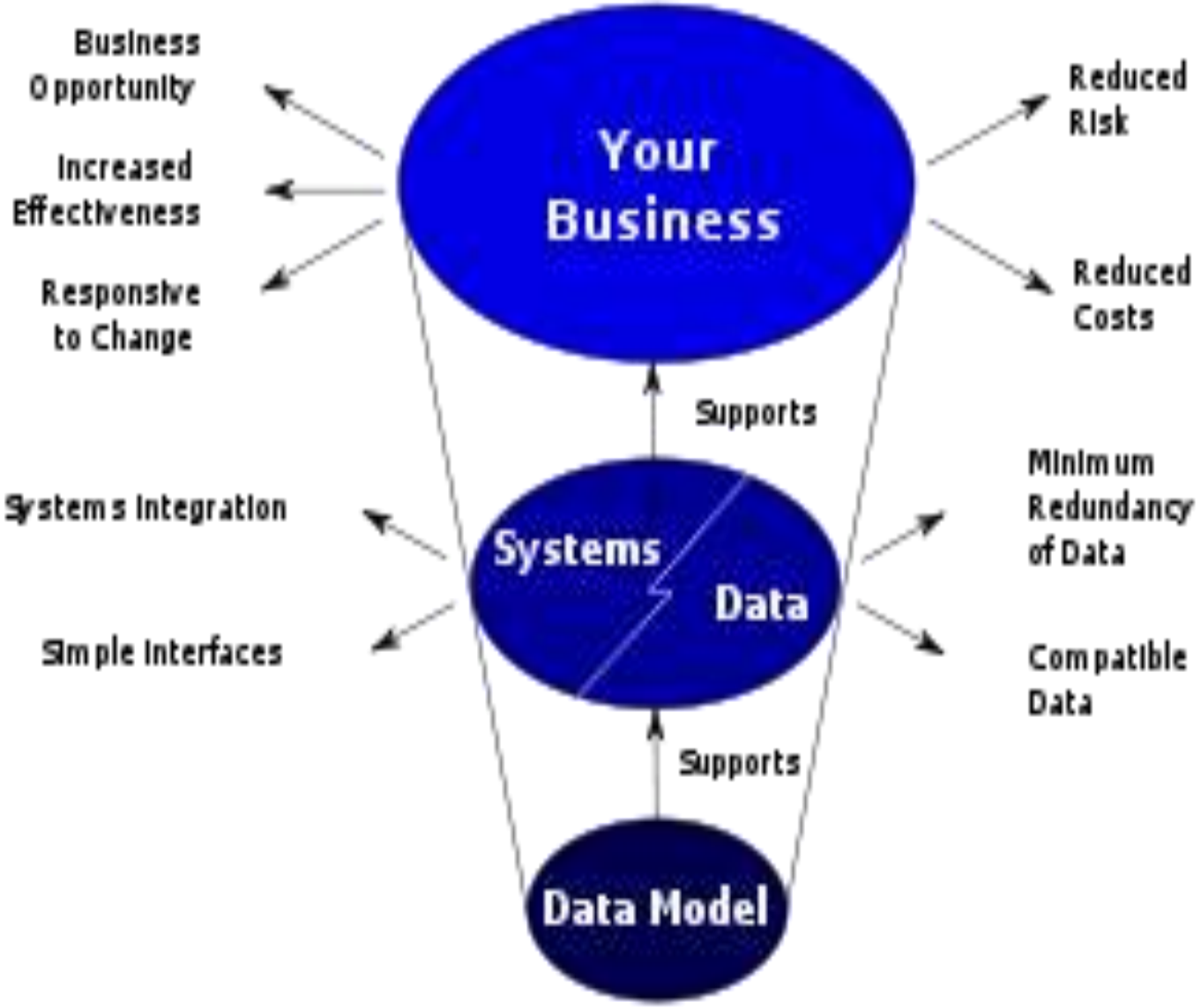
Data Models

- ✓ Managing large quantities of structured and unstructured data is a primary function of information systems. Data models describe structured data for storage in data management systems such as relational databases. They typically do not describe unstructured data, such as word processing documents, email messages, pictures, digital audio, and video.
- ✓ Data model plays an important role in database design.
- ✓ The physical or logical structure of database is spelt out by the data model.
- ✓ A collection of conceptual tools for describing ***data, data relationships, data semantics and data constraints.***
- ✓ The model should enable the designer to incorporate a major portion of semantics of the database in the schema.

The role of data models

- ✓ The main aim of data models is to support the development of information systems by providing the definition and format of data. According to West and Fowler (1999) "if this is done consistently across systems then compatibility of data can be achieved. If the same data structures are used to store and access data then different applications can share data.
- ✓ "Business rules, specific to how things are done in a particular place, are often fixed in the structure of a data model. This means that small changes in the way business is conducted lead to large changes in computer systems and interfaces".
- ✓ "Entity types are often not identified, or incorrectly identified. This can lead to replication of data, data structure, and functionality, together with the attendant costs of that duplication in development and maintenance".
- ✓ "Data models for different systems are arbitrarily different. The result of this is that complex interfaces are required between systems that share data.
- ✓ The reason for these problems is a lack of standards that will ensure that data models will both meet business needs and be consistent.

How data models deliver benefit.



Data Models

- ✓ Numerous data models have been proposed which can be classified in the following categories
  1. Object based data models
  2. Record based data models
  3. Physical Data models

Object based data models

- ✓ Object based data models are used in describing data and data relationships in accordance with concept.
- ✓ In general, the object-based models are gaining wide acceptance for their flexible structuring capabilities.
- ✓ Various data integrity constraints can be specified explicitly by using the object-based data models.
- ✓ A number of object-based models have been proposed, Some of the important ones are the following:

1. **Entity-Relationship model**
2. object-oriented model
3. Binary Model
4. Semantic Data model
5. Info-Logical model
6. Functional Data model

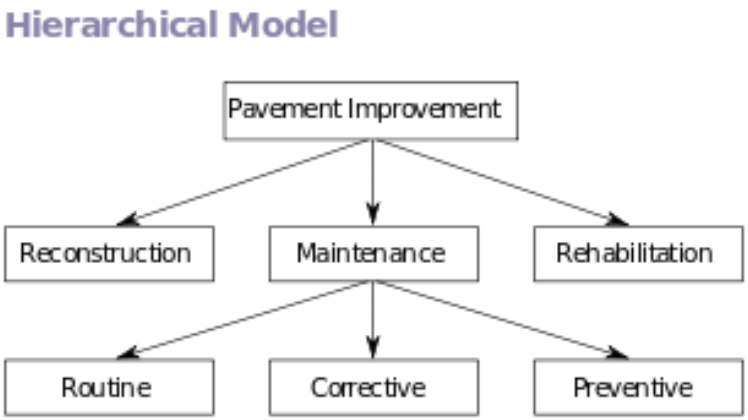
Record based Model

- ✓ Record based models describes data at the conceptual and view levels. Unlike object-oriented data models, these models specify overall logical structure of the database, and provide a higher-level description of the implementation.
- ✓ These models are called record-based because the database is structured in fixed-format of several types. Each record type defines a fixed number of fields, or attributes. Each field is usually of a fixed length.
- ✓ The three most widely-accepted record based models are:

1. **Network model**
2. **Hierarchical model**
3. **Relational model**

hierarchical database model

- A **hierarchical database model** is a data model in which the data is organized into a tree-like structure. The data is stored as **records** which are connected to one another through **links**. A record is a collection of fields, with each field containing only one value.
- The hierarchical database model mandates that each child record has only one parent, whereas each parent record can have one or more child records. In order to retrieve data from a hierarchical database the whole tree needs to be traversed starting from the root node. This model is recognized as the first database model created by IBM in the 1960s



Network Model

- The **network model** is a database model conceived as a flexible way of representing objects and their relationships. Its distinguishing feature is that the schema, viewed as a graph in which object types are nodes and relationship types are arcs, is not restricted to being a hierarchy or lattice.
- While the hierarchical database model structures data as a tree of records, with each record having one parent record and many children, the network model allows each record to have multiple parent and child records, forming a generalized graph structure.

Physical Data Model

- ✓ These models are used to have lower level description of the storage structure of the database and their access mechanism. With the physical models it is possible to implement the database at the system level.
- ✓ Physical data model includes all required tables, columns, relationships, database properties for the physical implementation of databases.
- ✓ Database performance, indexing strategy, physical storage and denormalization are important parameters of a physical model
- ✓ A very few physical data models have been proposed so far. Two of these well known models are:
  1. The unifying model
  2. The frame memory model

Entity-Relationship (E-R) Model

- ✓ In software engineering, an **entity-relationship model** (**ERM**) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system.
- ✓ In 1976, Chen developed the **Entity-Relationship (ER) model.**
- ✓ Creation of an ER diagram, which is one of the first steps in designing a database, helps the designer(s) to understand and to specify the desired components of the database and the relationships among those components.
- ✓ An ER model is a diagram containing entities or "items", relationships among them, and attributes of the entities and the relationships.
- ✓ Diagrams created by this process are called **entity-relationship diagrams**, **ER diagrams**, or **ERDs**.
- ✓ Overall logical structure of database can expressed graphically by an ER diagram which consists of the following components.

Entity-Relationship (E-R) Model

- ✓ Rectangles ---------- Represent entity set
- ✓ Ellipse ---------- Represent attributes
- ✓ Diamonds ---------- Relationship among entity set
- ✓ Lines ---------- Links attributes to entity sets to relationship

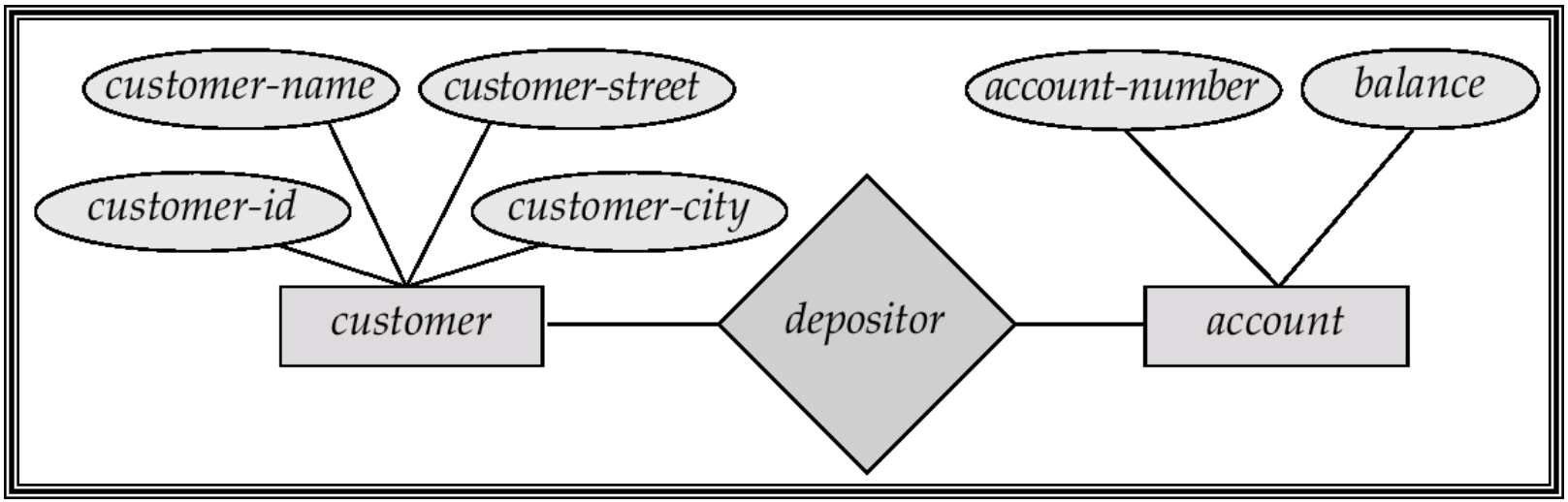
- ✓ Each entity has associated with it a set of attributes describing it.
- ✓ A relationship is an association among several entities.
- ✓ The set of all entities or relationships of the same type is called the entity set or relationship set.
- ✓ Widely used for database design.
- ✓ Database design in E-R model usually converted to design in the relational model which is used for storage and processing

Example:

Each customer deposit amount in own-account.

Entity-Relationship (E-R) Model (Cont.)

Example of schema in the entity-relationship model



Example:

1. Each student issue Books.
2. Each book is written by author.
3. Each book is published by some publisher.
4. Students write examinations.
5. Students attend classes.
6. Professors write books.
7. Driver drives a car.

Basic Concepts of E-R Model

The E-R data model employs three basic notions:

1. **Entities and Entity set**
2. **Relationships and Relationship sets**
3. **Attributes**

Entities and 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 i.e. an entity may be either concrete, such as customer or book, or it may be abstract, such as project or a concept.
  - ➢ Example: specific person, company, event, plant
- ✓ An *entity set* is a set of entities of the same type that share the same properties.
  - ➢ Example: The set of all employees of an organization can be defined as the entity set employees. Similarly the entity set projects represents the set of all projects undertaken by the organization.

Entities and Entity Sets (cont….)

- ✓ An entity set is represented by a set of **attributes**. Possible attributes of the employees entity set are emp_id, emp_name, address, sex, date_of_birth. Possible attributes of the project entity set are project_id, Project_name.
- ✓ For each attribute there is a set of permitted values, called the domain of that attribute, which can be assigned to the attribute.
- ✓ E.g. one instance of the entity set employee and one instance of the entity set project are shown as

Emp-id	Name	sex	address
E1	Ram Kumar	M	Newroad
E2	Shayam Shrestha	M	Kalanki
E3	Rohan Sherpa	M	Biratnagar
E4	Rojina Sah	F	Pokhara

Project-id	Project_Name
P1	RDBMS
P2	UNIX
P3	OS
P4	MIS

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

Relationships and Relationship Set

- ✓ A relationship is an association among several entities.
  - For example, a relationship which associates the employees Ram Kumar with a project with Project_id P2. This specifies that Ram Kumar is an employee who is working in the project P2.
- ✓ A relationship set is a set of relationship of the same type.
- ✓ A relationship may also have descriptive attributes for example, date_of_issue could be an attribute of the relationship set **Issue**.
- ✓ Consider the two entity set ***Employee*** and ***project***. The relationship set ***work*** to denote the association between ***employee and project*** that the employees have

Emp-id	Name	sex	address
E1	Ram Kumar	M	Newroad
E2	Shayam Shrestha	M	Kalanki
E3	Rohan Sherpa	M	Biratnagar
E4	Rojina Sah	F	Pokhara

Employee

Project-id	Project_Name
P1	RDBMS
P2	UNIX
P3	OS
P4	MIS

Project

Relationships and Relationship Set (cont….)

- ✓ The association between entity sets is referred to as participation; that is, the entity sets E_1, E_2, …E_n **participate** in relationship R.
- ✓ A relationship instance in an E-R schema represents an association between the named entities in the real world enterprise that is being modeled
  - ✓ E.g., in above, the individual employee entity **Ram Kumar**, who has the **emp_id E_1**, and the project entity P_1 participate in a relationship instance of works. This relationship instance represents that, in the real-world enterprise, the person called **Ram Kumar** who holds emp_id E_1 work in project P_1
- ✓ The number of entity sets that participate in a relationship set is also the degree of the relationship set.
- ✓ A binary relationship set is one that involves two entity sets. Most of the relationship sets in a database system are binary.
- ✓ However, there are relationship sets, which might involve more than two entity sets.

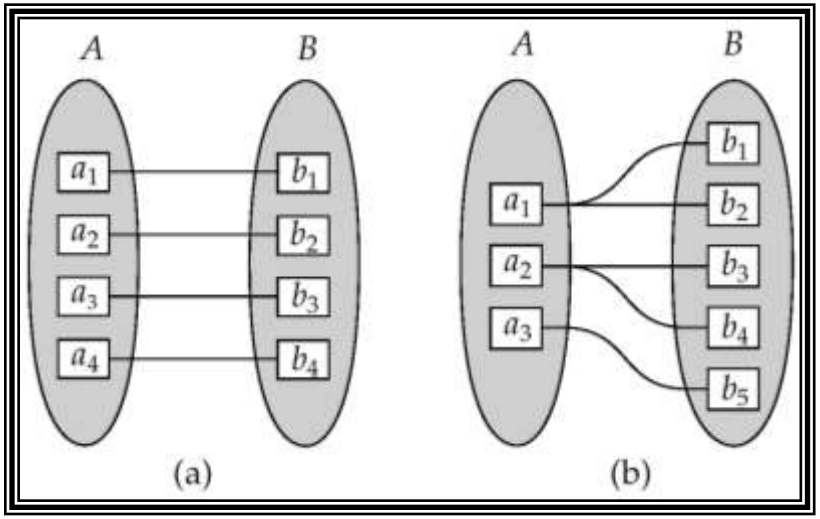
Constraints

- ✓ An E-R enterprise schema may define certain constraints to which the contents of a database most conform.
- ✓ **Mapping cardinalities** and **Participation constraints** are two most important types of constraints.

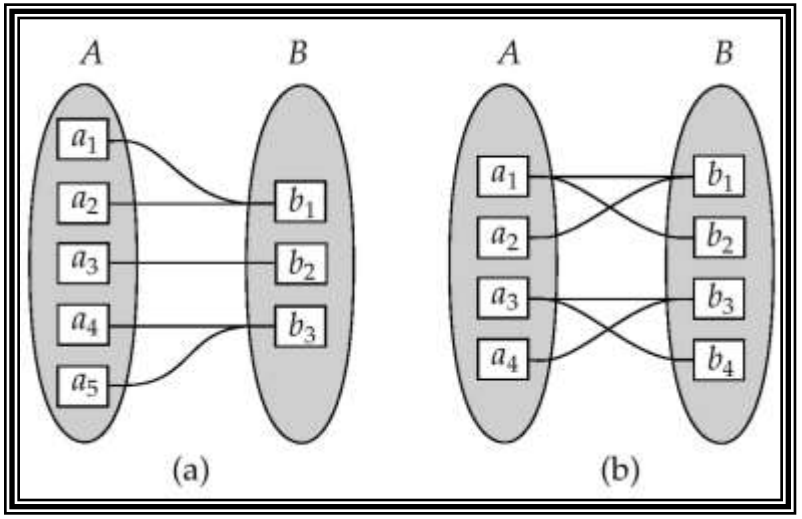
Mapping cardinalities

- ✓ **Mapping cardinalities,** or **cardinality ratios,** express the number of entities to which another entity can be associated via a relationship set.
- ✓ **Mapping cardinalities** are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets.
- ✓ For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:
  - **One to One** : An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
  - **One to many** : An entity in A is associated with any number (zero or more) of entities in B, and an entity in B, however, can be associated with at most one entity in A.
  - **Many to One** : An entity in A is associated with at most one entity in B, and an entity in B, however, can be associated with any number (zero or more) of entities in A.
  - **Many to Many** : An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.

Mapping cardinalities



One to one One to many



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 Constraints

- ✓ The participation of an entity set E in a relationship set R is said to be **total** if every entity in E participates in at least one relationship in R.

- ✓ If only some entities in E participate in relationship in R, the participation of entity set E in relationship R is said to be **partial**.

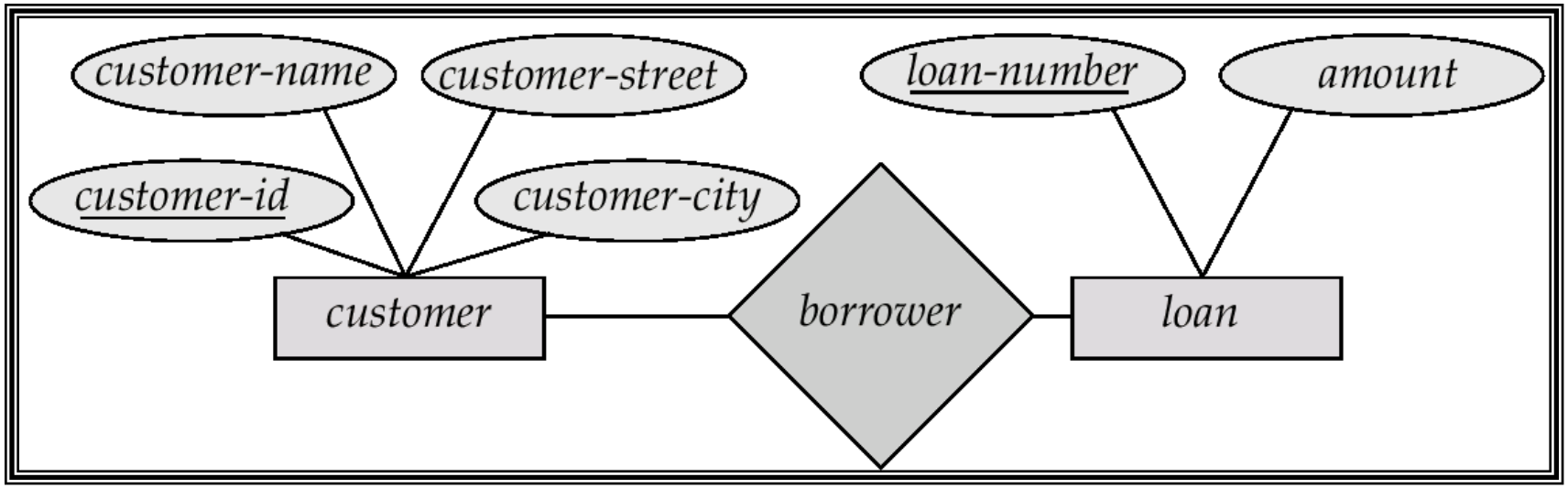
Keys

- ✓ A way to specify how entities within a given entity set are distinguished.
- ✓ Conceptually, individual entities are distinct; from a database perspective, however, the difference among them must be expressed in terms of their attributes.
- ✓ Therefore, the values of the attribute values of an entity must be such that they can uniquely identify the entity. In other words, *no two entities in an entity set are allowed to have exactly the same value for all attributes*.
- ✓ A key allows us to identify a set of attributes that suffice to distinguish entities from each other.
- ✓ Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

Keys (cont….)

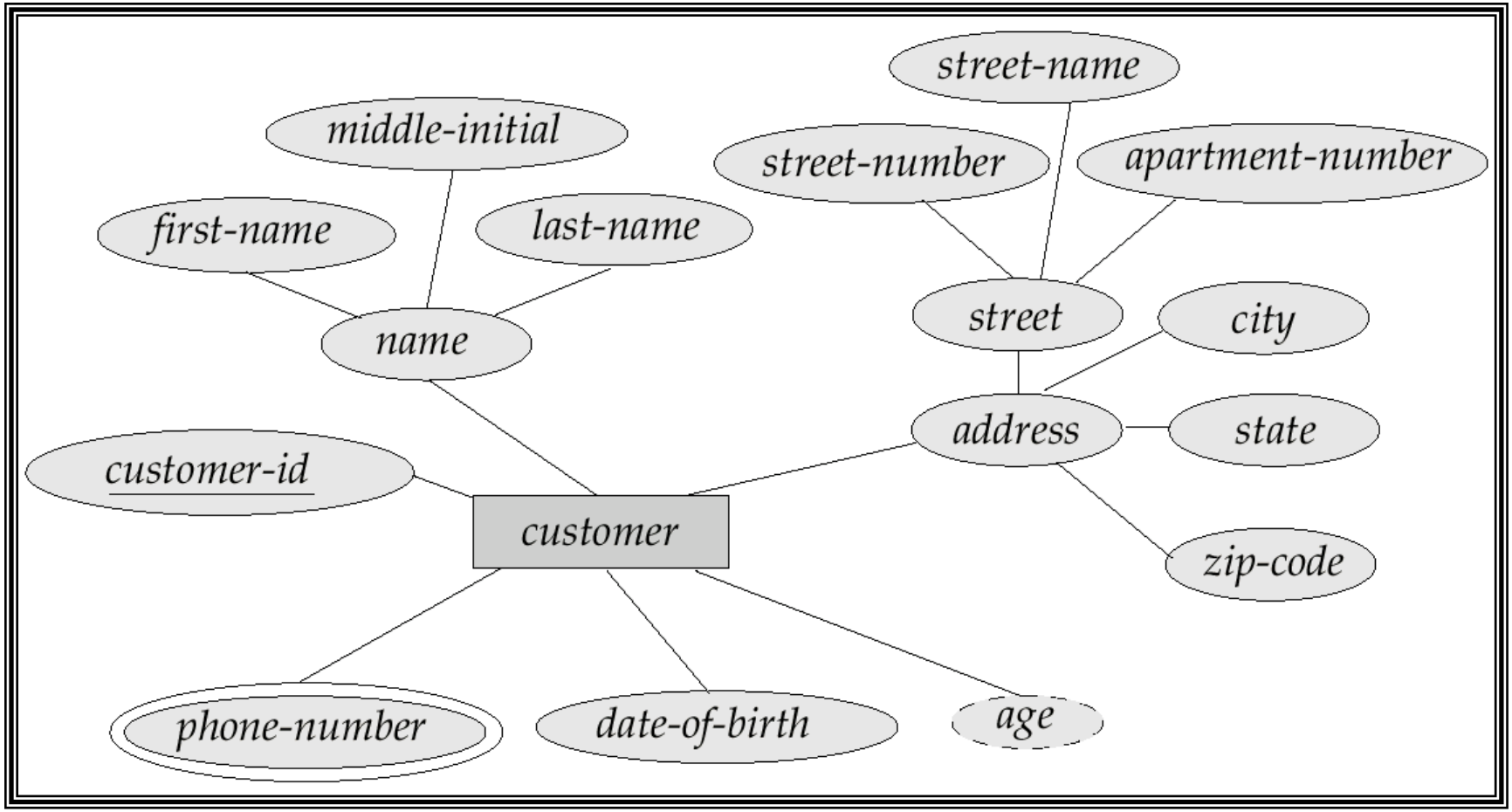
- ✓ A ***superkey*** of an entity set is a set of one or more attributes whose combined values uniquely determine each entity in the entity set.
  - For example, for an entity set employees the set of attributes **{emp_id, emp_name, address}** can be considered to be a super key. If we assume that there are no two employees with same name and same address, then the set **{ emp_name, address}** is another super key.
- ✓ A ***candidate key*** of an entity set is a minimal super key, i.e. a superkey which does not have any proper subset which is also a superkey.
  - For example, {emp_id} and {emp_name, address} are two candidate keys for the entity set **Employee**.
- ✓ A **primary key** is a candidate key that is chosen by the database designer as the principal means of uniquely identifying entities within an entity set.
  - For example, the candidate key **{emp_id}** can be considered to be the primary key for the entity set employees.

E-R Diagrams

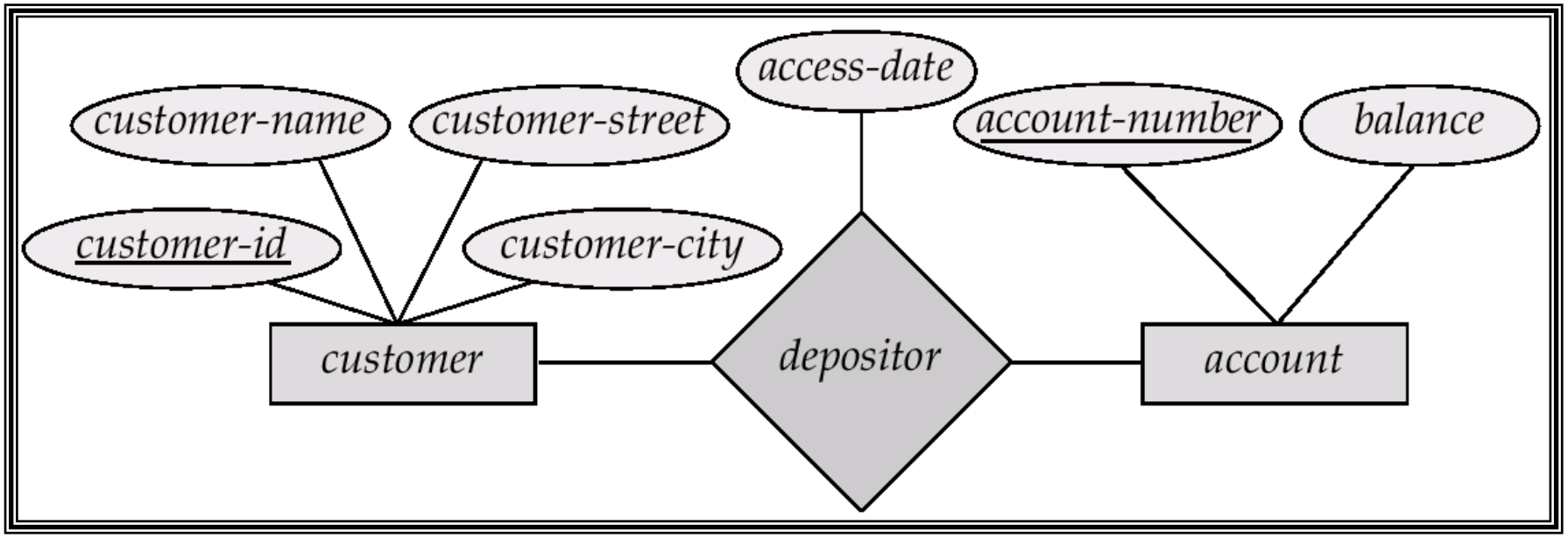


- **Rectangles** represent entity sets.
- **Diamonds** represent relationship sets.
- **Lines** link attributes to entity sets and entity sets to relationship sets.
- **Ellipses** represent attributes
  - **Double ellipses** represent multivalued attributes.
  - **Dashed ellipses** denote derived attributes.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

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

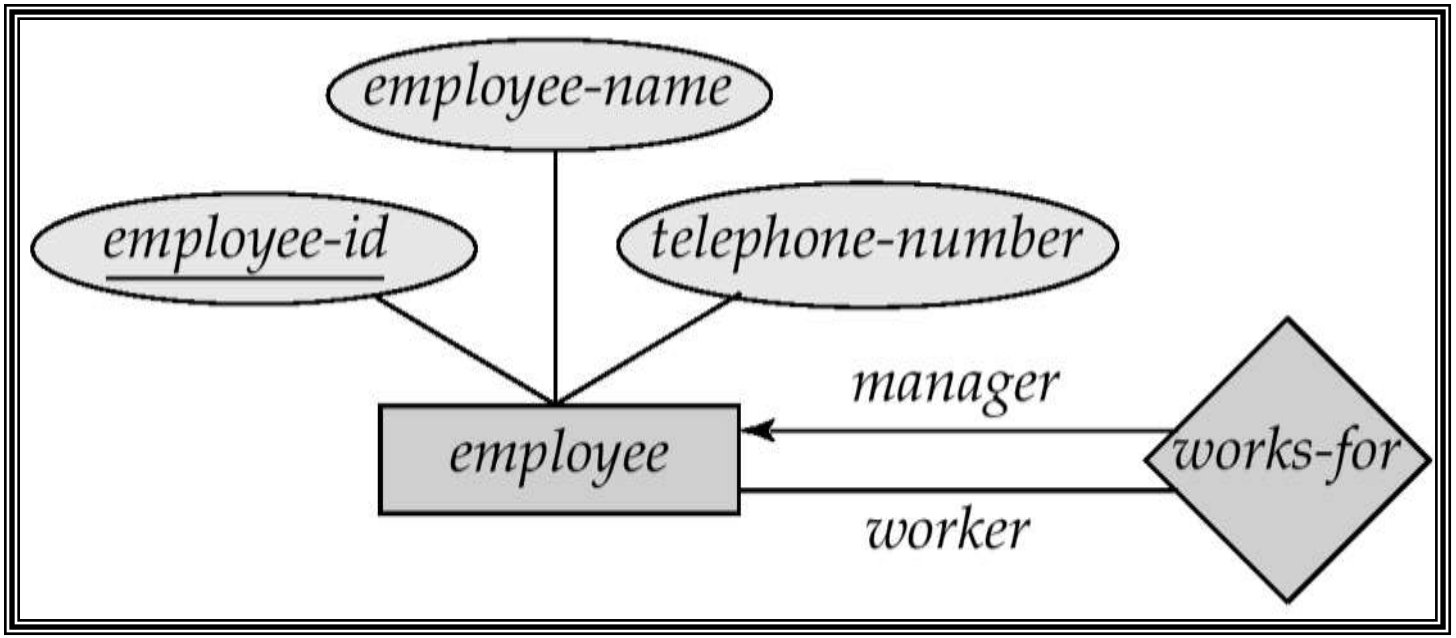


Relationship Sets with Attributes



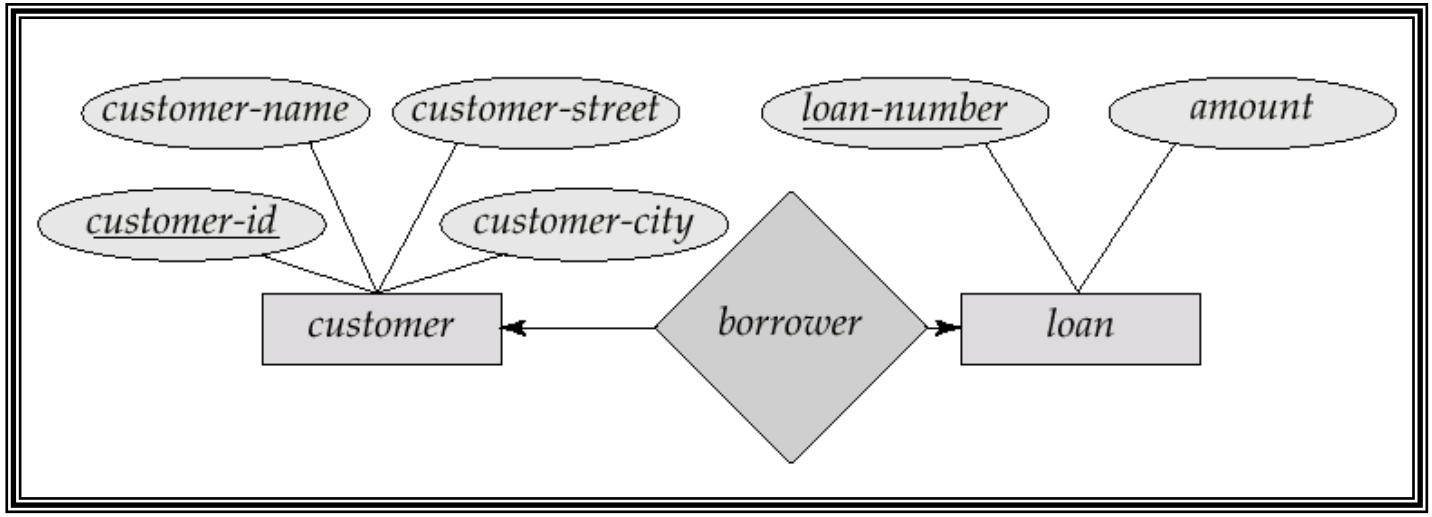
Roles

- ✓ 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



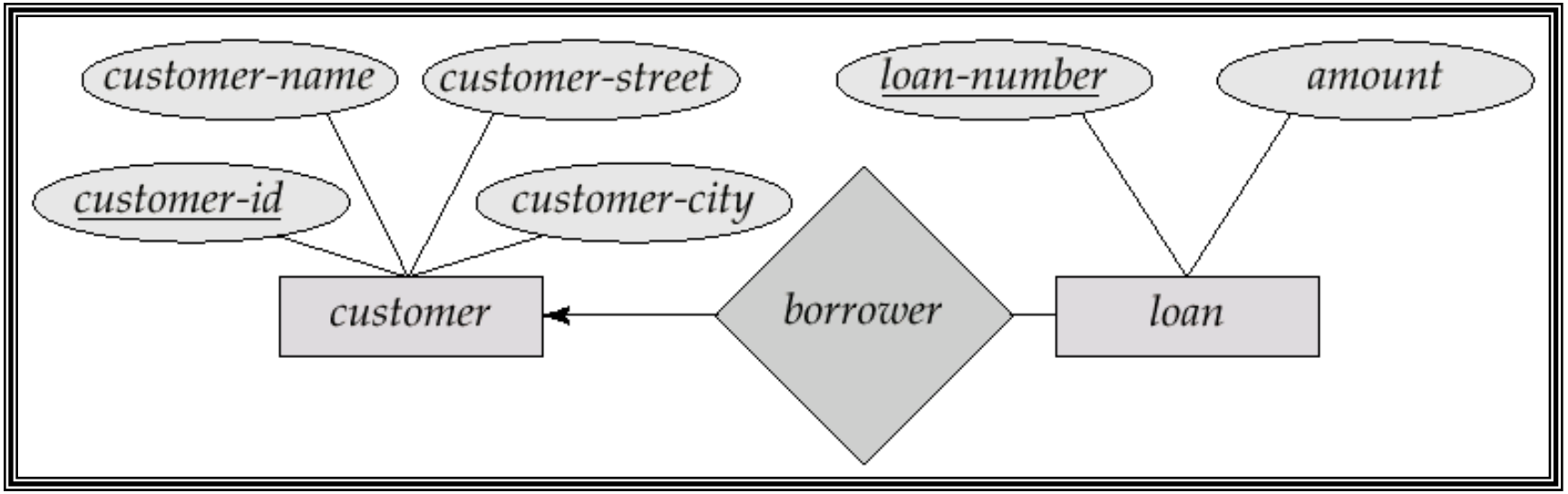
Cardinality Constraints

- ✓ 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*



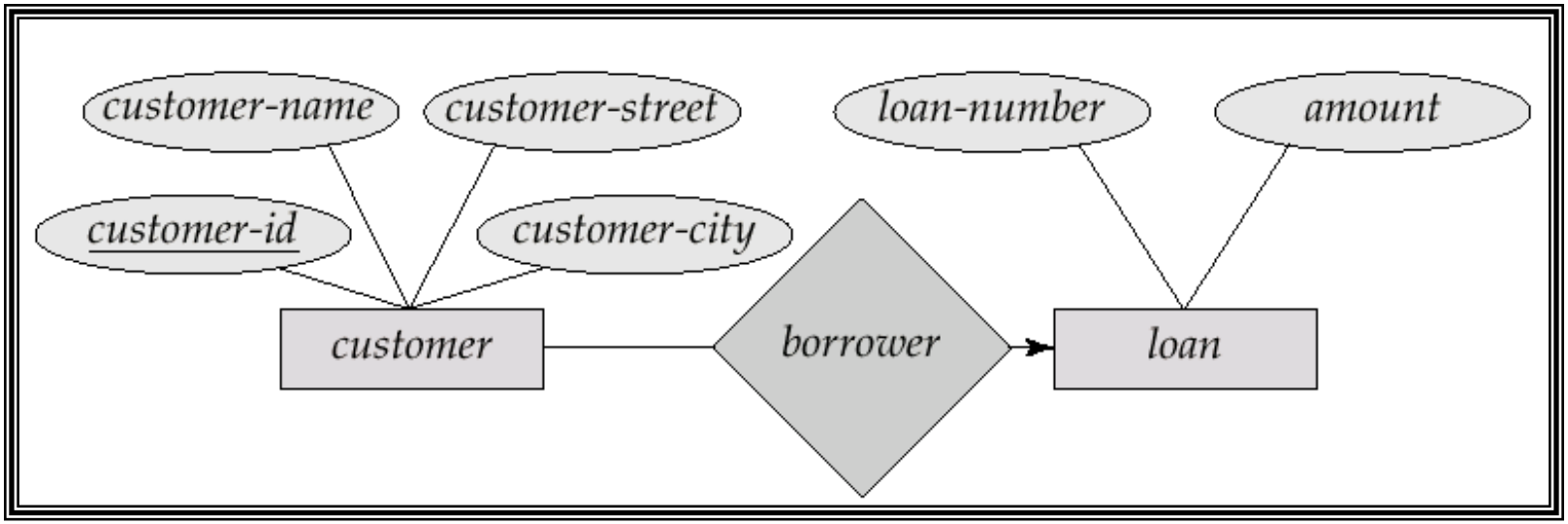
One-To-Many Relationship

- ✓ 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*

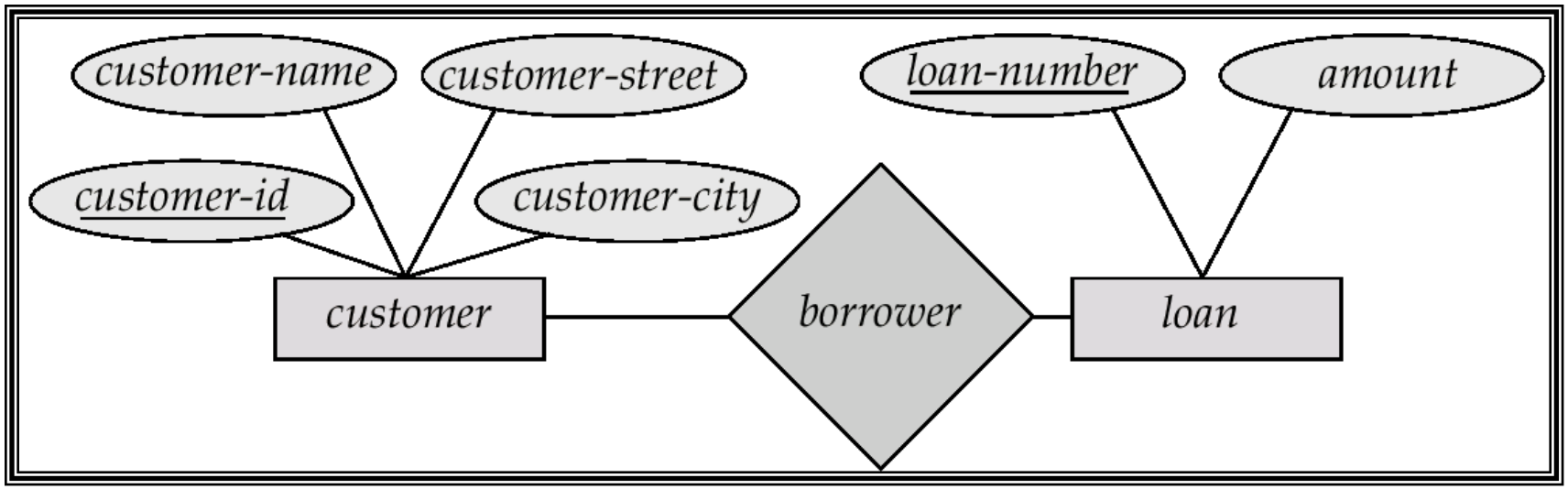


Many-To-One Relationships

- ✓ 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*



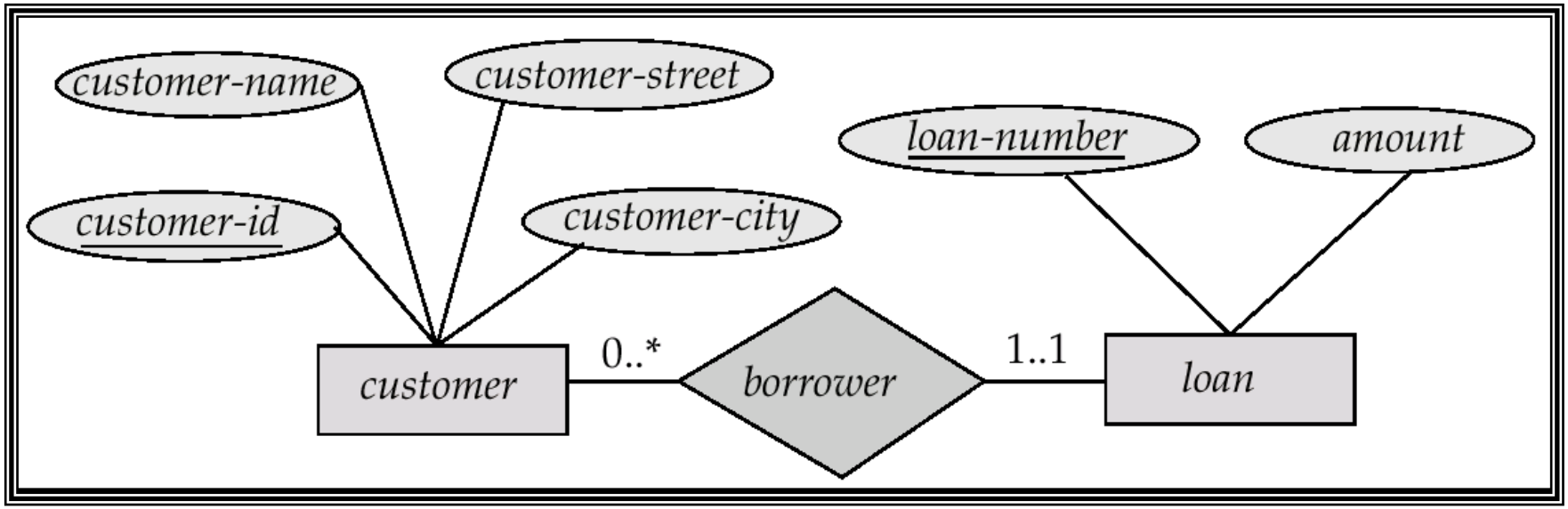
Many-To-Many Relationship



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

Alternative Notation for Cardinality Limits

- ✓ Cardinality limits can also express participation constraints

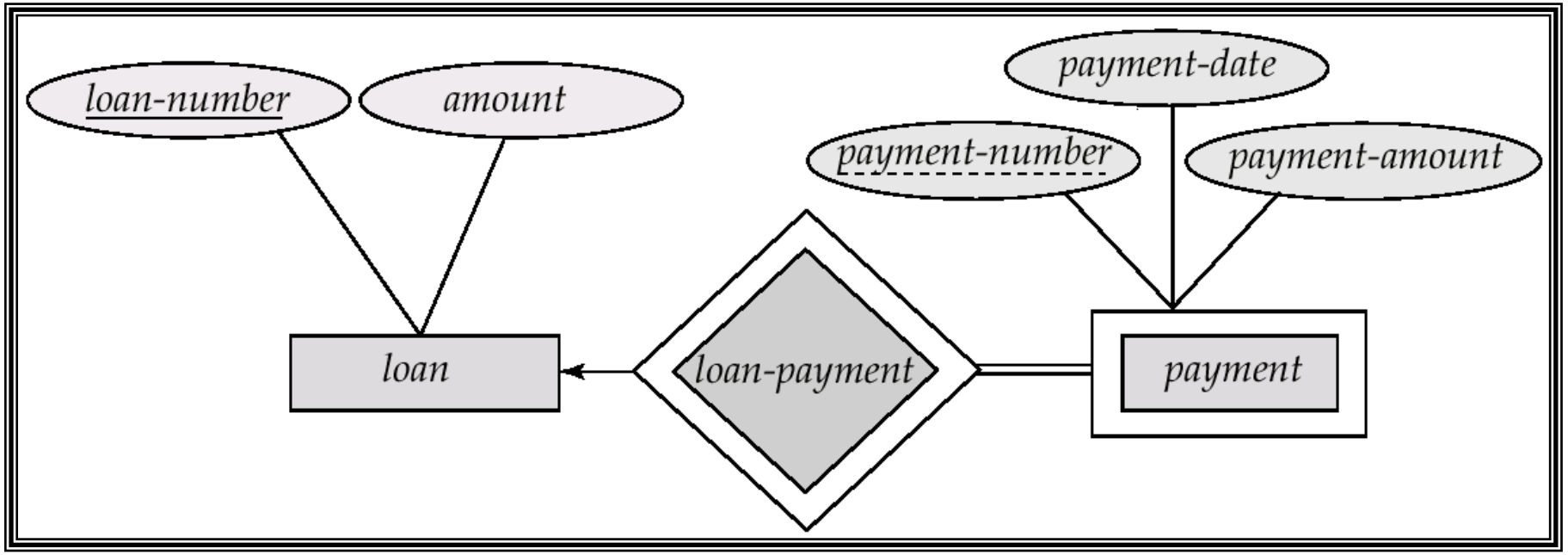


Weak Entity Sets

- ✓ 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*

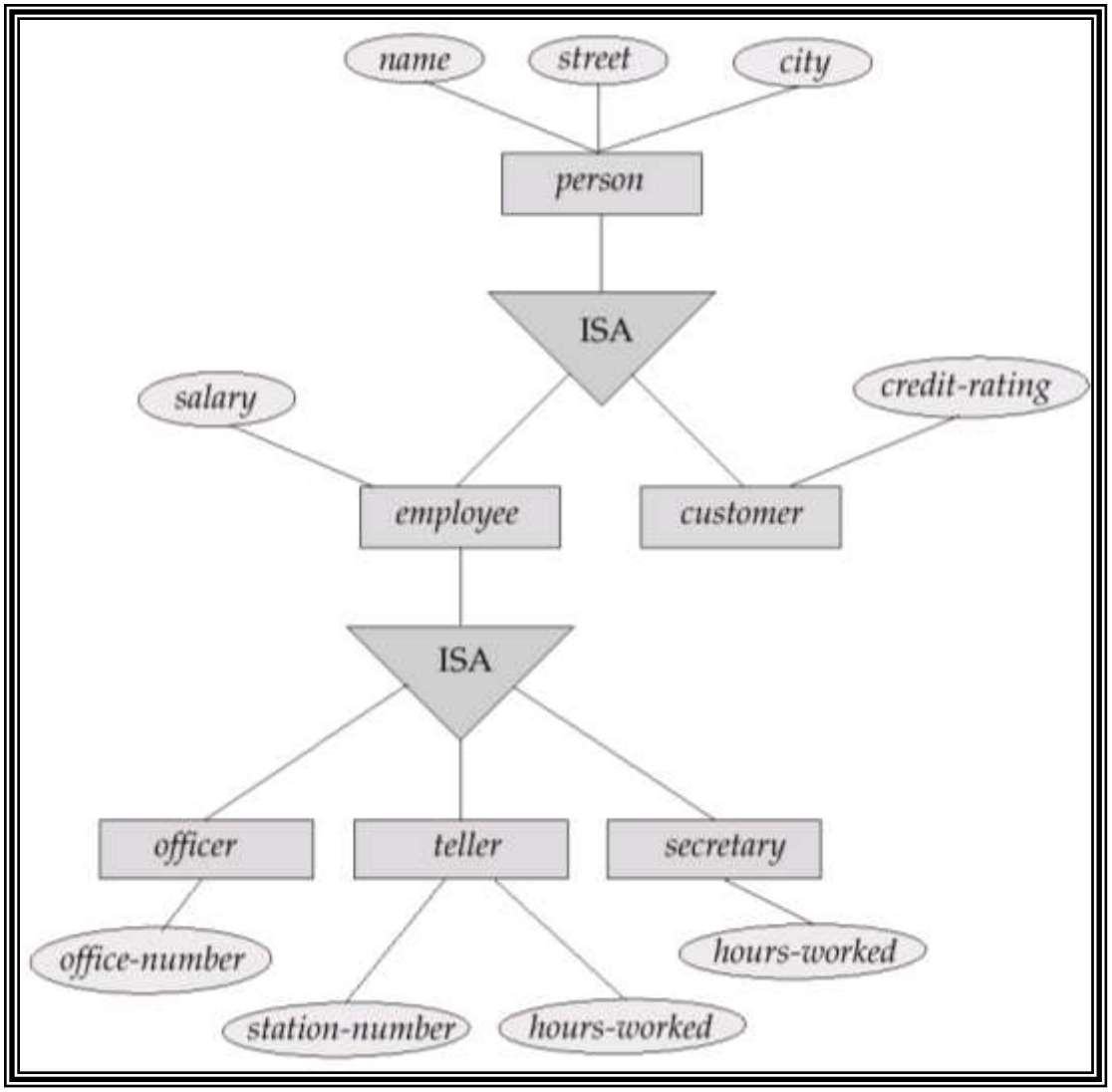
More Weak Entity Set Examples

- ✓ 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

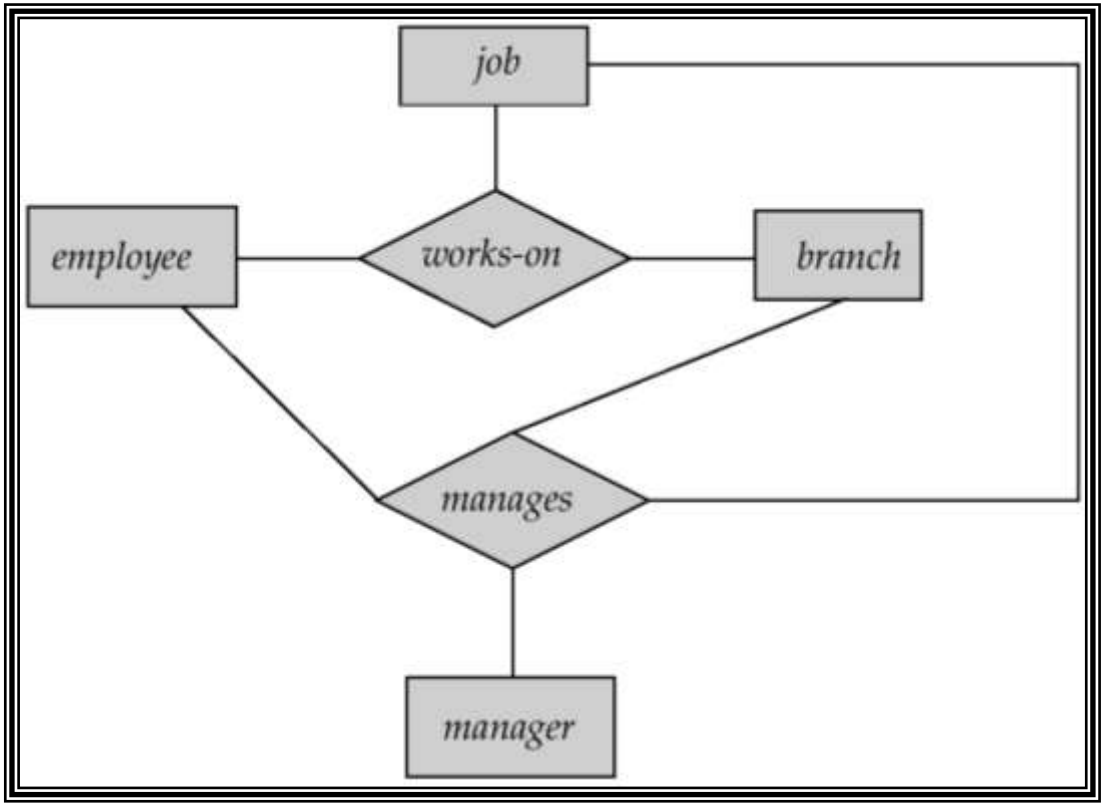
- ✓ Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - E.g. all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
  - user-defined
- ✓ Constraint on whether or not entities may belong to more than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

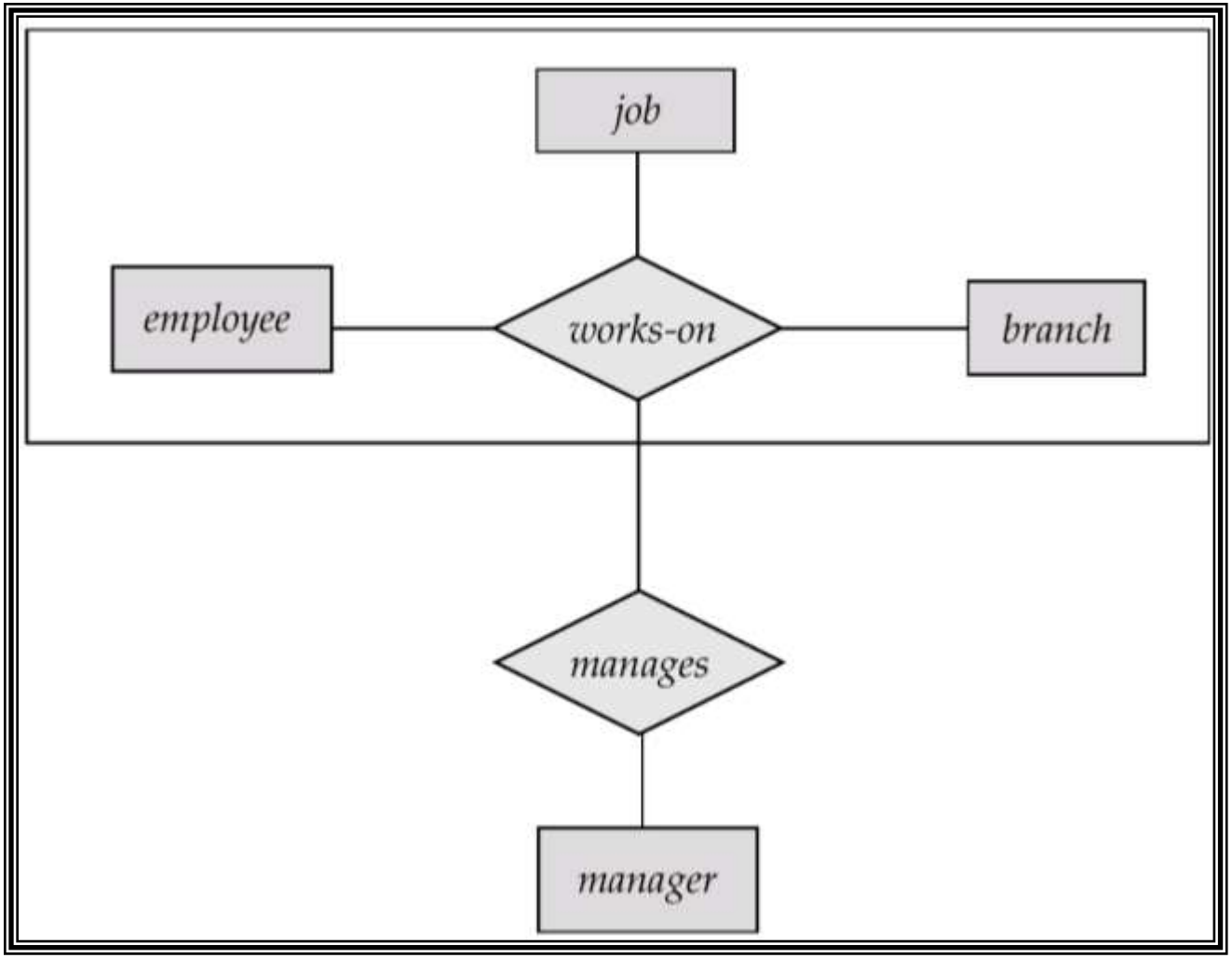
- ✓ Consider the ternary relationship *works-on*, which we saw earlier
- ✓ Suppose we want to record managers for tasks performed by an employee at a branch



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

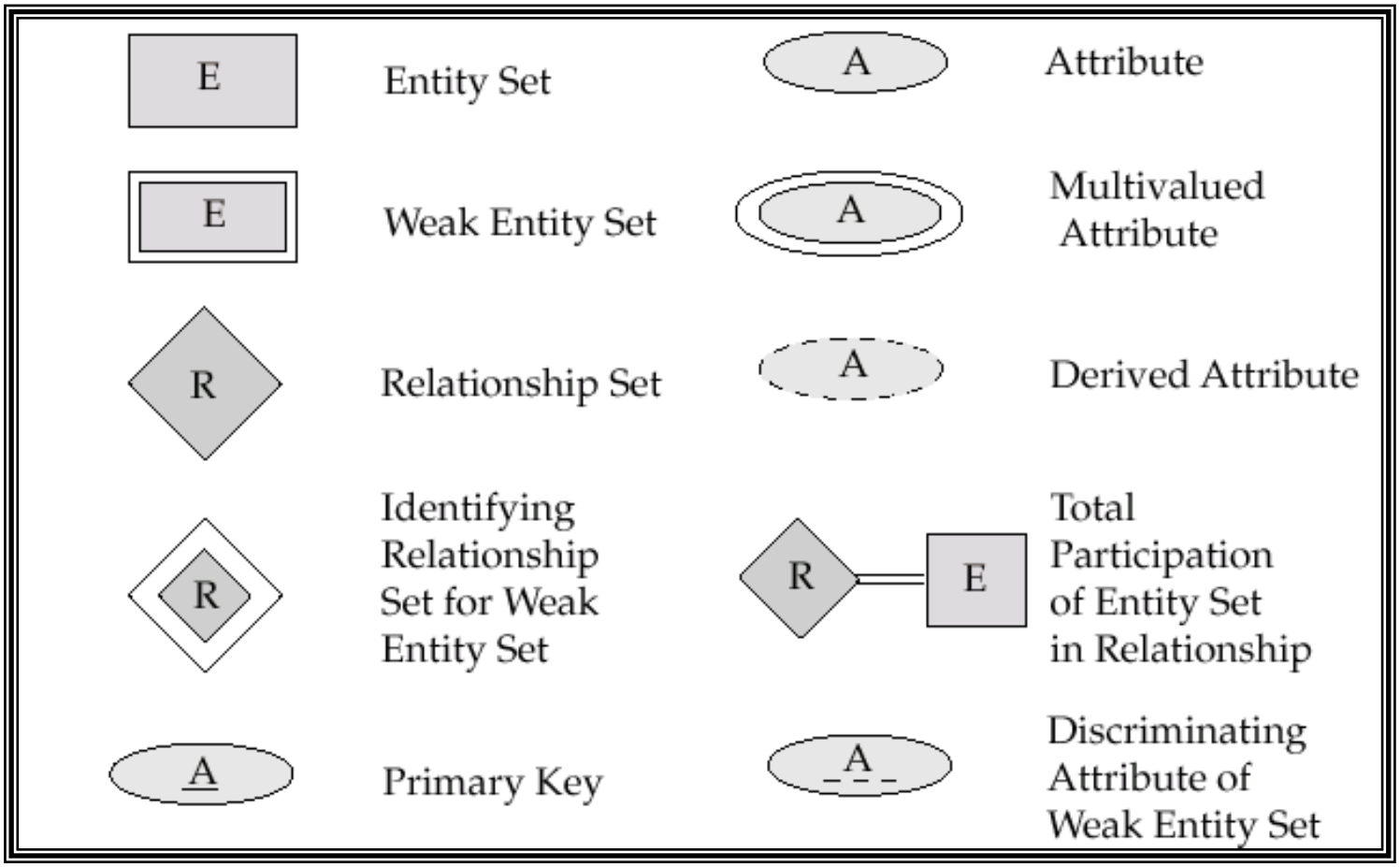
E-R Diagram With Aggregation



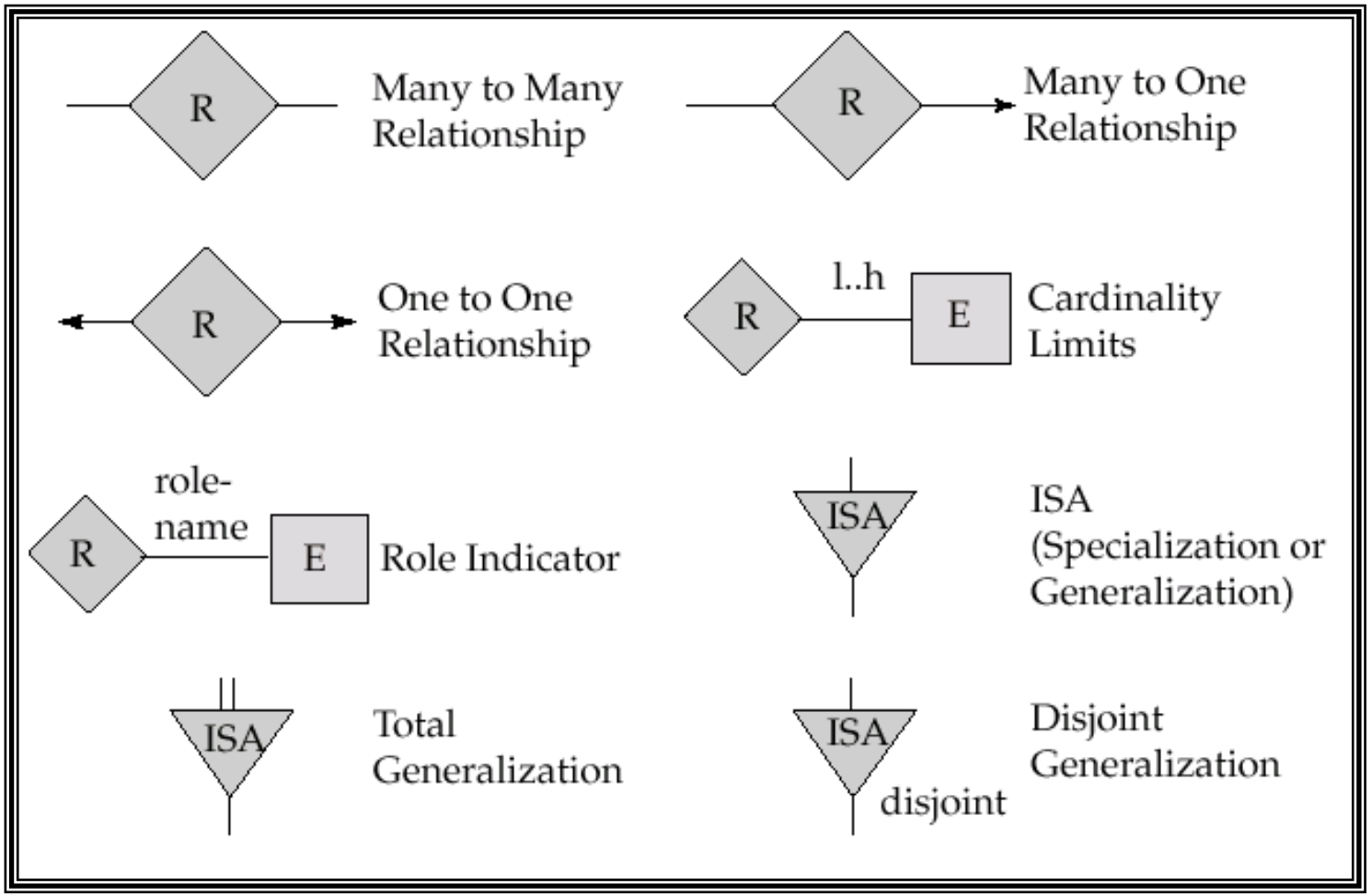
E-R Design Decisions

- ✓ 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.

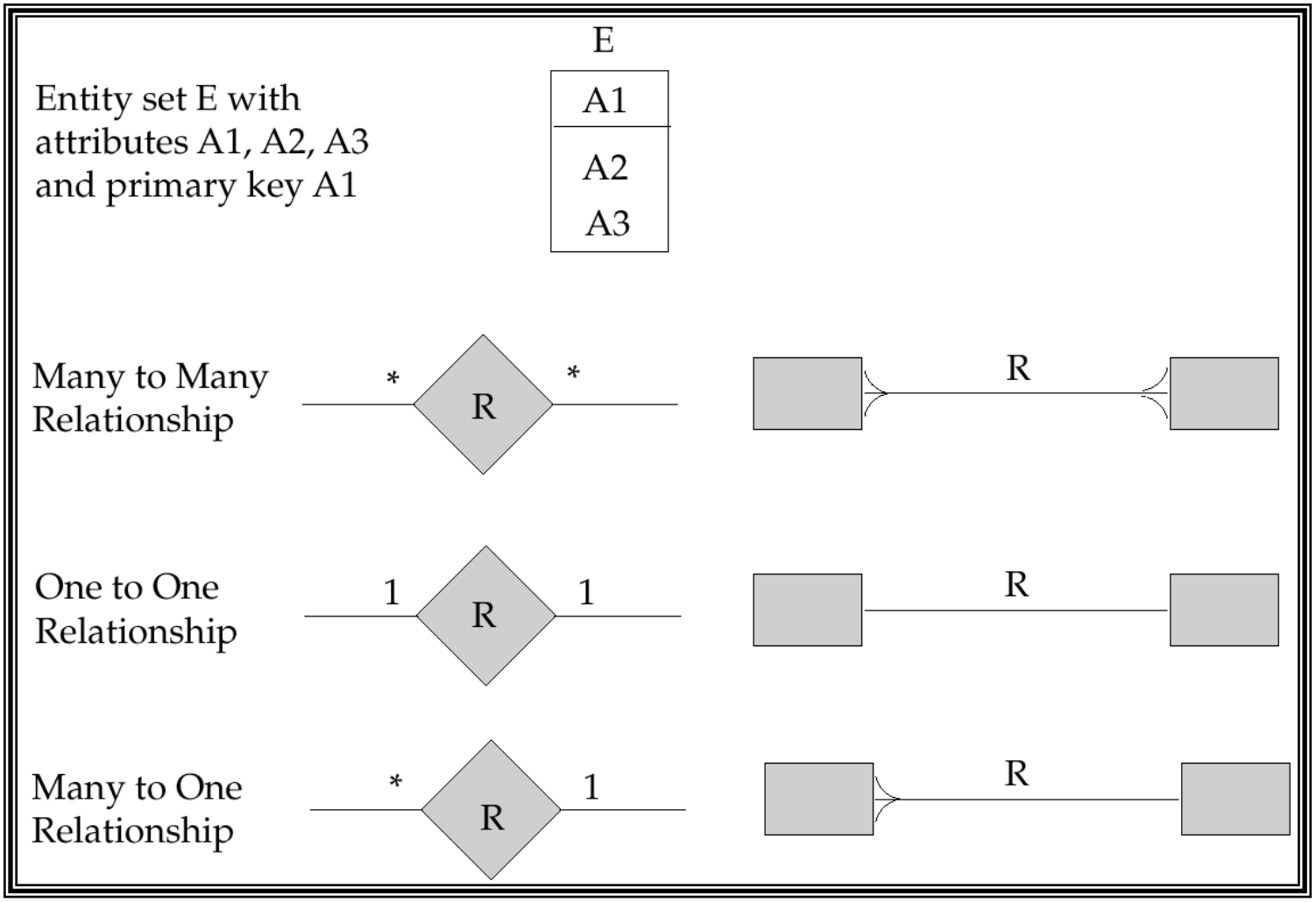
Summary of Symbols Used in E-R Notation



Summary of Symbols (Cont.)



Alternative E-R Notations



E-R Design: Exercise

For each of the projects, do the following:

a. Draw an ER diagram. Make sure each entity type has at least one key attribute. Document any assumptions you make. Avoid diagrams with a single entity.

b. **b.** For each attribute, determine its domain of values, whether null is an acceptable value, and, if acceptable, what null indicates.

c. **c.** If you think the description is incomplete, list other data the database should store.

d. Adjust your design to incorporate these additions.

1. A chemistry department wants to have a database of all chemicals in the stockroom. The information includes the name, molecular formula, amount on hand, date purchased, supplier, and supplier contact information.
2. A space agency wants to develop a database of all satellites that humans have launched into space. Data includes the satellite identification, date of launch, destruction date, purpose, maximum orbital altitude, launching location, launching agency, and contact information for agency.

E-R Design: Exercise

3. An environmental agency wants to catalogue all the plants in an area that is susceptible to acid rain. Data should include genus, species, quantity, date, quadrant identification number, quadrant location, average altitude of quadrant, and botanist.
4. A psychological study requires participants to answer a number of questions related to personality. The database should store the multiple choice answers (A, B, C, D) to the questions and information about each participant, such as participant id, age, and sex. The database should compute a score based on the individual's answers. The score indicates one of 8 personality categories. Each category has an identifying name and specifies that each of three qualities is either true or false.