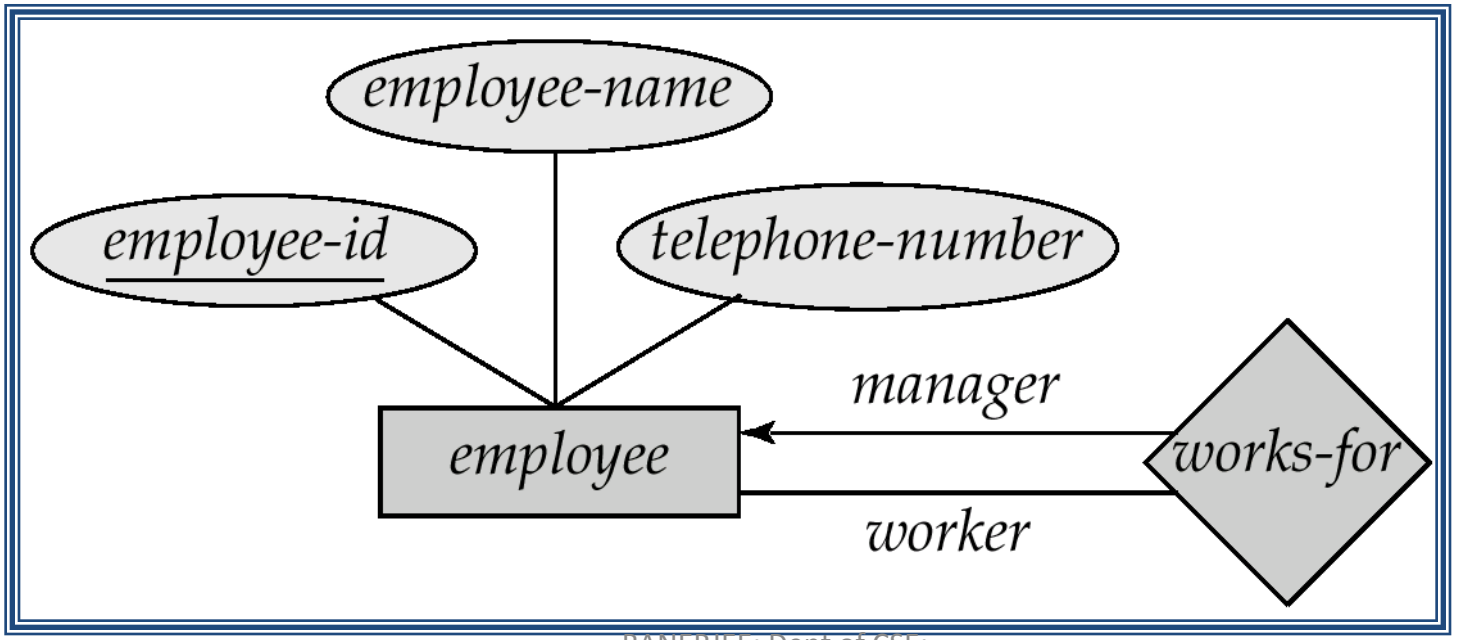
Entity-Relationship Model

1. Entity
2. Attributes
3. Entity Sets
4. Relationship Sets
5. Design Issues
6. Mapping Constraints
7. Weak Entity
8. Keys
9. E-R Diagram
10. Extended E-R Features
11. Design of an E-R Database Schema
12. Reduction of an E-R Schema to Tables

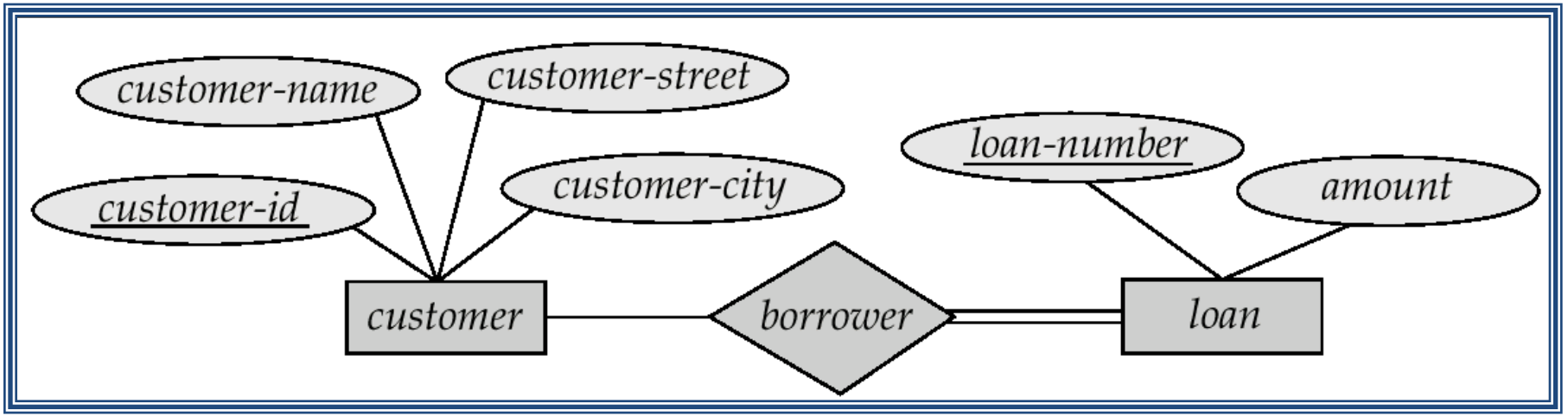
Roles

- 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

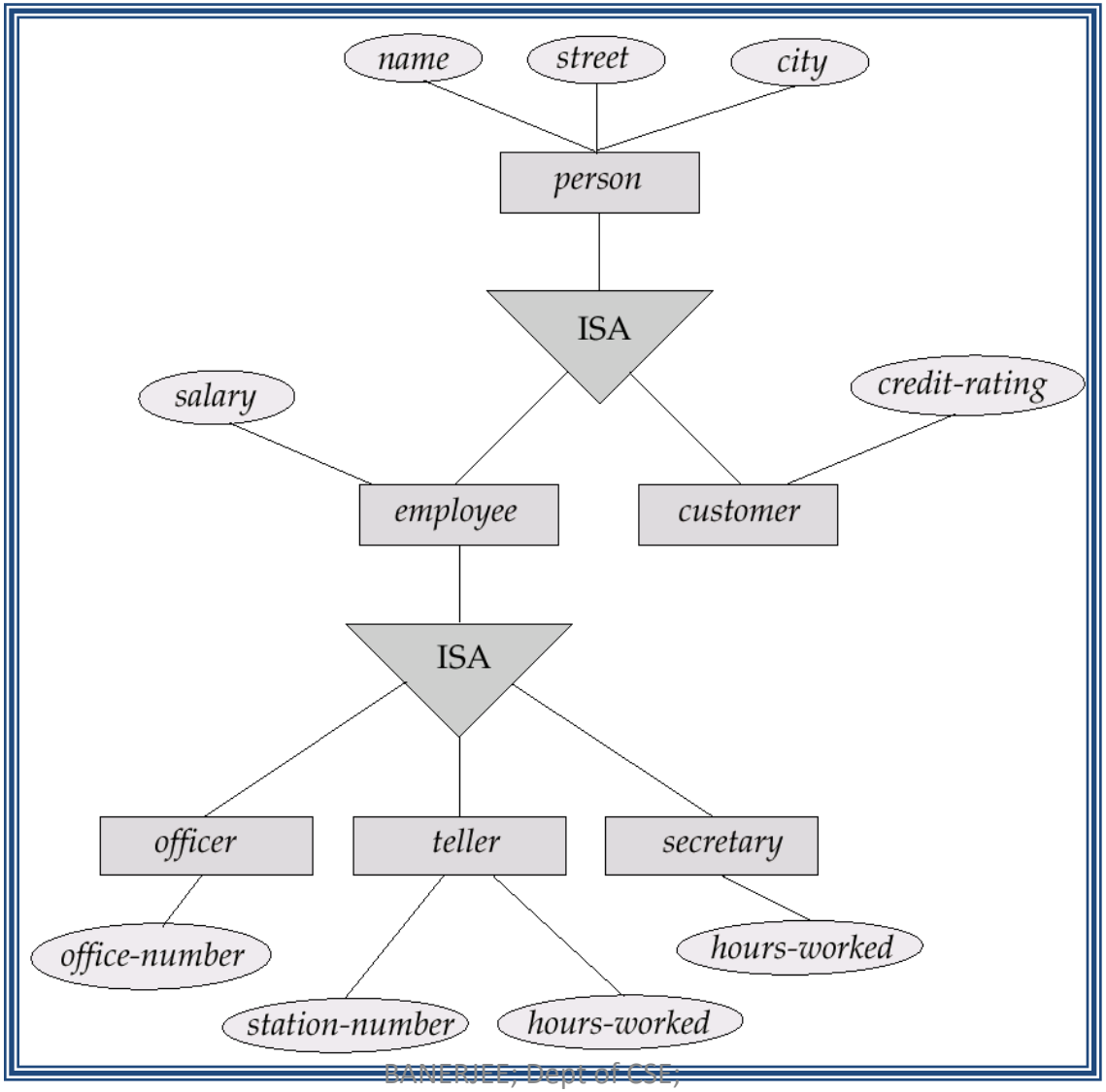
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g. participation of *loan* in *borrower* is total
    - every loan must have a customer associated to it via borrower
- Partial participation: some entities may not participate in any relationship in the relationship set
  - E.g. participation of *customer* in *borrower* is partial



Specialization

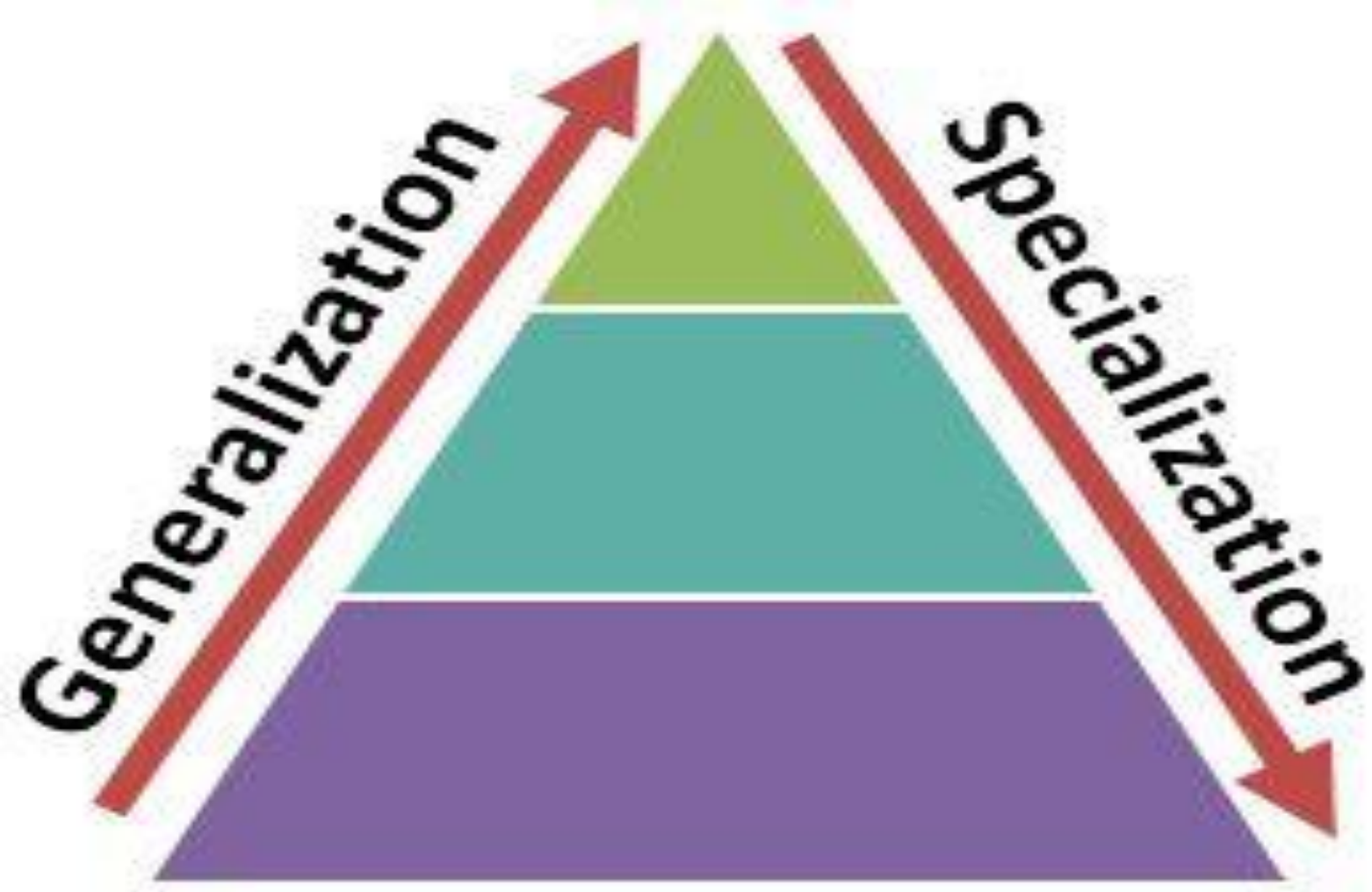
- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle*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

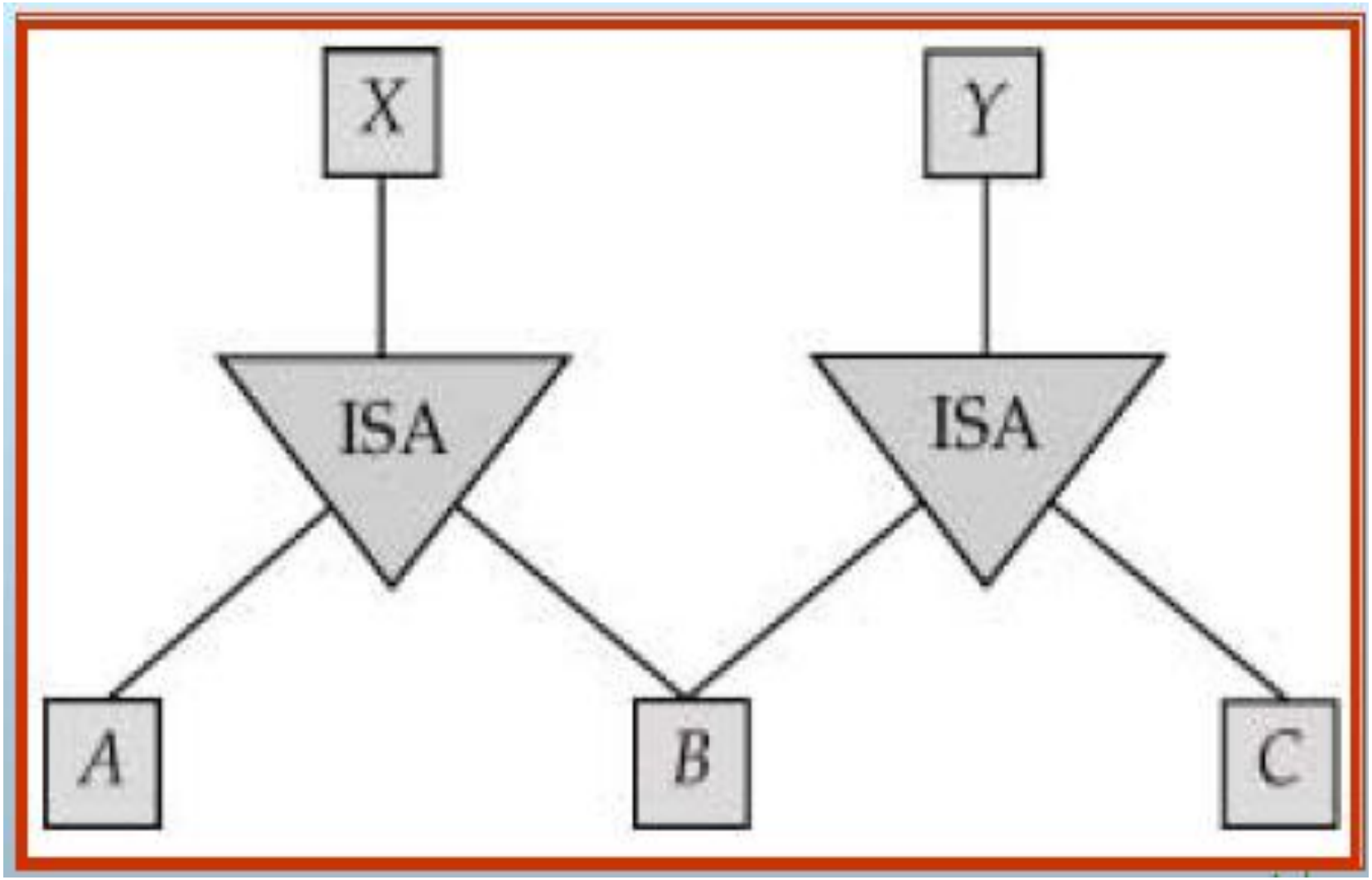


Generalization

- 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.


Generalization
Specialization

Complex Specialization and Generalization



BASIS FOR COMPARISON	GENERALIZATION	SPECIALIZATION
Basic	It proceeds in a bottom-up manner.	It proceeds in a top-down manner.
Function	Generalization extracts the common features of multiple entities to form a new entity.	Specialization splits an entity to form multiple new entities that inherit some feature of the splitting entity.
Entities	The higher level entity must have lower level entities.	The higher level entity may not have lower level entities.
Size	Generalization reduces the size of a schema.	Specialization increases the size of a schema.
Application	Generalization entities on group of entities.	Specialization is applied on a single entity.
Result	Generalization results in forming a single entity from multiple entities.	Specialization results in forming the multiple entity from a single entity.

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

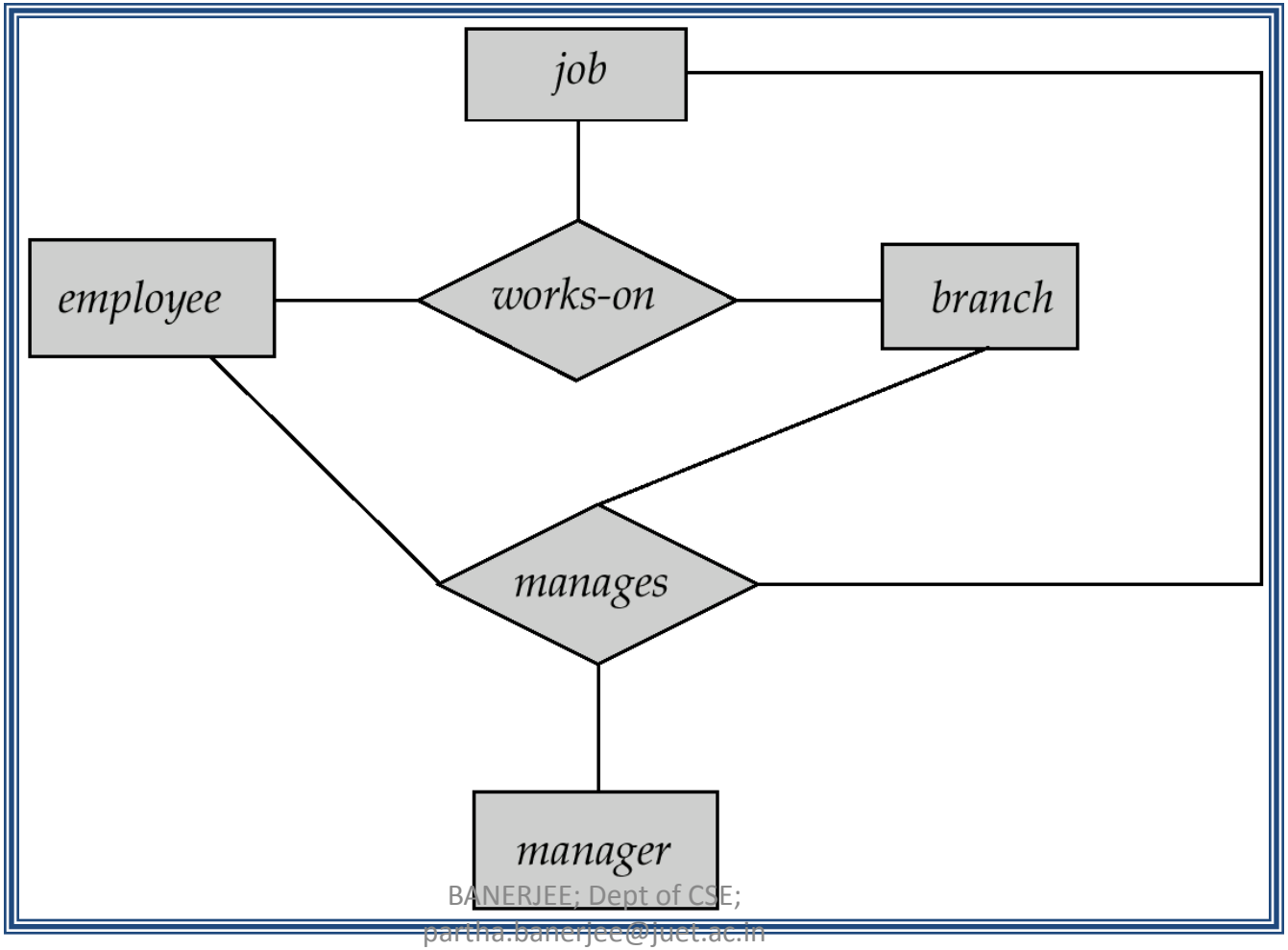
- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - E.g. all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
  - user-defined
- Constraint on whether or not entities may belong to more than one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

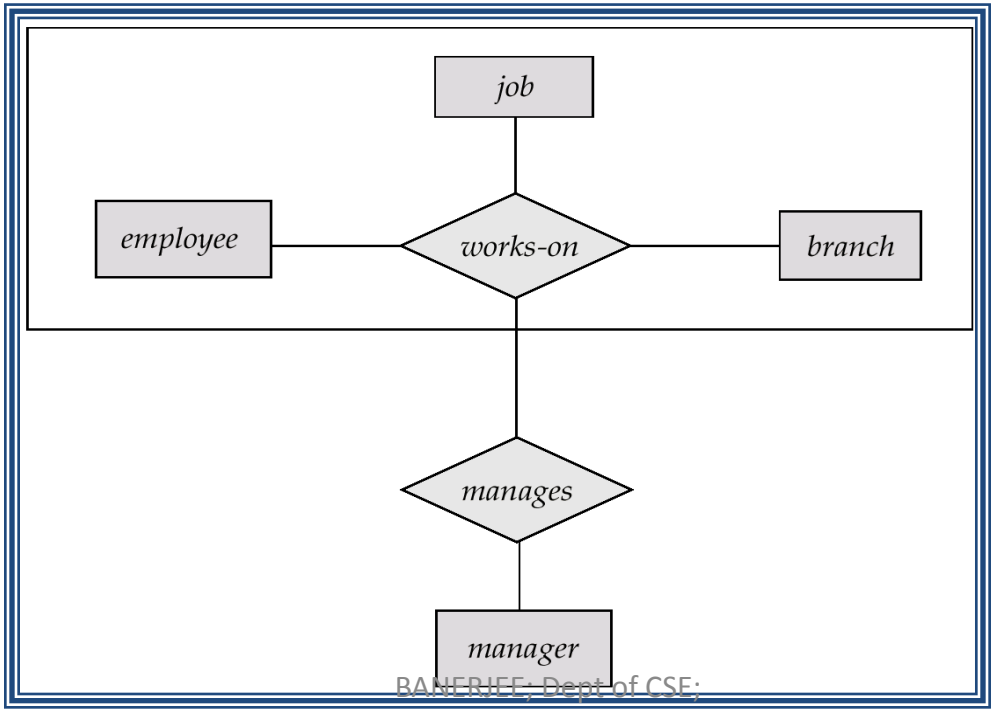
- Relationship sets *works-on* and *manages* represent overlapping information
  - Every *manages* relationship corresponds to a *works-on* relationship
  - However, some *works-on* relationships may not correspond to any *manages* relationships
    - So we can't discard the *works-on* relationship
- 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

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*



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.)

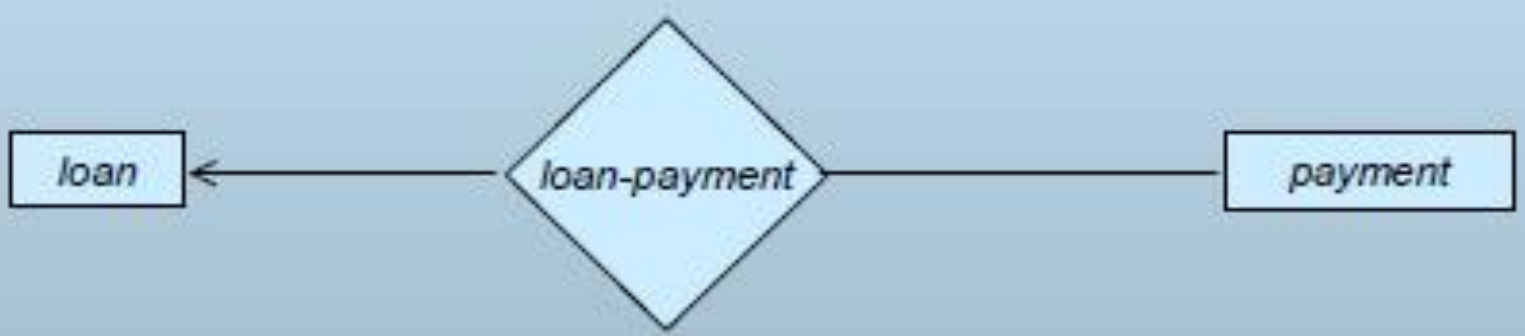
- Method 2:
  - Form a table for each entity set with all local and inherited attributes

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

  - If specialization is total, table for generalized entity (*person*) not required to store information
    - Can be defined as a "view" relation containing union of specialization tables
    - But explicit table may still be needed for foreign key constraints
  - Drawback: street and city may be stored redundantly for persons who are both customers and employees

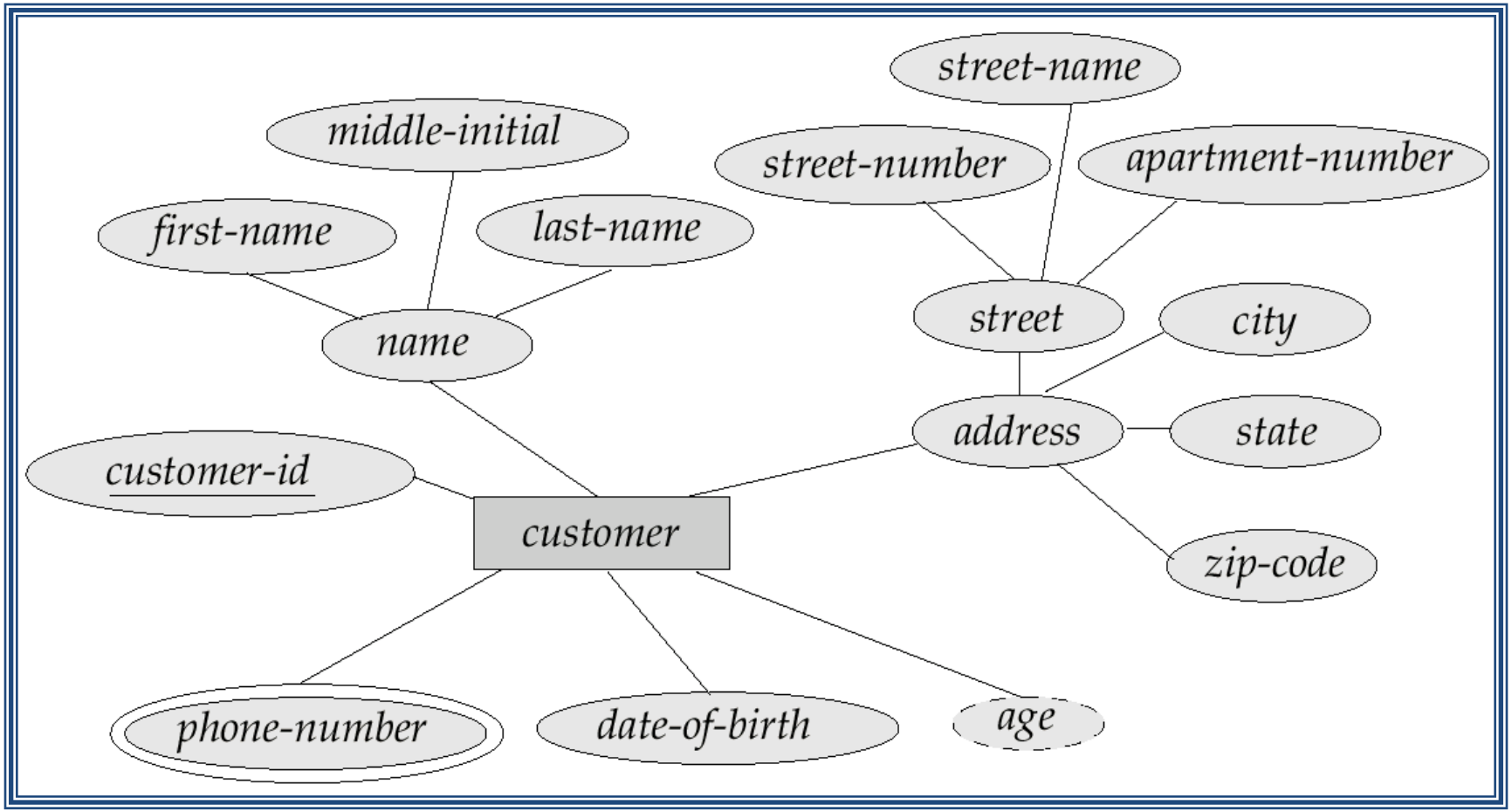
Existence Dependencies

- 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*)

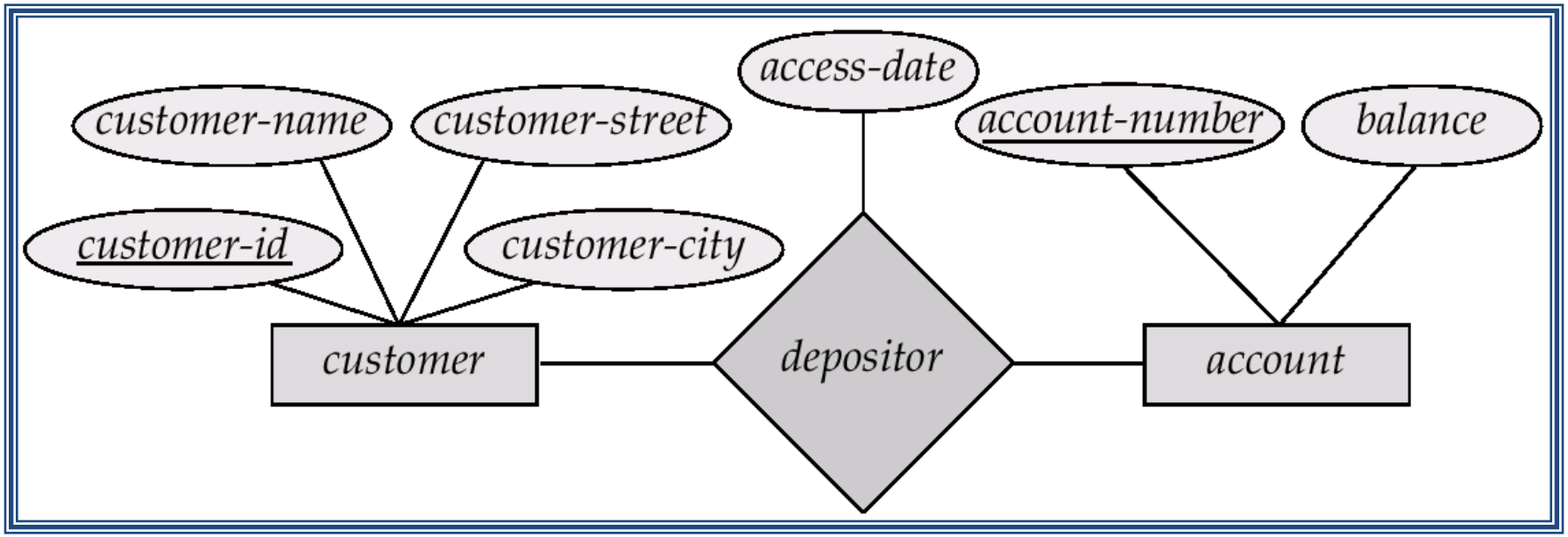


If a *loan* entity is deleted, then all its associated *payment* entities must be deleted also.

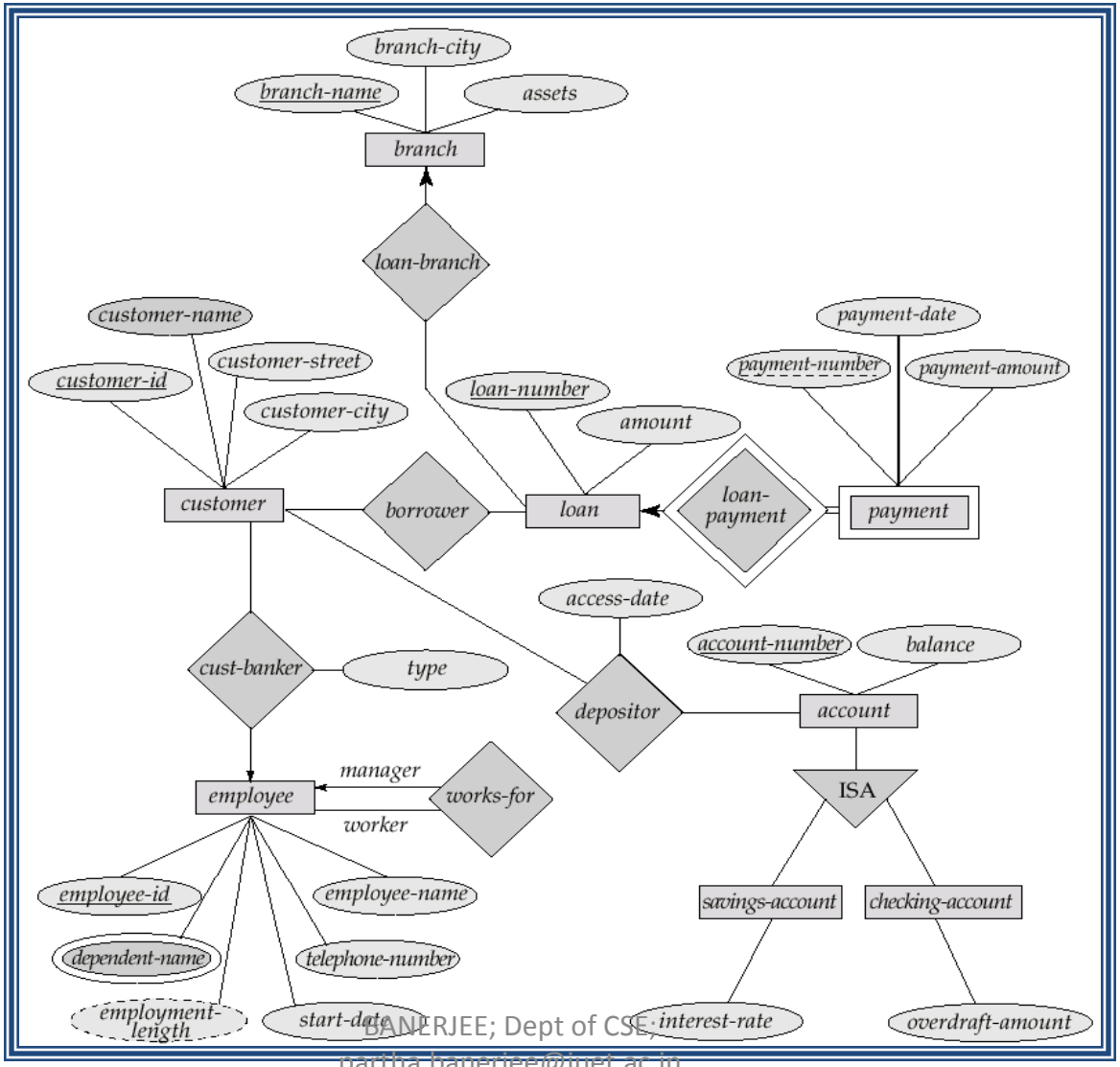
E-R Diagram With Composite, Multivalued, and Derived Attributes



Relationship Sets with Attributes

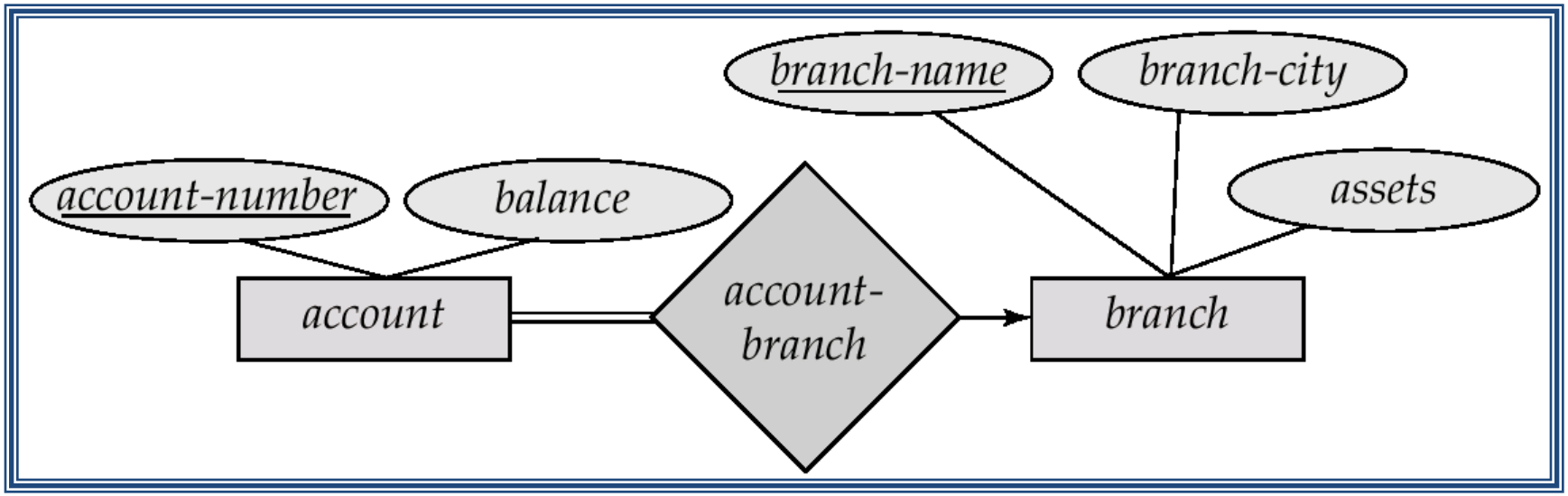


E-R Diagram for a Banking Enterprise



Redundancy of Tables

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the many side, containing the primary key of the one side
- E.g.: Instead of creating a table for relationship *account-branch*, add an attribute *branch* to the entity set *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, replacing a table by an extra attribute in the relation corresponding to the “many” side could result in null values
- The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - E.g. The *payment* table already contains the information that would appear in the *loan-payment* table (i.e., the columns loan-number and *payment-number*).