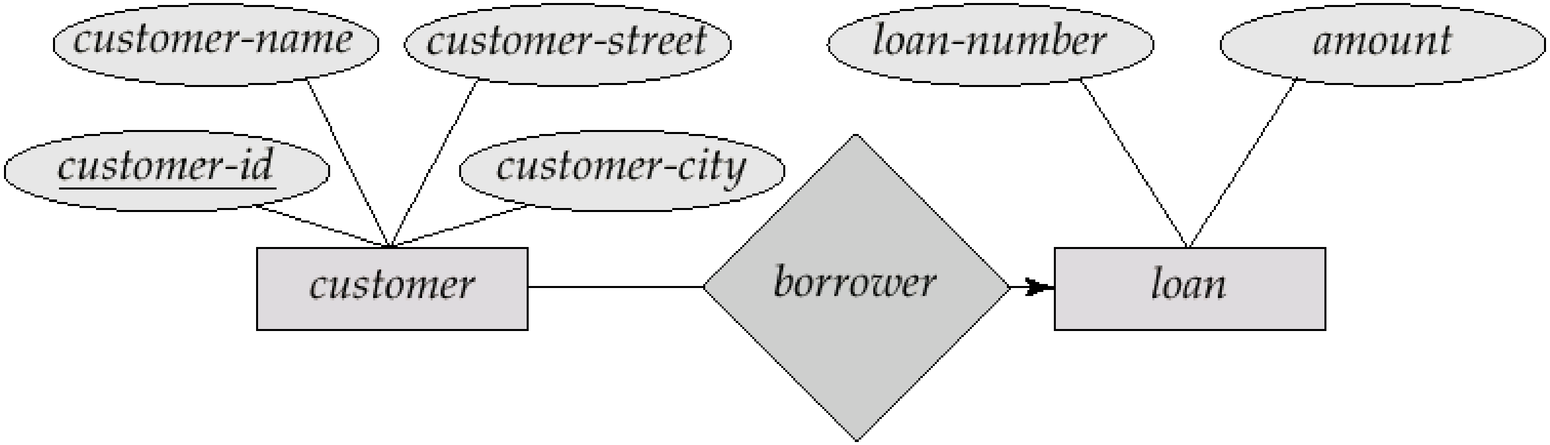
One-To-Many Relationship

- 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*



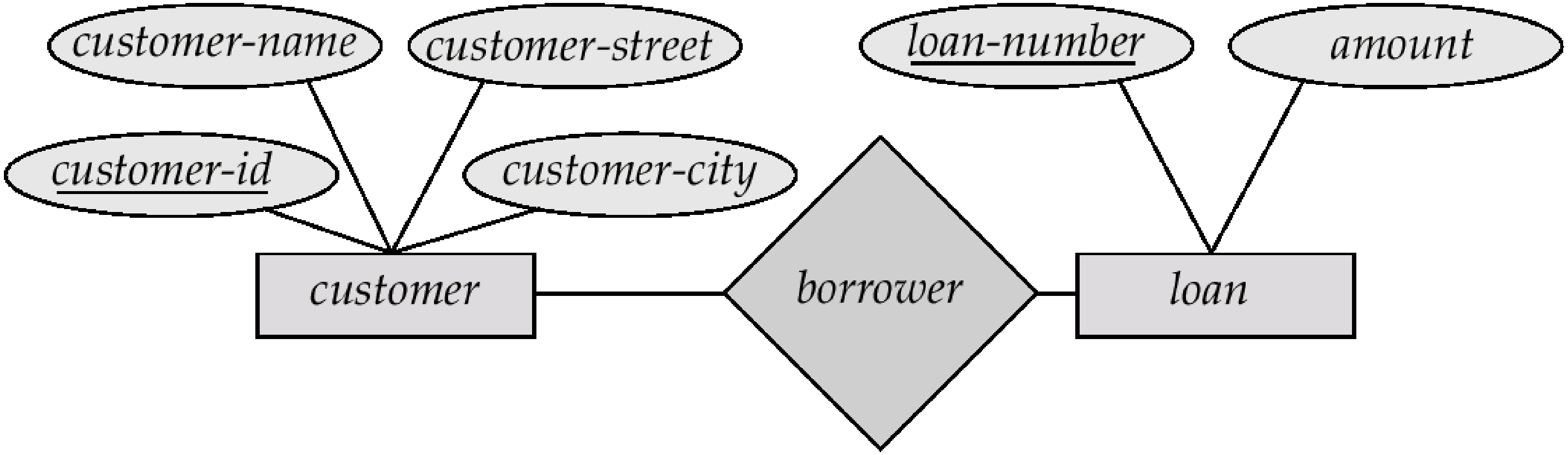
Many-To-One Relationships

- 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*



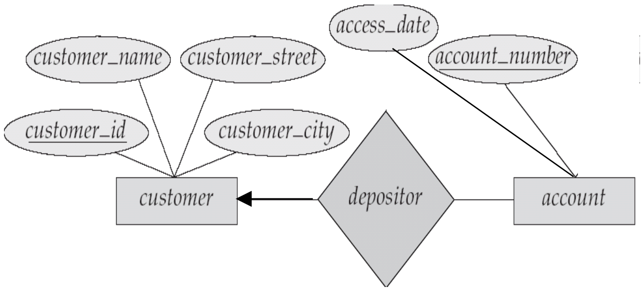
Many-To-Many Relationship

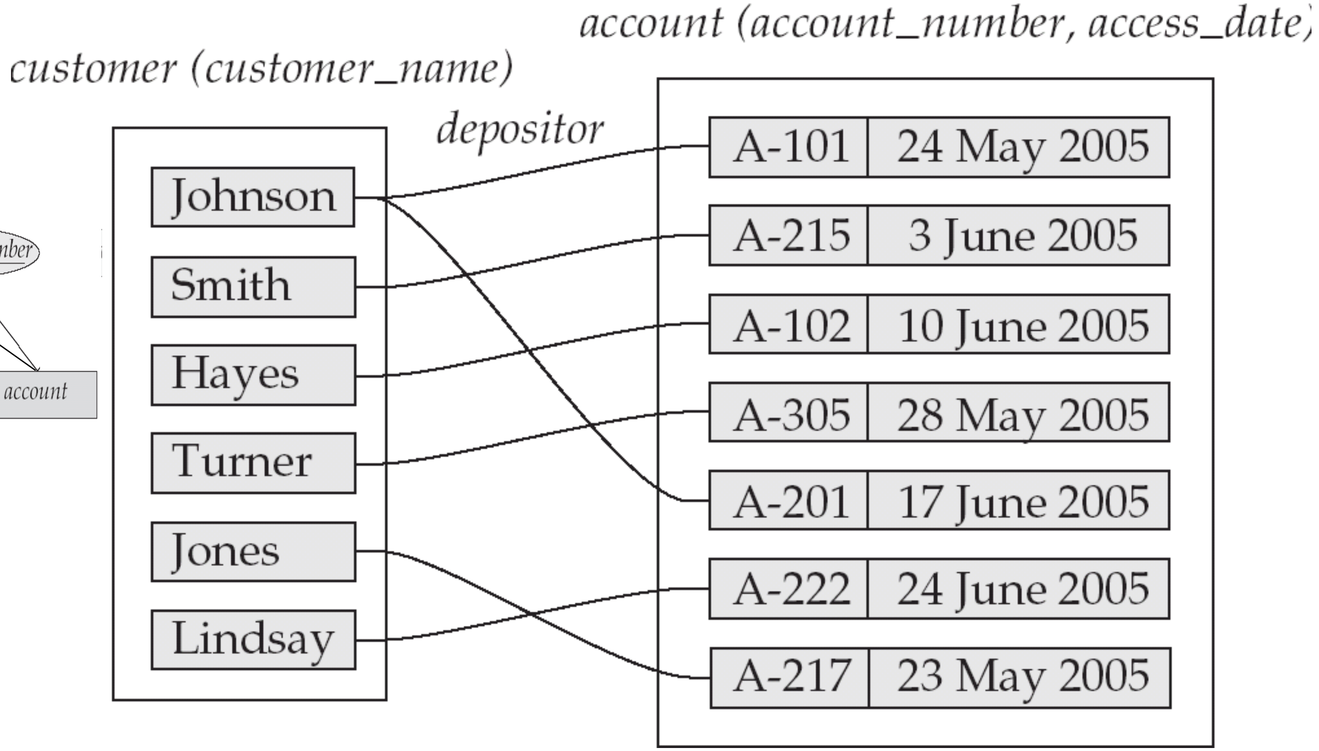
- A customer is associated with several (possibly zero) loans via borrower
- A loan is associated with several (possibly zero) customers via borrower



Mapping Cardinalities affect ER Design

- Especially, attributes of relationships
- 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





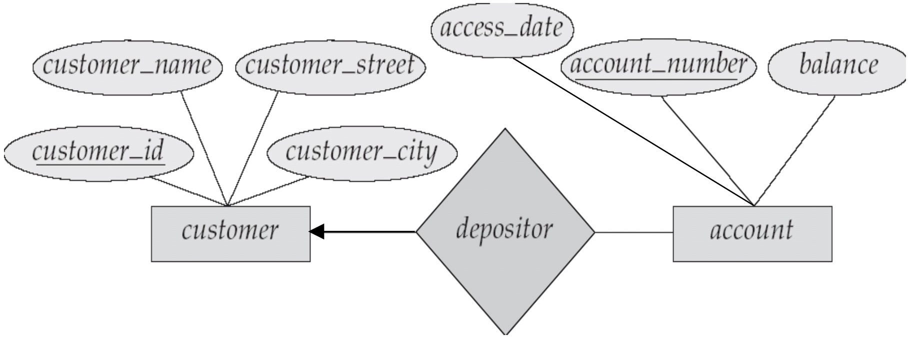
Keys

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

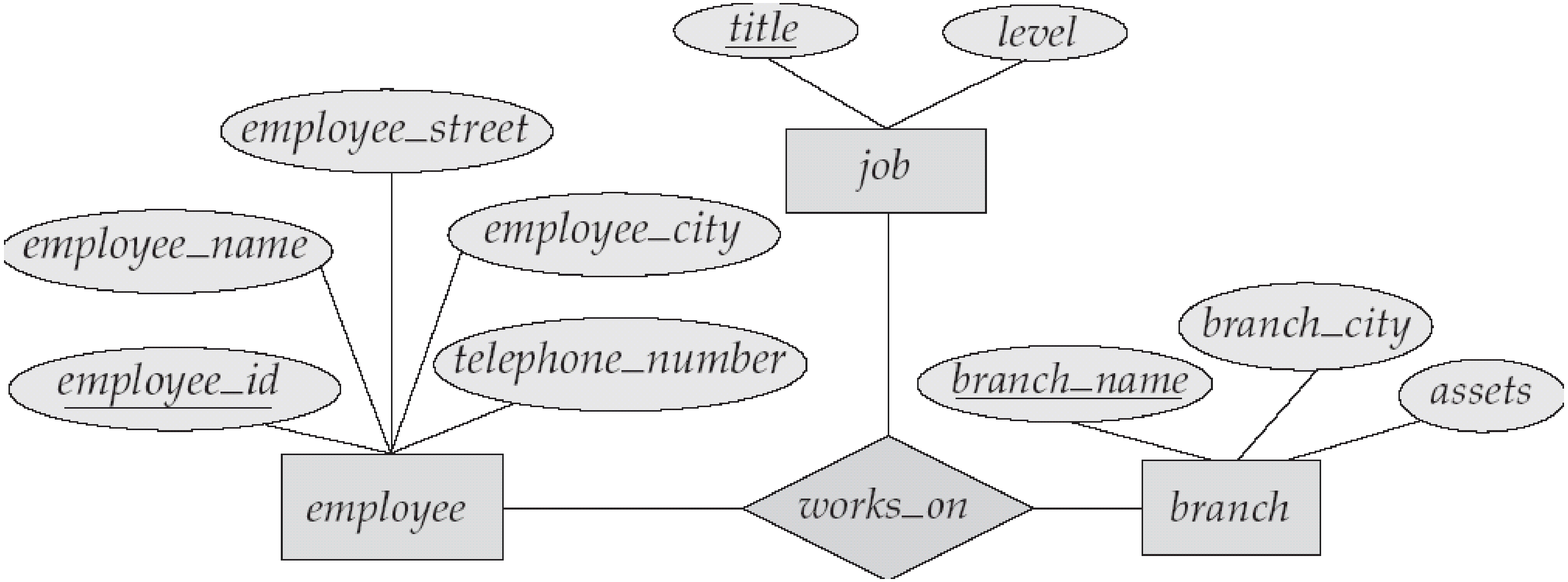
- “Database key” (primary key constraint)
  - Defined for unique identification of each entity and/or relationship
- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A **candidate key** of an entity set is a minimal super key
  - *customer_id* 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

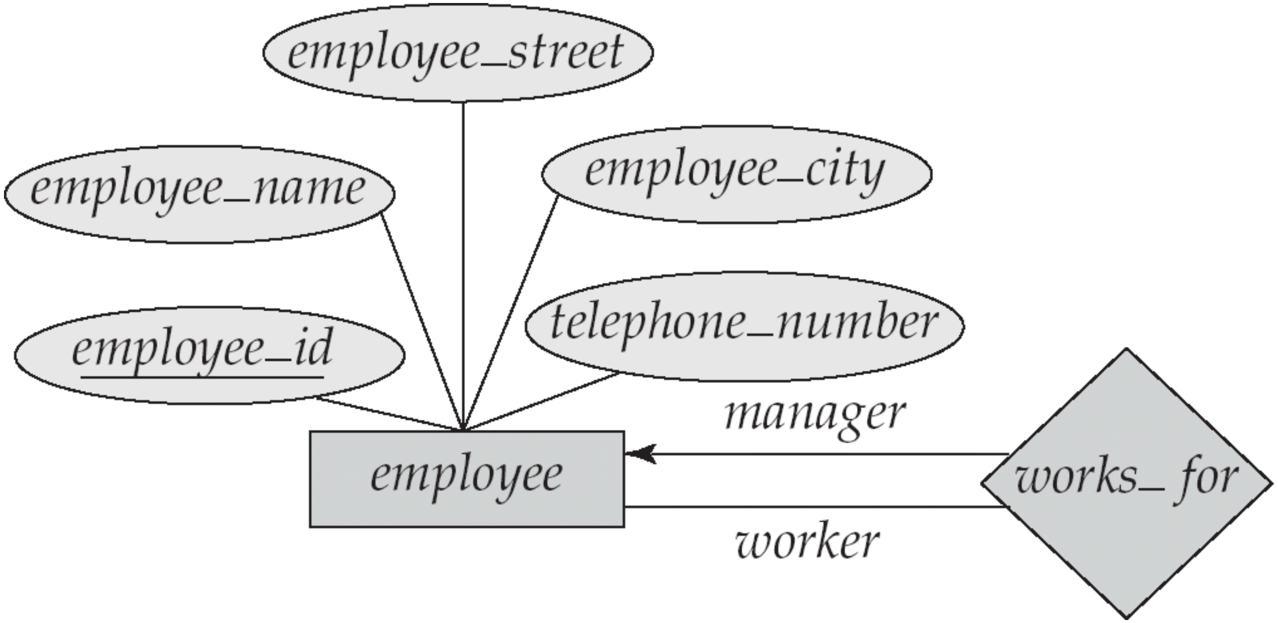


Cardinality Constraints on Ternary Relationship

- 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

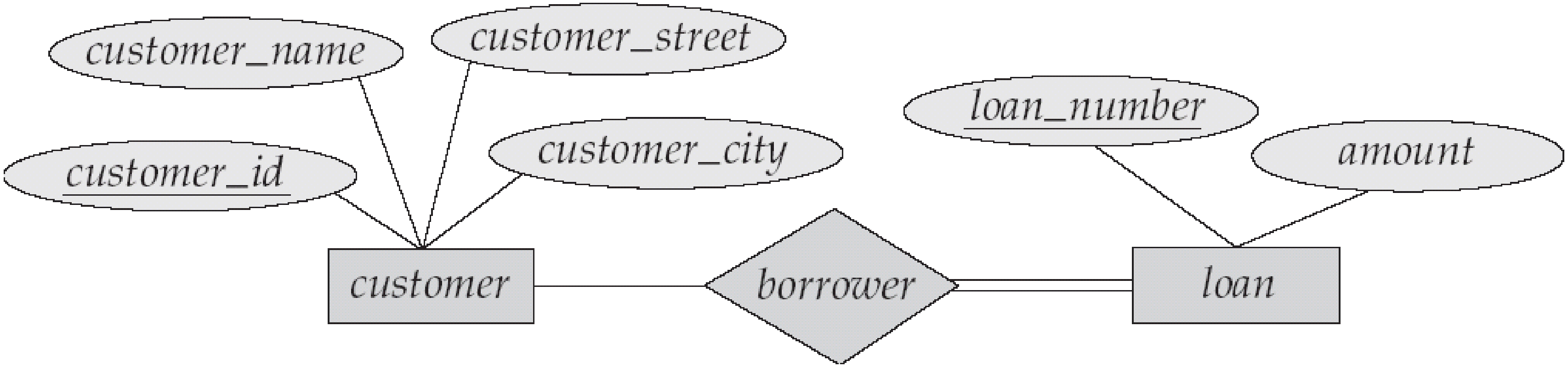
Roles

- 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



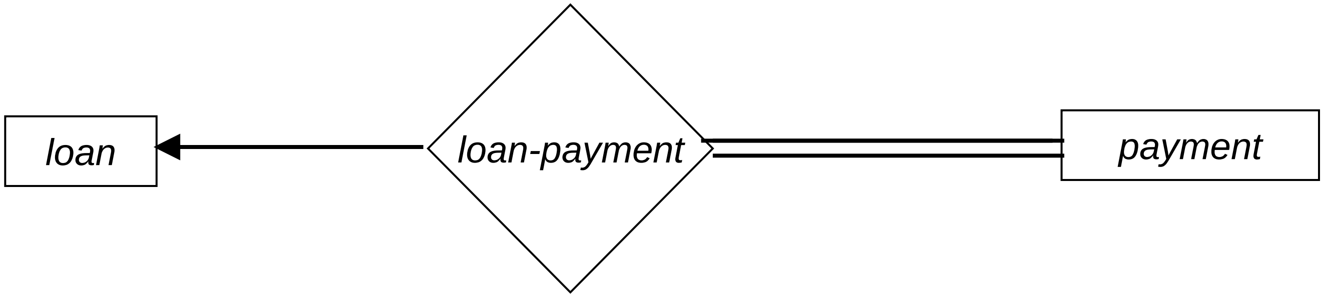
Participation of an Entity Set in a Relationship Set

- 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*)



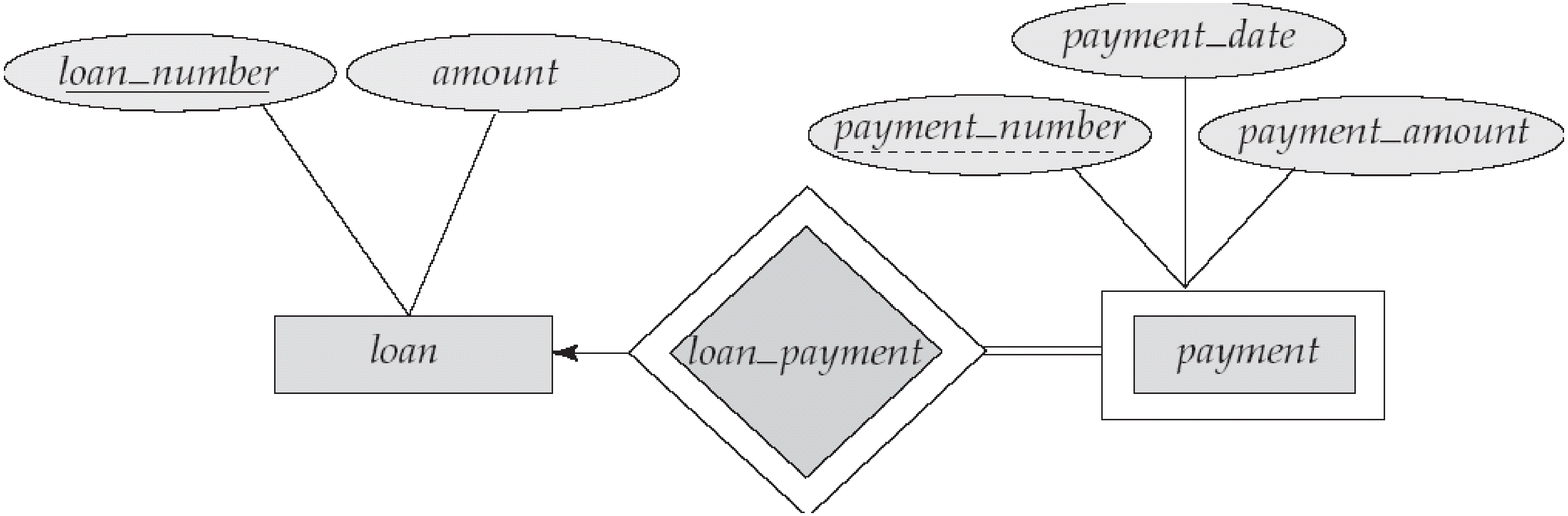
- If a *loan* entity is deleted, then all its associated *payment* entities must also be deleted.

Weak Entity Sets

- Models the existence dependency
- The existence of a weak entity set depends on the existence of an **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 set
  - **Identifying relationship** depicted using a double diamond
- Keys:
  - An entity set that does not have a primary key is referred to as a **weak entity set**.
  - The **discriminator** (*or partial key)* of a weak entity set is the key that distinguishes among all the weak entities corresponding to a specific identifying entity.
  - 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
  - So, it can represent the order of individual payments of a loan.
- Primary key for *payment* is (*loan_number, payment_number*)

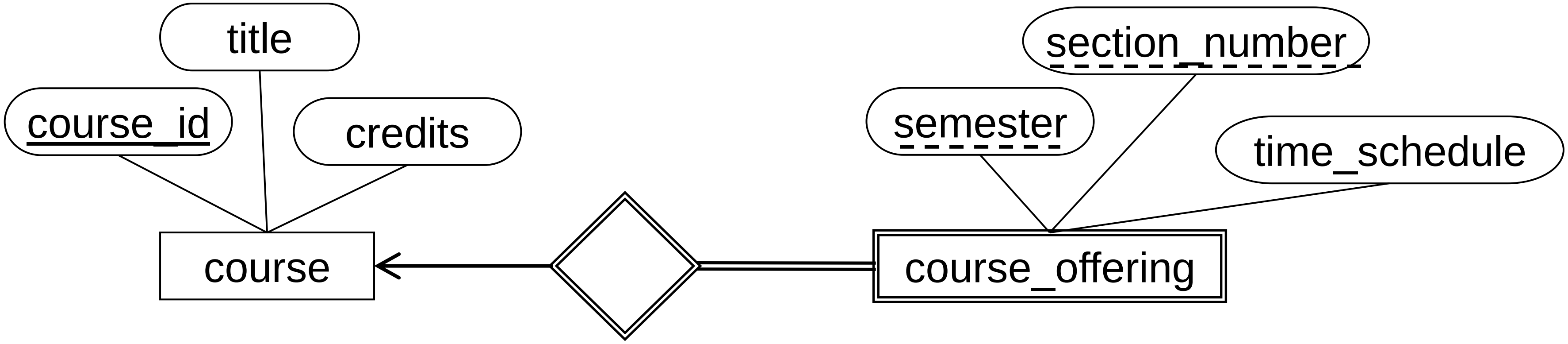


Weak Entity Sets (Cont.)

- Note: the primary key of the strong entity set is not explicitly added to the weak entity set, since it is implicit via the identifying relationship.
- If *loan_number* was 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*

More Weak Entity Set Examples

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



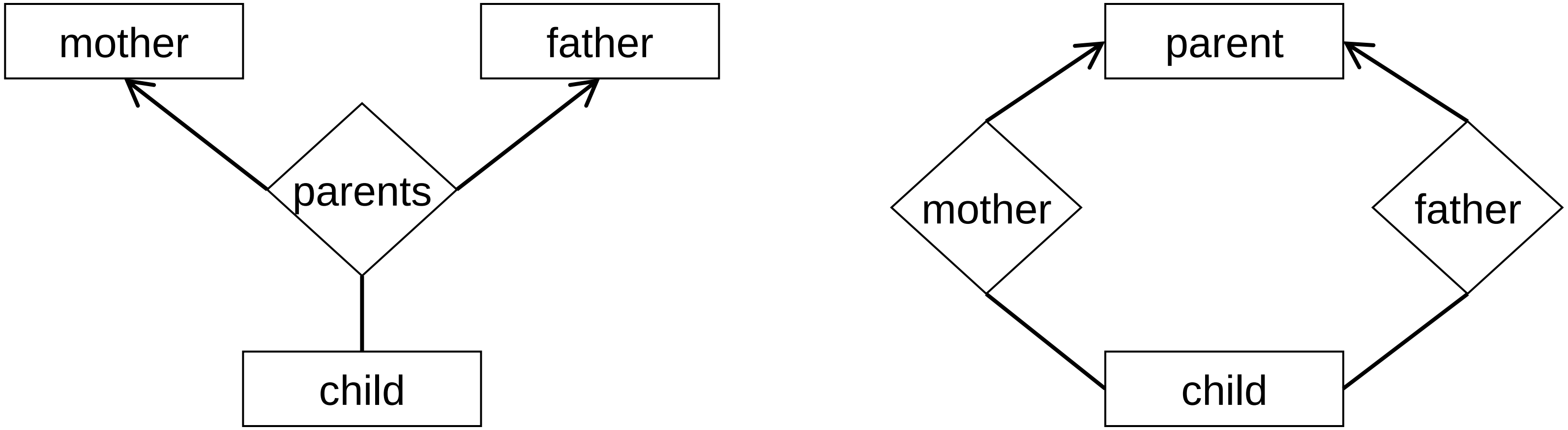
- If we model *course_offering* as a strong entity we would model *course_number* as an attribute.
  - Then the relationship with *course* would be implicit in the *course_number* attribute.

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*

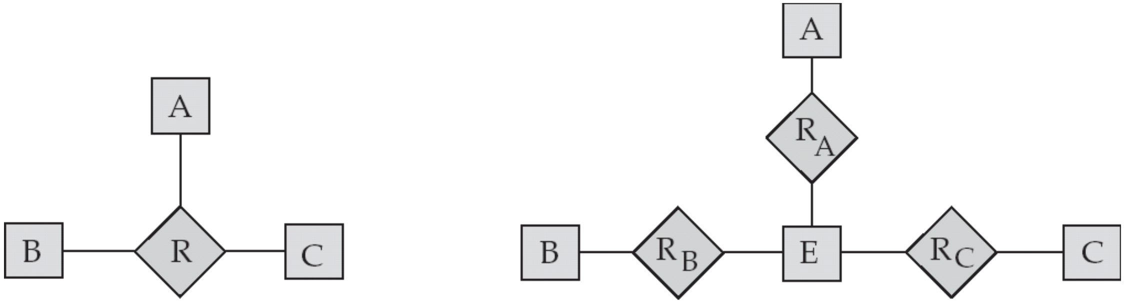


Converting Non-Binary Relationships to Binary Form

- 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



(a) (b)

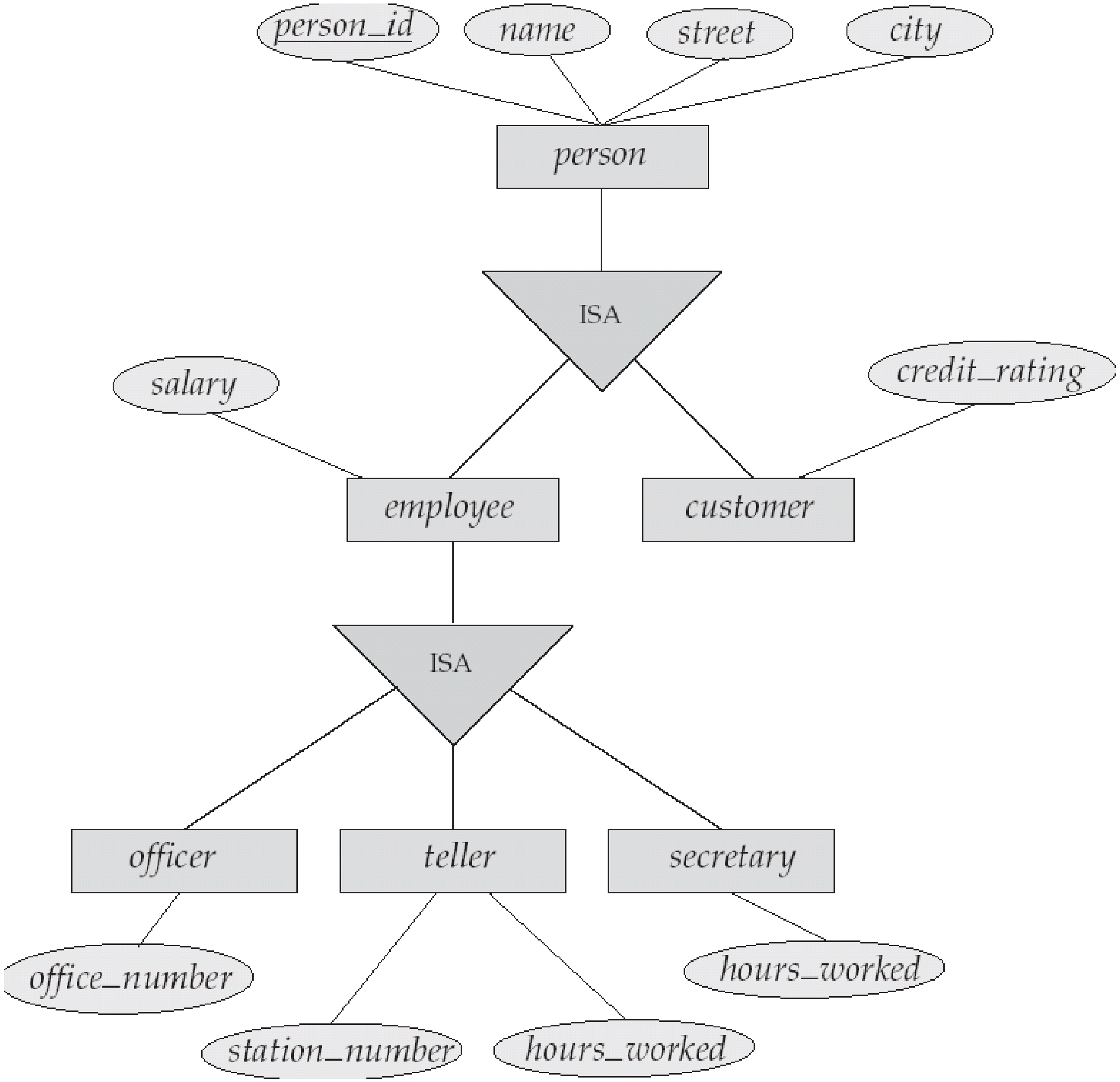
Converting Non-Binary Relationships (Cont.)

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

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



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

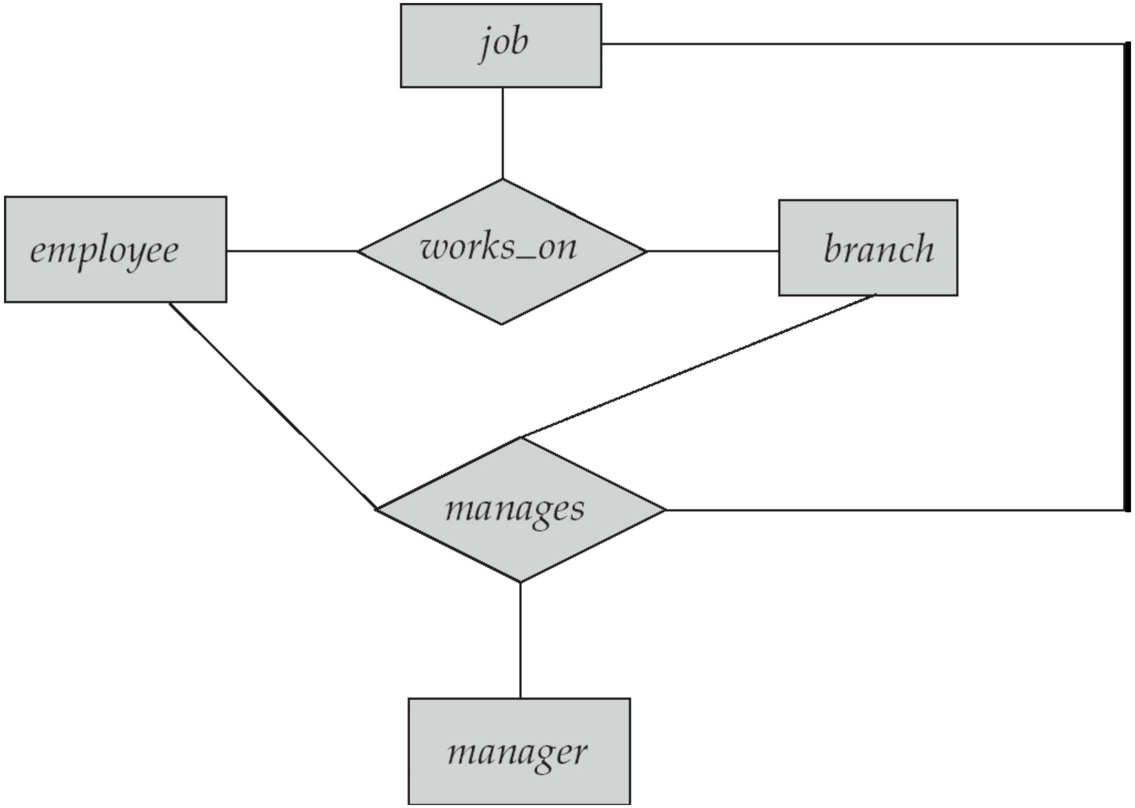
- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
  - user-defined
- Constraint on whether or not entities may belong to more than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

Aggregation

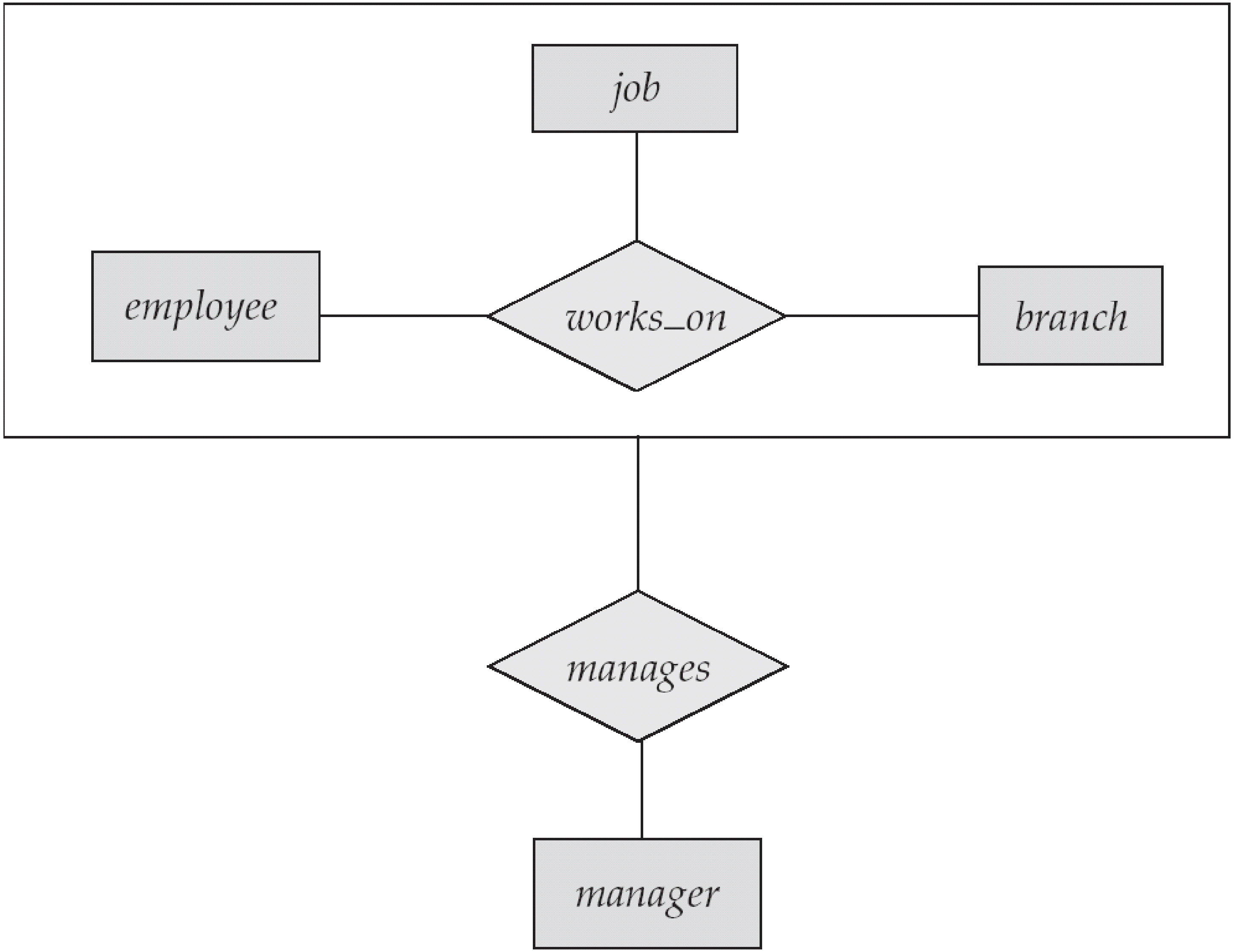
- Consider the ternary relationship *works_on*, which we saw earlier
- Suppose we want to record managers for some tasks performed by an employee at a branch



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

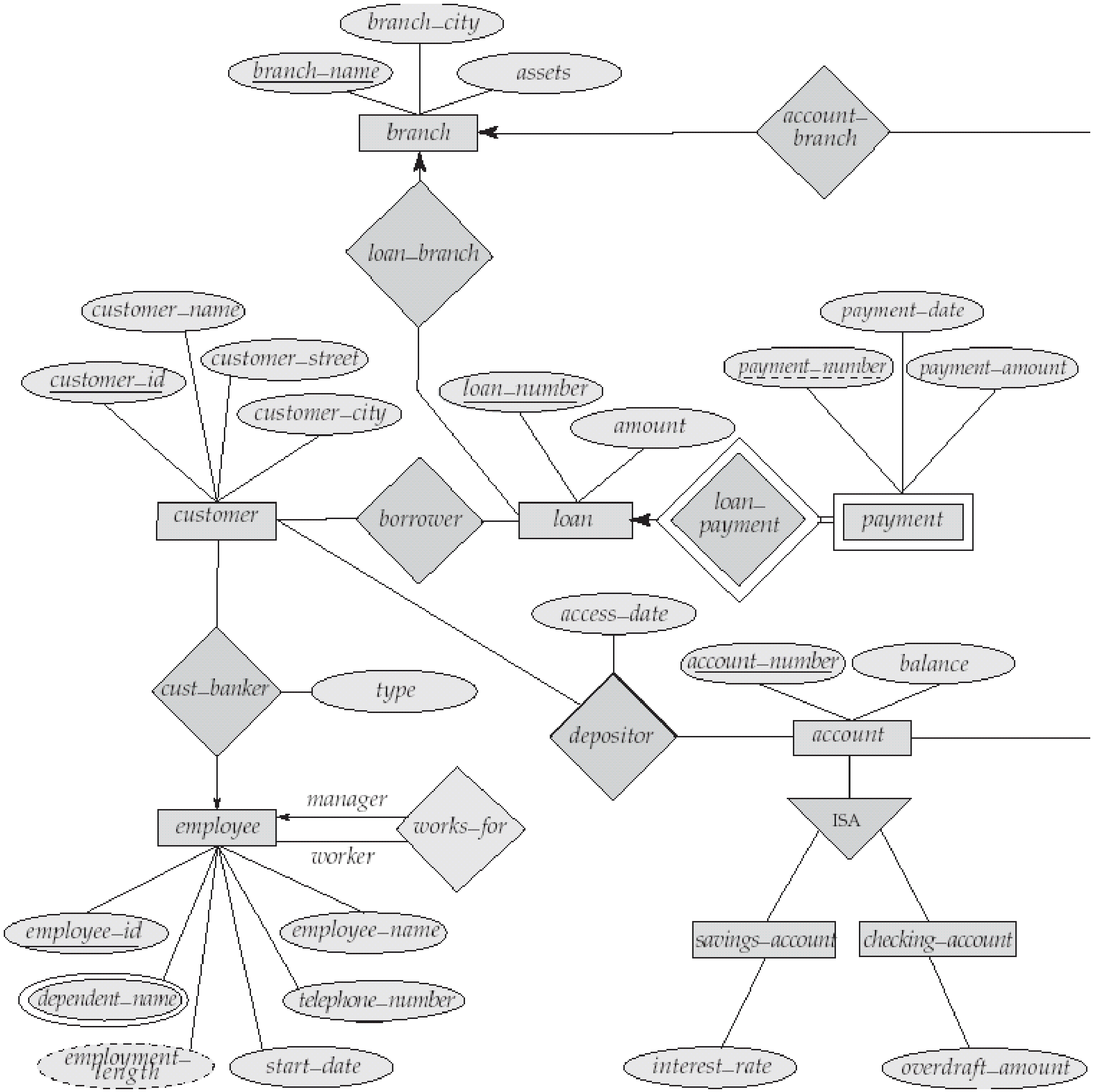


E-R Design Decisions

- 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.

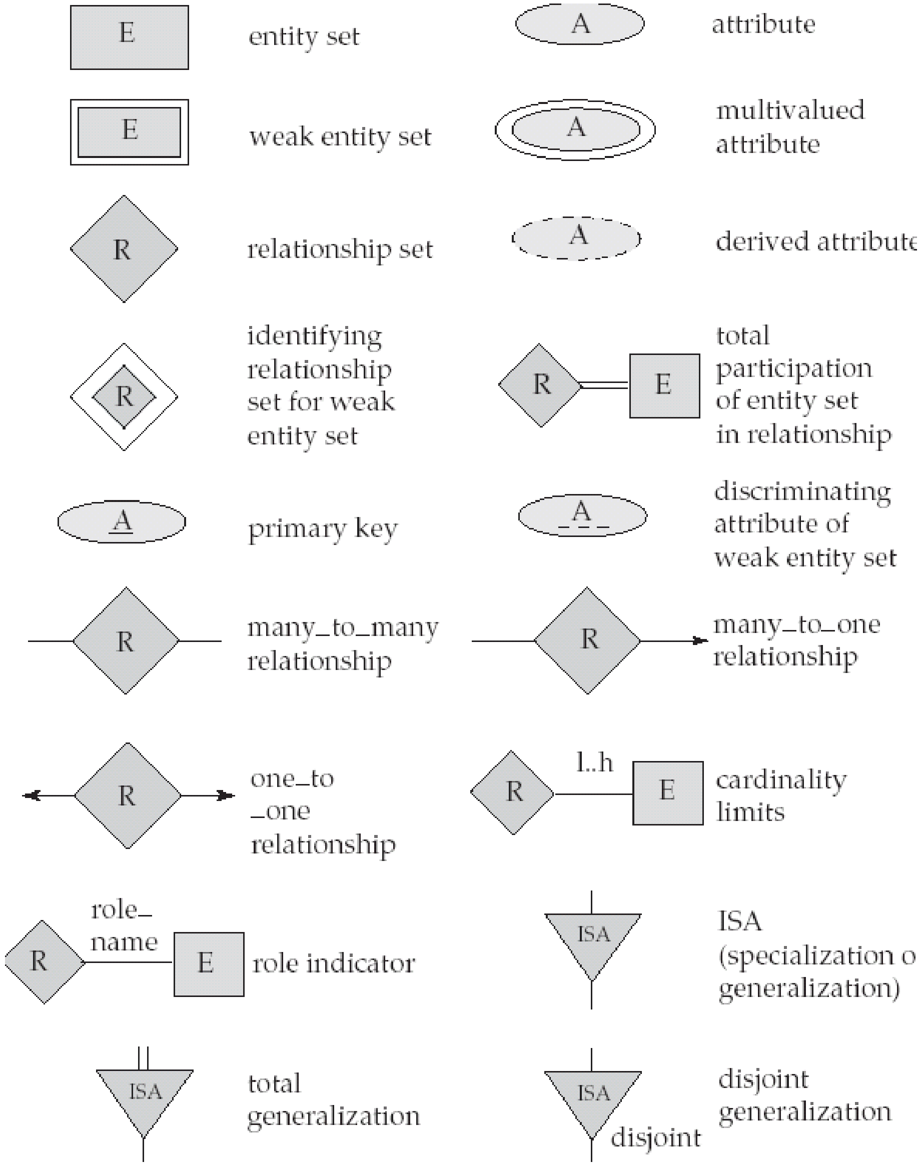
- The use of a strong or weak entity set.
- The use of specialization/generalization
  - contributes to modularity in the design.
- The use of aggregation
  - can treat the aggregate entity sets as a single unit without concern for the details of its internal structure.

E-R Diagram for a Banking Enterprise



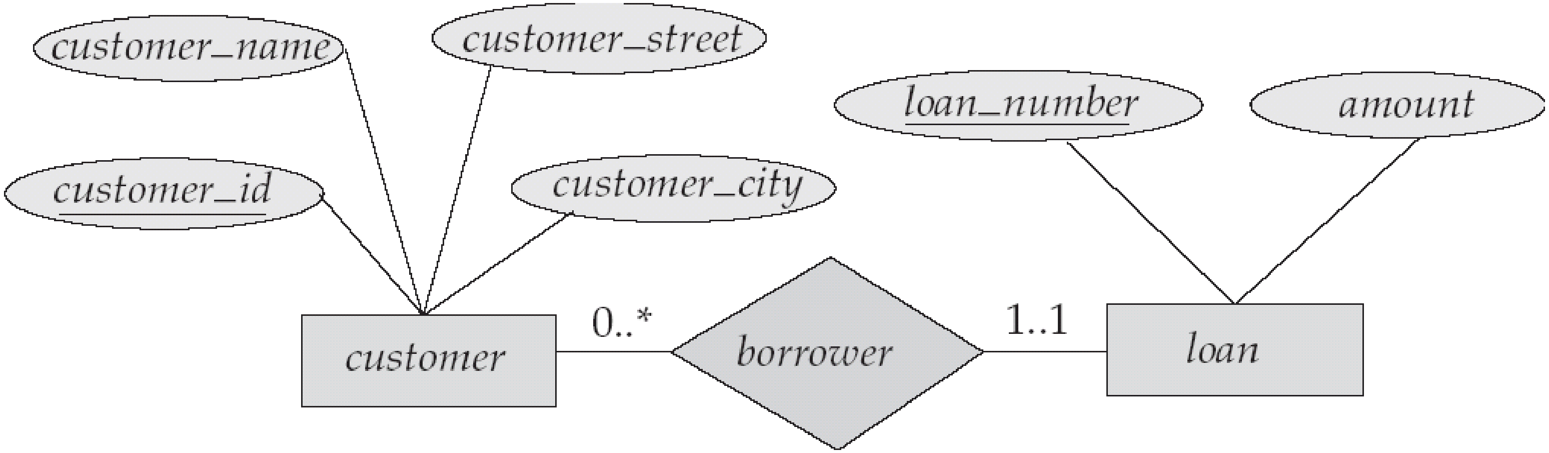
Summary of Symbols Used in E-R Notation

Chen’s E-R Notation

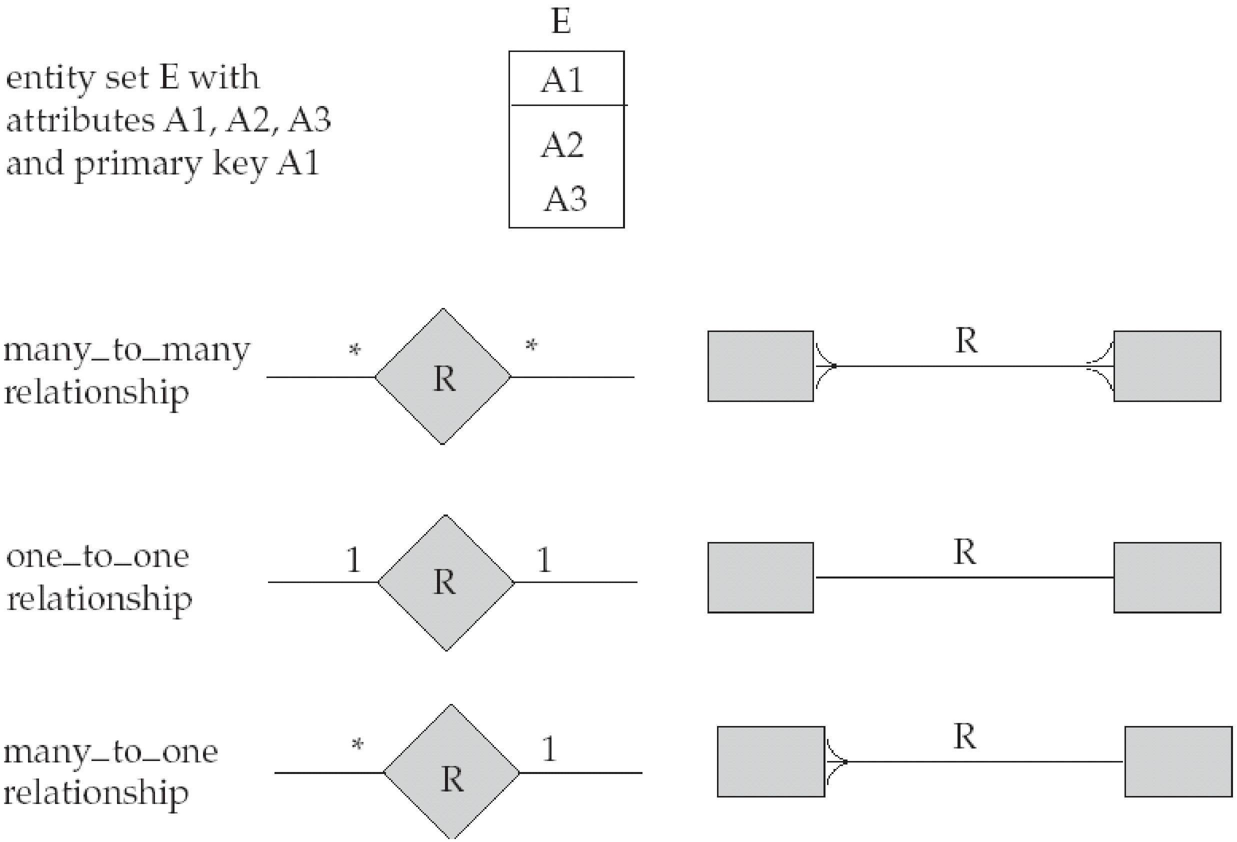


Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints
  - However, the other way around
    - It resembles Min-Max/ISO notation
  - This example expresses one-to-many relationship between customer (one) and loan (many)
    - Moreover, each loan must have a customer assigned (total participation)



Alternative E-R Notations



entity set E with
attributes A1, A2, A3
and primary key A1

many_to_many
relationship

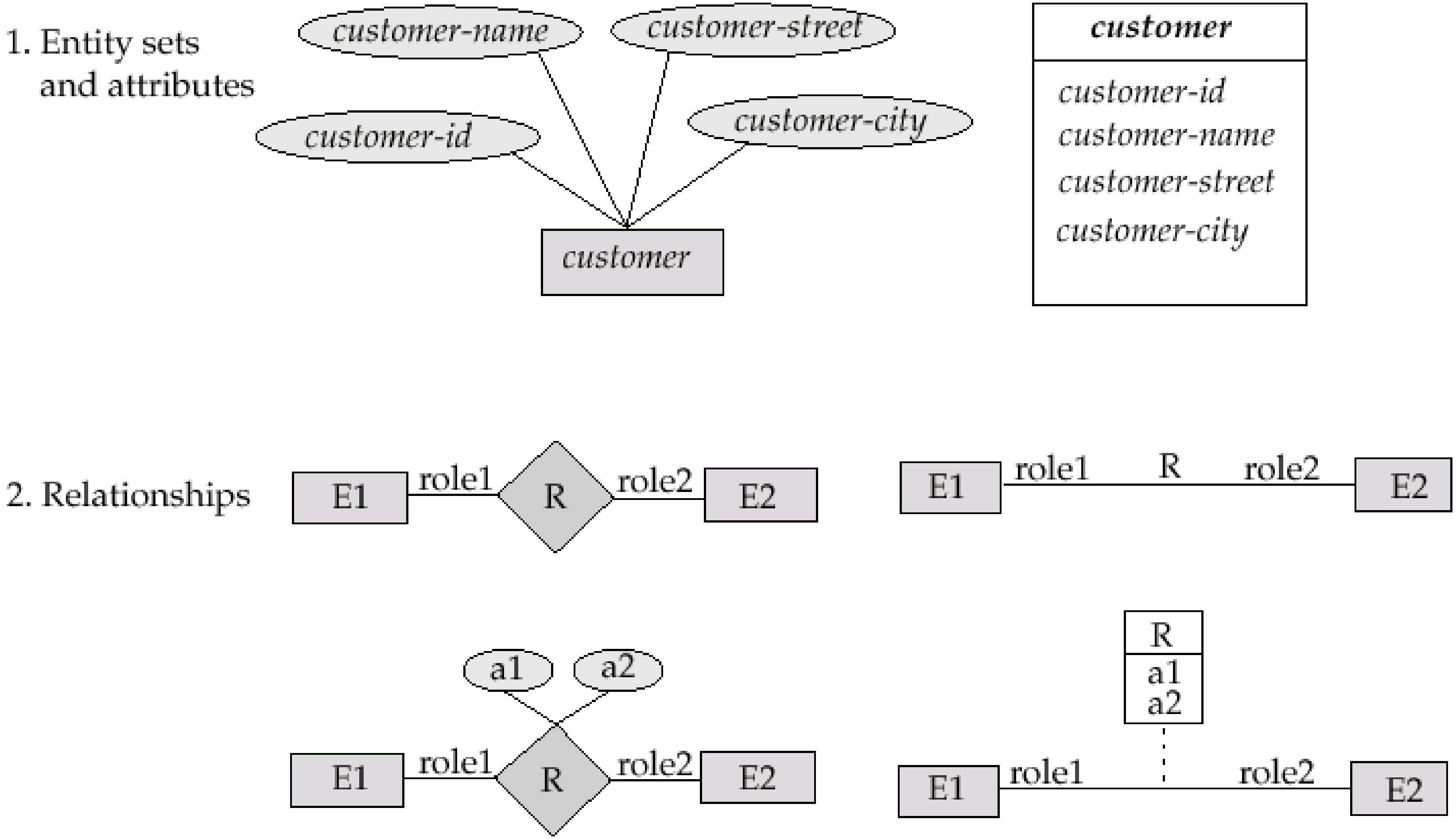
one_to_one
relationship

many_to_one
relationship

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 (work flows)
  - object modeling
  - component modeling
- UML Class Diagrams correspond to E-R Diagram
  - but there are several differences.

Summary of UML Class Diagram Notation



Chen’s notation

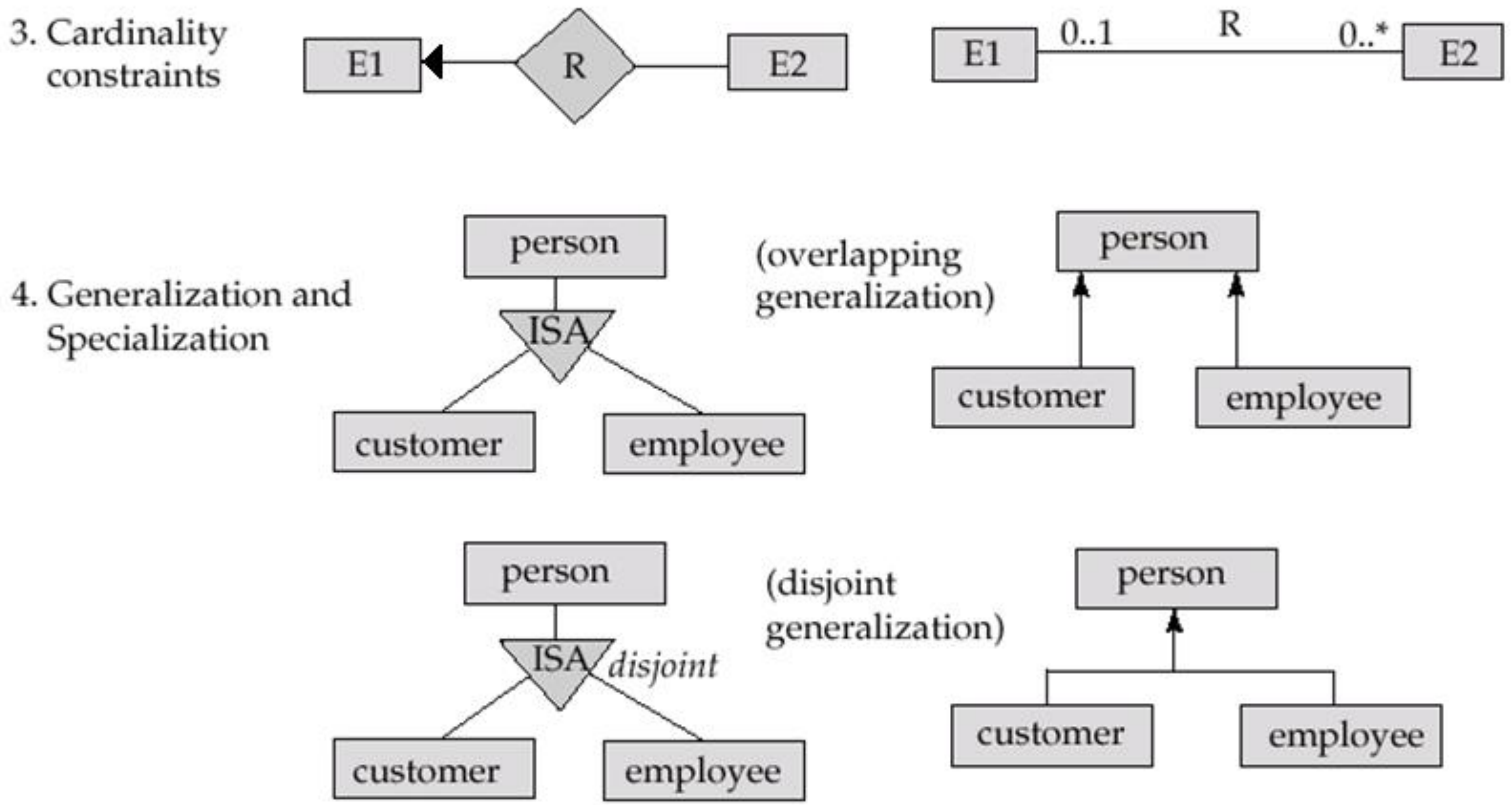
UML notation

UML Class Diagrams (Cont.)

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

UML Class Diagram Notation (Cont.)

Chen's notation UML notation



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

UML Class Diagrams (Cont.)

- Cardinality constraints are specified in the form *l..h*
  - *l* denotes the minimum and *h* the maximum number of relationships an entity can participate in.
- Beware: the positioning of the numeric constraints is exactly the reverse of the positioning of them in E-R diagrams (with numeric constraints).
  - But it is the same in case of arrows denoting 0..1.
- The constraint 0..* on the *E*2 side and 0..1 on the *E*1 side means
  - that each *E*2 entity can participate in at most one relationship,
  - whereas each *E*1 entity can participate in many relationships;
  - in other words, the relationship is many to one from *E*2 to *E*1.
- Single values, such as 1 or * may be written on edges;
  - The single value 1 on an edge is treated as equivalent to 1..1,
  - while * is equivalent to 0..*.

Takeaways

- Differences between ERD and DFD
- Create ERD from specification
  - Use the correct notation
  - Respect the rules of notation
    - do not forget the cardinality of the relationship
- Different notations
  - connection to UML
- Design decision rules