CSE 132B
Database Systems Principles

Alin Deutsch

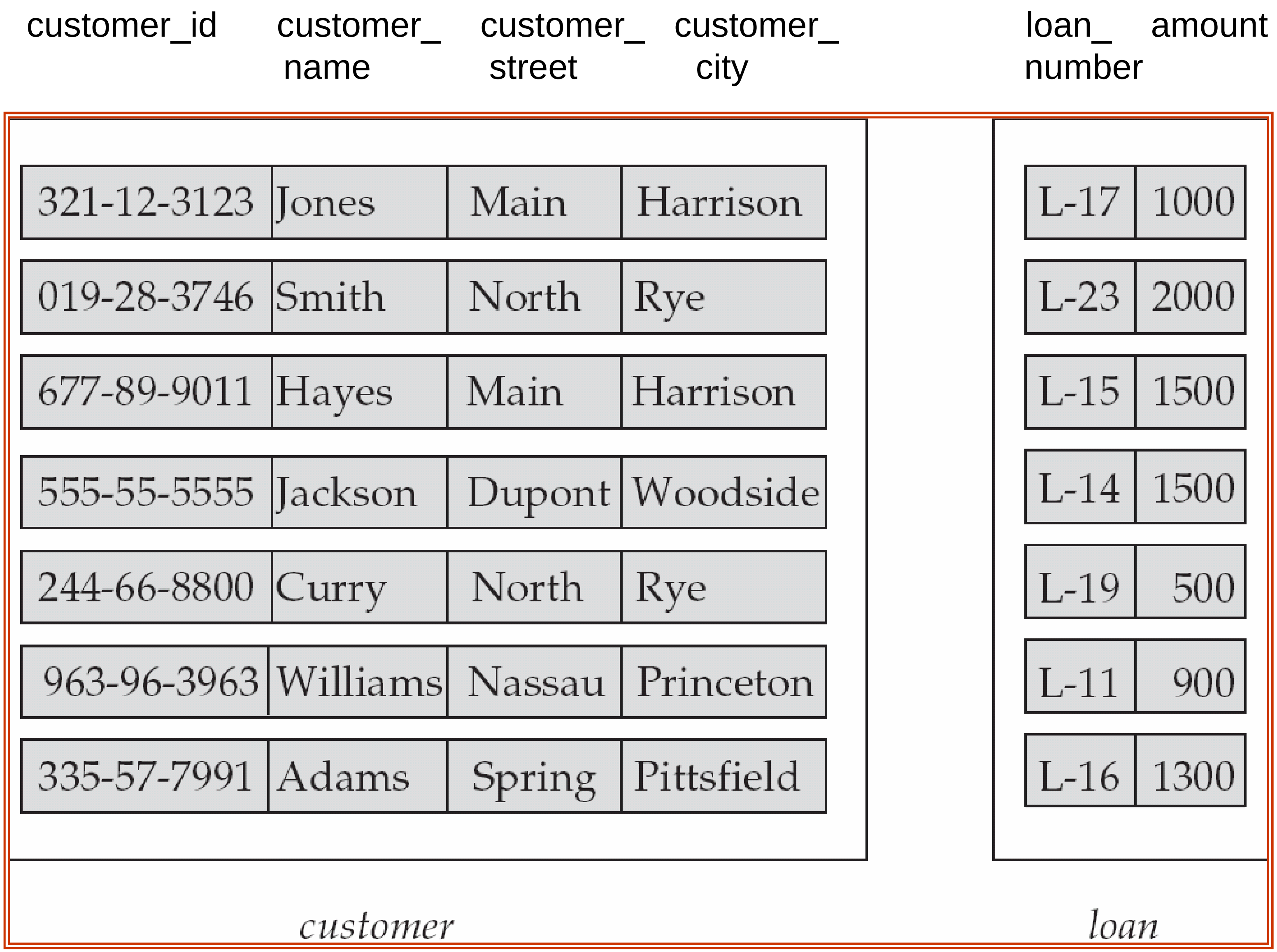
Database Design with the Entity-Relationship (ER) Model

Some slides are based or modified from originals by
Elmasri and Navathe,
Fundamentals of Database Systems, 4th Edition
© 2004 Pearson Education, Inc.
and
Database System Concepts, McGraw Hill 5th Edition
© 2005 Silberschatz, Korth and Sudarshan

Database 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

Attributes

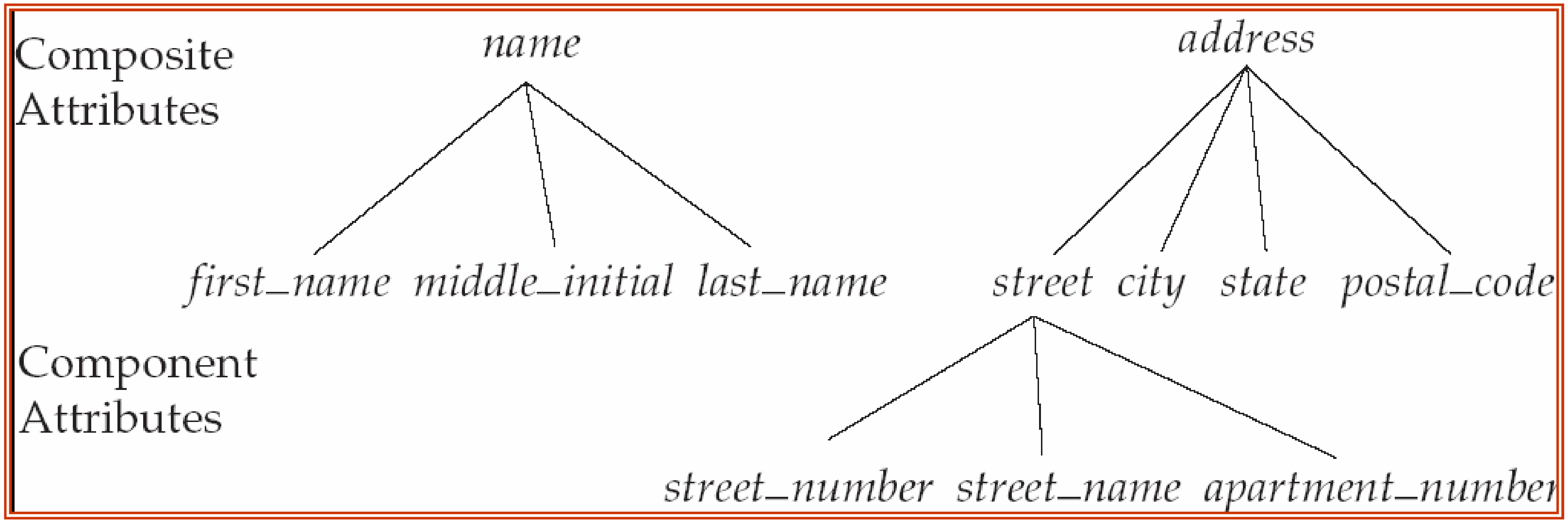
- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

  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
    - Example: multivalued attribute: *phone_numbers*
  - *Derived* attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth

Composite Attributes



Relationship Sets

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

  Example:

Hayes	*bottower*	L-15
customer entity	relationship set	*loan* entity

- A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets $E_1, E_2, \ldots, E_n$ (called the participating entity sets)

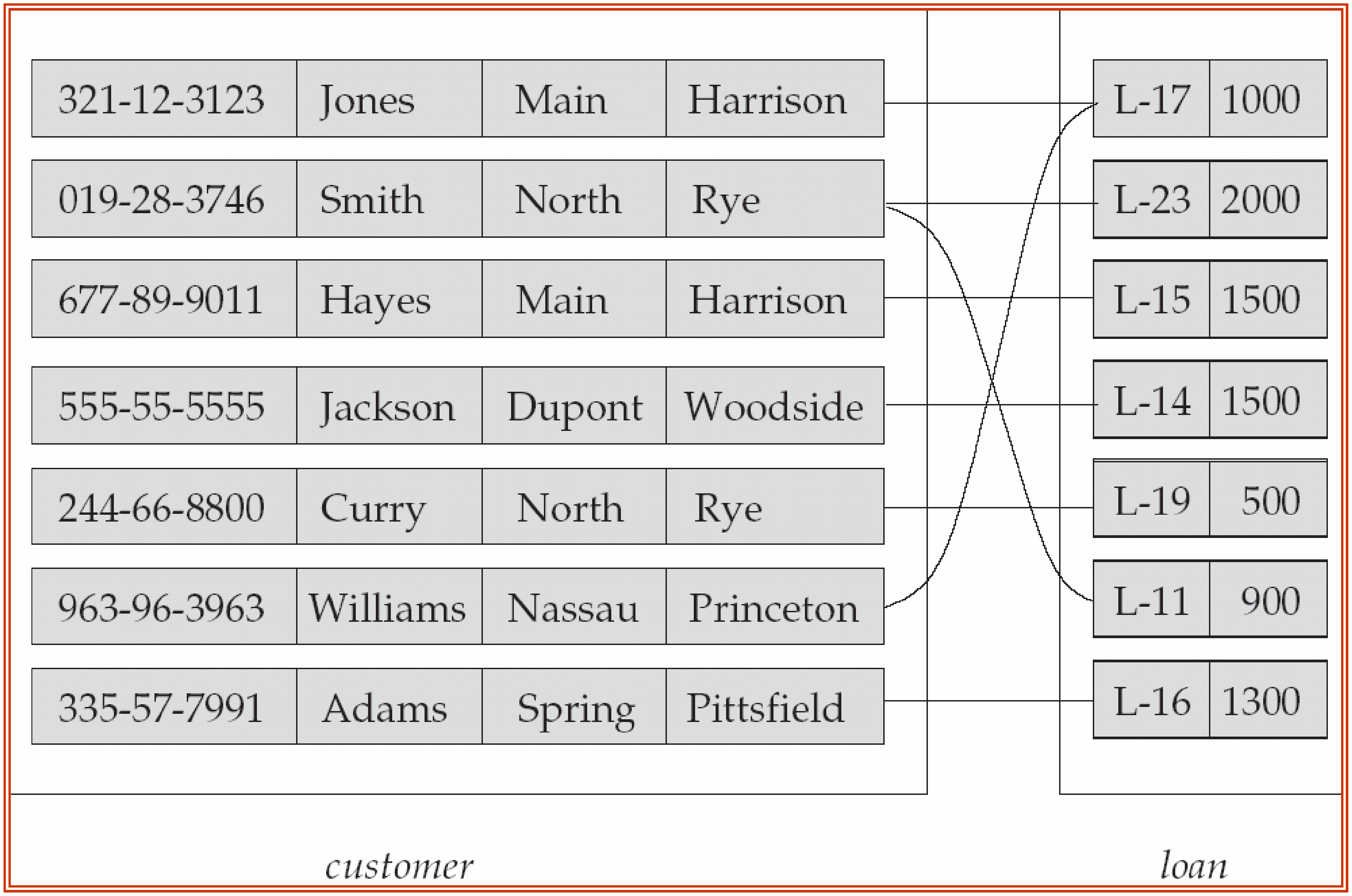
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

    (Hayes, L-15) $\in$ *borrower*

Relationship Set *borrower*

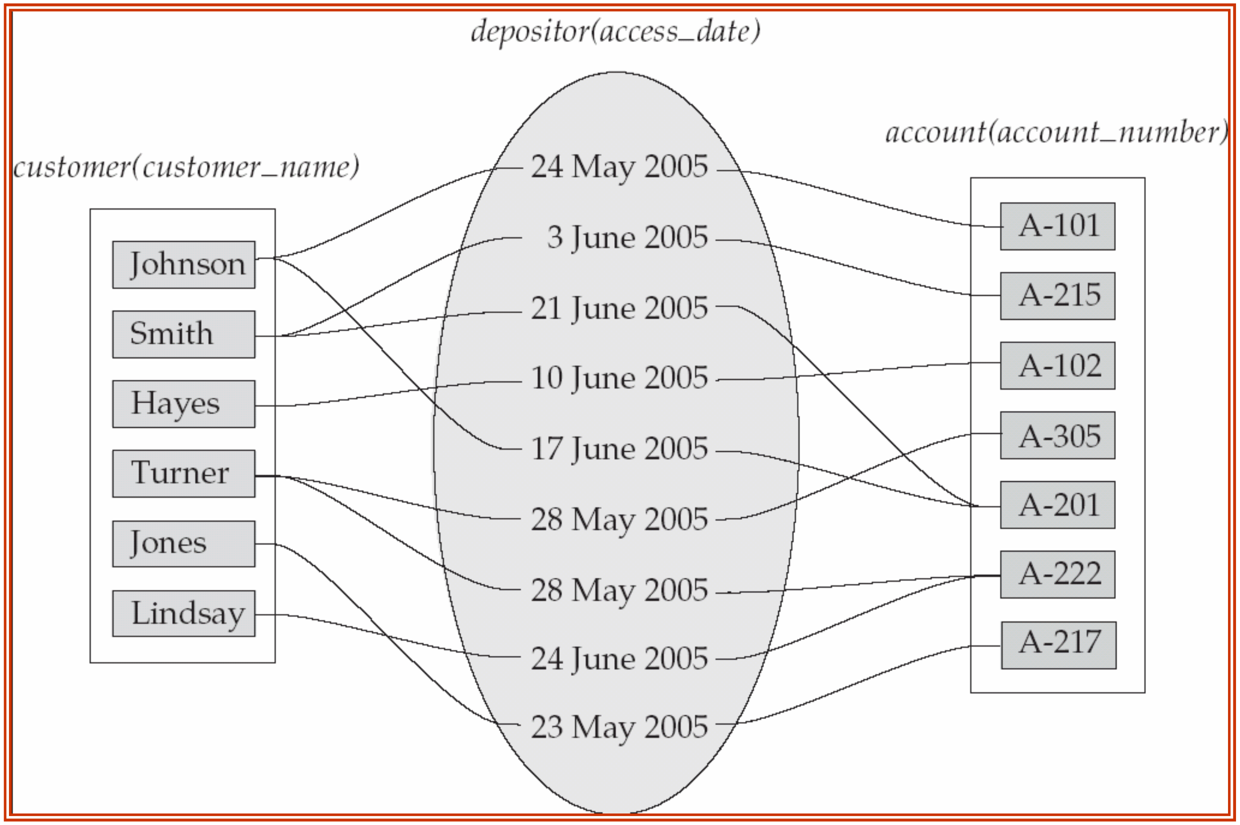


321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

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

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*



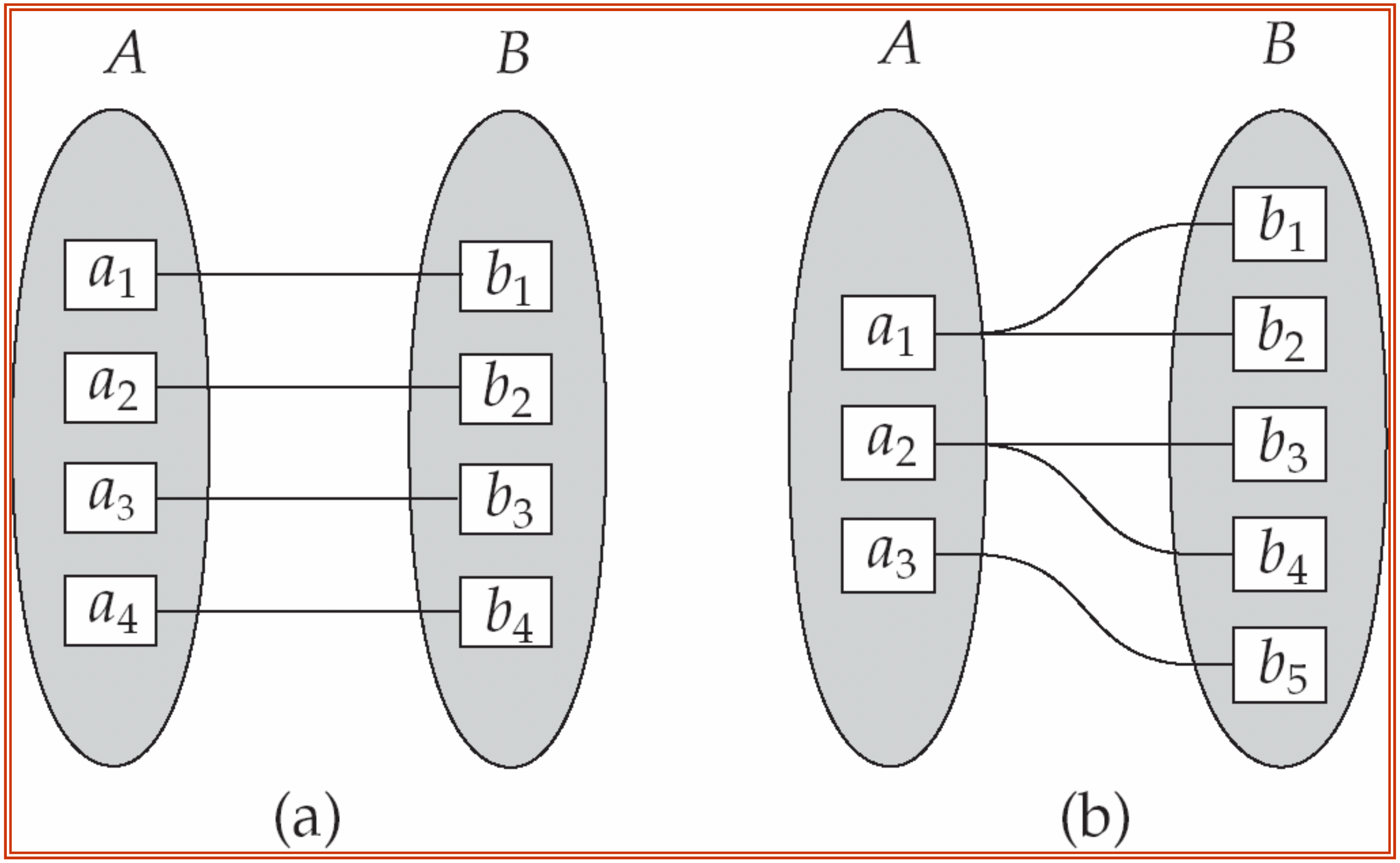
Degree of a Relationship Set

- 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.

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

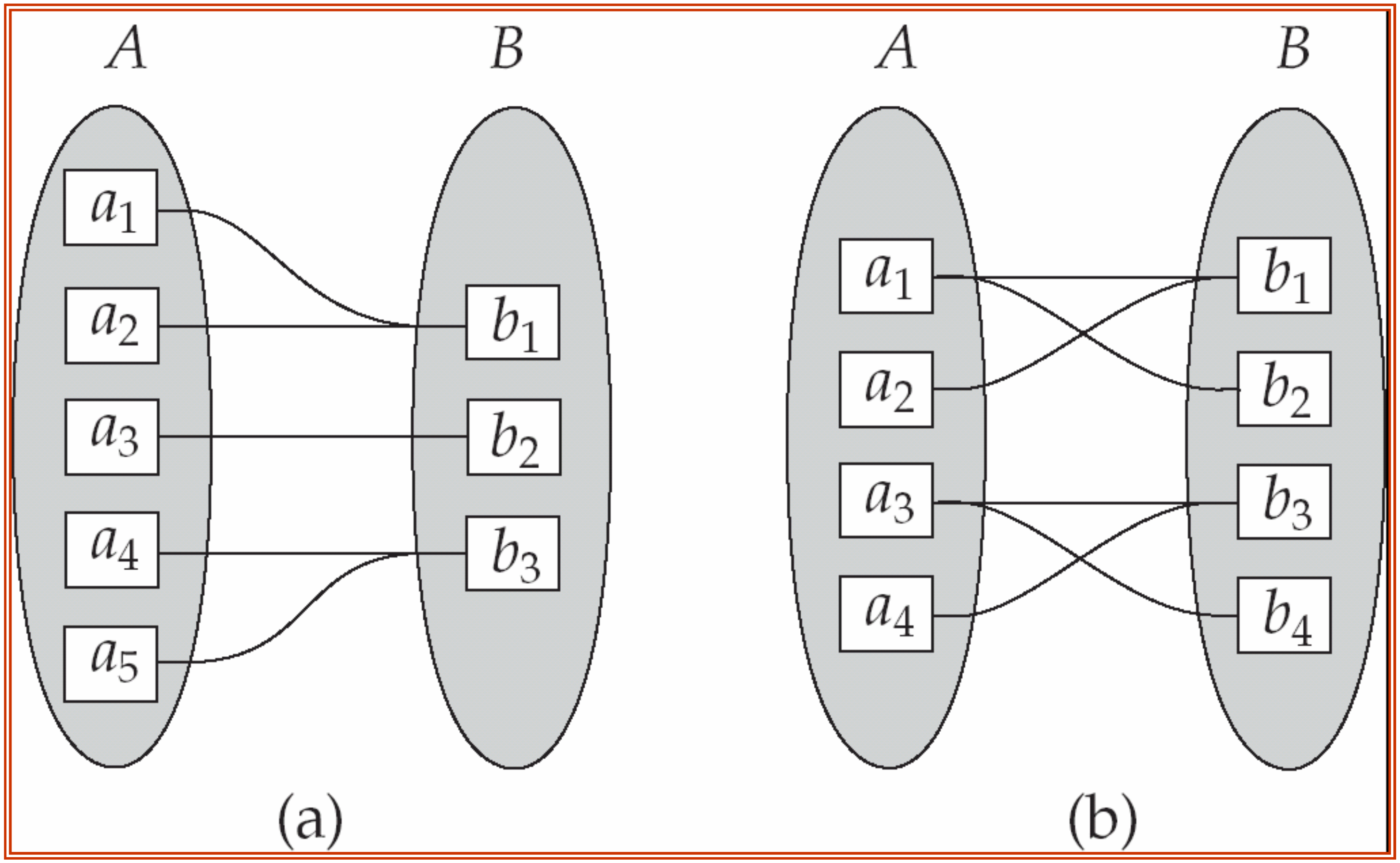


One to one

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

Keys

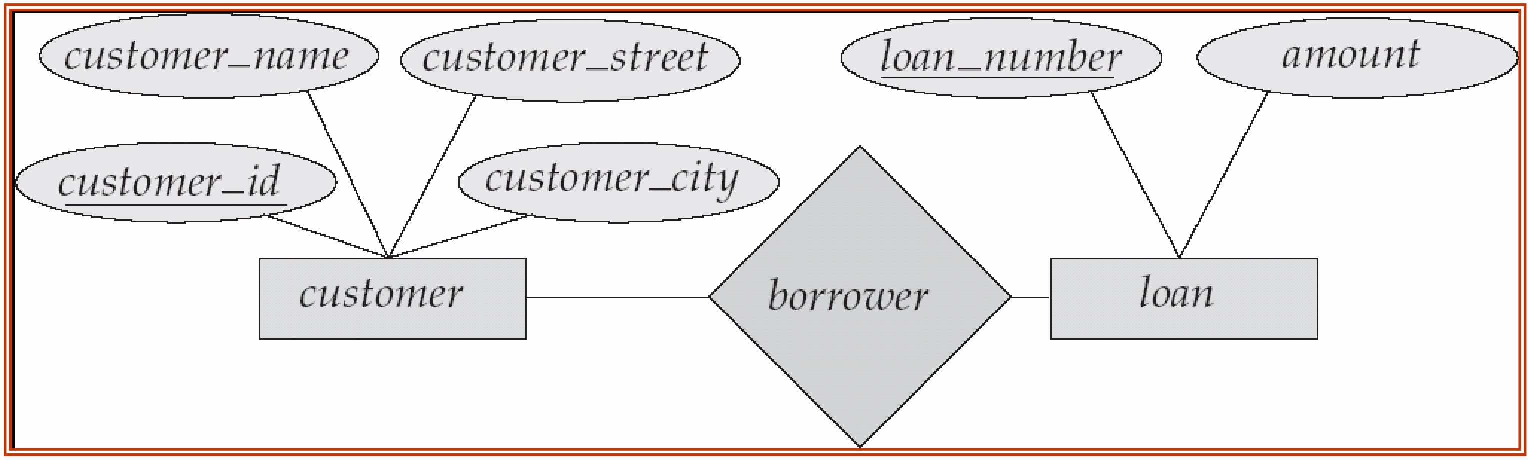
- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A **candidate key** of an entity set is a minimal super key
  - *Customer_id* is candidate key of *customer*
  - *account_number* is candidate key of *account*
- Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key**.

- Obs. Entities sometimes are considered to have only **keys**, without distinction about being primary or not!

Keys for Relationship Sets

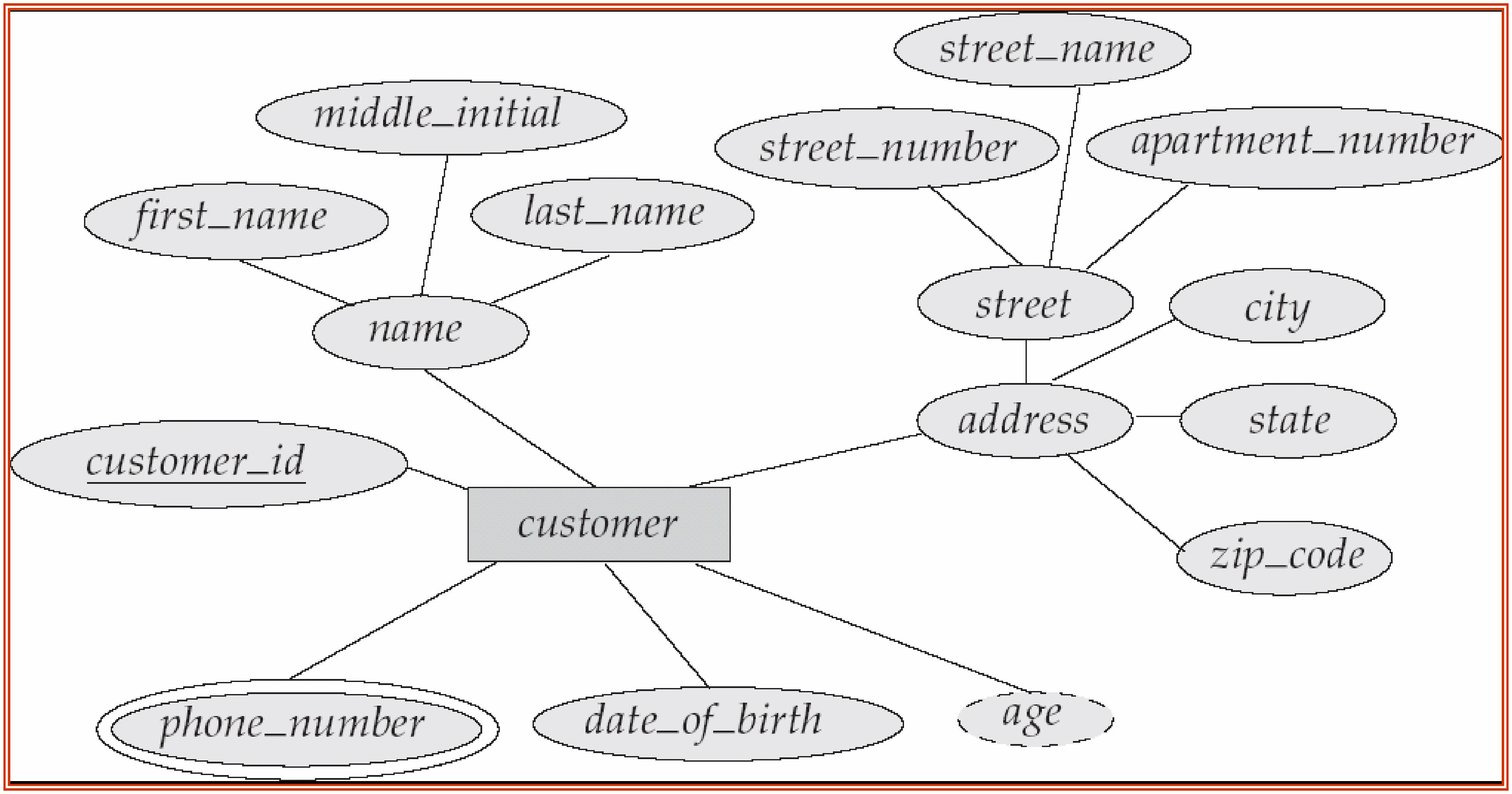
- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (*customer_id, account_number*) is the super key of *depositor*
  - *NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.*
    - Example: if we wish to track all access_dates to each account by each customer, we cannot assume a relationship for each access. We can use a multivalued attribute though
- Must consider the mapping cardinality of the relationship set when deciding the 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

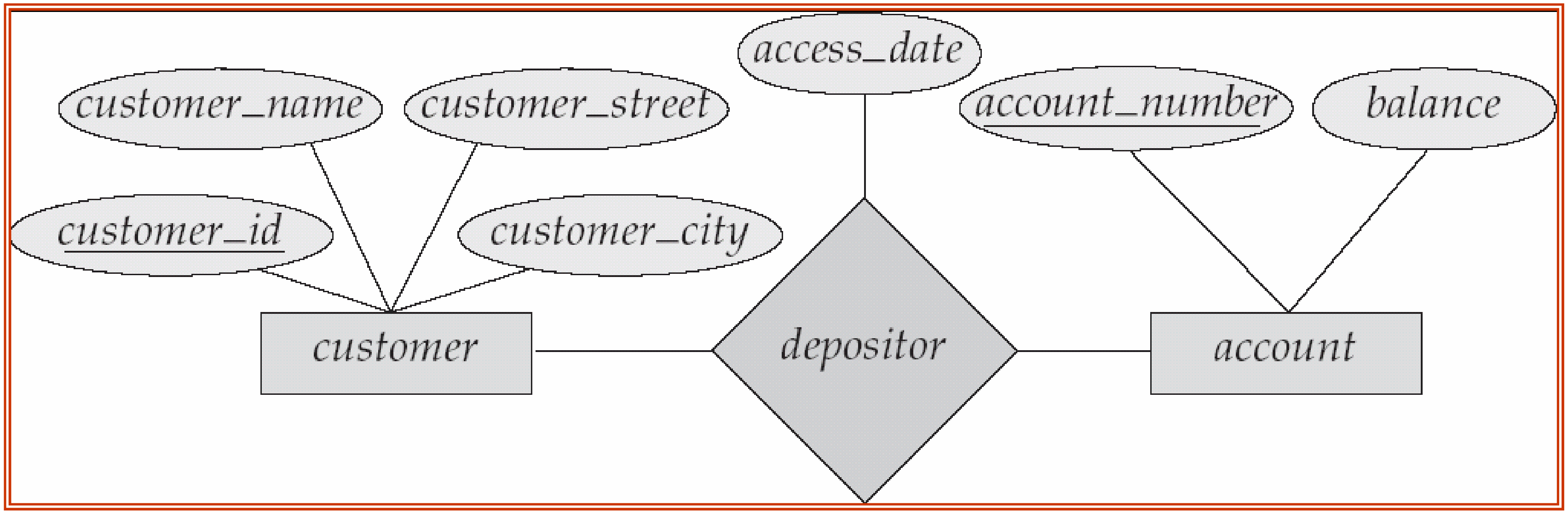


- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Ellipses represent attributes
  - Double ellipses represent multivalued attributes.
  - Dashed ellipses denote derived attributes.
- Underline indicates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

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

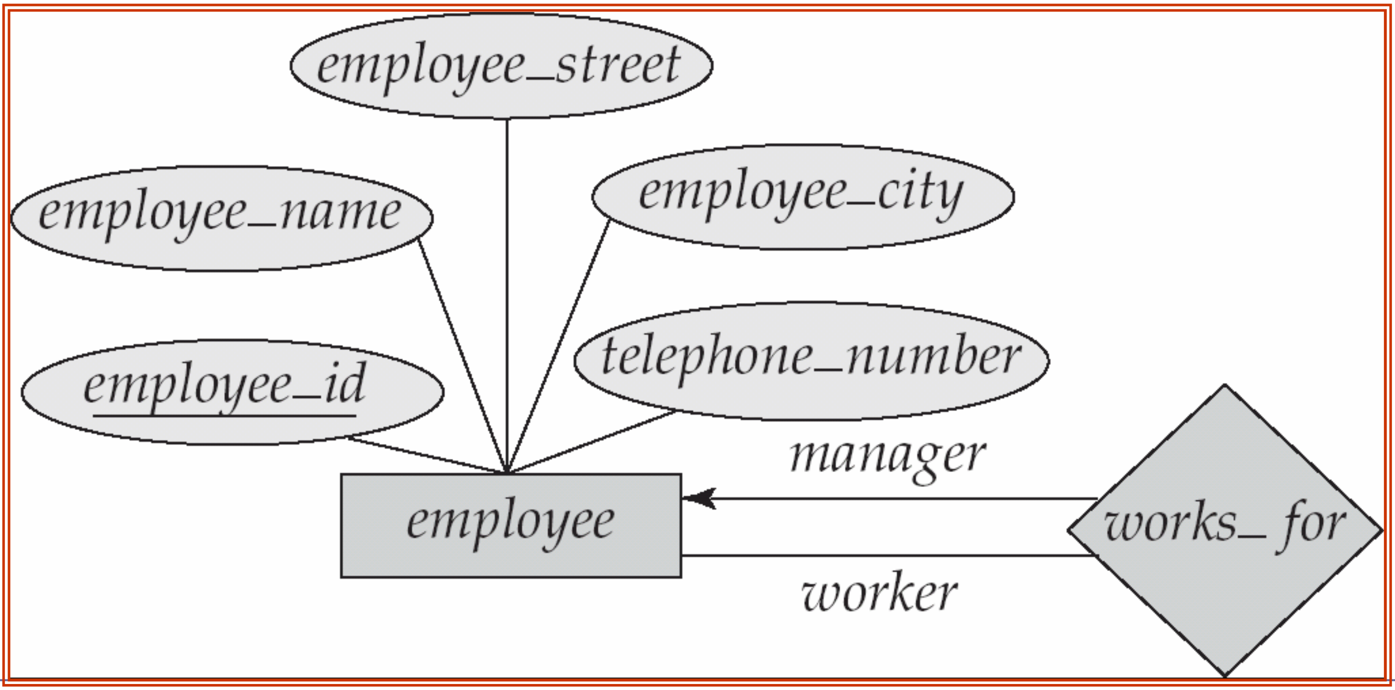


Relationship Sets with Attributes



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

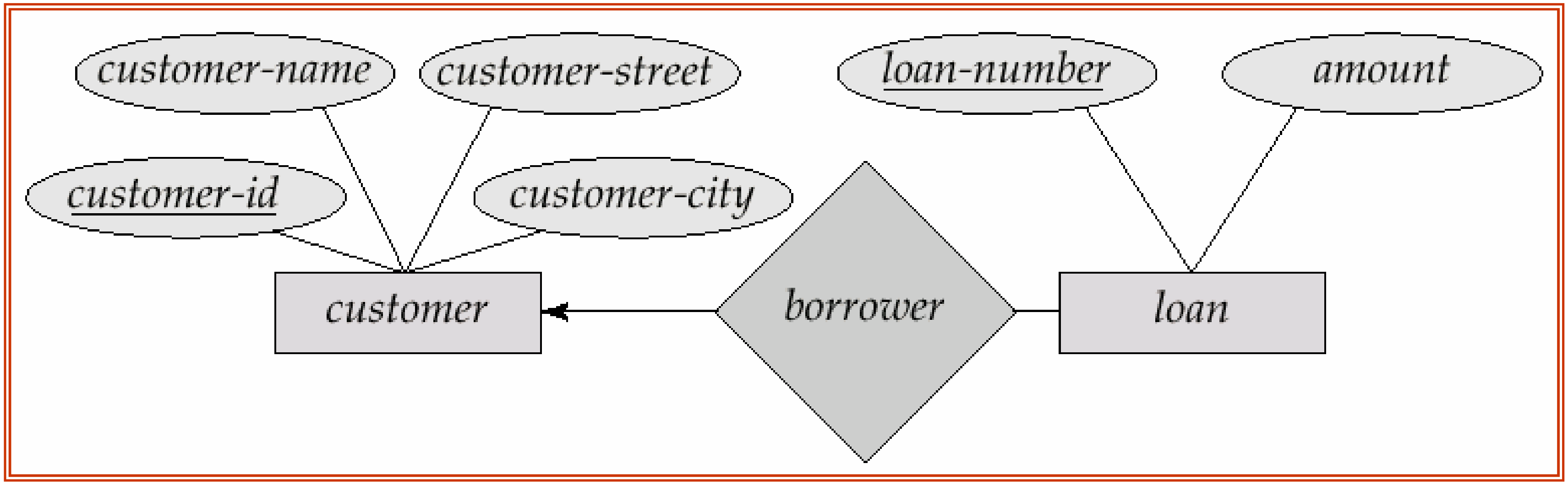


Cardinality Constraints

- 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*

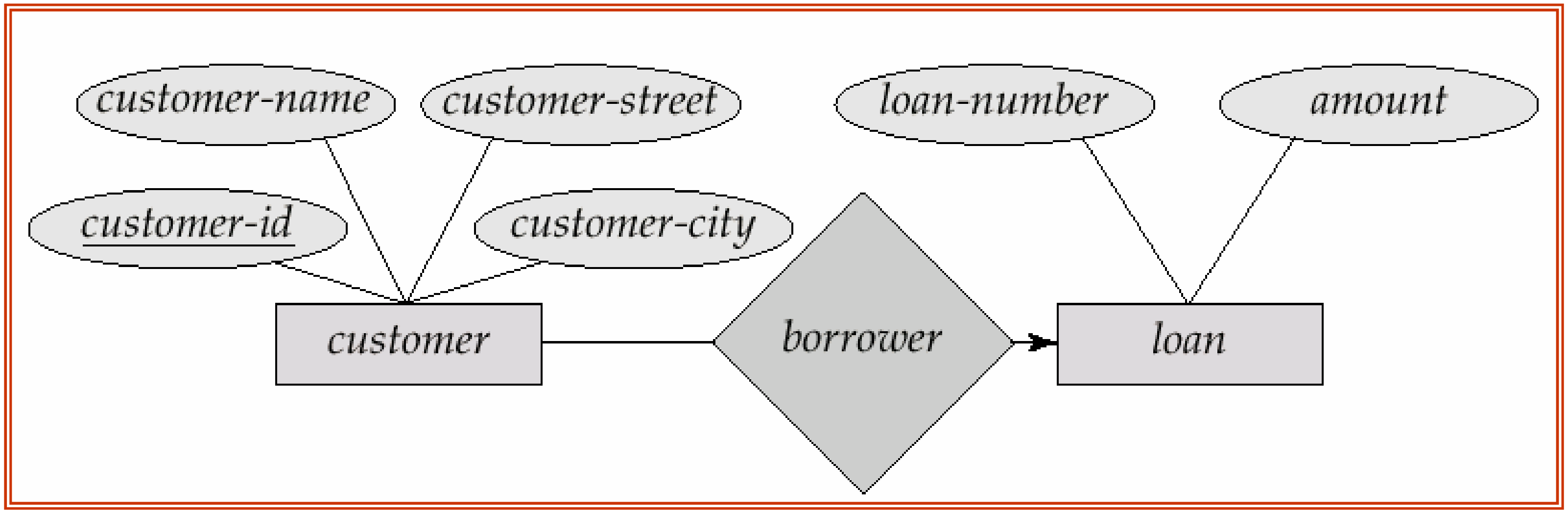
One-To-Many Relationship

- 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*



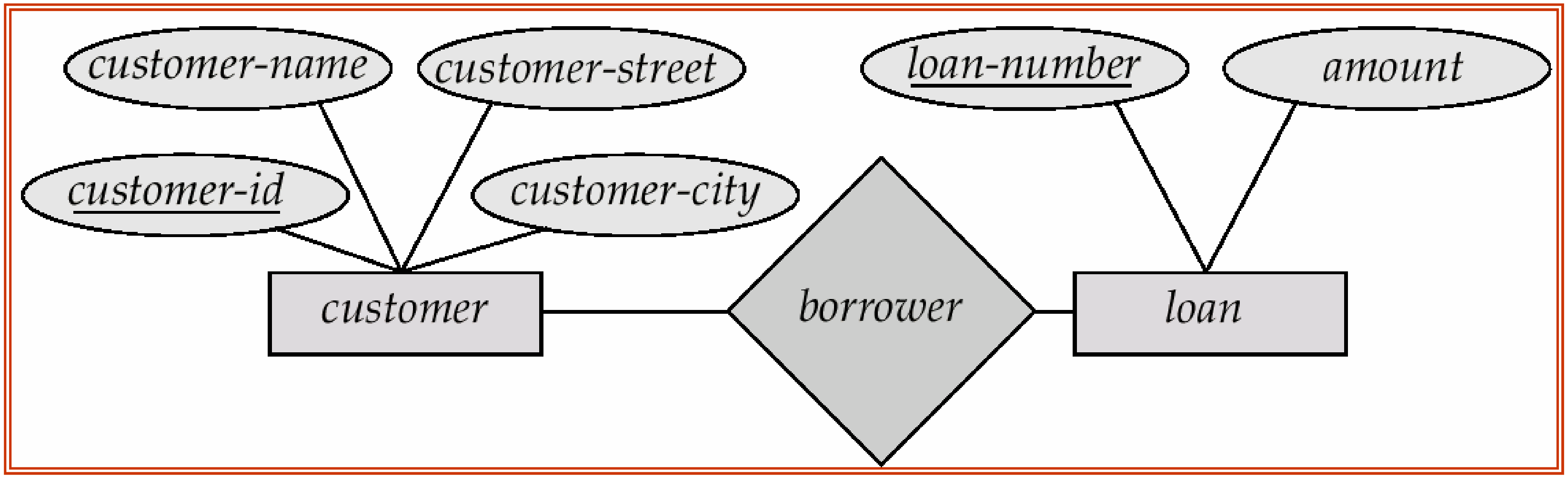
Many-To-One Relationships

- 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*



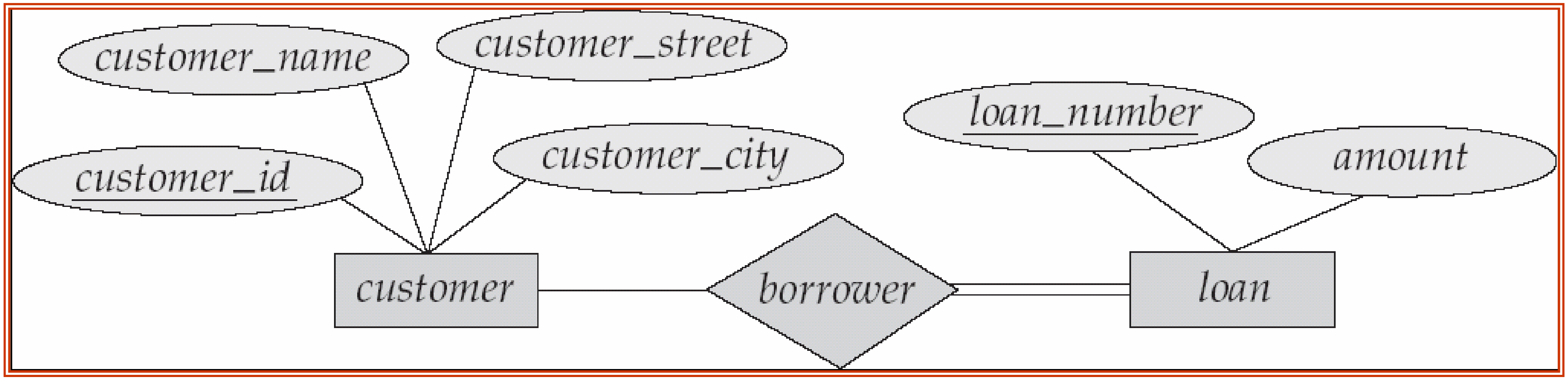
Many-To-Many Relationship

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



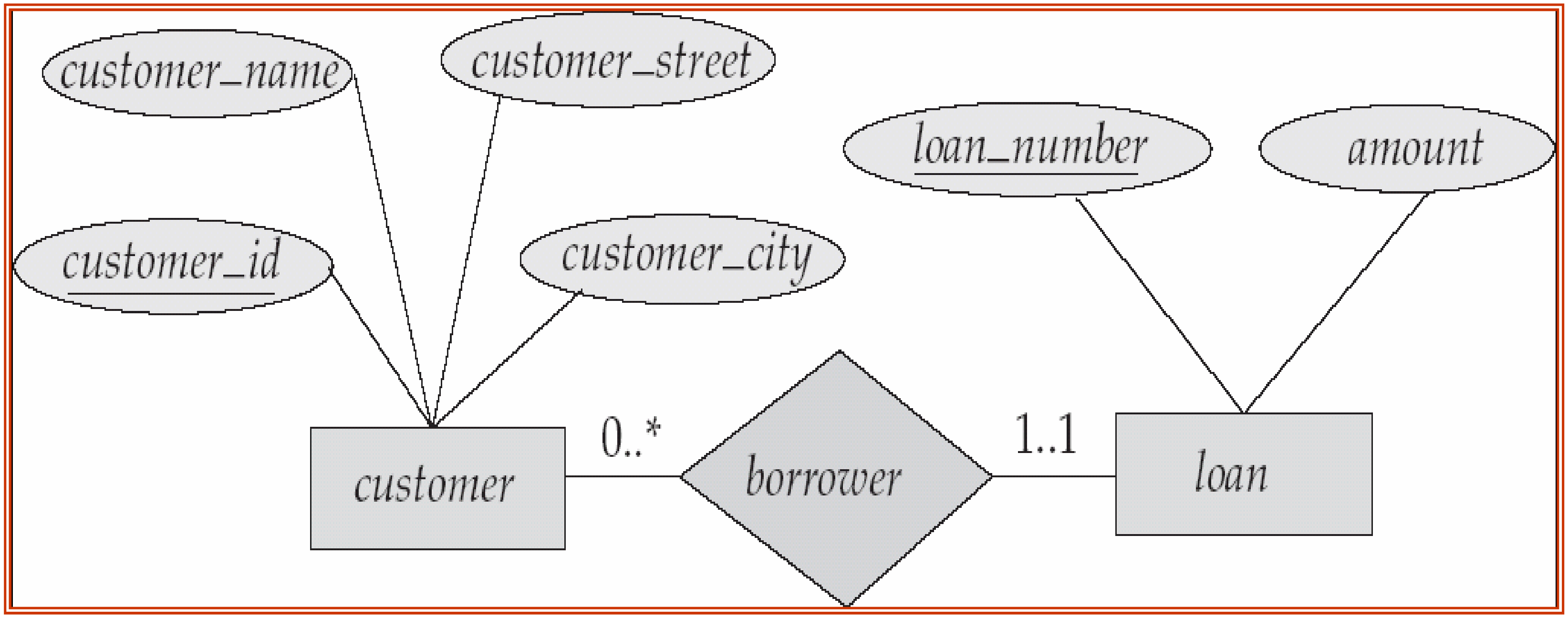
Participation of an Entity Set in a Relationship Set

- 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
  - Example: participation of customer in borrower is partial

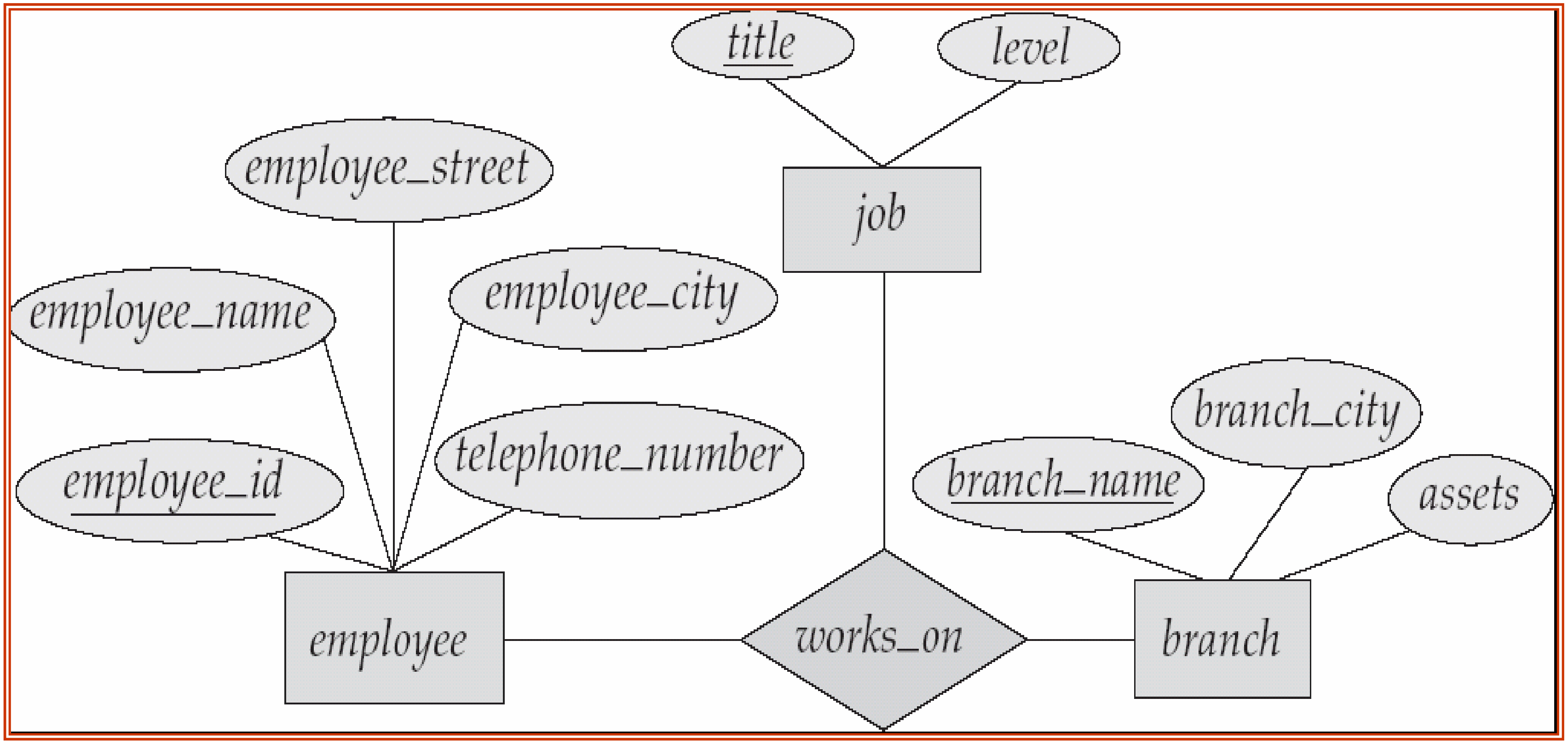


Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints



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

Design Issues

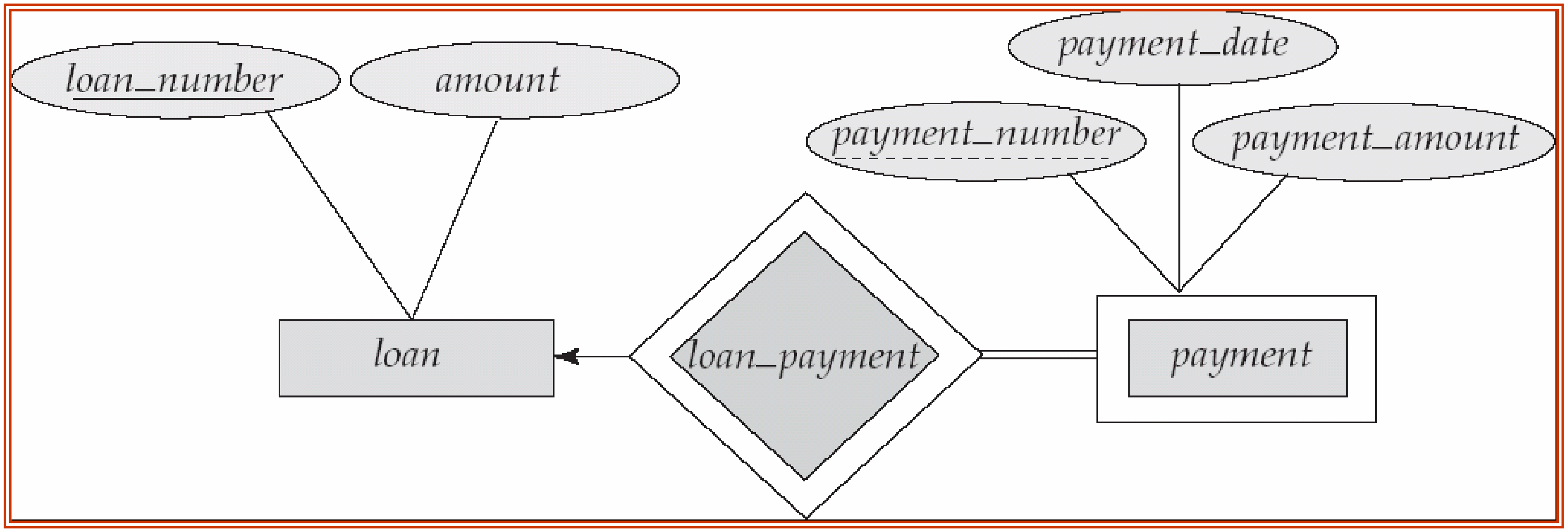
- **Use of entity sets vs. attributes**
  Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.
- **Use of entity sets vs. relationship sets**
  Possible guideline is to designate a relationship set to describe an action 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**

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.

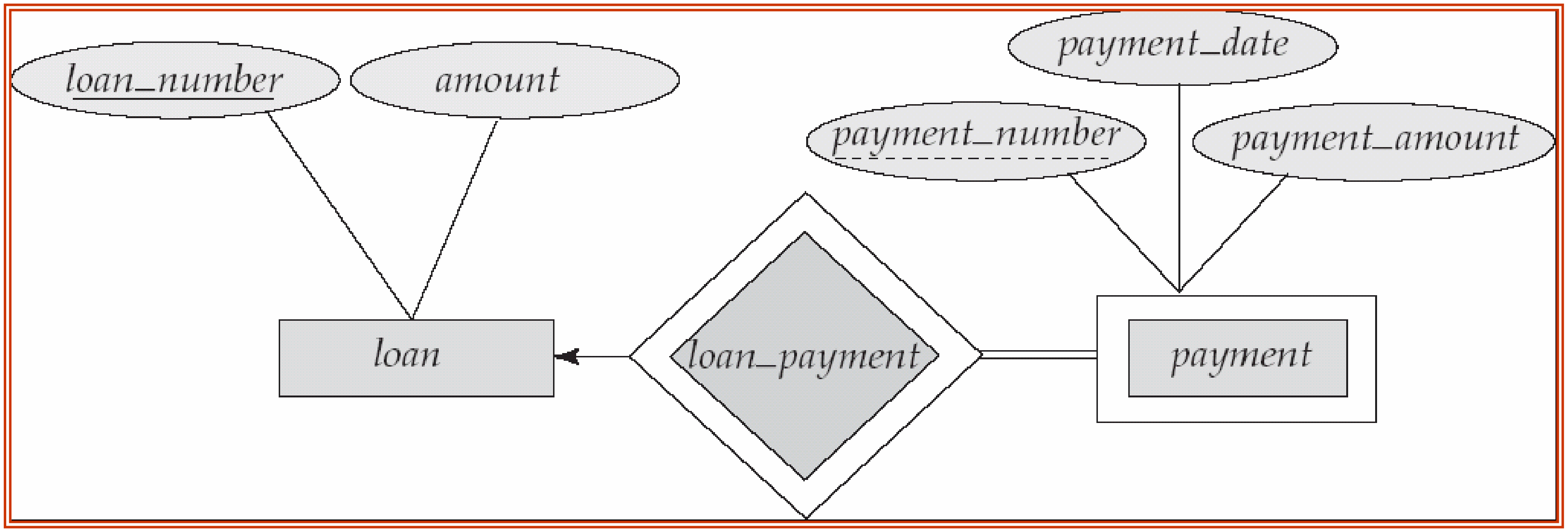
Weak Entity Sets (Cont.)

- 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

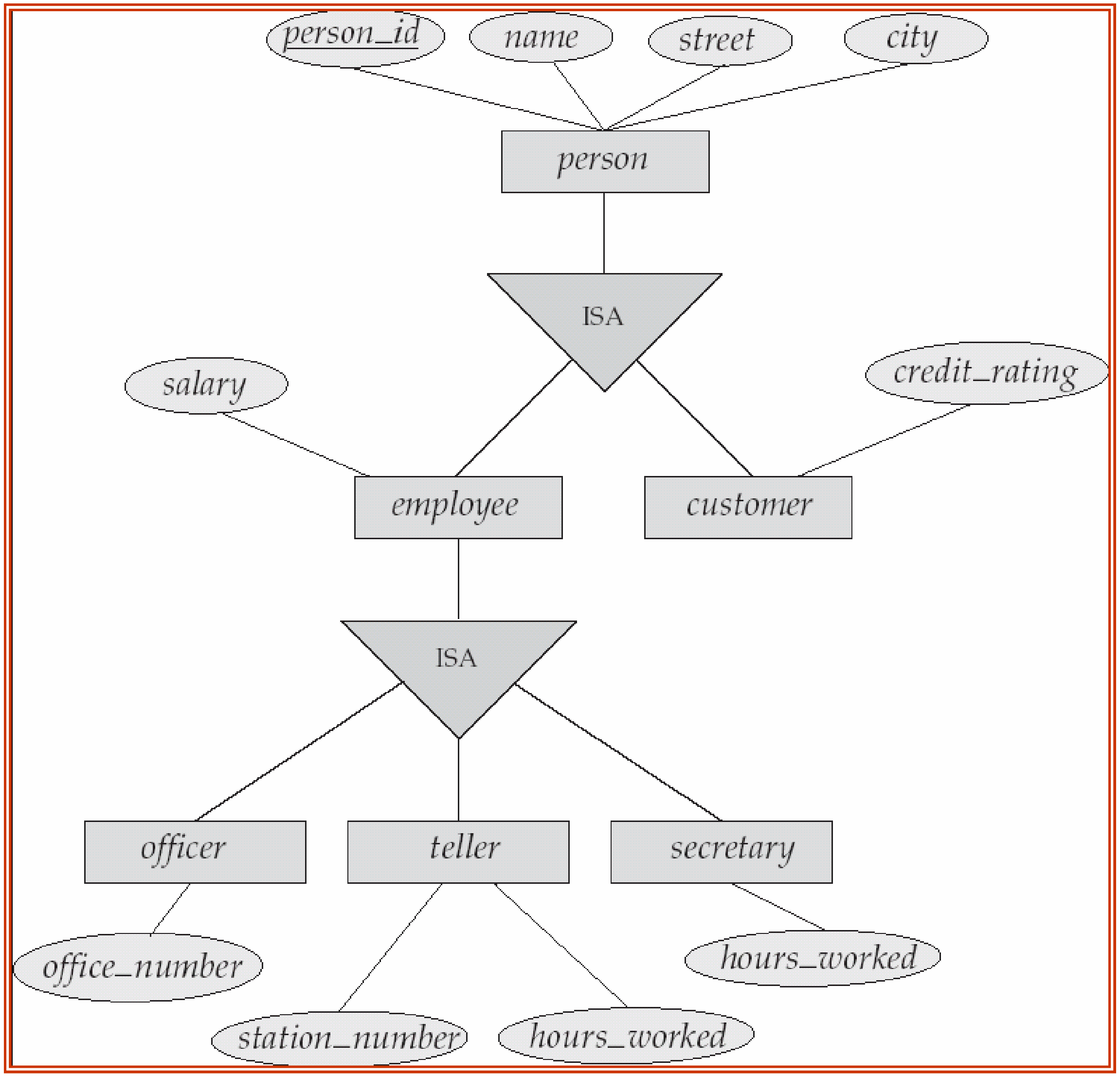
- In a university, a *course* is a strong entity and a *course_offering* can be modeled as a weak entity
- The discriminator of *course_offering* would be *semester* (including year) and *section_number* (if there is more than one section)
- If we model *course_offering* as a strong entity we would model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

  Then the relationship with *course* would be implicit in the *course_number* attribute

Extended E-R Features: 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



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

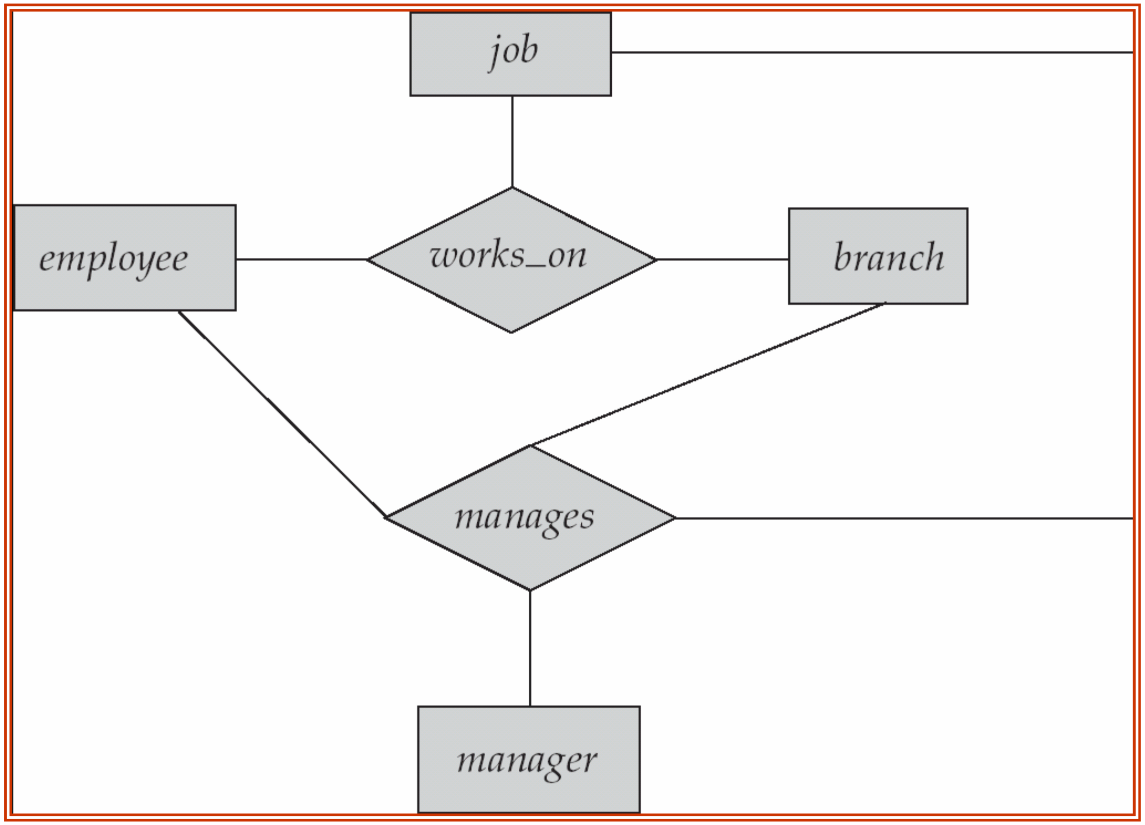
- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
  - user-defined
- Constraint on whether or not entities may belong to more than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

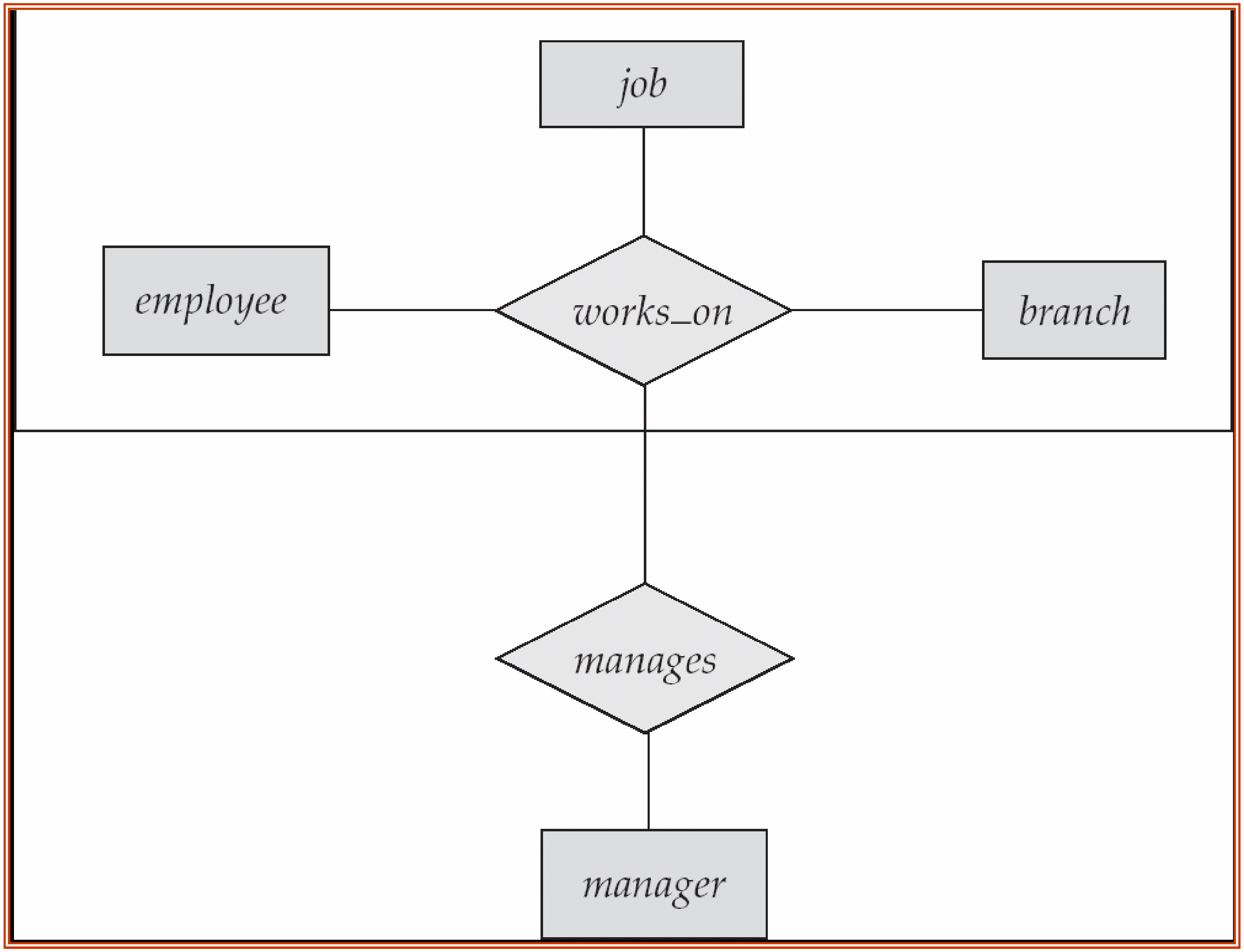
- Consider the ternary relationship *works_on*, which we saw earlier
- Suppose we want to record managers for tasks performed by an employee at a branch



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

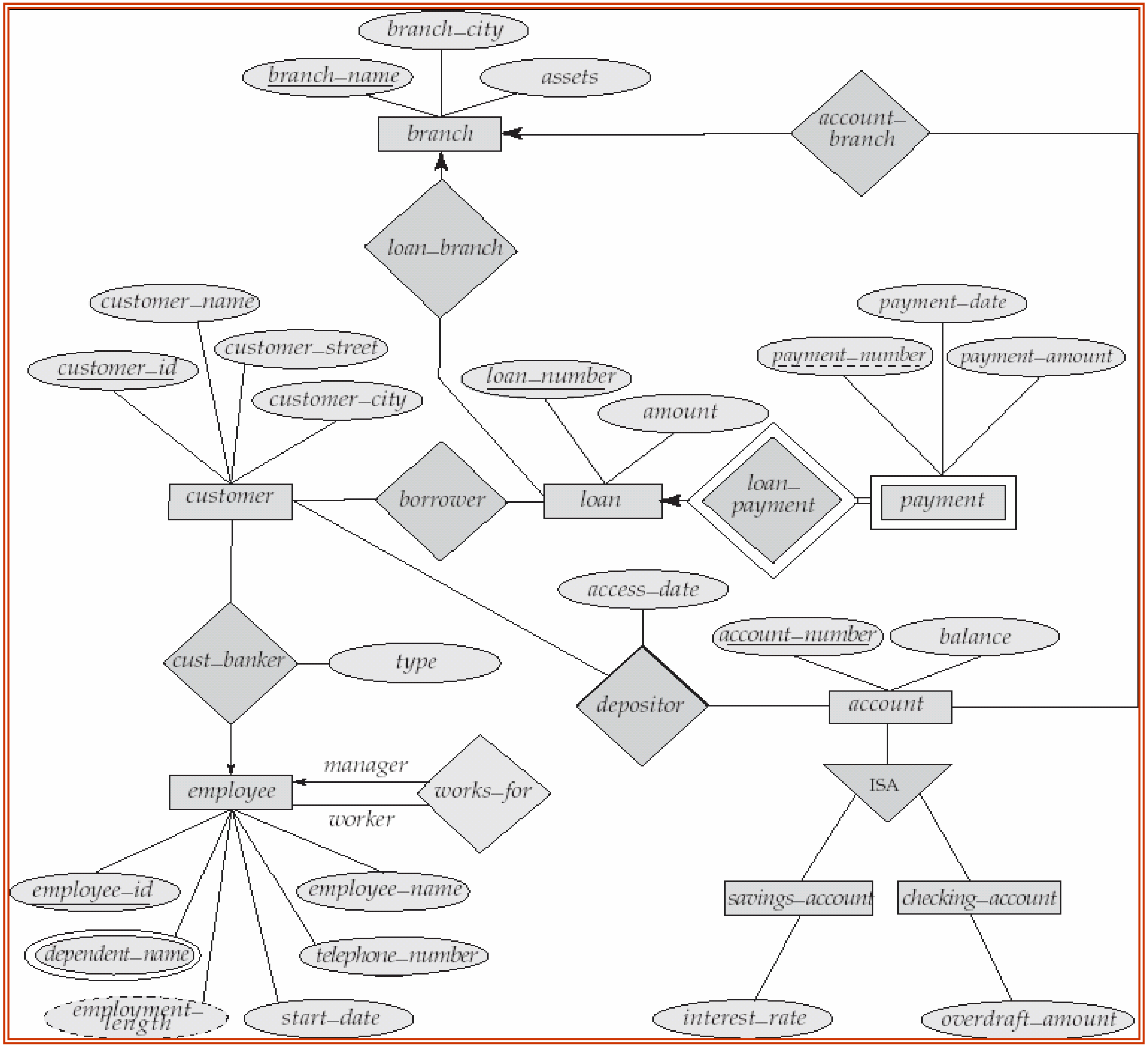
E-R Diagram With Aggregation



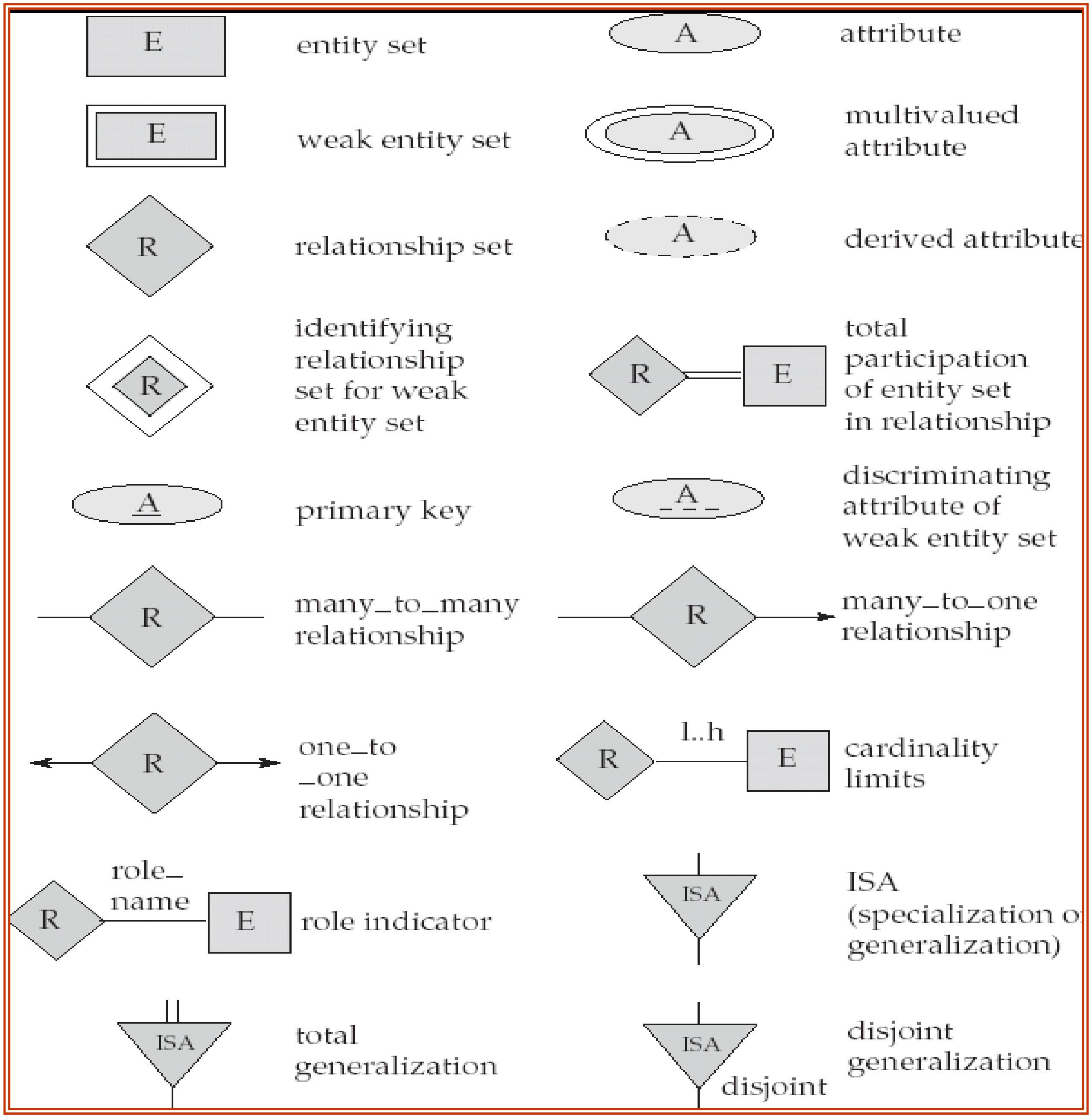
E-R Design Decisions

- 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.

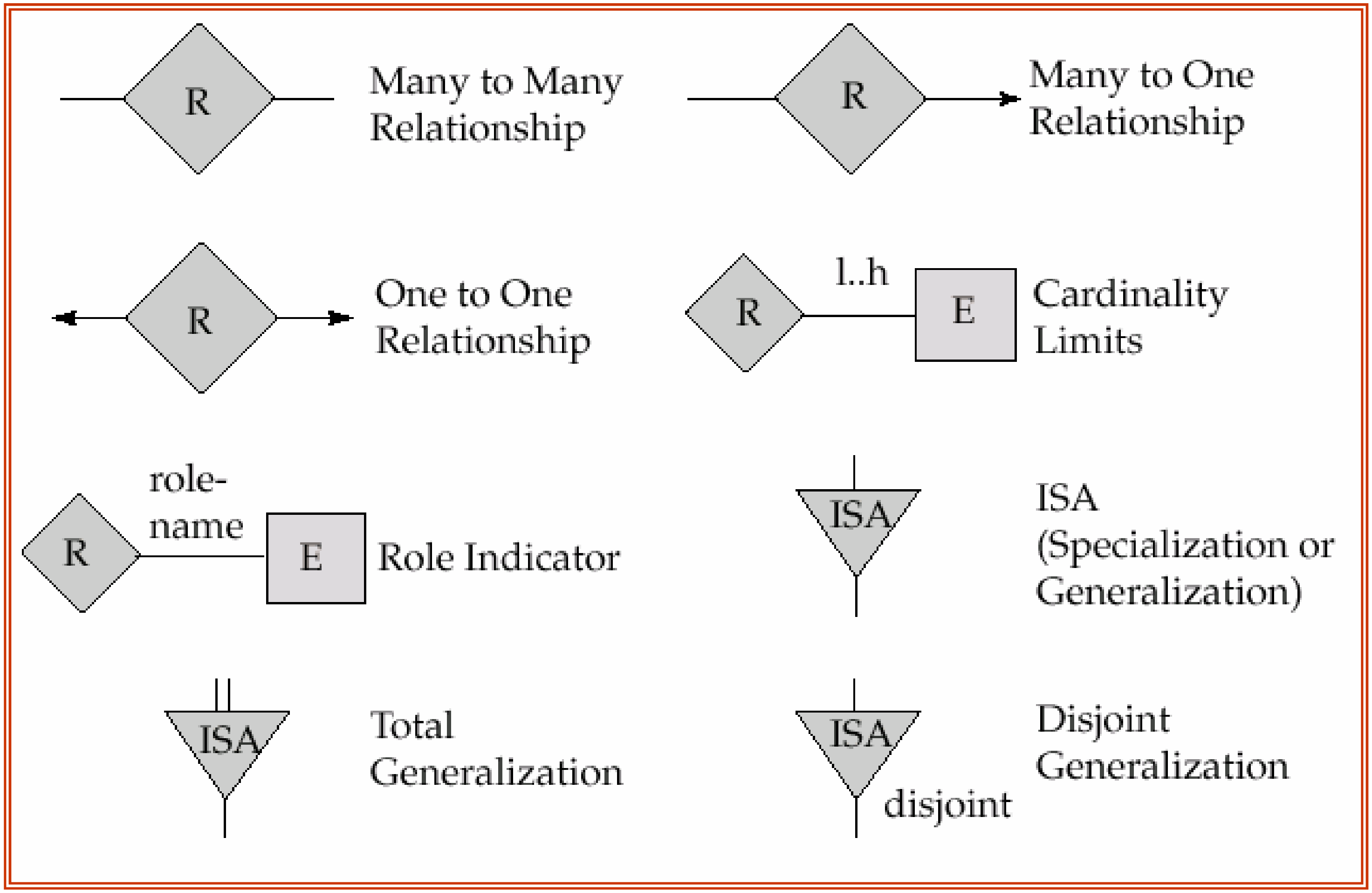
E-R Diagram for a Banking Enterprise



Summary of Symbols



Summary of Symbols (Cont.)



Example 2: COMPANY Database

- Requirements (oversimplified for illustrative purposes)
  - The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager.
  - Each department *controls* a number of PROJECTs*.* Each project has a name, number and is located at a single location.
  - We store each EMPLOYEE's social security number, address, salary, sex, and birthdate. Each employee *works for* one department but may *work on* several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the *direct supervisor* of each employee.
  - Each employee may *have* a number of DEPENDENTs. For each dependent, we keep track of their name, sex, birthdate, and relationship to employee.

ER Model Concepts

- Entities and Attributes
  - Entities are specific objects or things in the mini-world that are represented in the database.
    - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
  - Attributes are properties used to describe an entity.
    - For example an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate
  - A specific entity will have a value for each of its attributes.
    - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55‘
  - Each attribute has a *value set* (or data type) associated with it
    - e.g. integer, string, subrange, enumerated type, …

Types of Attributes

- Simple
  - Each entity has a single atomic value for the attribute.
    - For example, SSN or Sex.
- Composite
  - The attribute may be composed of several components.
    - For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName). Composition may form a hierarchy where some components are themselves composite.
- Multi-valued
  - An entity may have multiple values for that attribute.
    - For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.

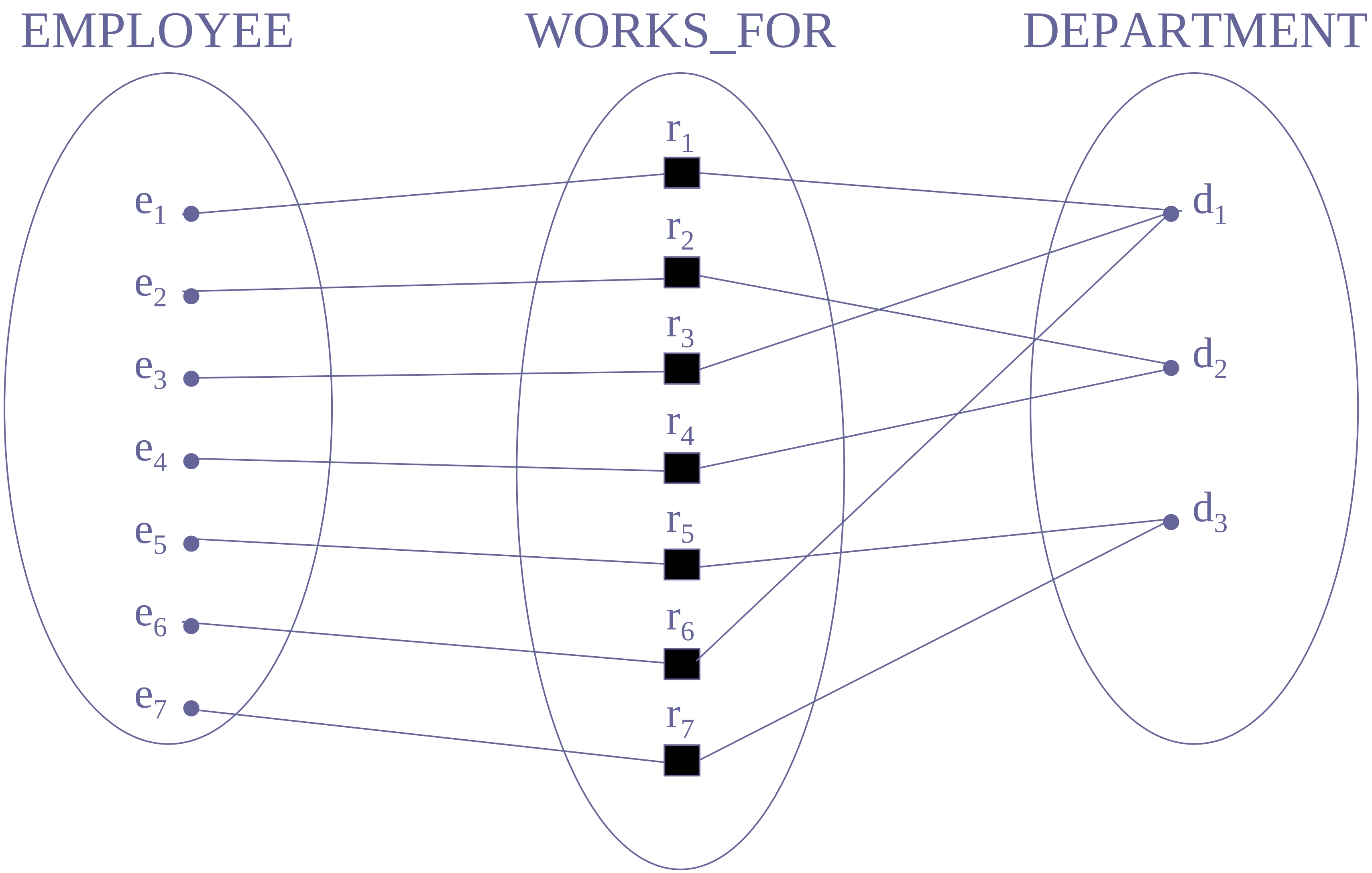
Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the EMPLOYEE entity type or the PROJECT entity type.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
  - For example, SSN of EMPLOYEE.
- A key attribute may be composite.
  - For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
  - For example, the CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN) and
    - VehicleTagNumber (Number, State), also known as license_plate number.

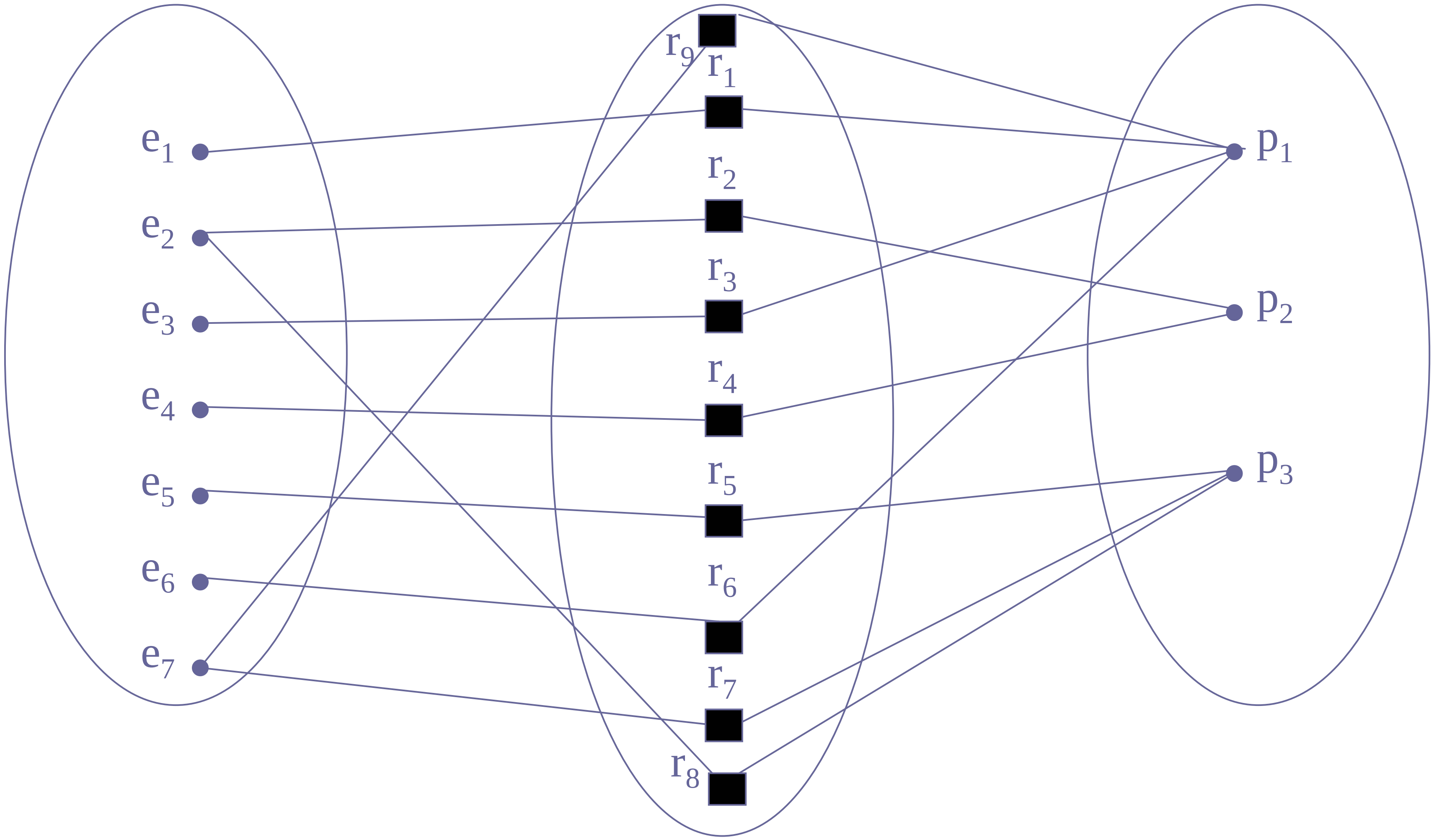
Relationships and Relationship Types

- A relationship relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped into a relationship type.
  - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types.
  - Both MANAGES and WORKS_ON are binary relationships.
- More than one relationship type can exist with the same participating entity types.
  - For example, MANAGES and WORKS_FOR are distinct relationships between EMPLOYEE and DEPARTMENT, but with different meanings and different relationship instances.

Example relationship instances of the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT



Example relationship instances of the WORKS_ON relationship between EMPLOYEE and PROJECT



Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

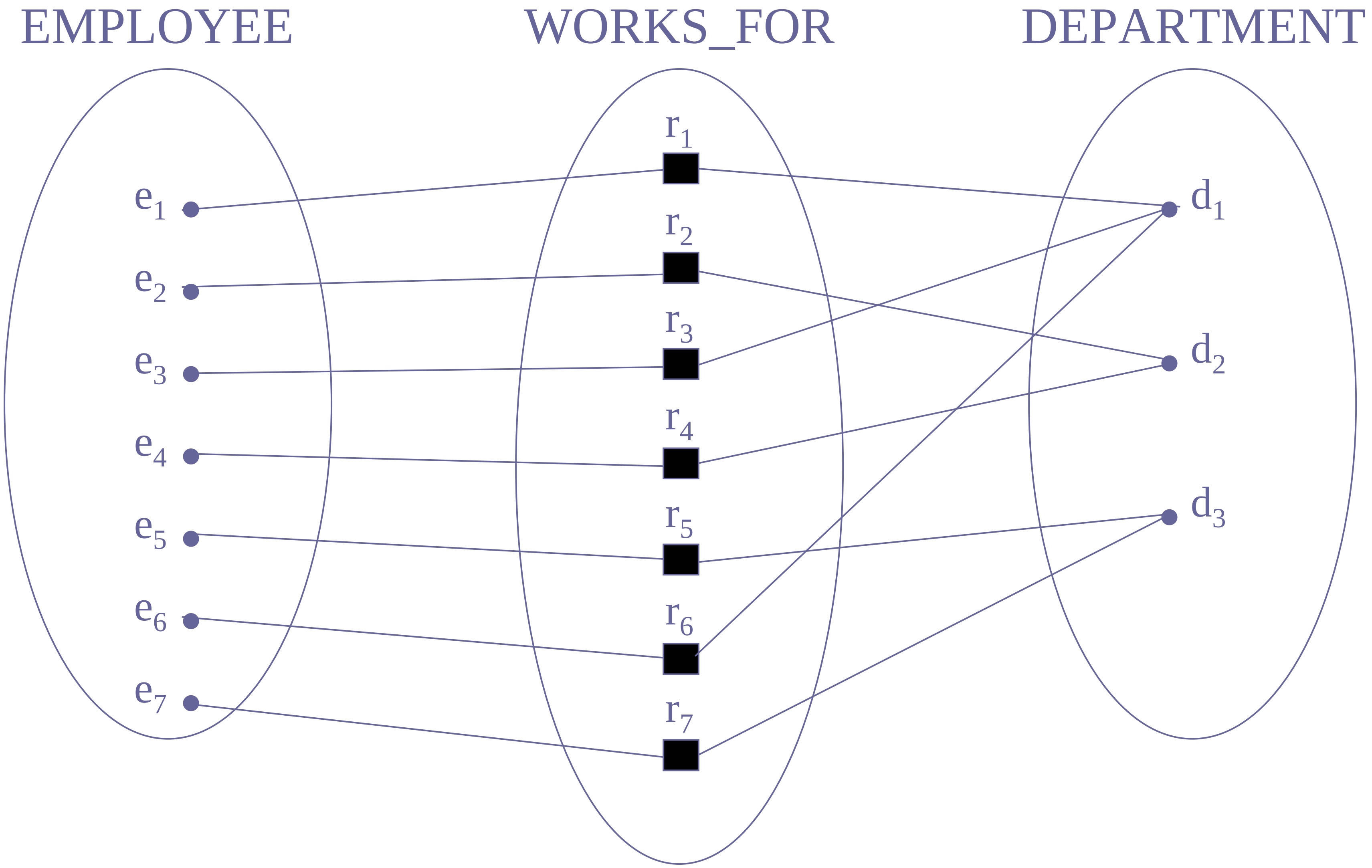
Example:

Suppose that a DEPENDENT entity is identified by the dependent's first name and birthdates, *and* the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

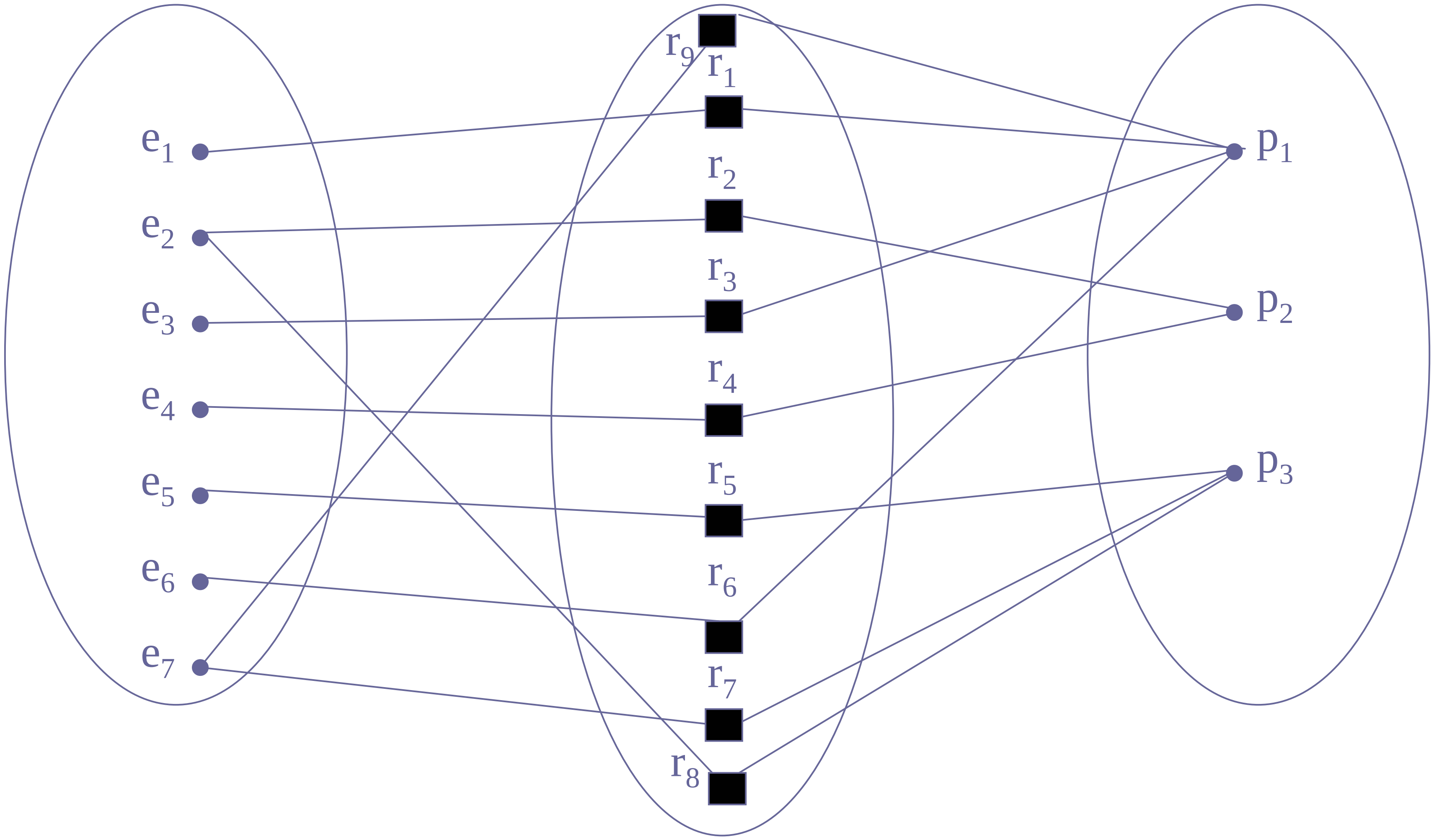
Constraints on Relationships

- Constraints on Relationship Types
  - (Also known as ratio constraints)
  - Maximum Cardinality
    - One-to-one (1:1)
    - One-to-many (1:N) or Many-to-one (N:1)
    - Many-to-many
  - Minimum Cardinality (also called participation constraint or existence dependency constraints)
    - zero (optional participation, not existence-dependent)
    - one or more (mandatory, existence-dependent)

Many-to-one (N:1) RELATIONSHIP



Many-to-many (M:N) RELATIONSHIP

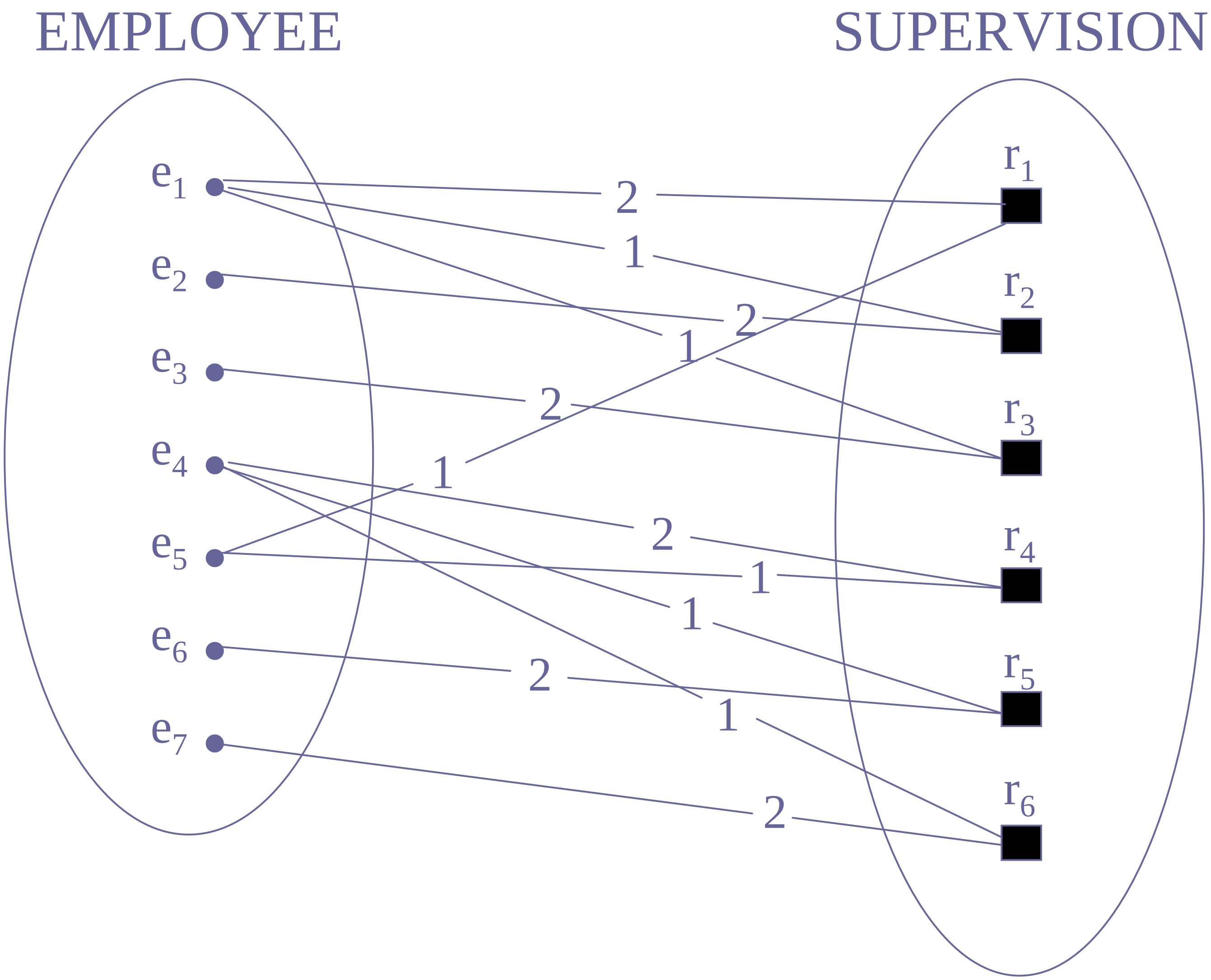


Relationships and Relationship Types (2)

- We can also have a **recursive** relationship type.
- Both participations are same entity type in different roles.
- For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).
- In following figure, first role participation labeled with 1 and second role participation labeled with 2.
- In ER diagram, need to display role names to distinguish participations.

- A relationship type can also have attributes
  - for example, HoursPerWeek of WORKS_ON; its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.

A RECURSIVE RELATIONSHIP SUPERVISION



Structural Constraints – one way to express semantics of relationships

Structural constraints on relationships:

- **Cardinality ratio** (of a binary relationship): 1:1, 1:N, N:1, or M:N

  SHOWN BY PLACING APPROPRIATE NUMBER ON THE LINK.

- **Participation constraint** (on each participating entity type): total (called *existence dependency*) or partial.

  SHOWN BY DOUBLE LINING THE LINK

NOTE: These are easy to specify for Binary Relationship Types.

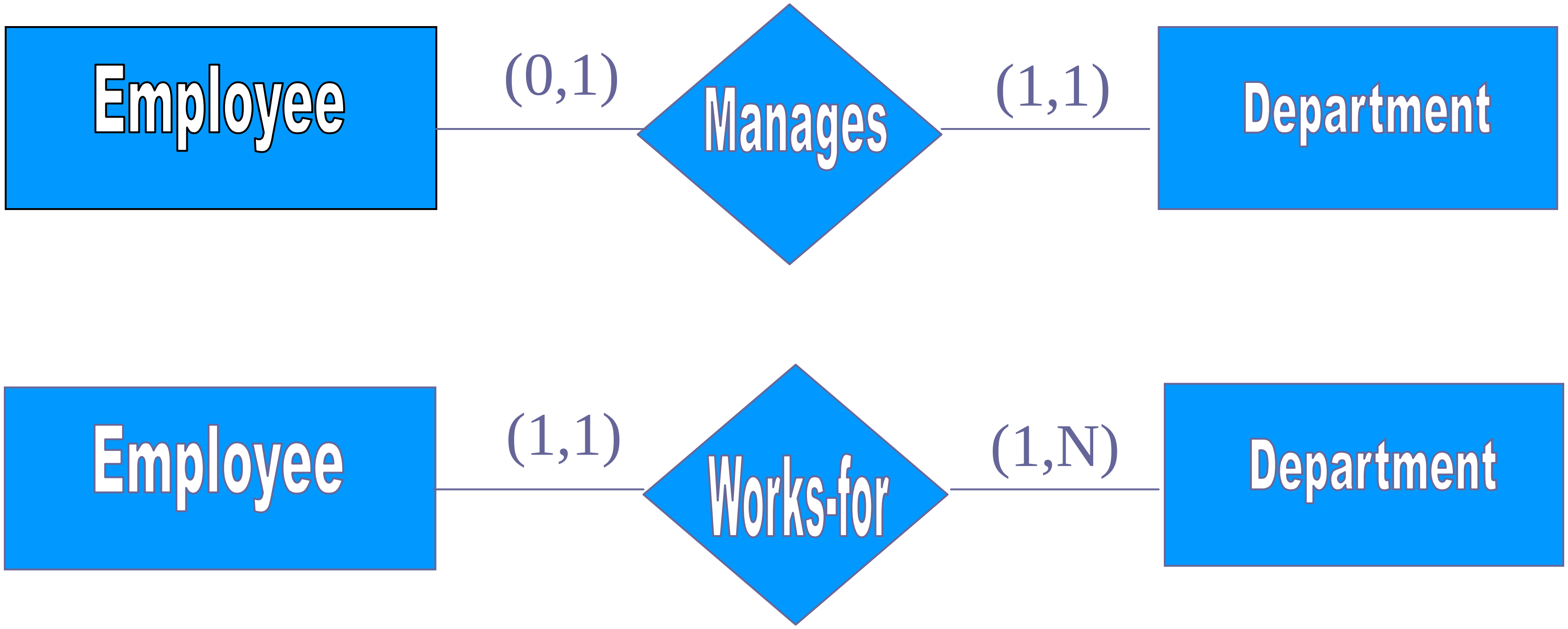
Alternative (min, max) notation for relationship structural constraints

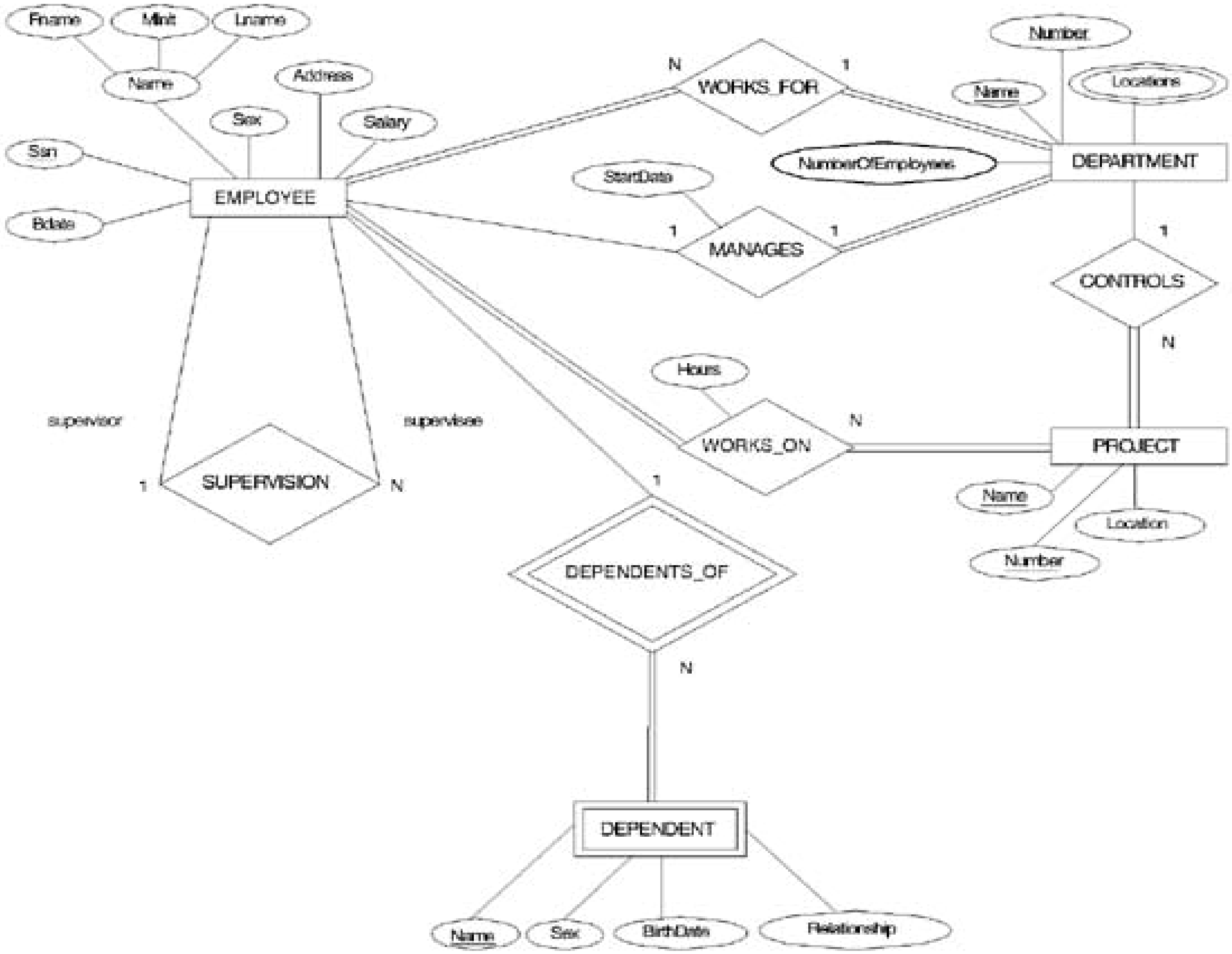
- Specified on *each participation* of an entity type E in a relationship type R
- Specifies that each entity e in E participates in *at least* min and *at most* max relationship instances in R
- Default(no constraint): min=0, max=n
- Must have min≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints

<u>Examples:</u>

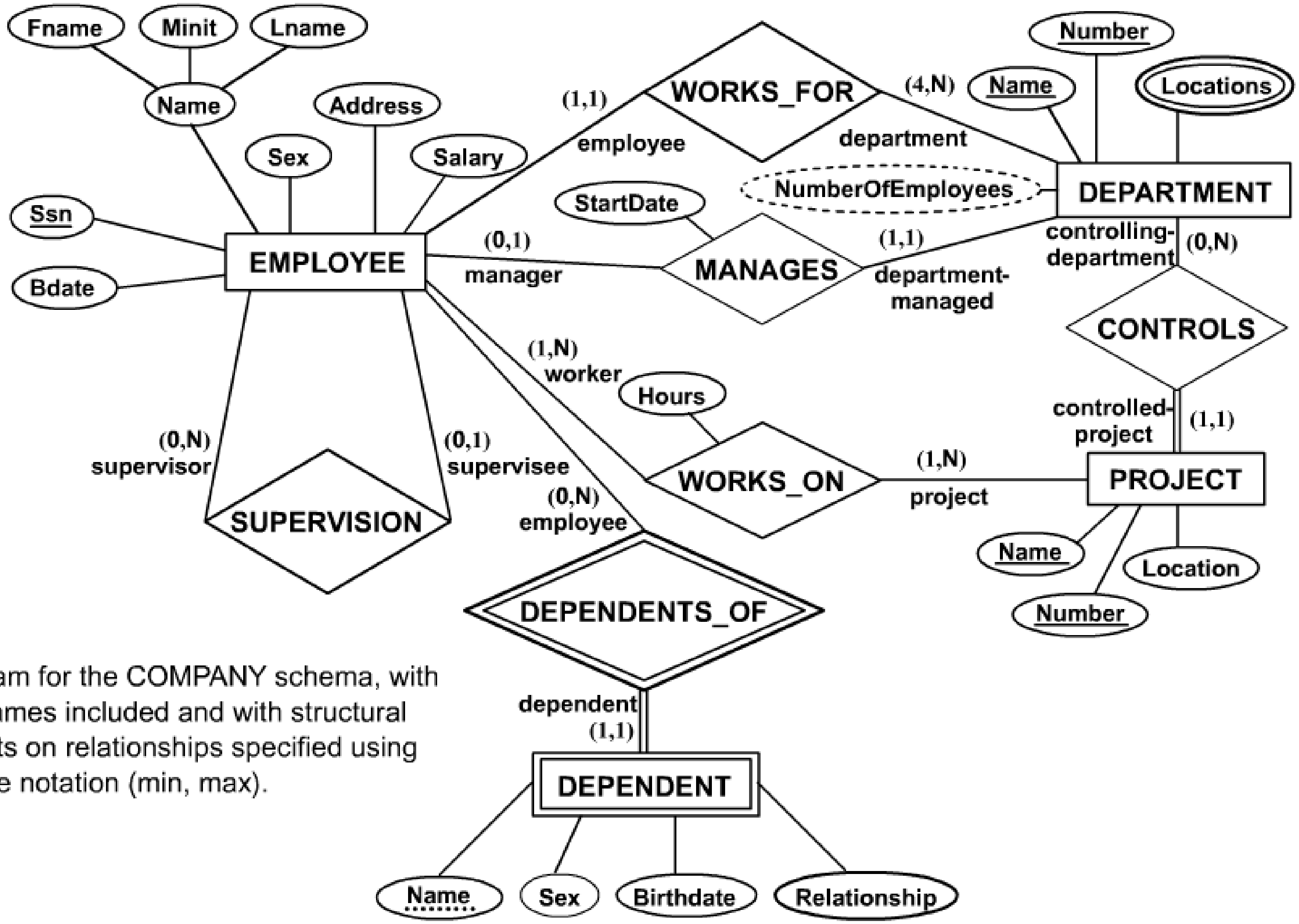
- A department has *exactly one* manager and an employee can manage *at most one* department.
  - Specify (0,1) for participation of EMPLOYEE in MANAGES
  - Specify (1,1) for participation of DEPARTMENT in MANAGES
- An employee can work for *exactly one* department but a department can have *any number of employees*.
  - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
  - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

The (min,max) notation relationship constraints




Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
NumberOfEmployees
DEPARTMENT
StartDate
1
MANAGES
1
1
CONTROLS
N
Hours
supervisor
supervisee
WORKS_ON
N
PROJECT
1
SUPERVISION
N
1
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

Alternative ER Notations



ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).

Reduction to Relation Schemas

- Primary keys allow entity sets and relationship sets to be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

Representing Entity Sets as Schemas

- A strong entity set reduces to a schema with the same attributes.
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

payment =

(*<u>loan_number</u>*, *<u>payment_number</u>*, *payment_date*, *payment_amount*)

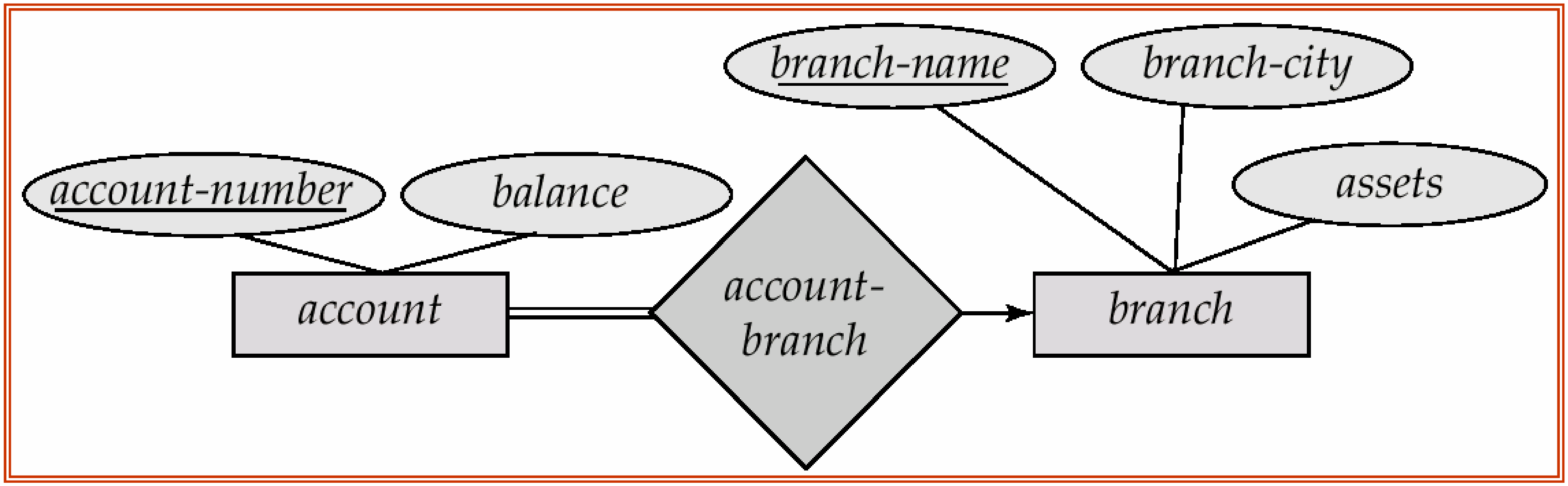
Representing Relationship Sets as Schemas

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

borrower = (<u>*customer_id, loan_number*</u>)

Redundancy of Schemas

- 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
- Example: Instead of creating a schema for relationship set *account_branch*, add an attribute *branch_name* to the schema arising from entity set *account*



Redundancy of Schemas (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 schema by an extra attribute in the schema corresponding to the “many” side could result in null values
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - Example: The *payment* schema already contains the attributes that would appear in the *loan_payment* schema (i.e., *loan_number* and *payment_number*).

Composite and Multivalued Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *custome*r with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes
    name.first_name and *name.last_name*
- A multivalued attribute *M* of an entity *E* is represented by a separate schema *EM*
  - Schema *EM* has attributes corresponding to the primary key of *E* and an attribute corresponding to multivalued attribute *M*
  - Example: Multivalued attribute *dependent_names* of *employee* is represented by a schema: *employee_dependent_names* = (*<u>employee_id</u>, dname*)
  - Each value of the multivalued attribute maps to a separate tuple of the relation on schema *EM*
    - An employee entity with primary key 123-45-6789 and dependents Jack and Jane maps to two tuples:
      (123-45-6789 , Jack) and (123-45-6789 , Jane)

Representing Specialization via Schemas

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

schema	attributes
person	*name, street, city*
customer	*name, credit_rating*
employee	*name, salary*

  - Drawback: getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

Representing Specialization as Schemas (Cont.)

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

schema	attributes
person	*name, street, city*
customer	*name, street, city, credit_rating*
employee	*name, street, city, salary*

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

Schemas Corresponding to Aggregation

To represent aggregation, create a schema containing

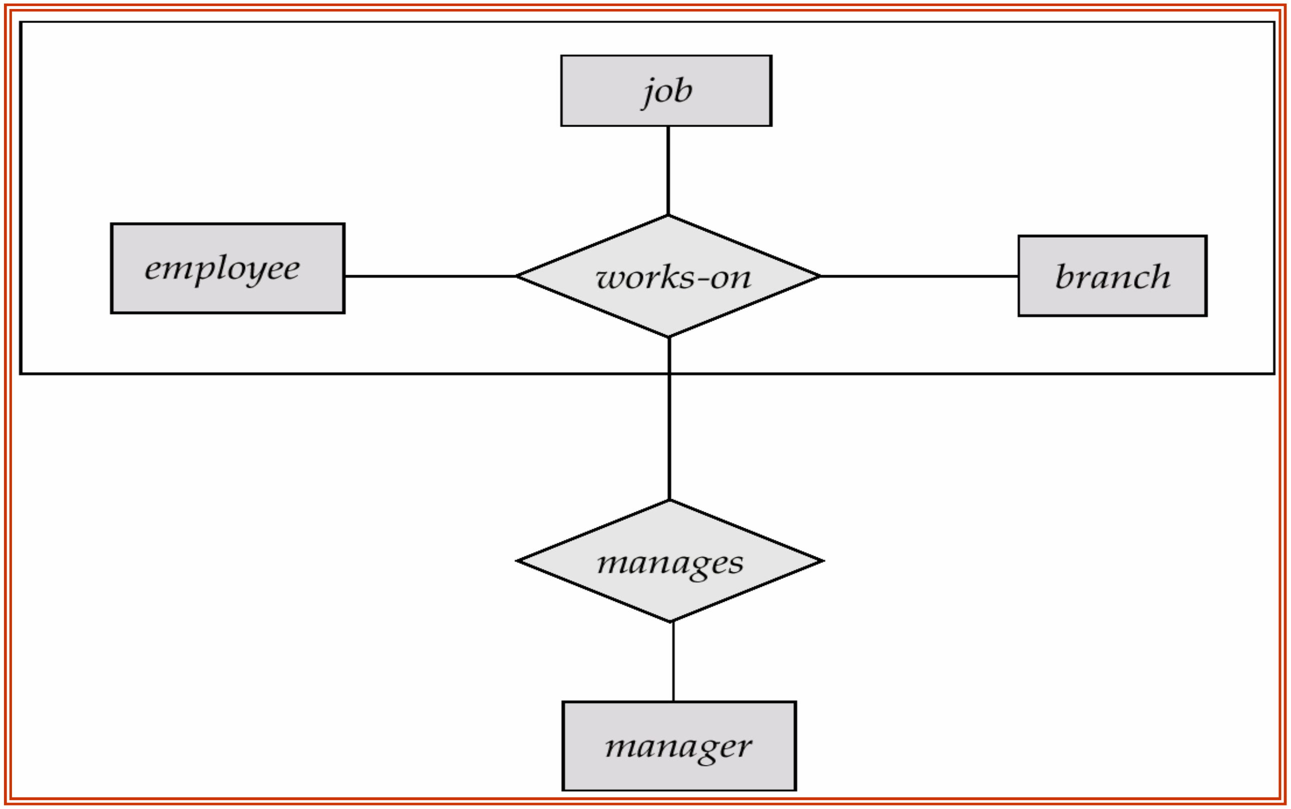
- primary key of the aggregated relationship,
- the primary key of the associated entity set
- any descriptive attributes

For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema

manages (*employee_id, branch_name, title, manager_name*)

Schemas Corresponding to Aggregation (Cont.)

- Schema *works_on* is redundant provided we are willing to store null values for attribute *manager_name* in relation on schema *manages*


job
employee
works-on
branch
manages
manager