Data Modeling

Entity-Relationship Model

Databases Model the Real World

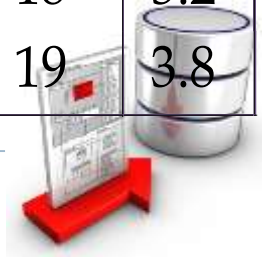
- "Data Model" translates real world things into structures computers can store
- Relational
  - Rows & Columns
  - Keys & Foreign Keys to link Relations

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8



Problems with Relational Model

```
CREATE TABLE Enrolled
      (sid CHAR(20),
       cid CHAR(20),
       grade CHAR(2))
```

```
CREATE TABLE Students
      (sid CHAR(20),
       name CHAR(20),
       login CHAR(10),
       age INTEGER,
       gpa FLOAT)
```

With complicated schemas, it may be hard for a person to understand the structure from the data definition.

Enrolled

cid	grade	sid
Carnatic101	C	53666
Reggae203	B	53666
Topology112	A	53650
History105	B	53666

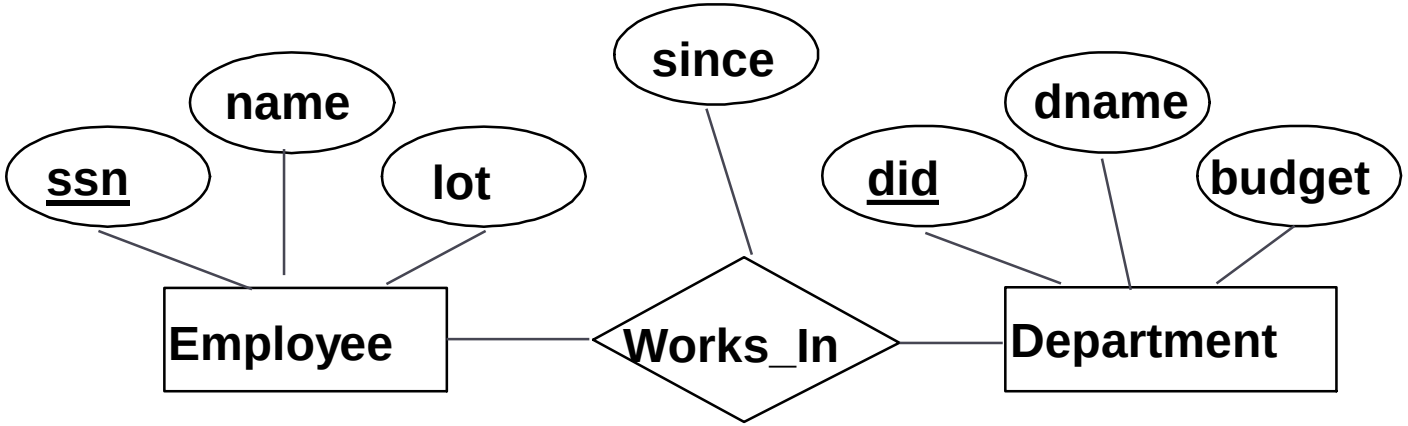
Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

One Solution: The E-R Model

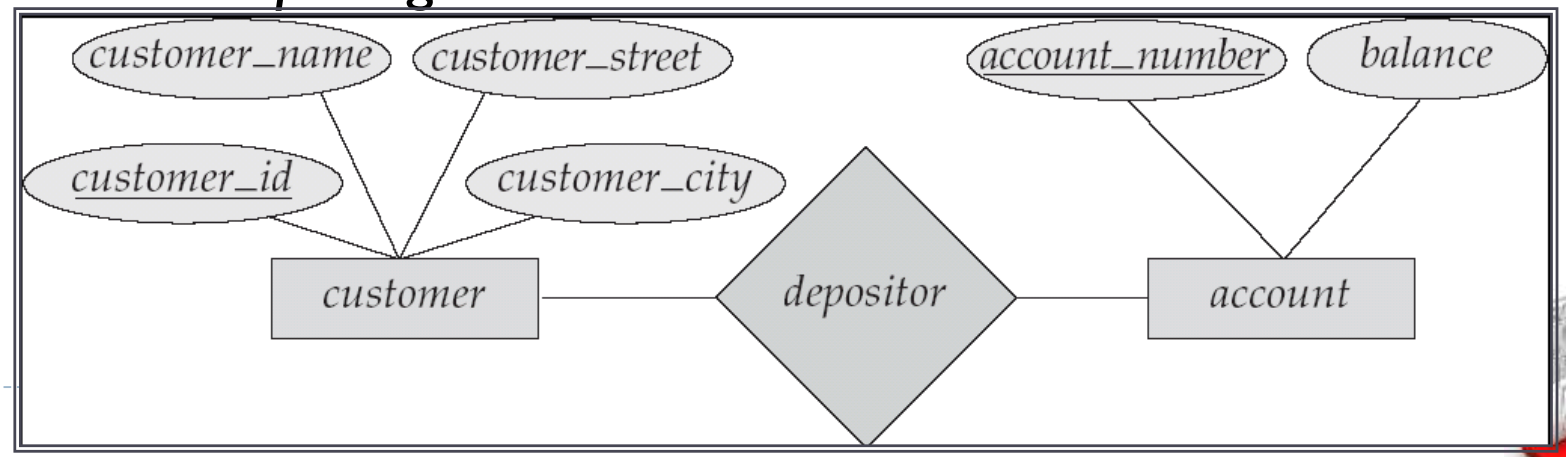
- Instead of relations, it has:
  - Entities and Relationships

- These are described with diagrams
  - both structure, notation more obvious to humans



The Entity-Relationship Model

- Models an enterprise as a collection of *entities* and *relationships*
  - Entity: a "thing" or "object" in the enterprise that is distinguishable from other objects
    - Described by a set of *attributes*
  - Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram:*



Steps in Database Design

- **Requirements Analysis**
  - user needs; what must database do?
- **Conceptual Design**
  - high level descr (often done w/ER model)
- **Logical Design**
  - translate ER into DBMS data model
- **Schema Refinement**
  - consistency, normalization
- **Physical Design**
  - indexes, disk layout,clustering
- **Application and Security Design**
  - who accesses what, and how

Example: DBA for Bank of America

- Requirements Specification
  - Determine the requirements of clients (Database to store information about customers, accounts, loans, branches, transactions, …)
- Conceptual Design
  - Express client requirements in terms of E/R model.
  - Confirm with clients that requirements are correct.
  - Specify required data operations
- Logical Design
  - Convert E/R model to relational, object-based, XML-based,…
- Physical Design
  - Specify file organizations, build indexes

E-R Modeling

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have *attributes*
  - Example: people have *names* and *addresses*
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays

Entity Sets *customer* and *loan*

customer_id	customer_ name	customer_ street	customer_ city
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

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

loan

Relationship Sets

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

  Example:

  Hayes *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

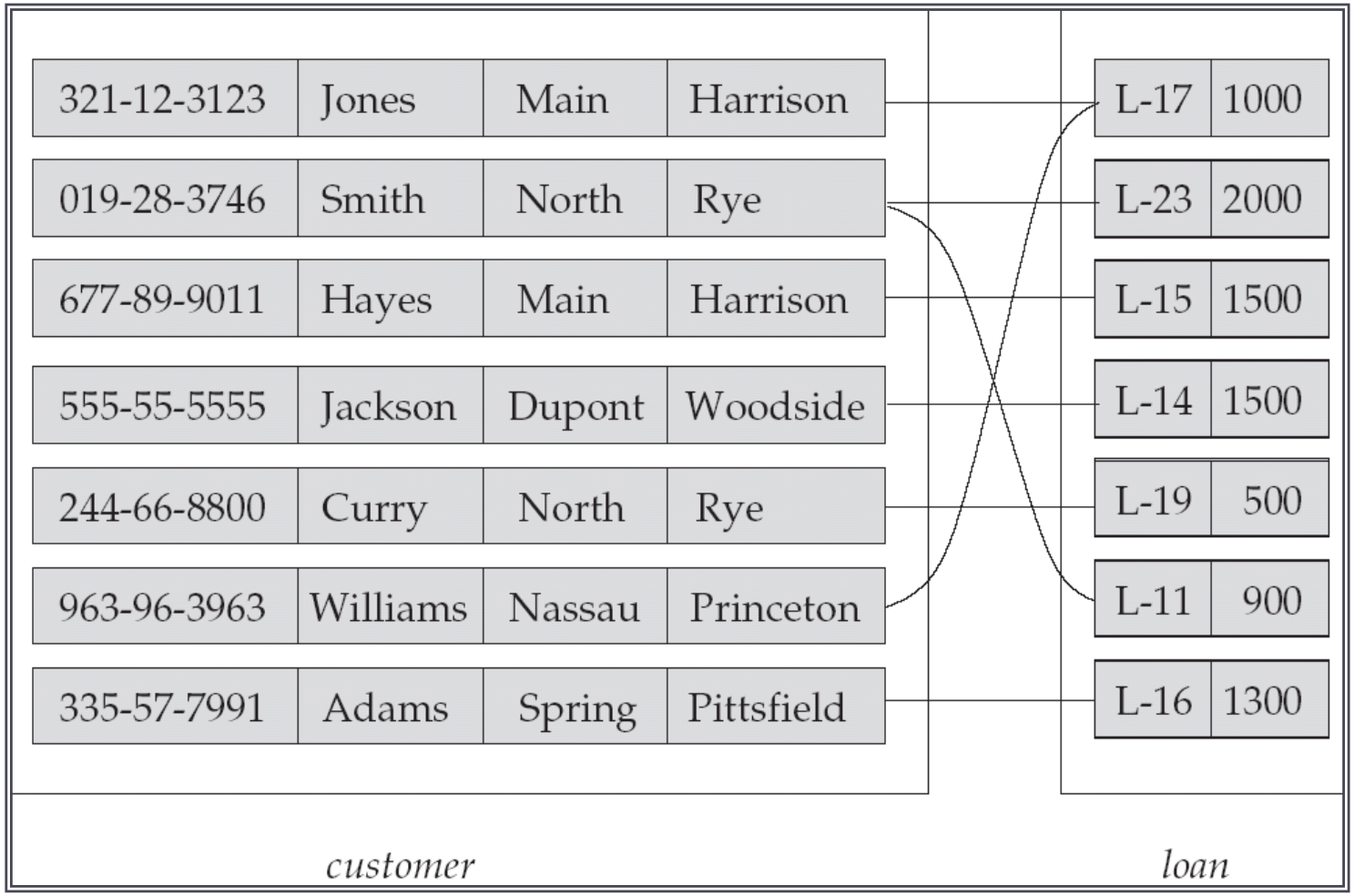
  $$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

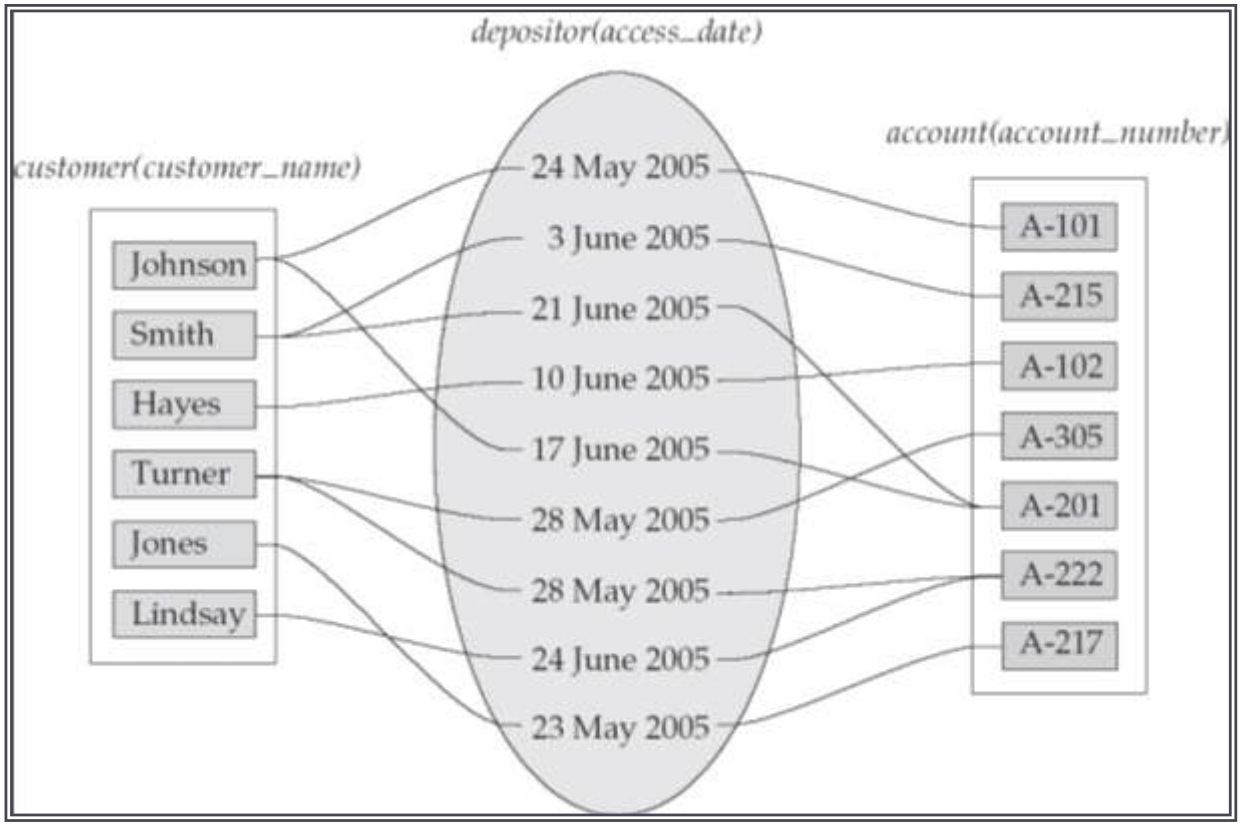
    (Hayes, A-102) $\in$ *depositor*

Relationship Set *borrower*



Relationship Sets (Cont.)

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



Degree of a Relationship Set

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

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

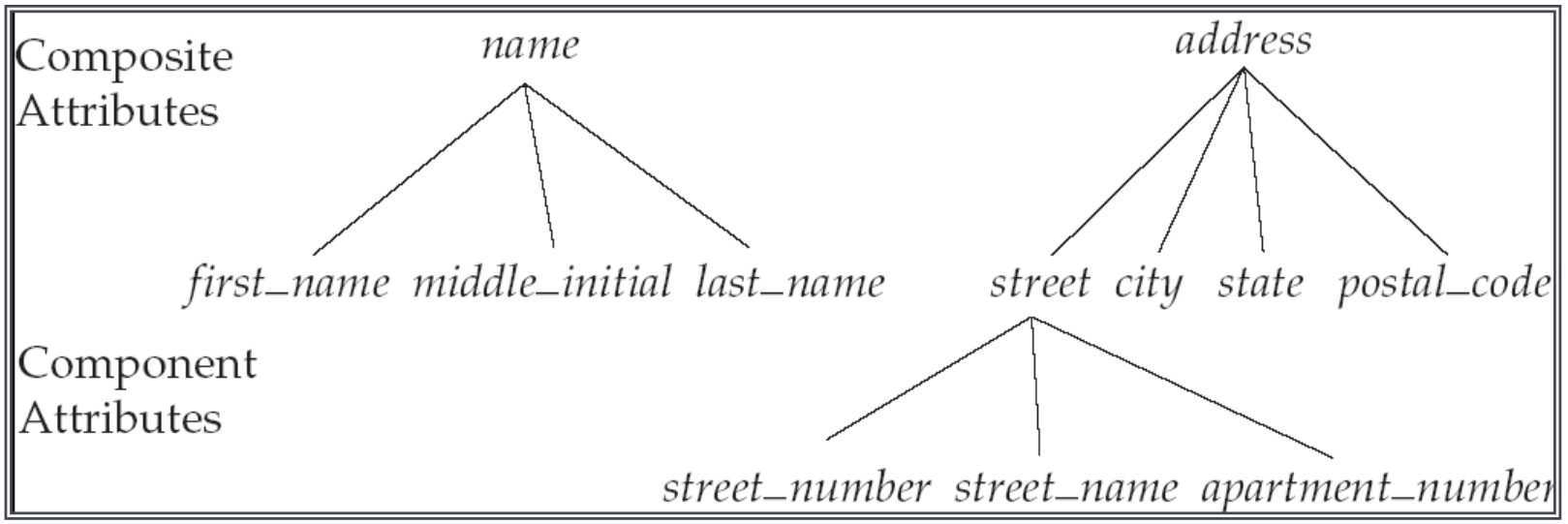
)

- **Domain** – the set of permitted values for each attribute

  loan = (*loan_number, amount*)

- Attribute types:
  - *Simple* and *composite* attributes.
  - *Single-valued* and *multi-valued* attributes
    - Example: multivalued attribute: *phone_numbers*
  - *Derived* attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth

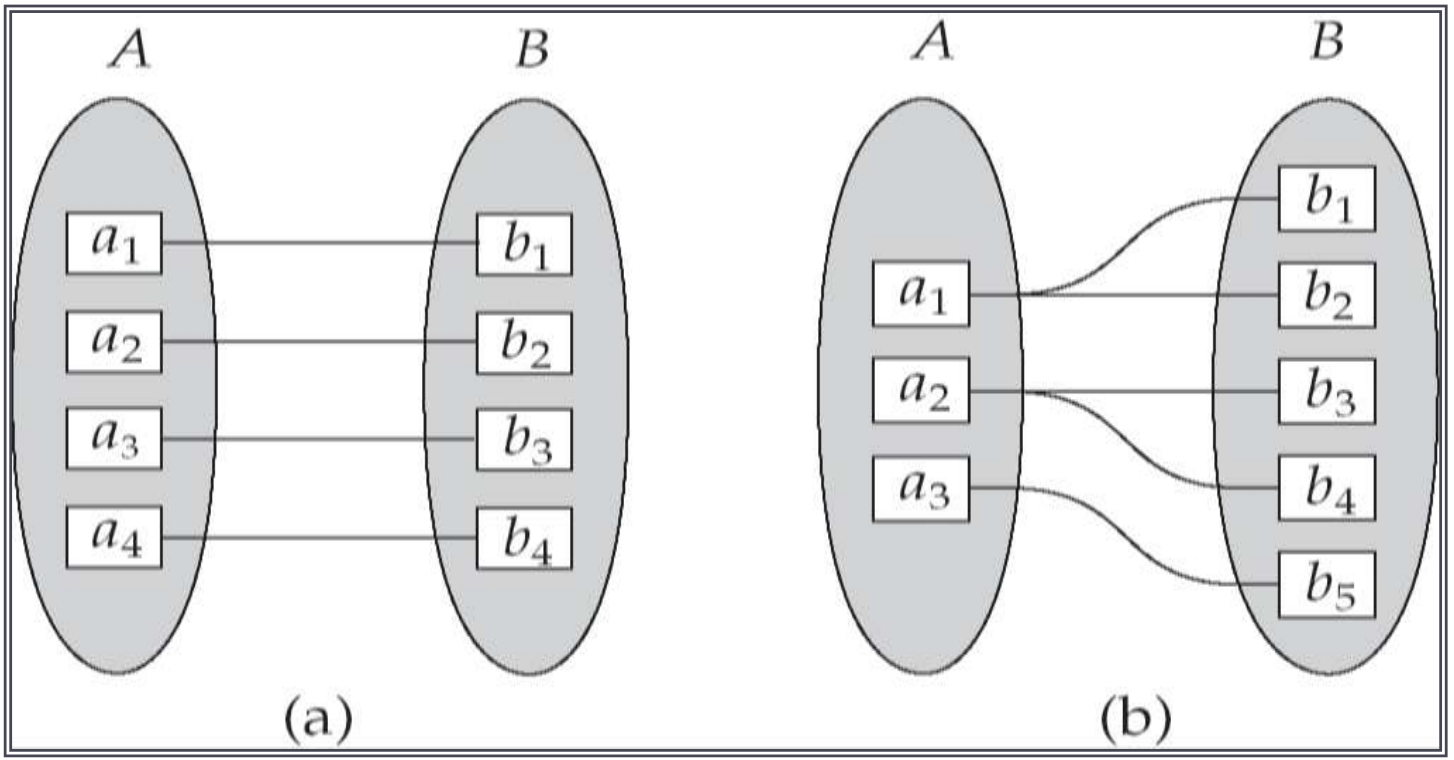
Composite Attributes



Mapping Cardinality Constraints

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

Mapping Cardinalities

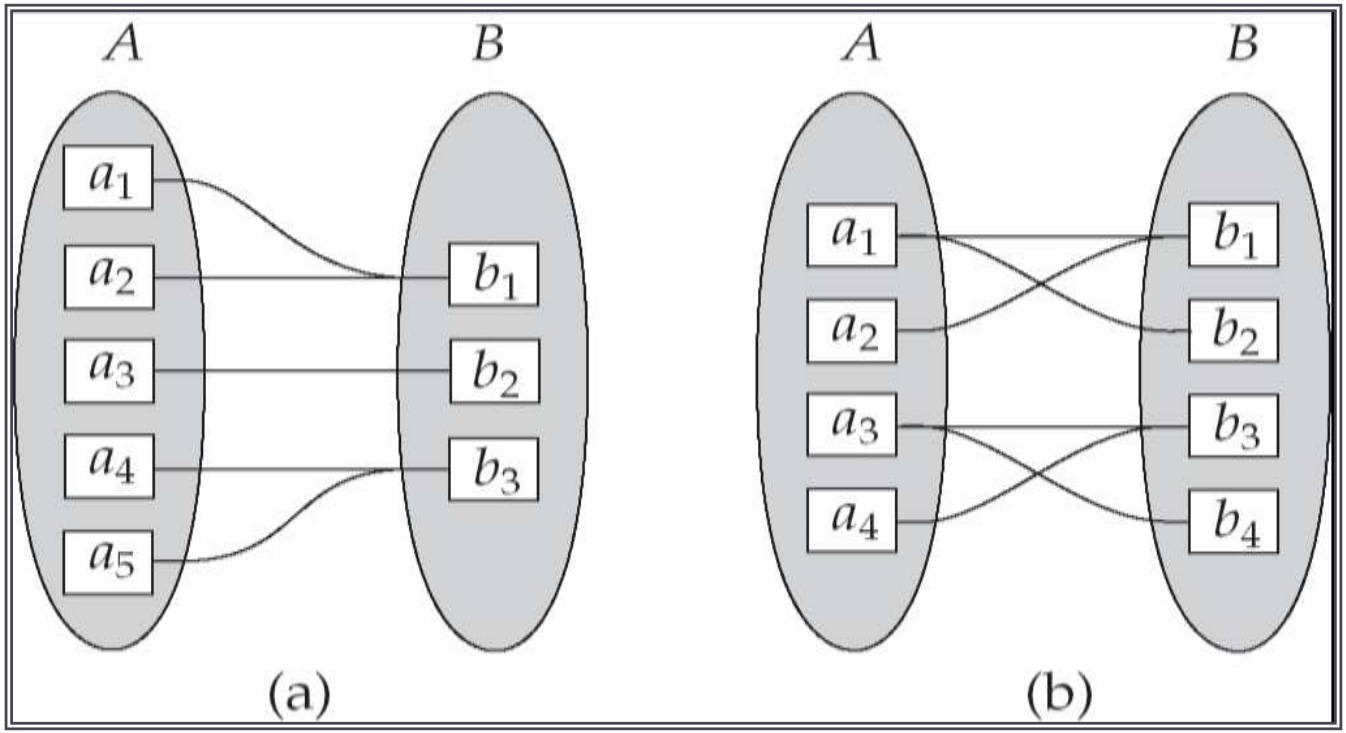


One to one

One to many

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

Mapping Cardinalities



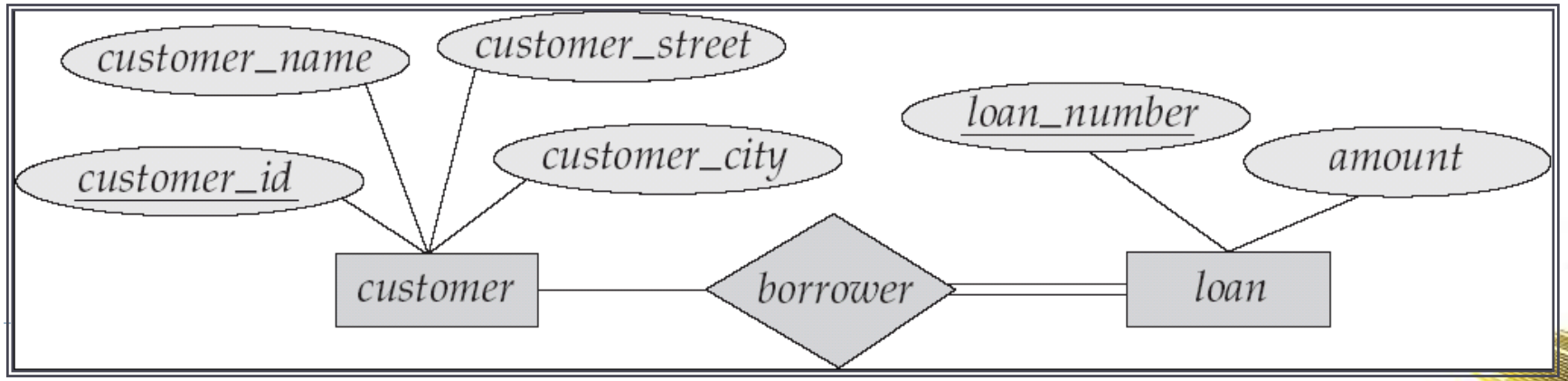
Many to one

Many to many

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

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



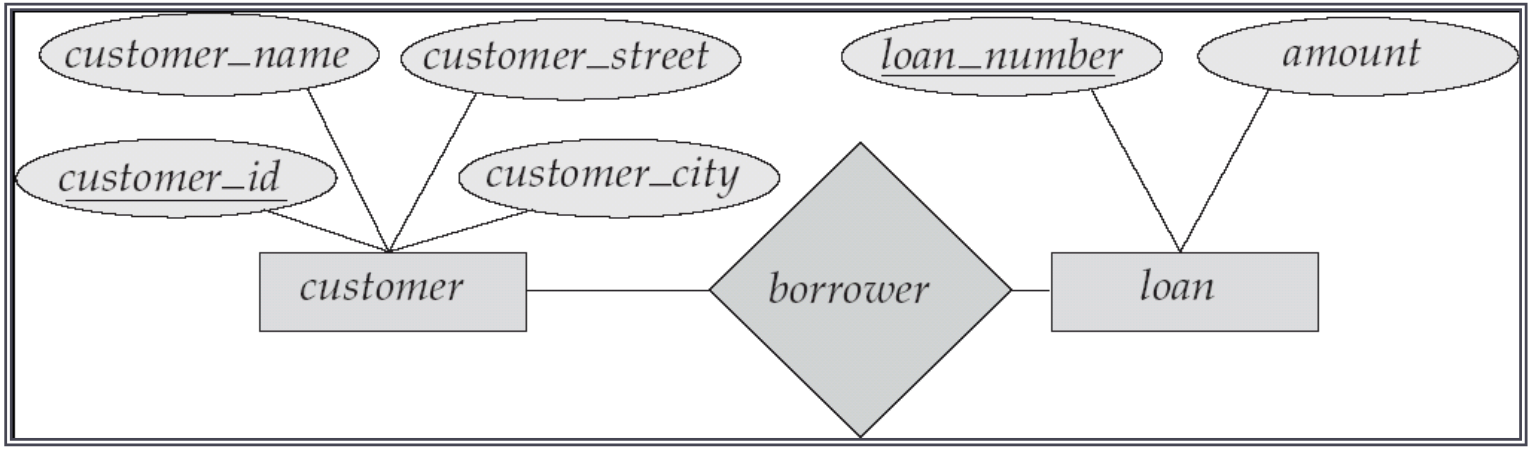
Keys

- 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

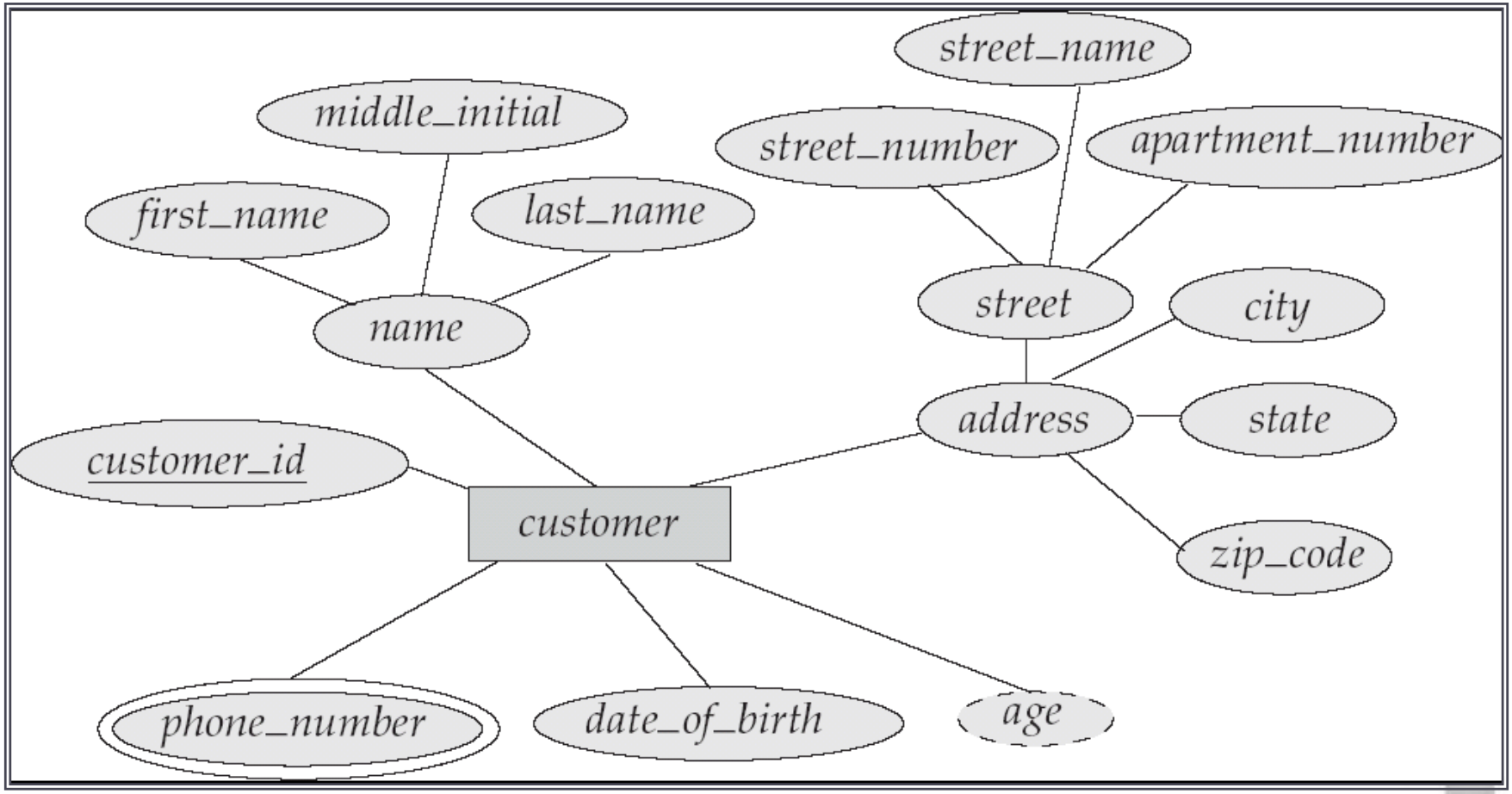
- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (*customer_id, account_number*) is the super key of *depositor*
  - *NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.*
- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys
- Need to consider semantics of relationship set in selecting the *primary key* in case of more than one candidate key

E-R Diagrams

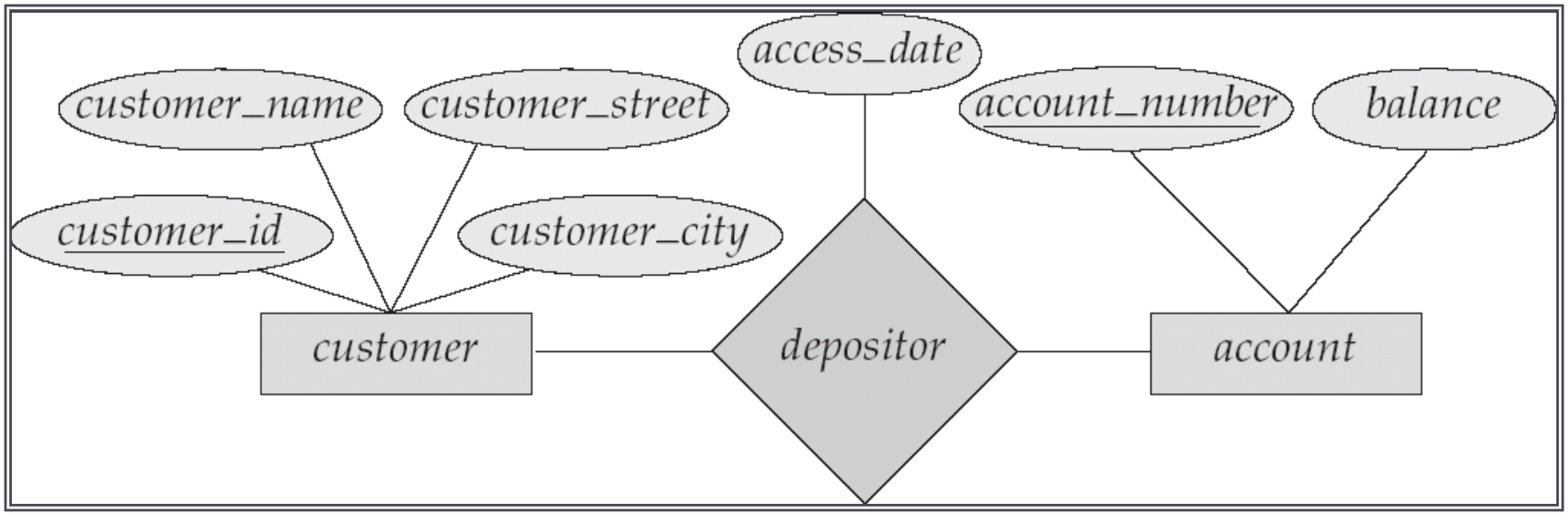


- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Ellipses represent attributes
  - Double ellipses represent multivalued attributes.
  - Dashed ellipses denote derived attributes.
- Underline indicates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

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

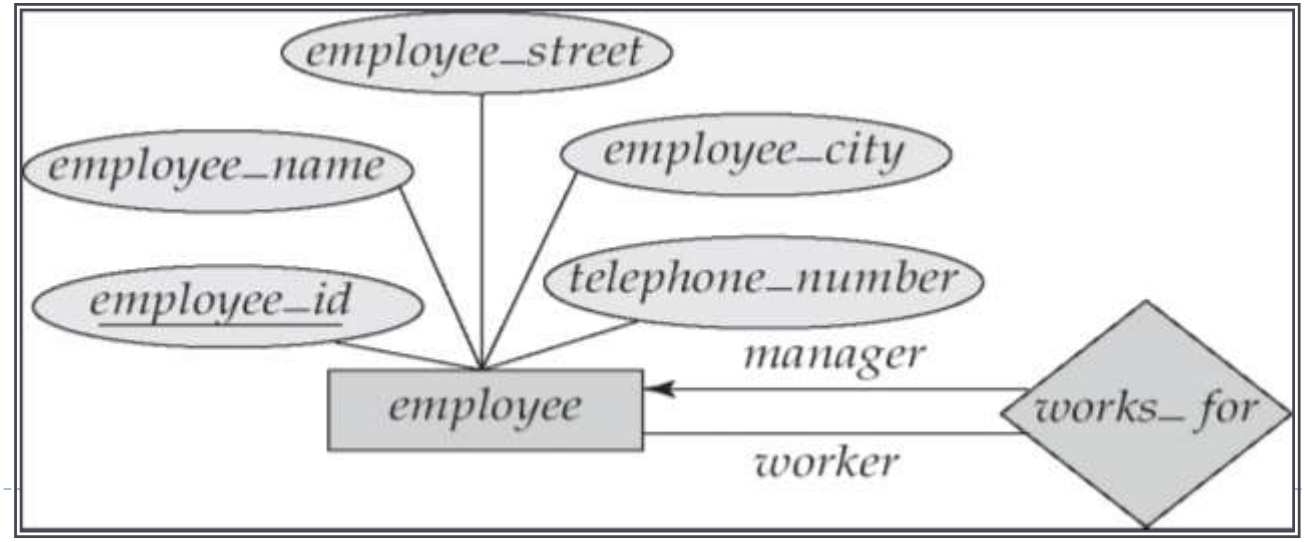


Relationship Sets with Attributes



Roles

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

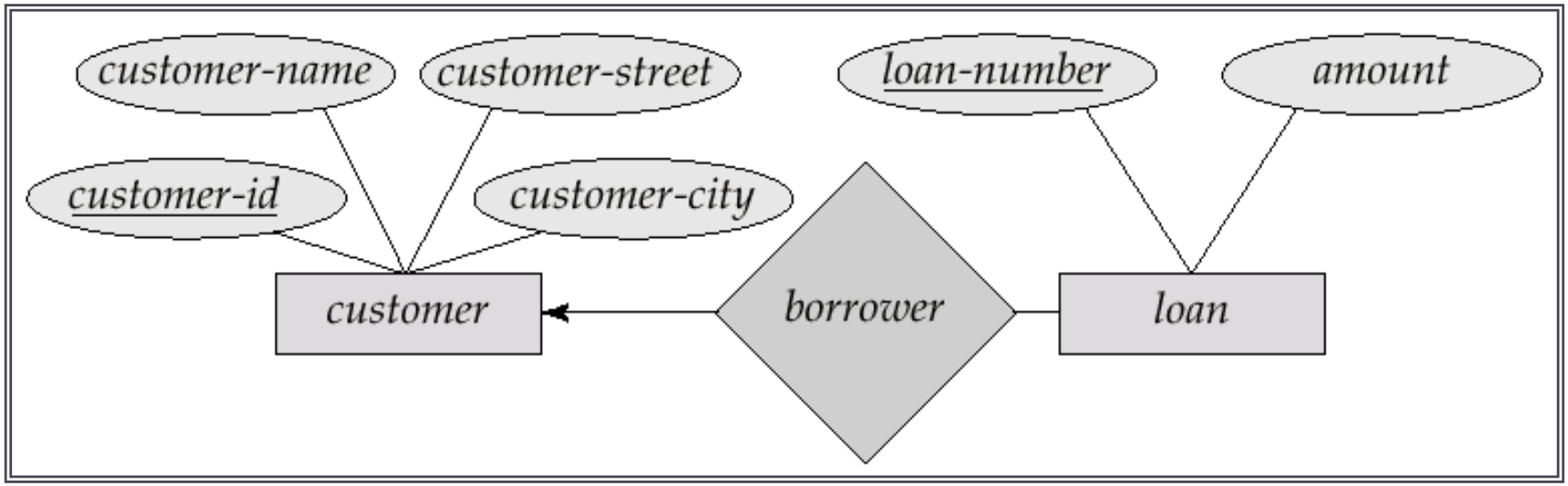


Cardinality Constraints

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

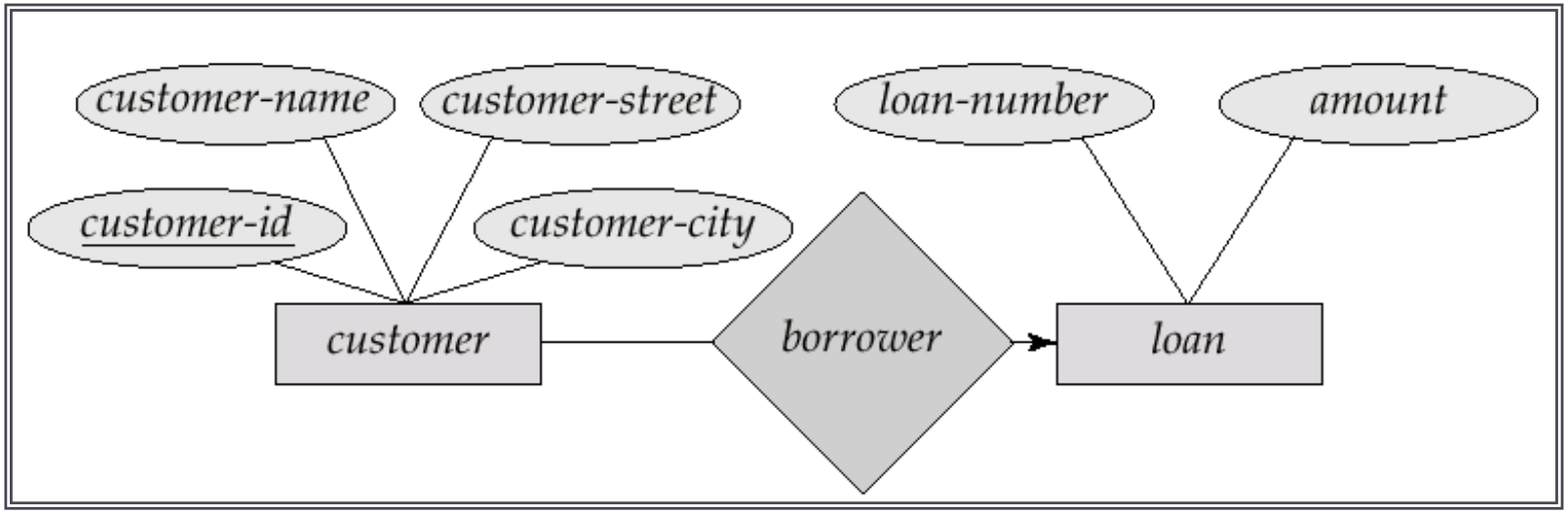
One-To-Many Relationship

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



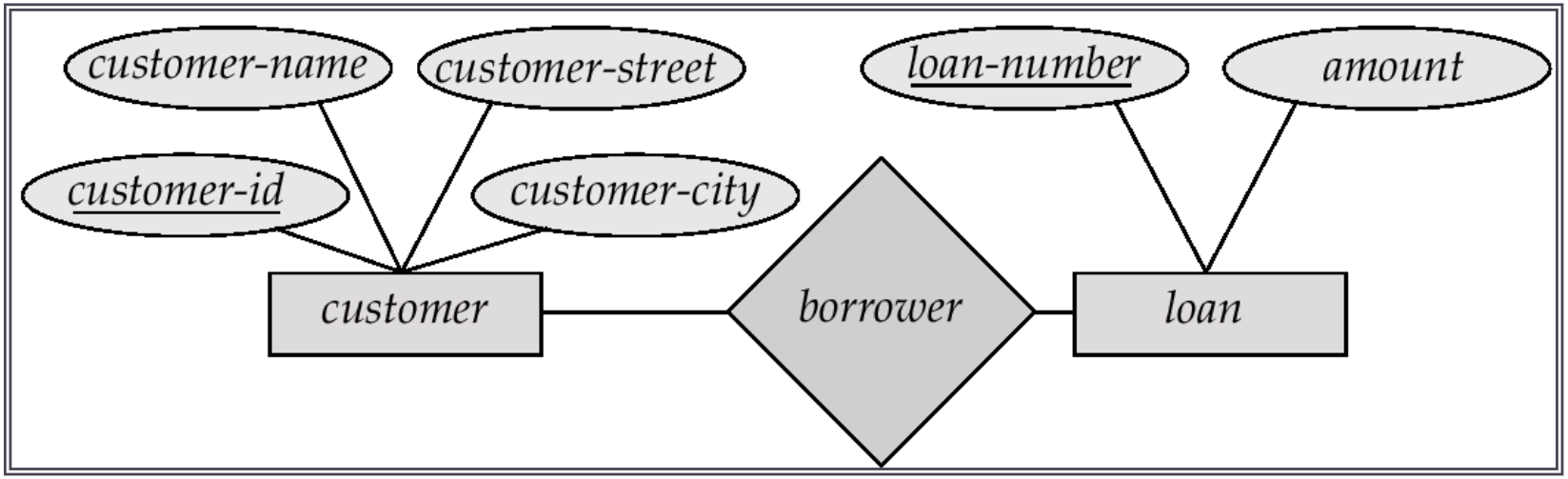
Many-To-One Relationships

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



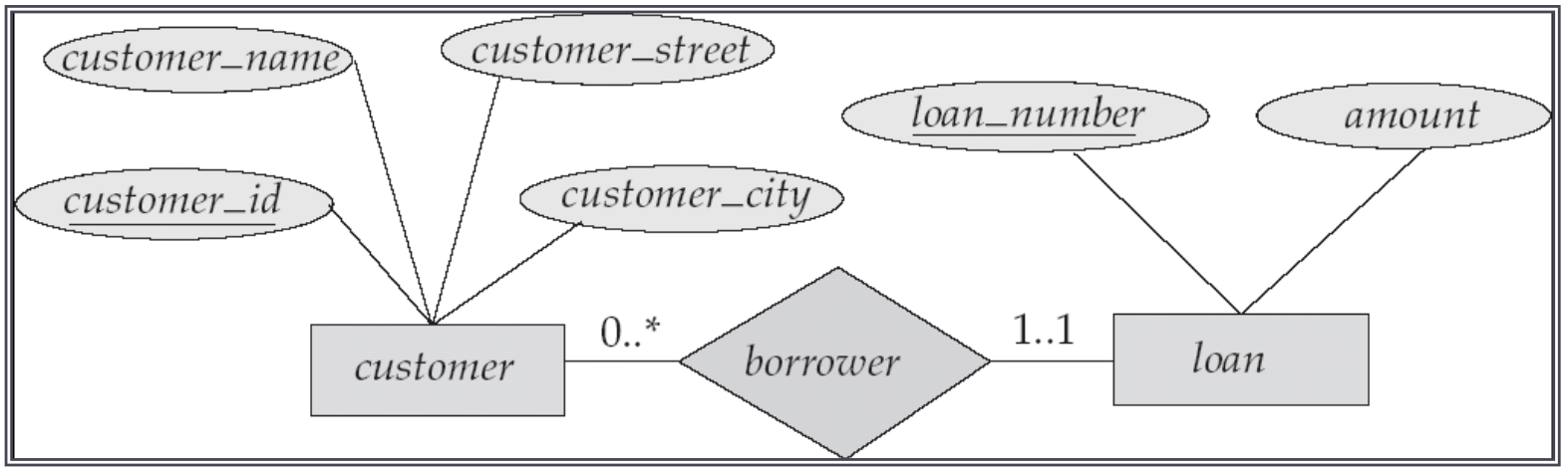
Many-To-Many Relationship

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

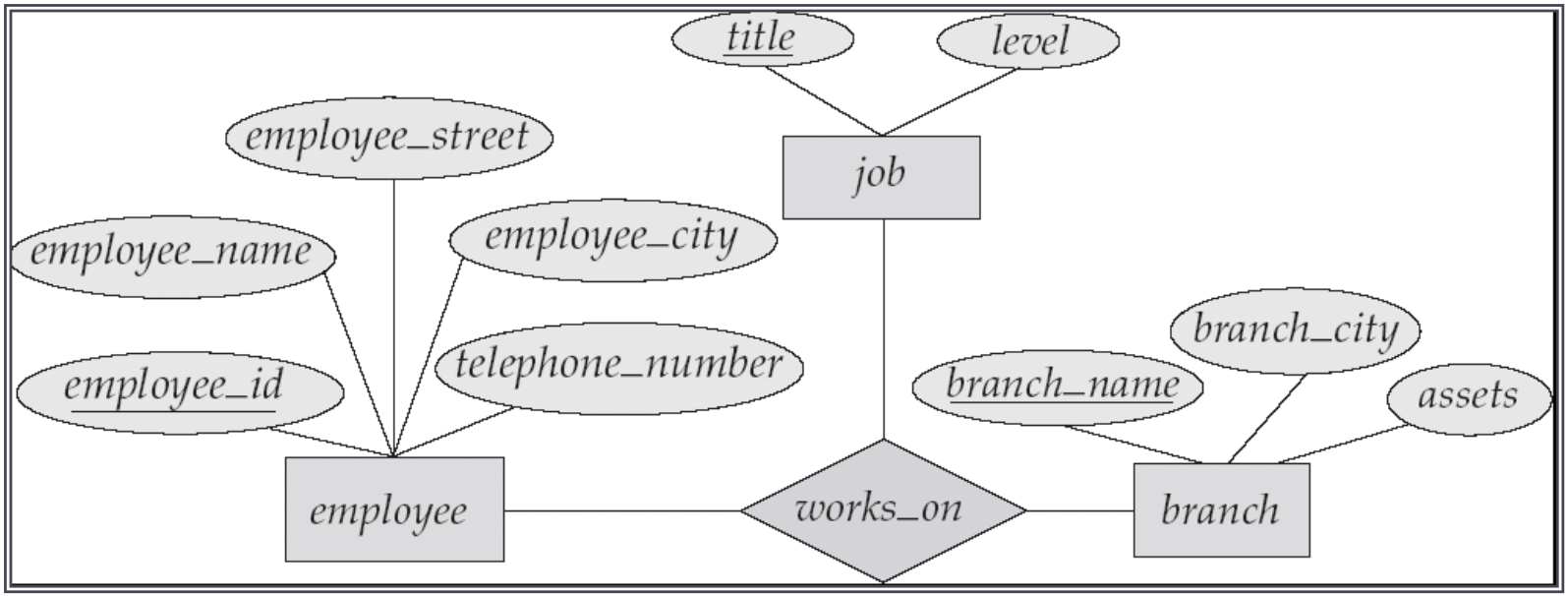


Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints



E-R Diagram with a Ternary Relationship



Design Issues

- **Use of entity sets vs. attributes**
  Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.
- **Use of entity sets vs. relationship sets**
  Possible guideline is to designate a relationship set to describe an action that occurs between entities
- **Binary versus n-ary relationship sets**
  Although it is possible to replace any nonbinary (*n*-ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- **Placement of relationship attributes**

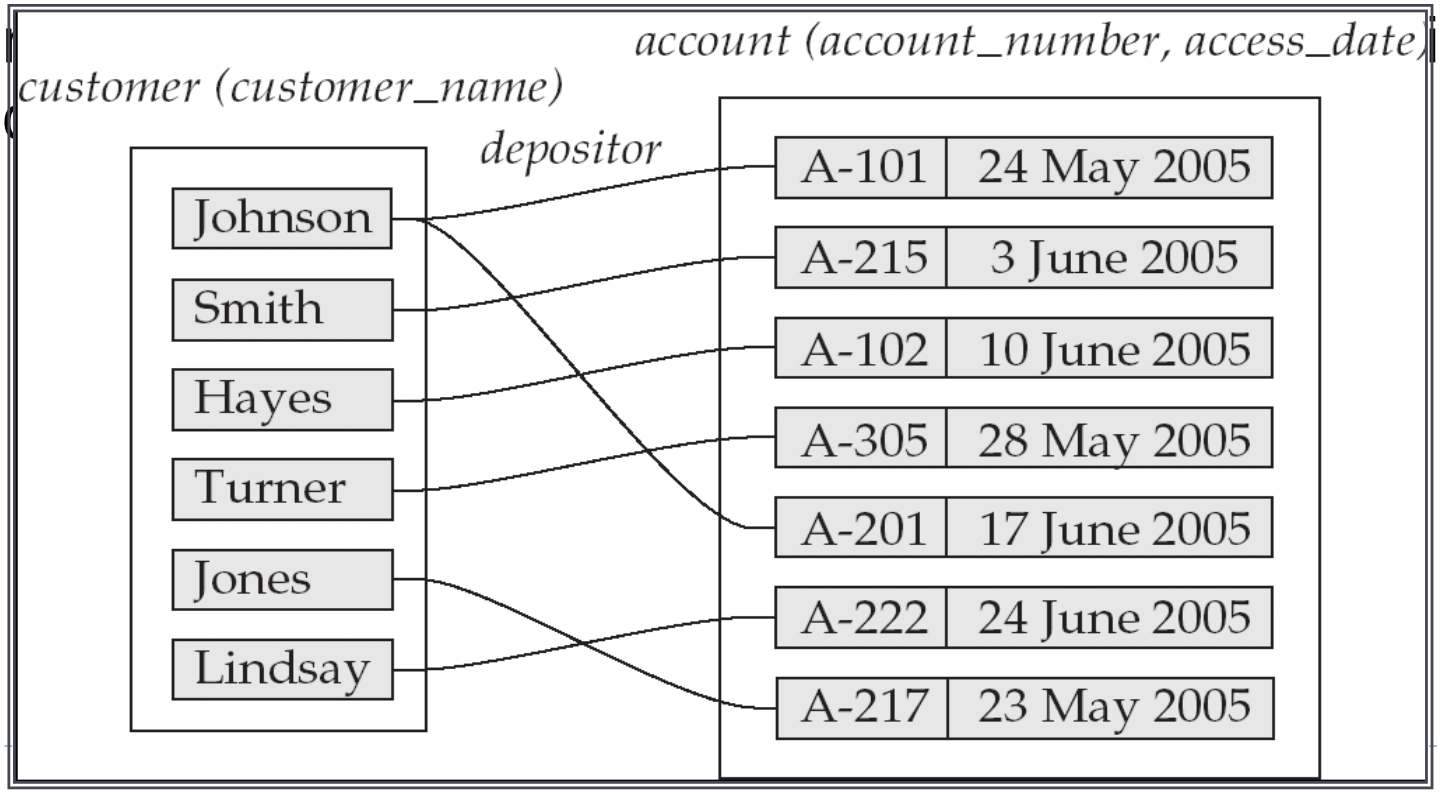
Binary Vs. Non-Binary Relationships

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

  Example: *works_on*

Mapping Cardinalities affect ER Design

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

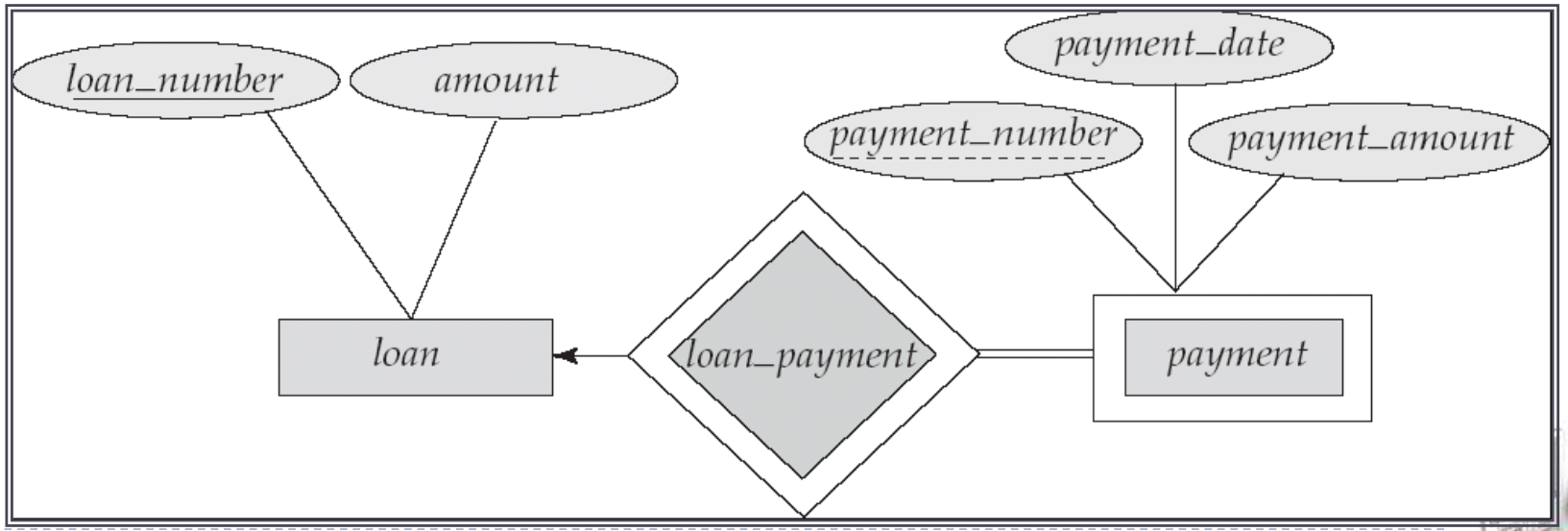


Weak Entity Sets

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

Weak Entity Sets (Cont.)

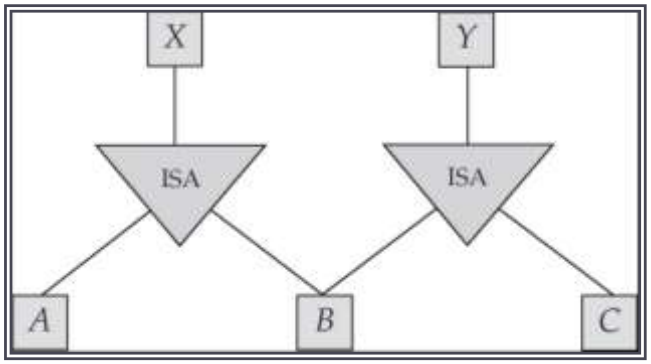
- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dashed line.
- payment_number – discriminator of the *payment* entity set
- Primary key for *payment* – (*loan_number, payment_number*)

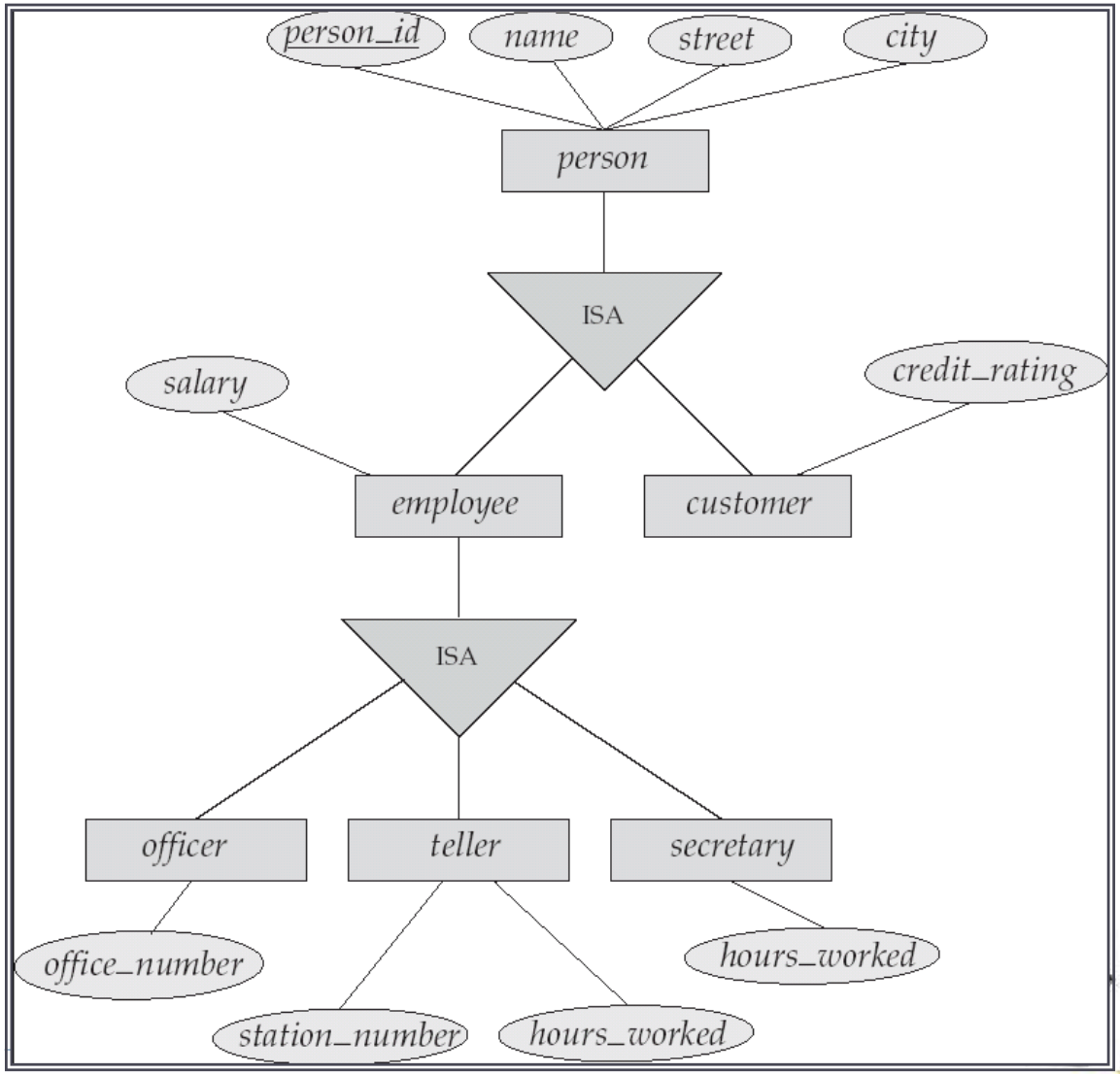


Weak Entity Sets (Cont.)

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

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

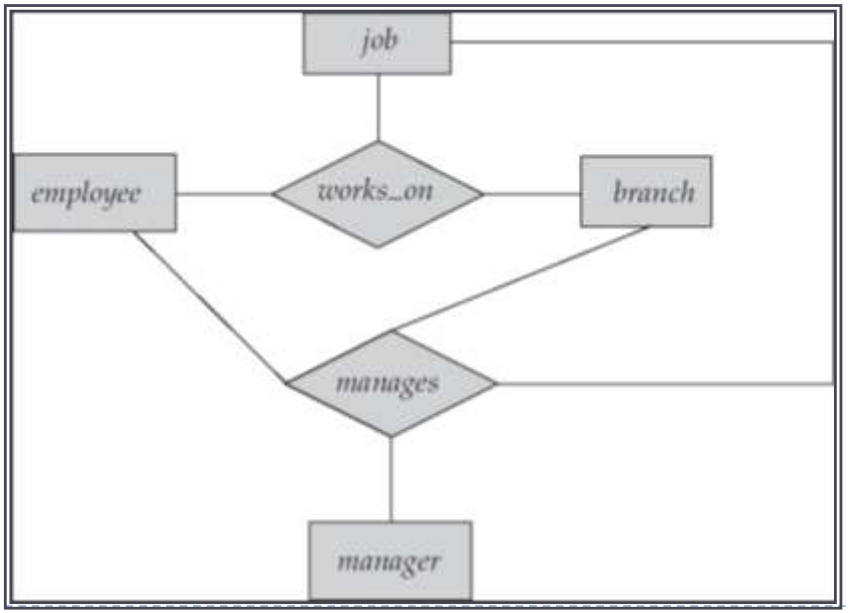
- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
  - user-defined
- Constraint on whether or not entities may belong to more than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

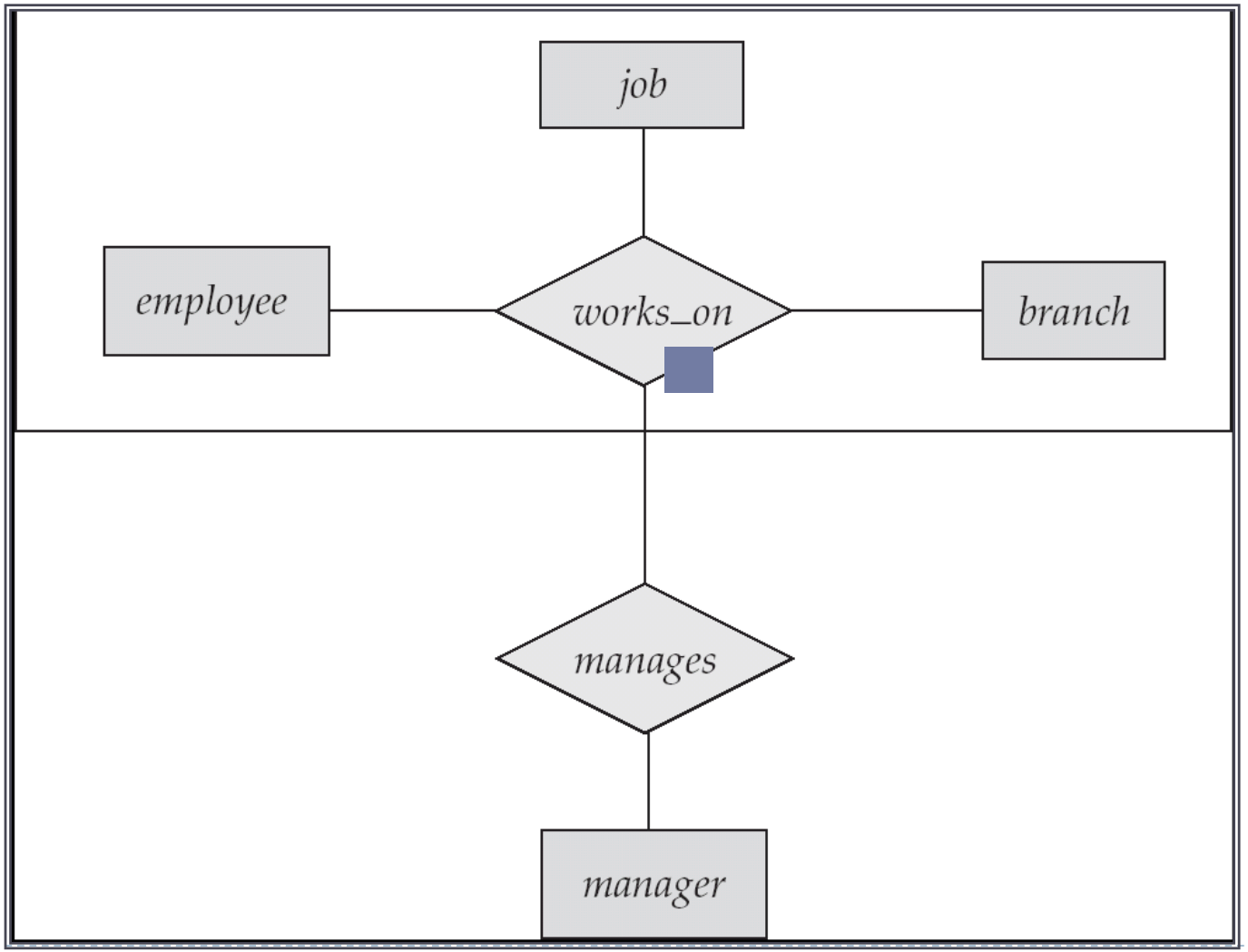
- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total** : an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong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



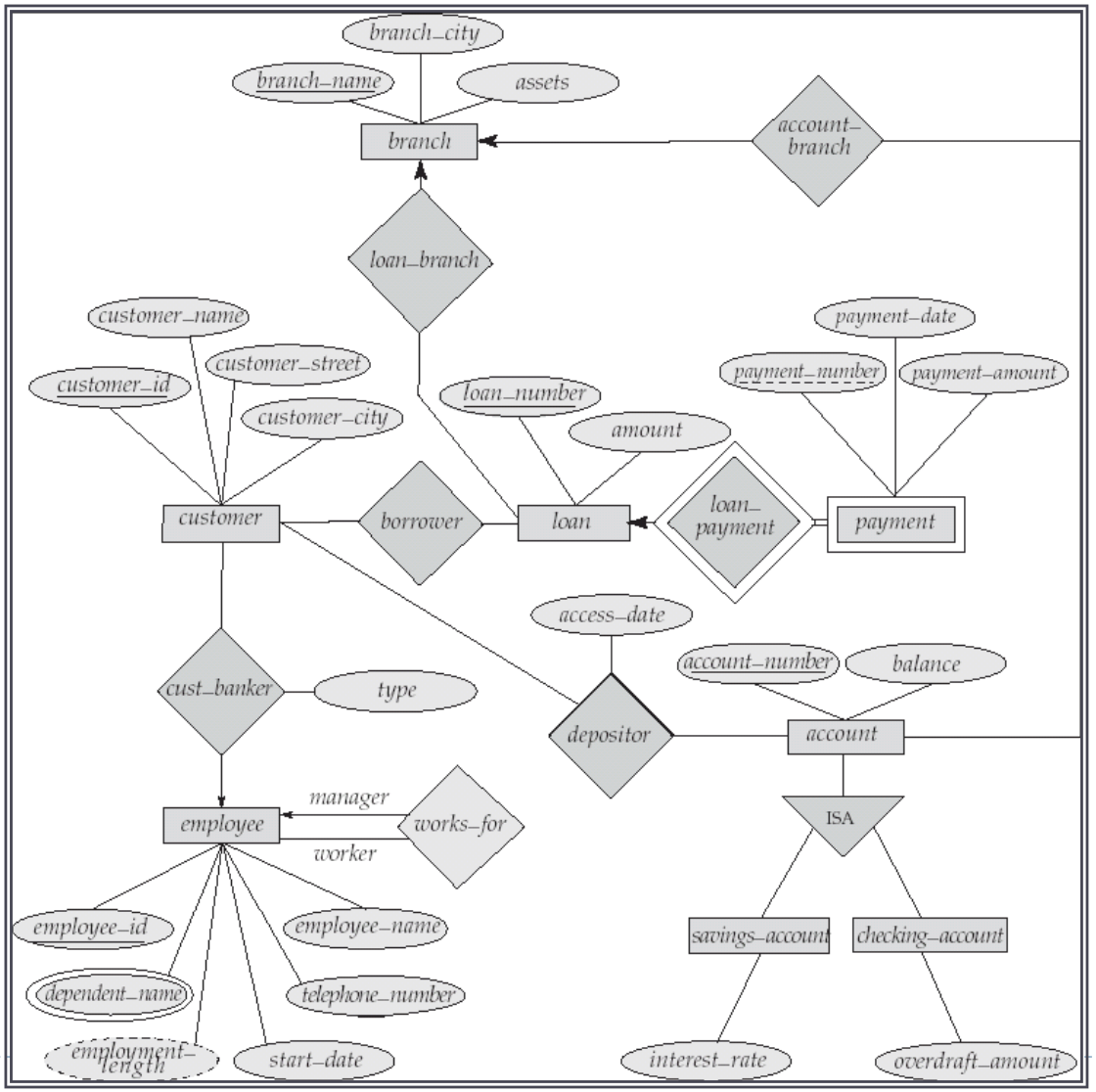
E-R Diagram With Aggregation



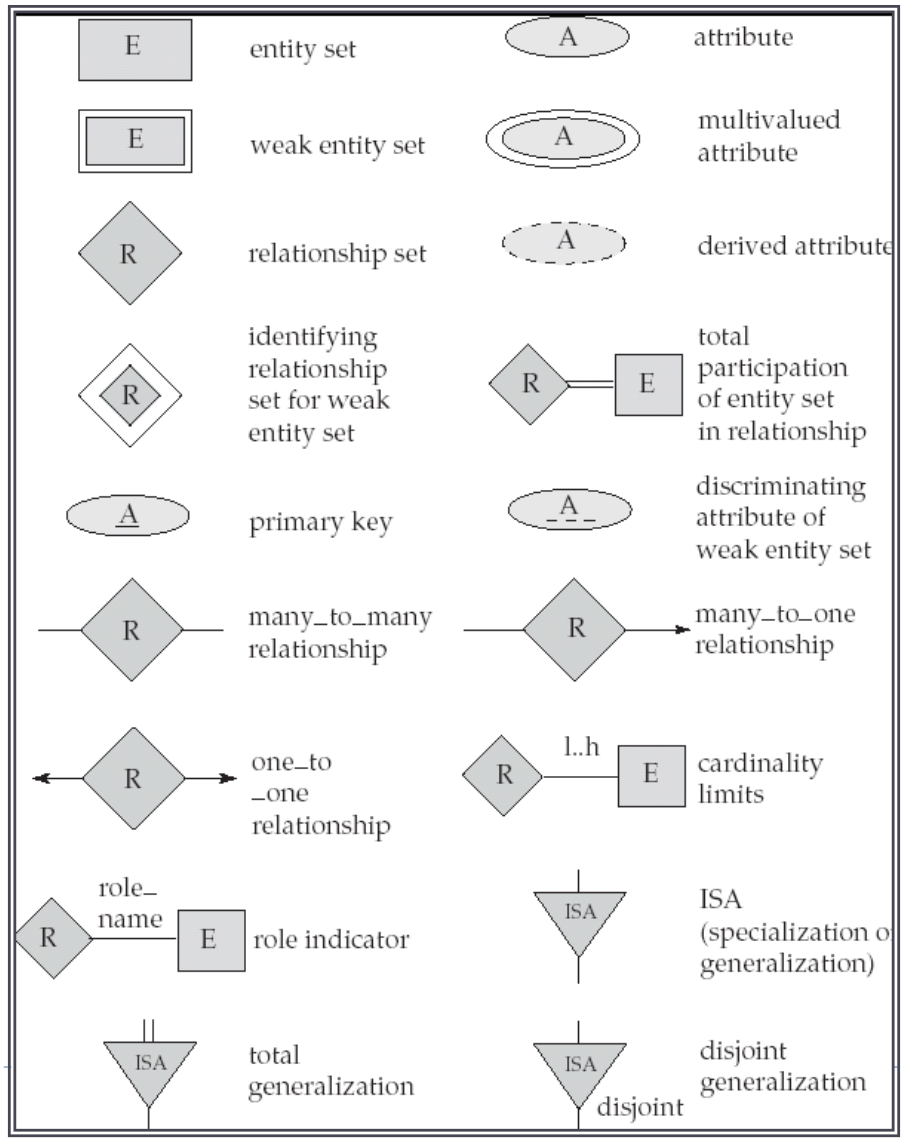
E-R Design Decisions

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

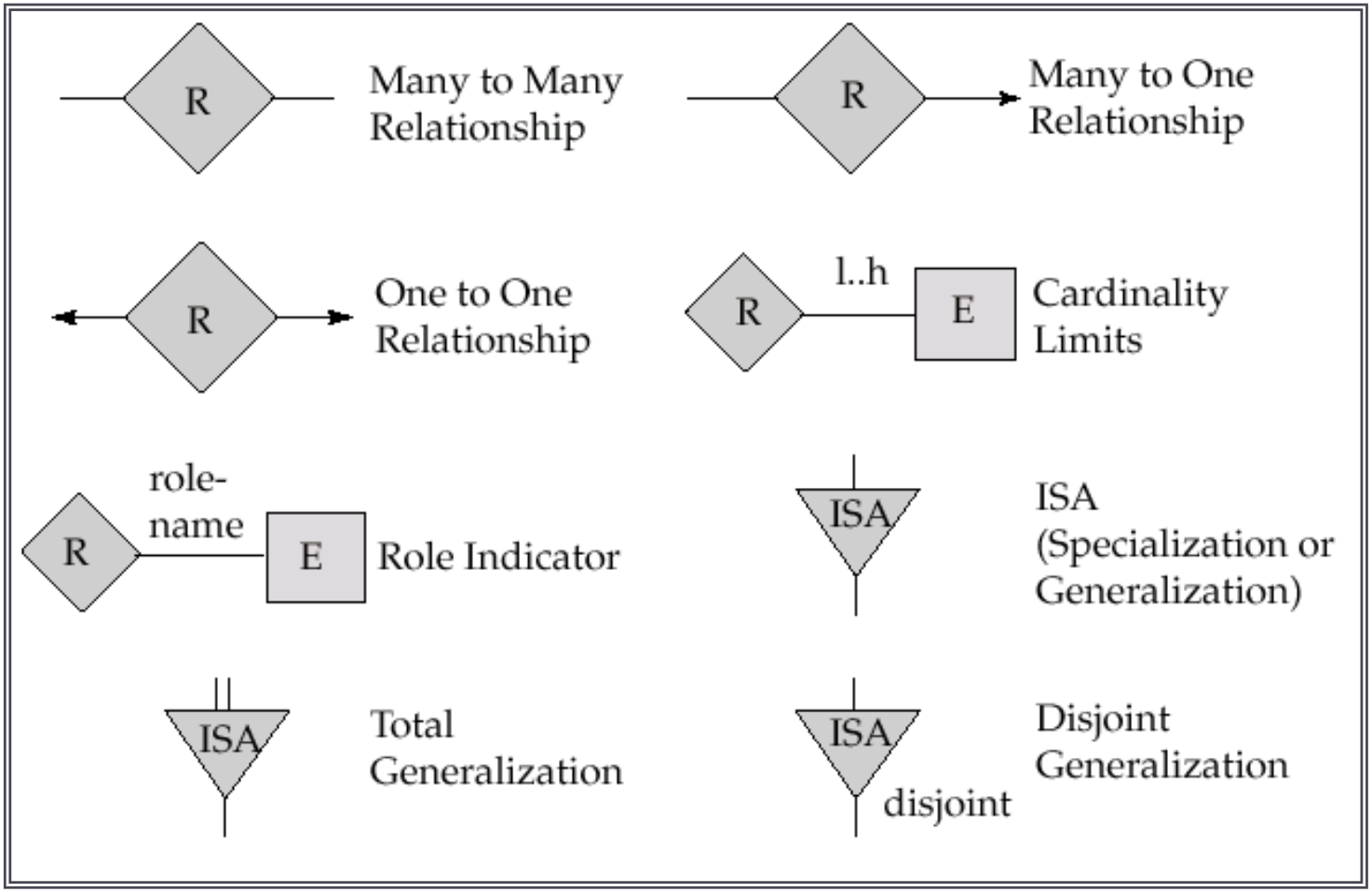
E-R Diagram for a Banking Enterprise



Summary of Symbols Used in E-R Notation

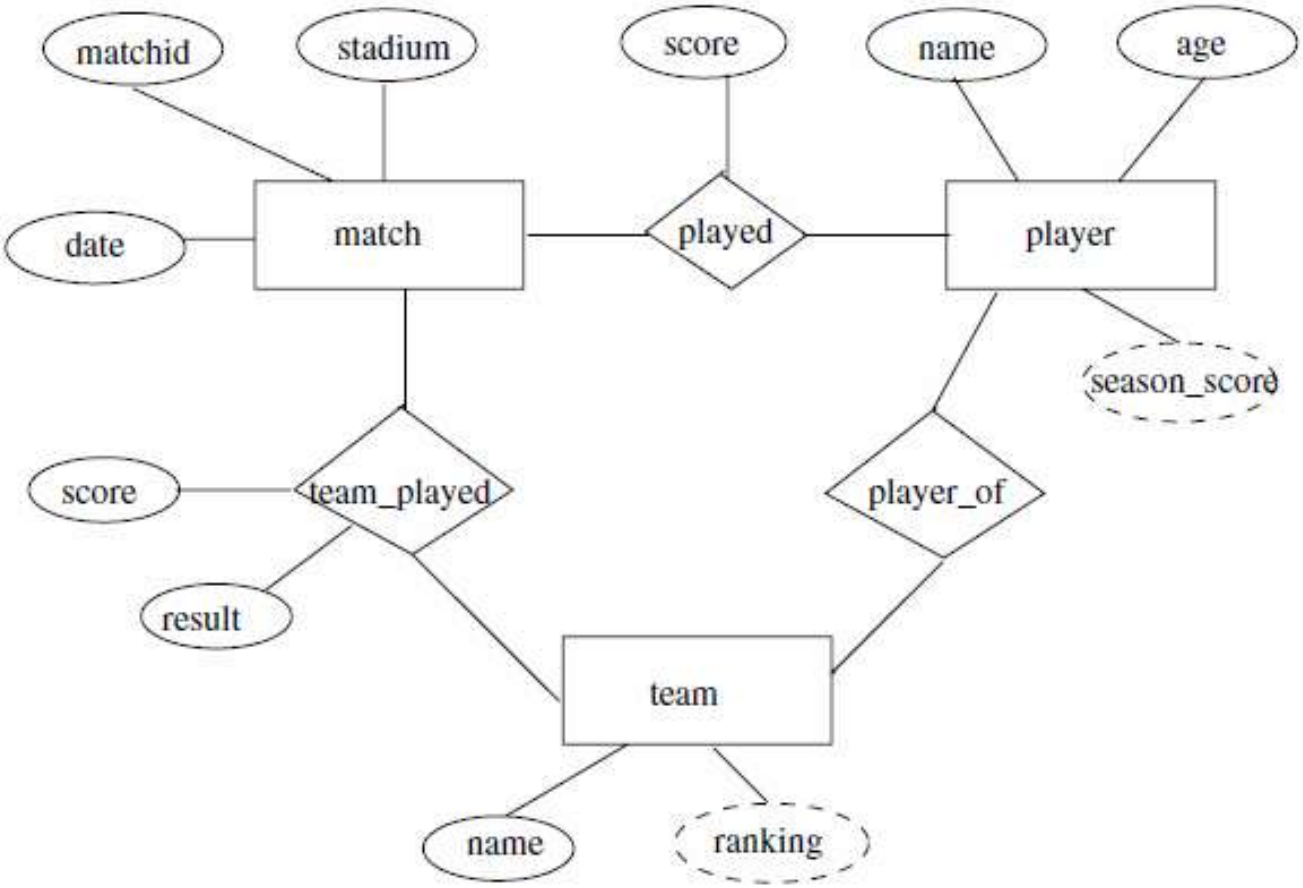


Summary of Symbols (Cont.)



Exercise

Design an E-R diagram for keeping track of the exploits of your favourite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modeled as derived attributes.


matchid
stadium
score
name
age
match
played
player
date
season_score
score
team_played
player_of
result
team
name
ranking