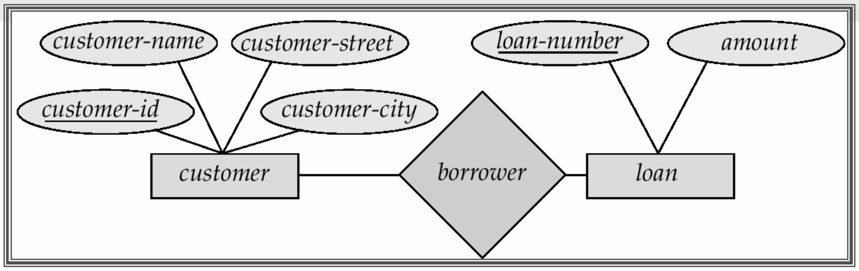
Chapter 2: Entity-Relationship Model

- What's the use of the E-R 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

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)

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

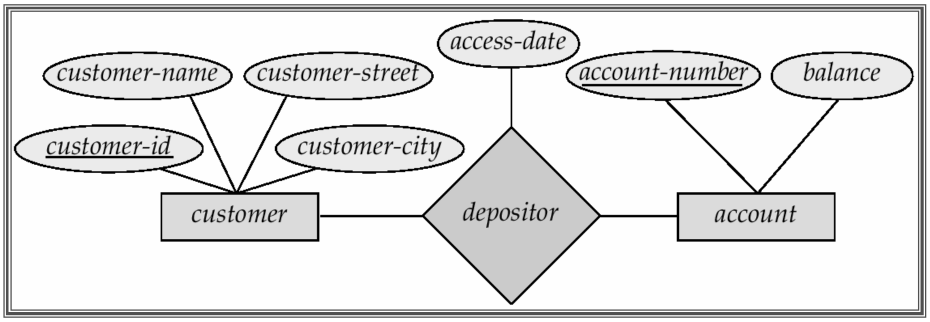
Entity Sets

- 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.

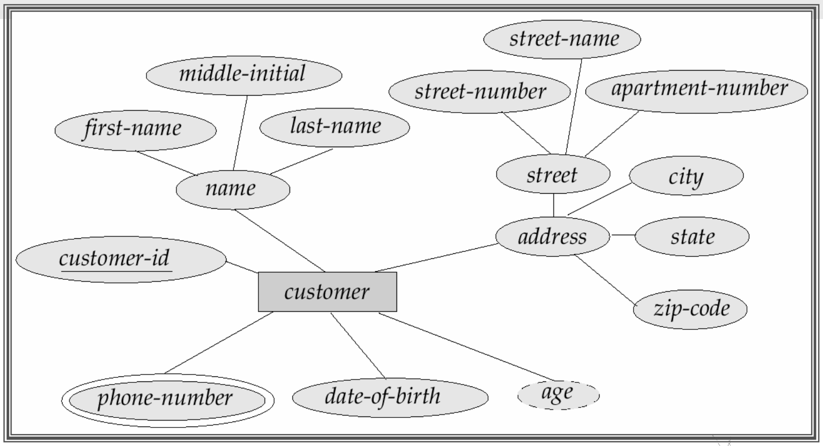
  Example: specific person, company, event, plant
- Entities have *attributes*
  Example: people have *names* and *addresses*
- An *entity set* is a set of entities of the same type that share the same properties.

  Example: set of all persons, companies, trees, holidays

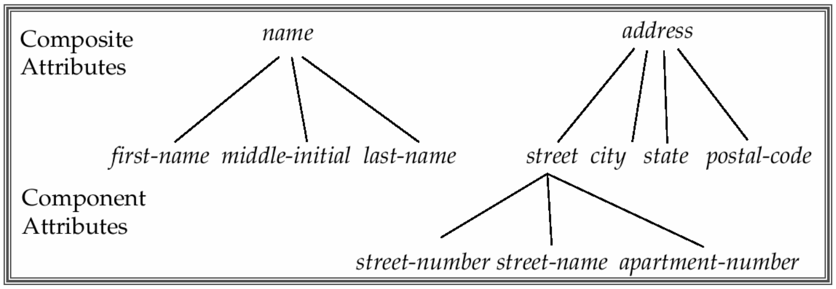
Relationship Sets with Attributes



E-R Diagram With Composite, Multivalued, and Derived Attributes—try to avoid them

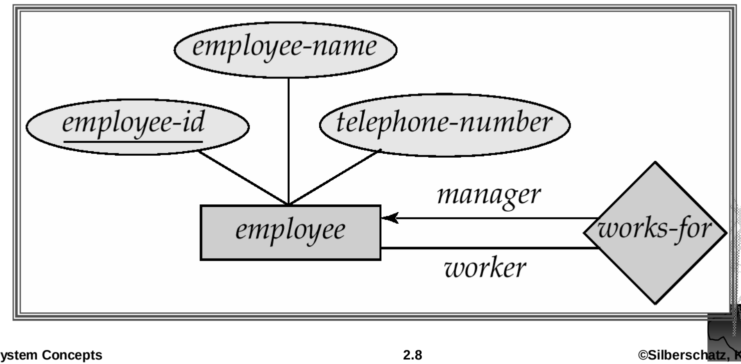


Composite Attributes



Roles

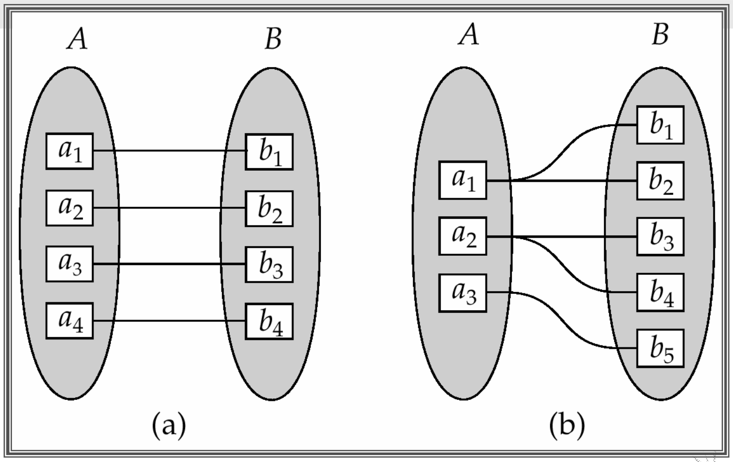
- Entity sets of a relationship need not be distinct
- The labels "manager" and "worker" are called roles; they specify how employee entities interact via the works-for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are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



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

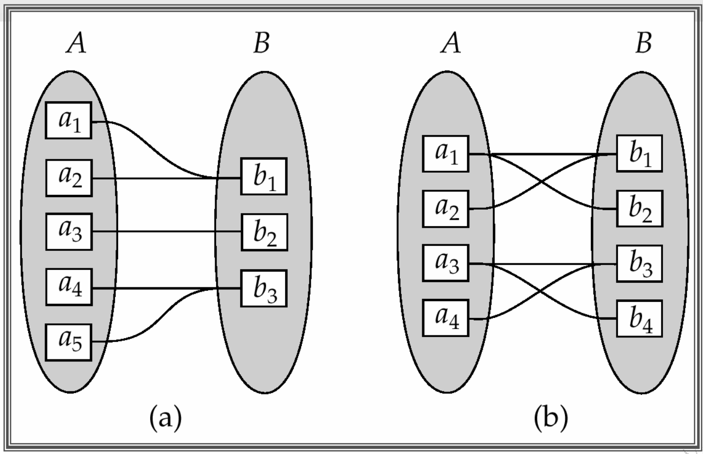


One to one

One to many

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

Mapping Cardinalities



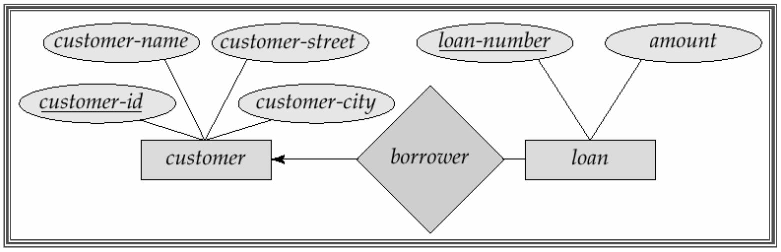
Many to one

Many to many

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

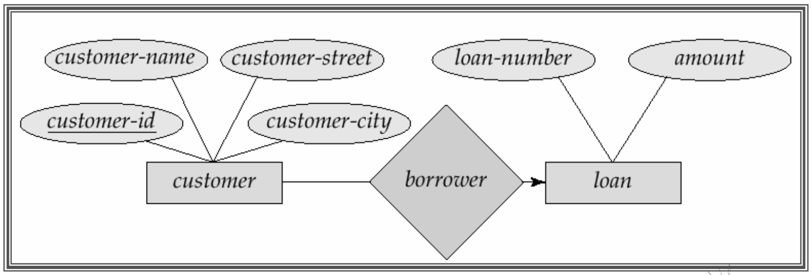
One-To-Many Relationship

- 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*



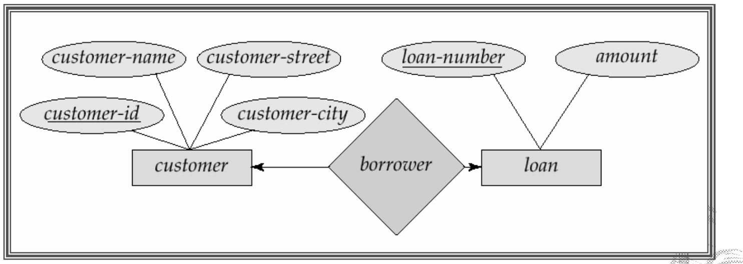
Many-To-One Relationships

- Example of many-to-one relationships: a loan is associated with several (including 0) customers via *borrower*, a customer is associated with at most one loan via *borrower*

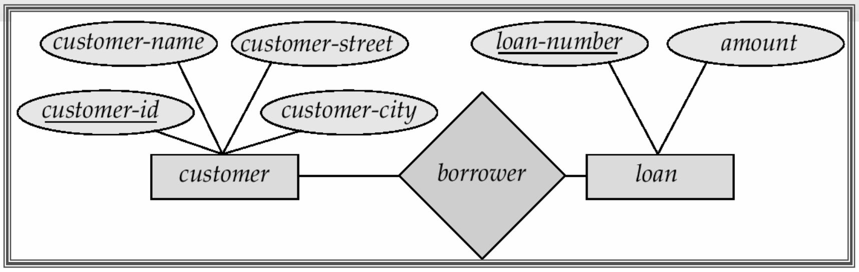


Cardinality Constraints

- 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.
- Example of 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*



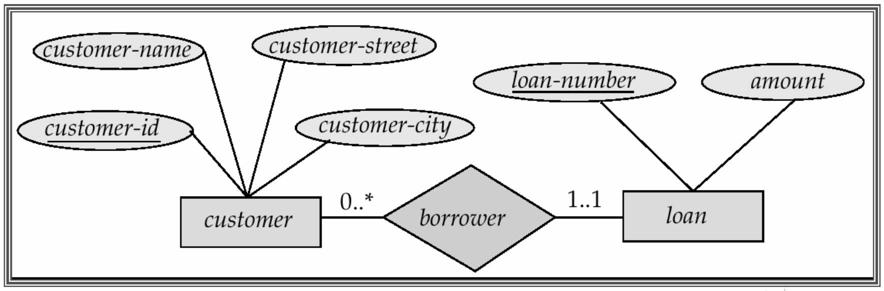
Many-To-Many Relationship



- Example of Many to Many Relationships:
  - 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



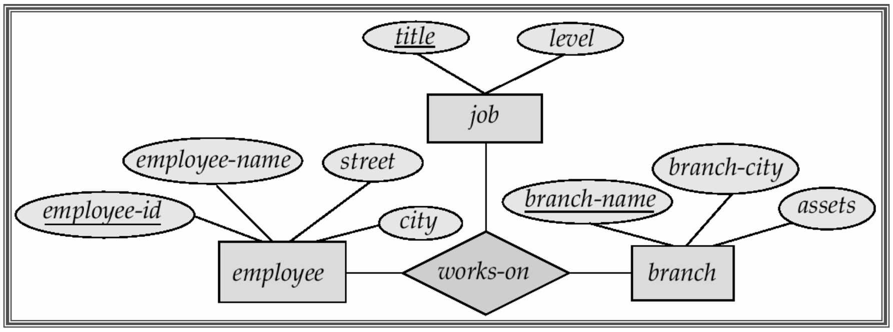
Keys

- 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*.

Degree of a Relationship Set

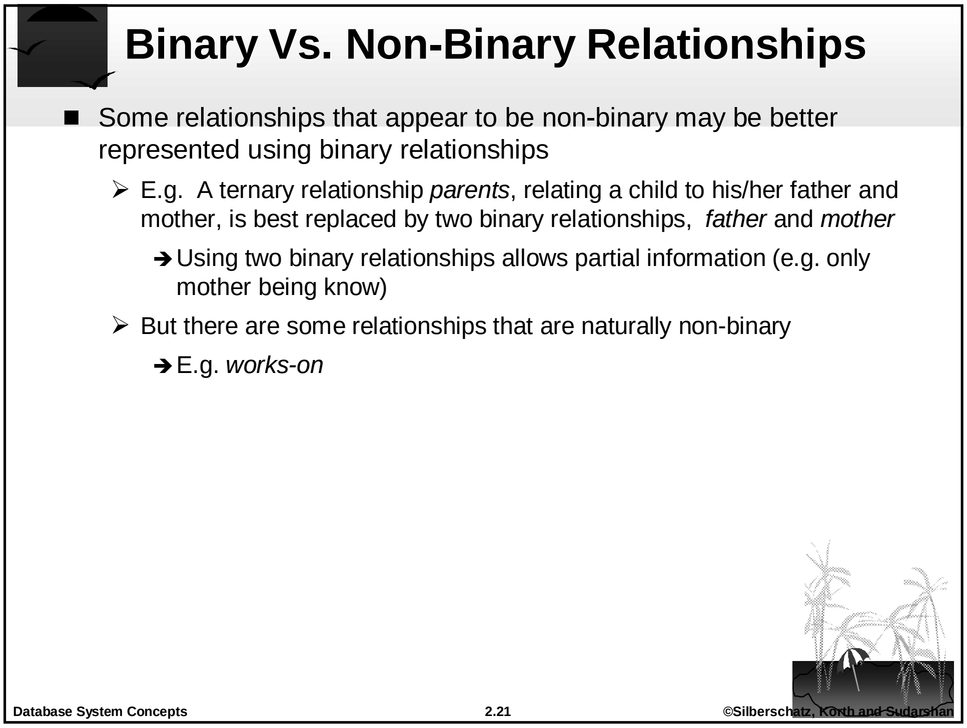
- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are *binary* (or degree two). Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets.
  - E.g. Suppose employees 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 not as common as binary ones.

E-R Diagram with a Ternary Relationship



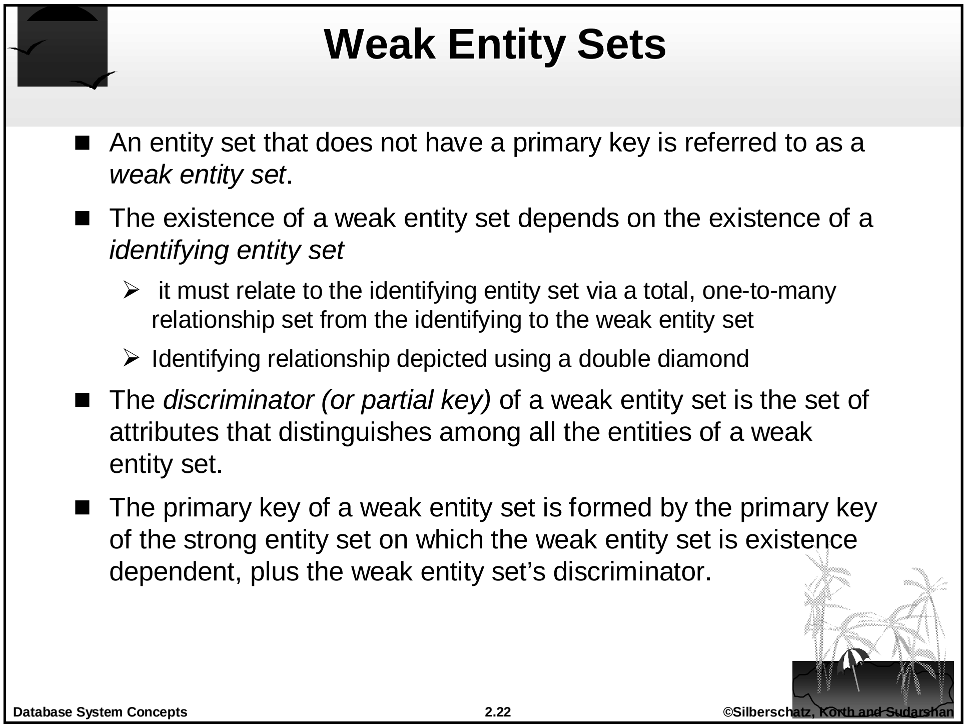
Cardinality Constraints on Ternary Relationship

- 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 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. 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
    - ➔ E.g. *works-on*

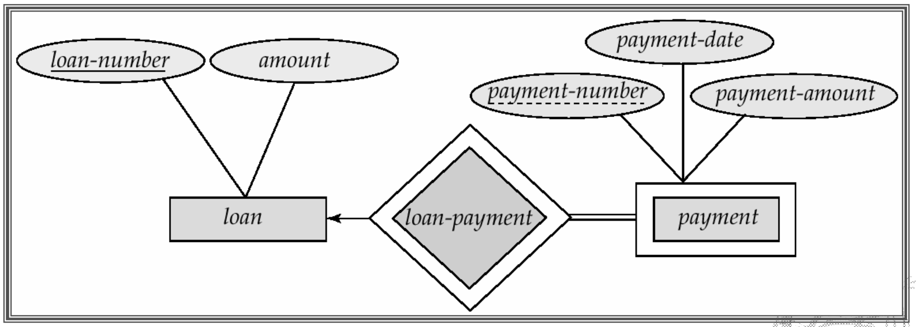


Weak Entity Sets

- 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*

More Weak Entity Set Examples

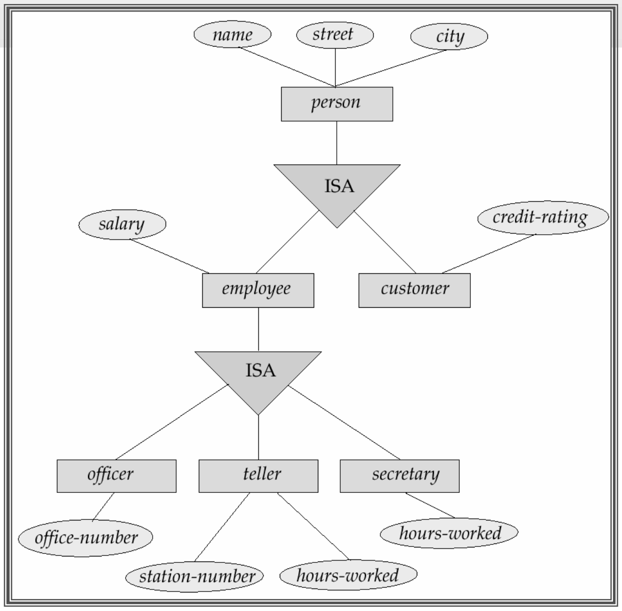
- In a university, a *course* is a strong entity and a *course-offering* can be modeled as a weak entity
- The discriminator of *course-offering* would be *semester* (including year) and *section-number* (if there is more than one section)
- If we model *course-offering* as a strong entity we would model *course-number* as an attribute.

  Then the relationship with *course* would be implicit in the *course-number* attribute

Specialization

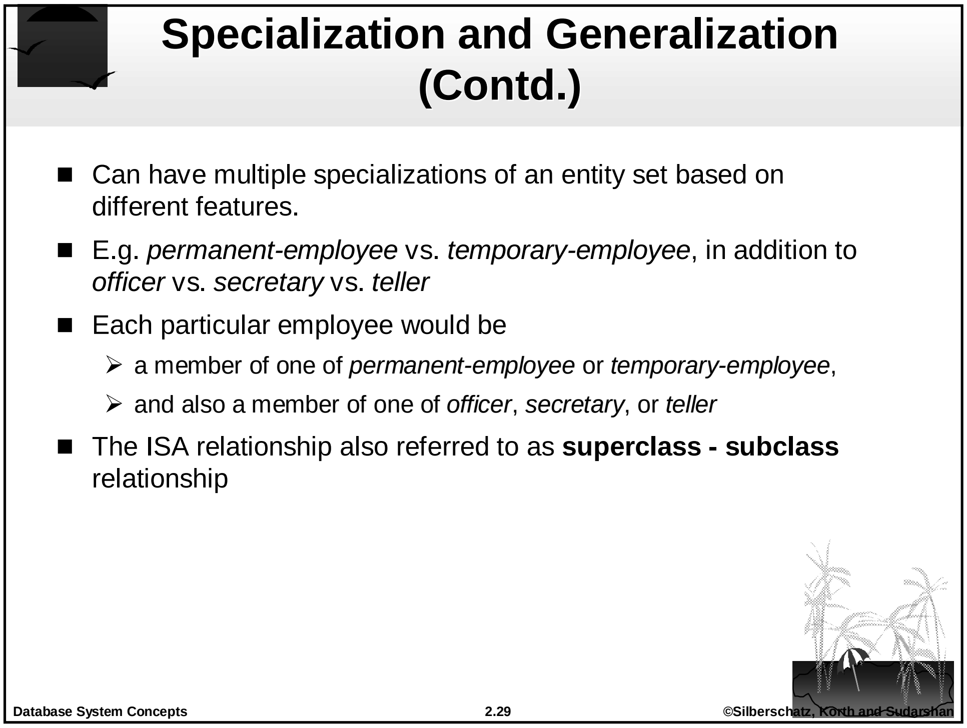
- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* 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



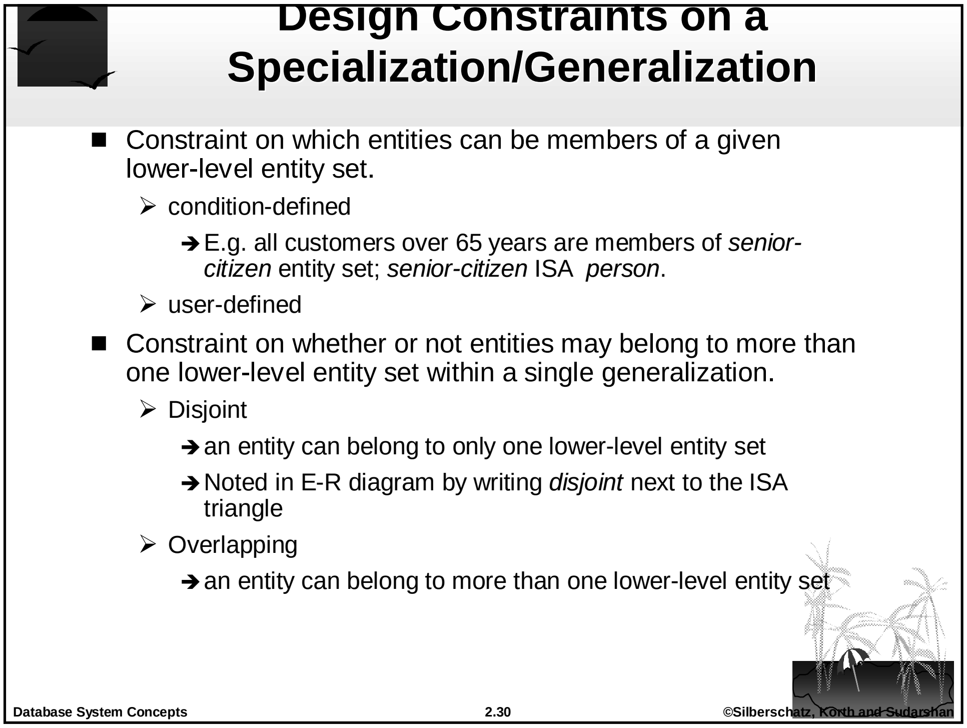
Generalization

- 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

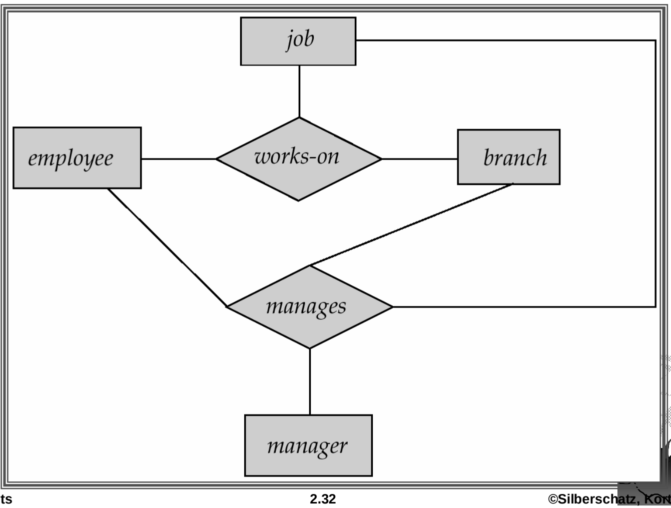
- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - ➔ E.g. all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
  - user-defined
- Constraint on whether or not entities may belong to more than one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

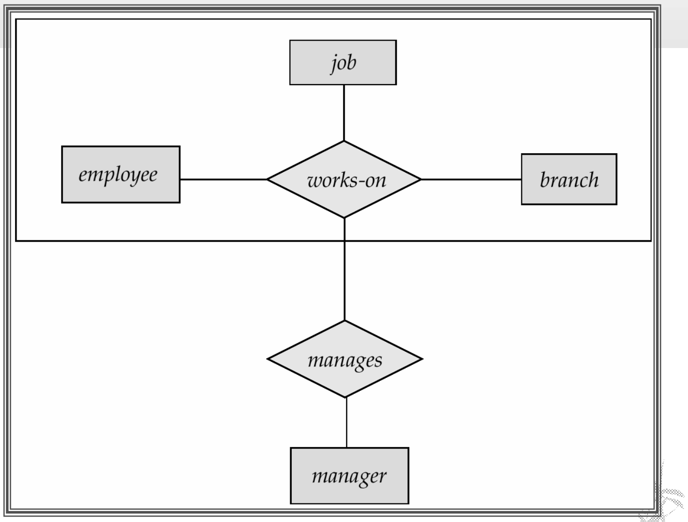
- Consider the ternary relationship *works-on*, which we saw earlier
- Suppose we want to record managers for tasks performed by an employee at a branch



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

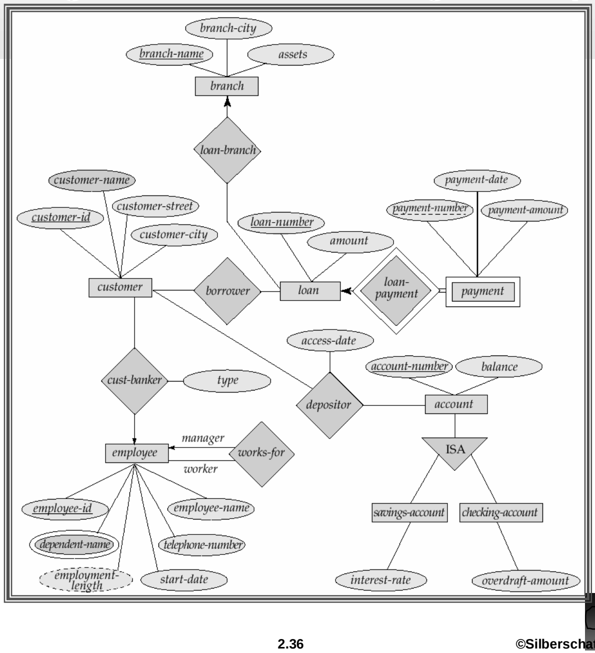
E-R Diagram With Aggregation



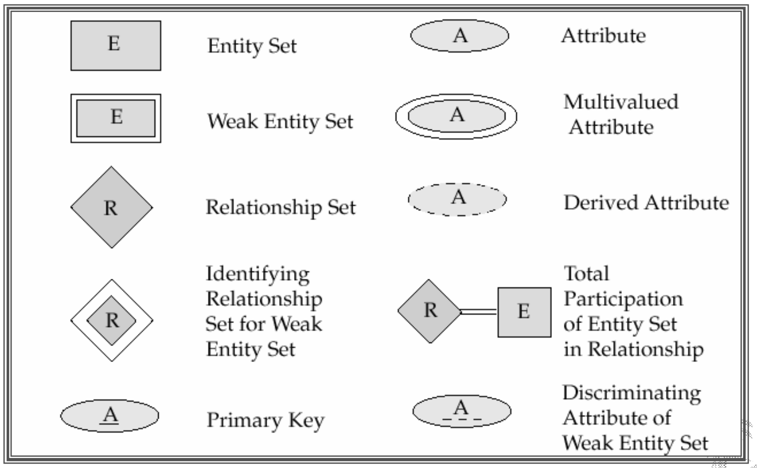
E-R Design Decisions

- 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



Entity Set

Attribute

Weak Entity Set

Multivalued Attribute

Relationship Set

Derived Attribute

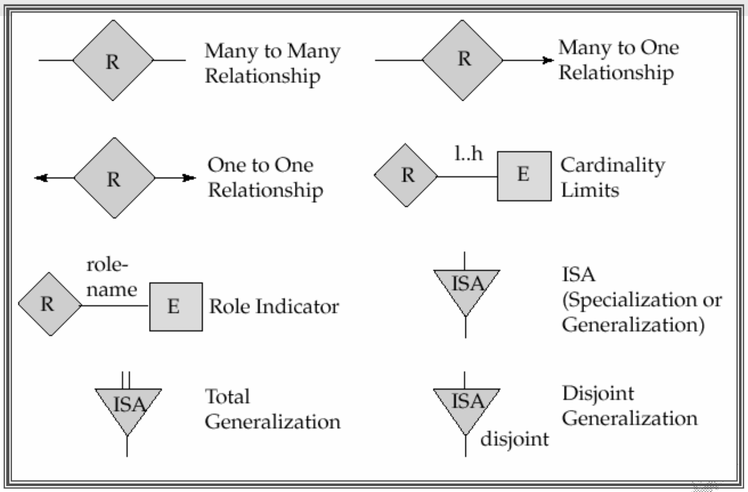
Identifying Relationship Set for Weak Entity Set

Total Participation of Entity Set in Relationship

Primary Key

Discriminating Attribute of Weak Entity Set

Summary of Symbols (Cont.)



Many to Many Relationship

Many to One Relationship

One to One Relationship

Cardinality Limits

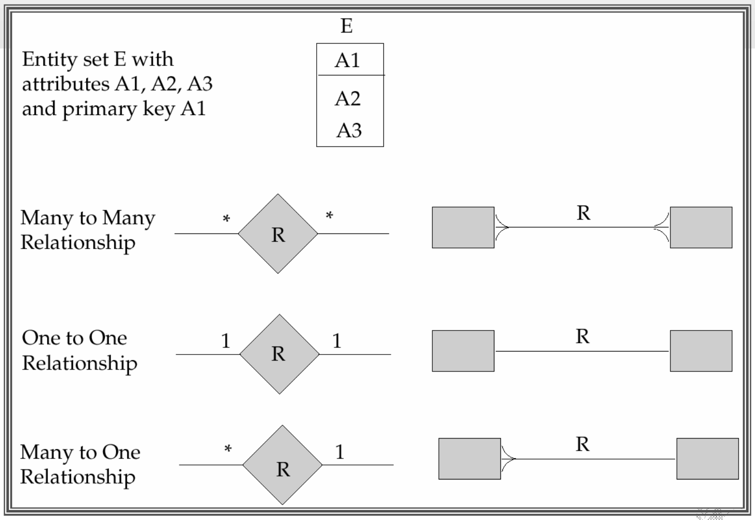
Role Indicator

ISA (Specialization or Generalization)

Total Generalization

Disjoint Generalization

Alternative E-R Notations

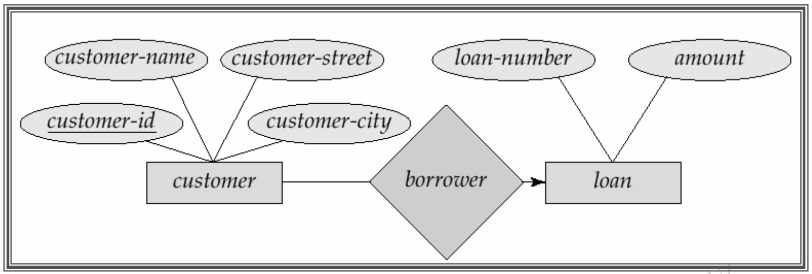

E
Entity set E with attributes A1, A2, A3 and primary key A1
A1
A2
A3
Many to Many Relationship
*
R
*
R
One to One Relationship
1
R
1
R
Many to One Relationship
*
R
1
R

Reduction of an E-R Schema to Tables

1. A database which conforms to an E-R diagram can be represented by a collection of tables
2. For each (strong) entity set there is a table having as candidate key the key of the entity set
3. For relationship set there is a table having as columns the keys of the participating entities. The candidate key for the table is determined by the cardinality constraints among participating entities.
4. A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
5. Inheritance to be discussed later …

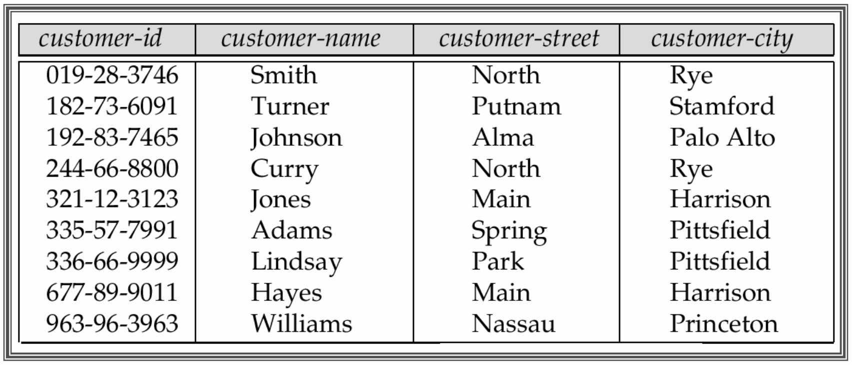
Many-To-One Relationships

- Example of many-to-one relationships: a loan is associated with several (including 0) customers via *borrower*, a customer is associated with at most one loan via *borrower*



Representing Entity Sets as Tables

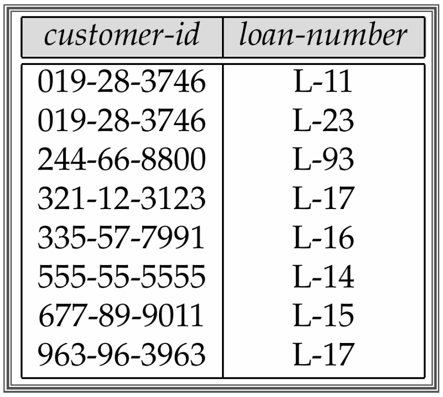
- A strong entity set reduces to a table with the same attributes.



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

Representing Relationship Sets as Tables

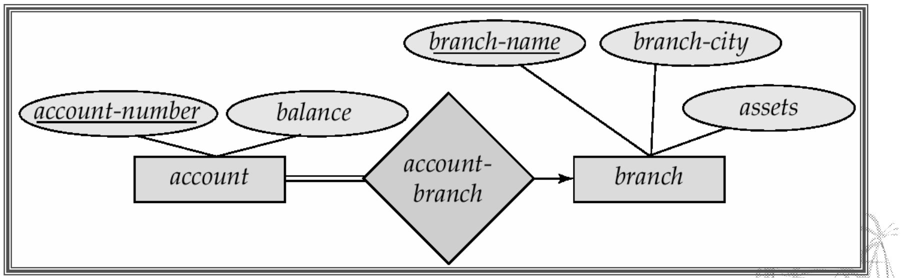
- A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- E.g.: table for relationship set *borrower*

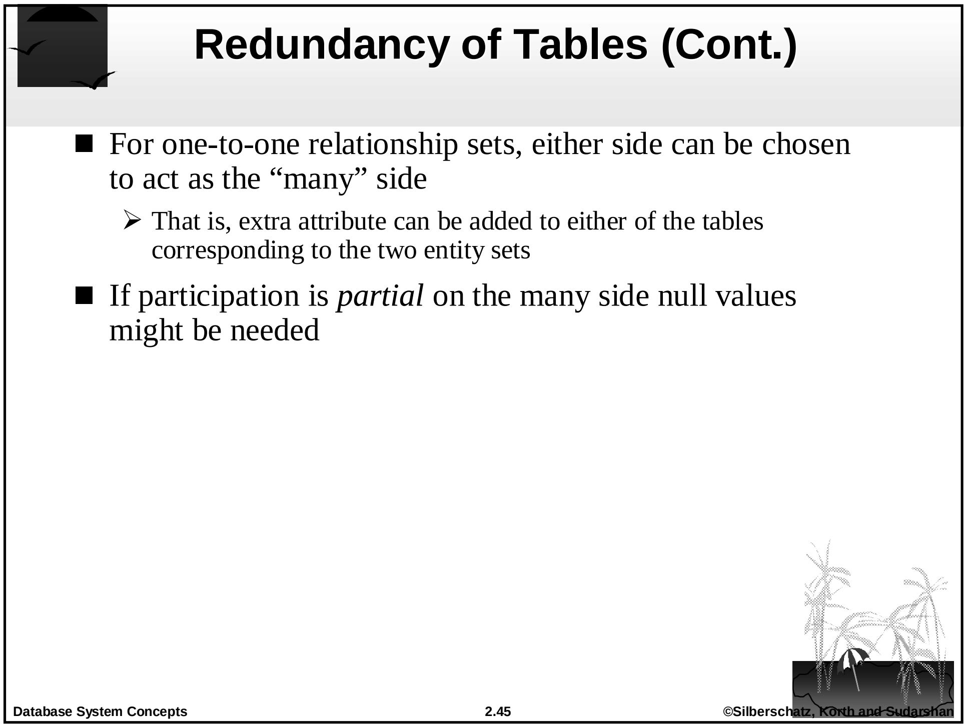


customer-id	*loan-number*
019-28-3746	L-11
019-28-3746	L-23
244-66-8800	L-93
321-12-3123	L-17
335-57-7991	L-16
555-55-5555	L-14
677-89-9011	L-15
963-96-3963	L-17

Redundancy of Tables

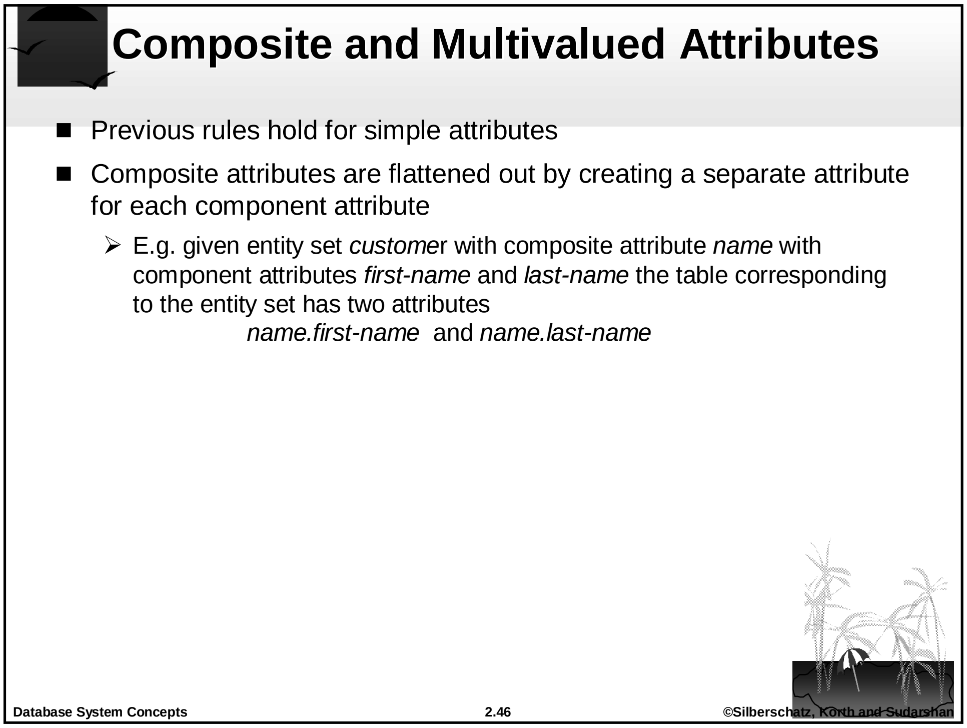
- Table with equivalent keys can be merged together---as in the 3NF design algorithm
- E.g.: Merge the tables *account-branch* with *account*





Redundancy of Tables (Cont.)

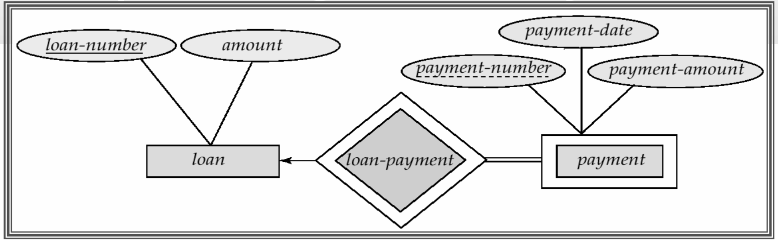
- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is *partial* on the many side null values might be needed



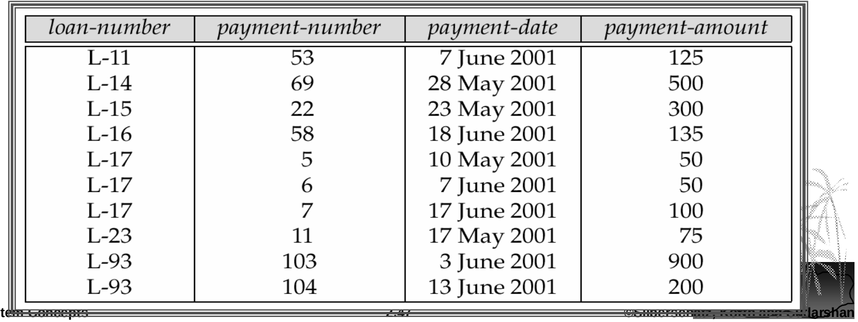
Composite and Multivalued Attributes

- Previous rules hold for simple attributes
- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - E.g. given entity set *custome*r with composite attribute *name* with component attributes *first-name* and *last-name* the table corresponding to the entity set has two attributes
    name.first-name and *name.last-name*

Representing Weak Entity Sets



- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set:



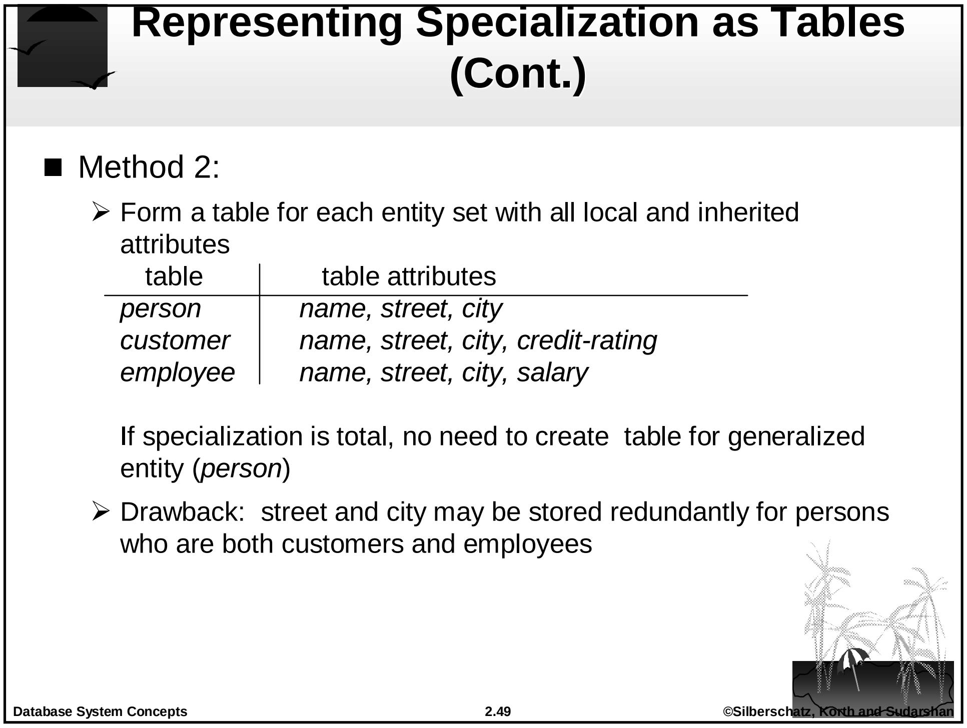
loan-number	*payment-number*	*payment-date*	*payment-amount*
L-11	53	7 June 2001	125
L-14	69	28 May 2001	500
L-15	22	23 May 2001	300
L-16	58	18 June 2001	135
L-17	5	10 May 2001	50
L-17	6	7 June 2001	50
L-17	7	17 June 2001	100
L-23	11	17 May 2001	75
L-93	103	3 June 2001	900
L-93	104	13 June 2001	200

Representing Specialization as Tables

- Method 1:
  - Form a table for the higher level entity
  - Form a table for each lower level entity set, include primary key of higher level entity set and local attributes

table	table attributes
person	*name, street, city*
customer	*name, credit-rating*
employee	*name, salary*

  - Drawback: getting information about, e.g., *employee* requires accessing two tables



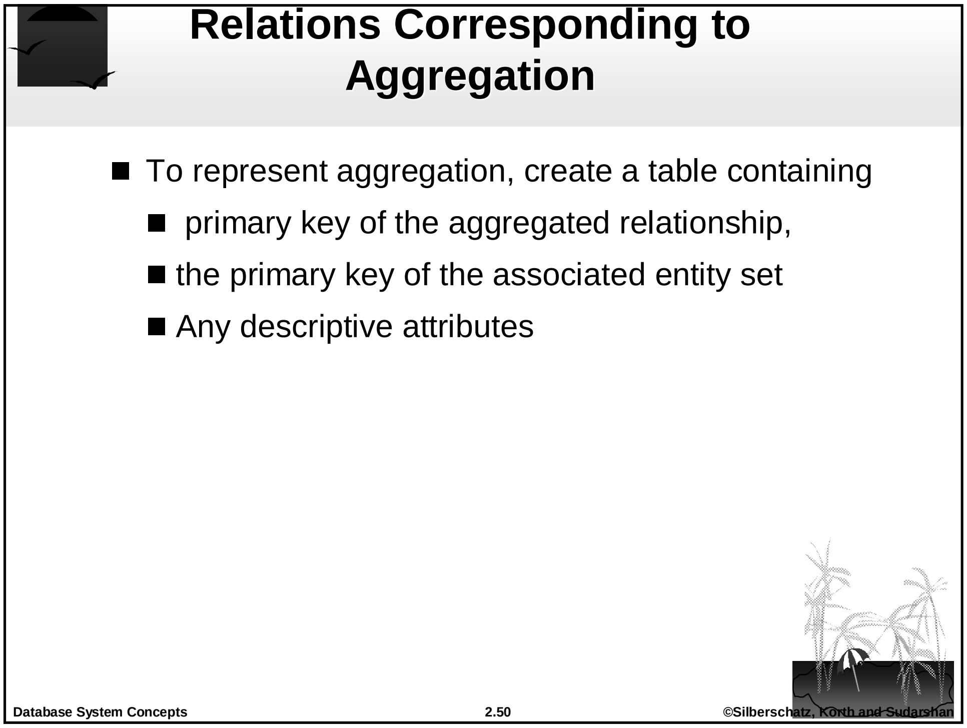
Representing Specialization as Tables (Cont.)

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

table	table attributes
person	*name, street, city*
customer	*name, street, city, credit-rating*
employee	*name, street, city, salary*

  If specialization is total, no need to create table for generalized entity (*person*)

  - Drawback: street and city may be stored redundantly for persons who are both customers and employees

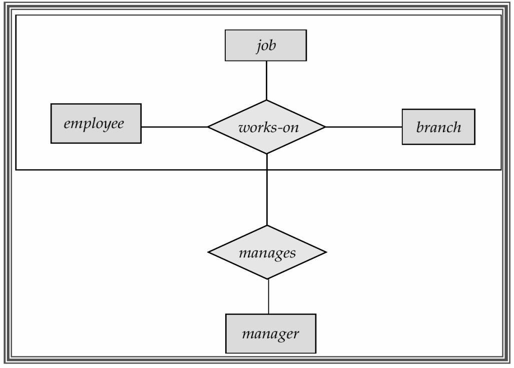


Relations Corresponding to Aggregation

- To represent aggregation, create a table containing
  - primary key of the aggregated relationship,
  - the primary key of the associated entity set
  - Any descriptive attributes

Relations Corresponding to Aggregation (Cont.)

- E.g. to represent aggregation *manages* between relationship *works-on* and entity set *manager*, create a table
  manages(*employee-id, branch-name, title, manager-name*)
- Table *works-on* is redundant **provided** we are willing to store null values for attribute *manager-name* in table *manages*



End of Chapter 2