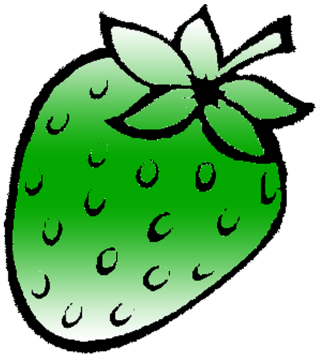
STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com

UNIT II – A. **ENTITY RELATIONSHIP MODEL**

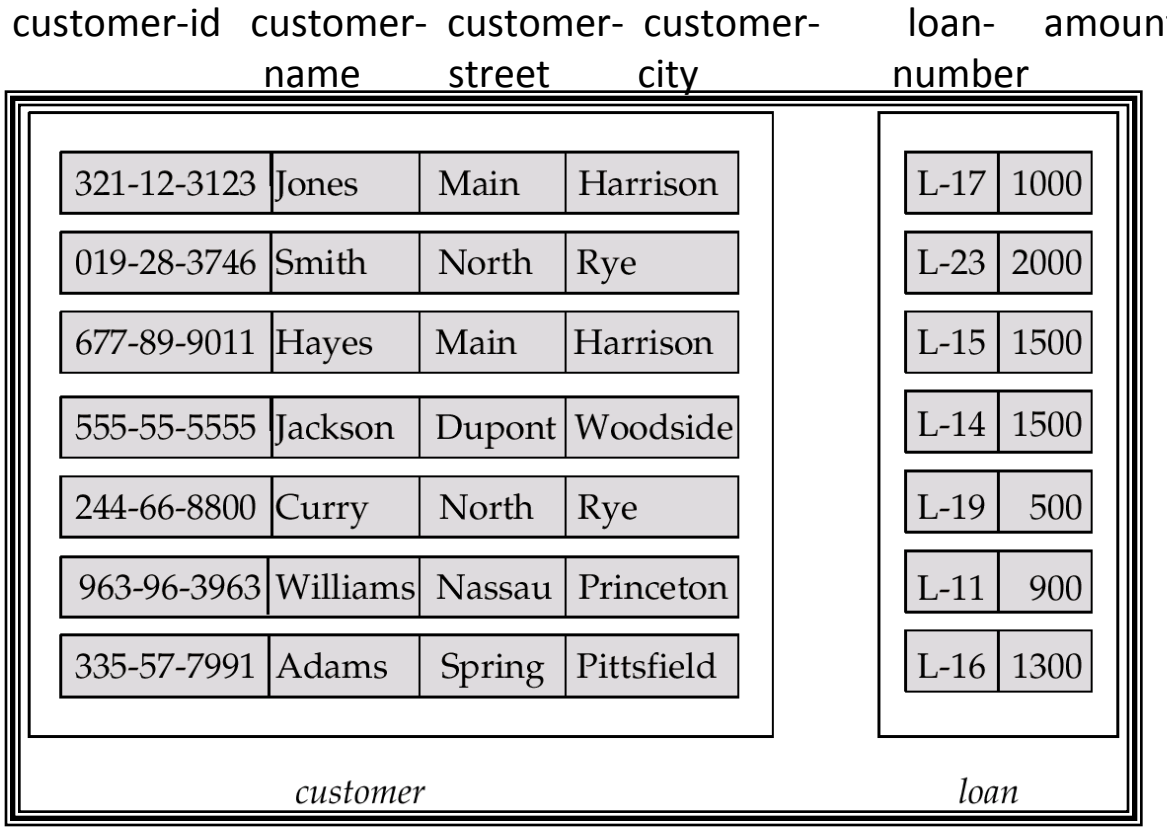
Agenda

- Entity & Entity Sets
- Attributes
- Relationship & Relationship Sets
- Constraints – Mapping Cardinalities, Participation Constraints, Keys
- E-R Diagrams & Design of Database schema
- Design Issues
- Weak Entity Sets
- Extended E-R Features – Specialization, Generalization, Aggregation
- Reduction of an E-R schema to Tables

Entity & Entity Sets

- A ***database*** can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An **entity** is an object in the real world that is distinguishable from another objects.
- Every **attribute** is defined by its set of values called domain. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- 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



Attributes

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

Example:

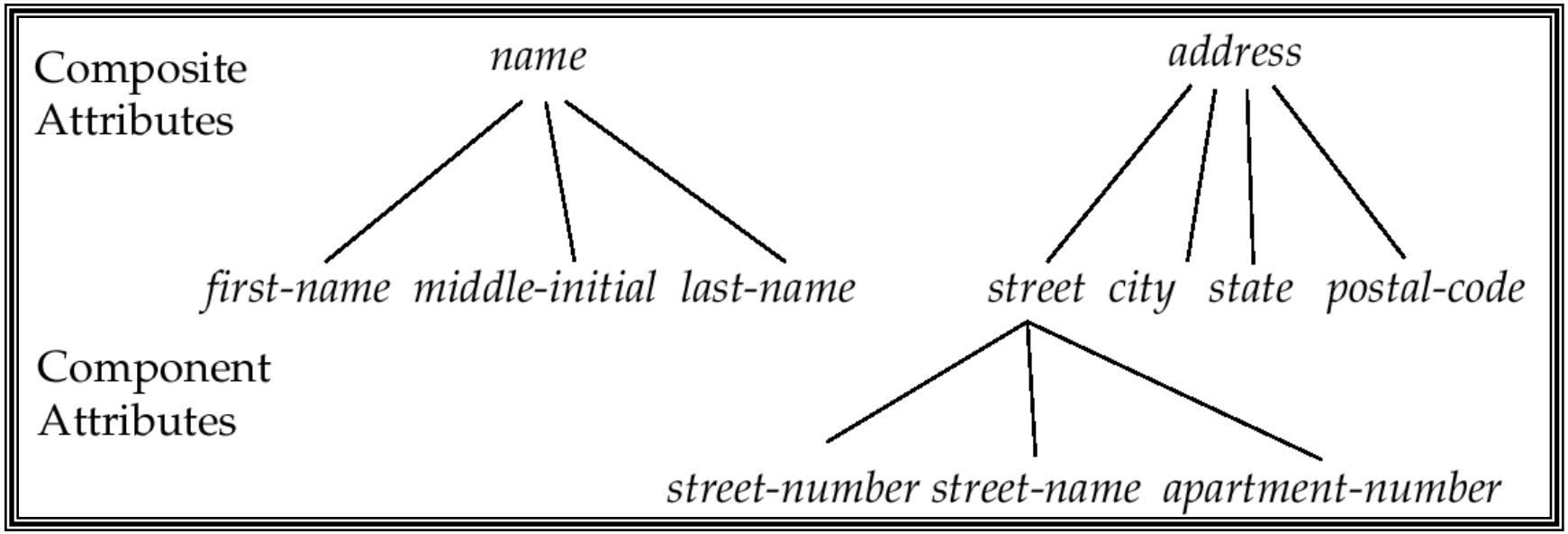
customer = (customer-id, customer-name,
customer-street, customer-city)
loan = (loan-number, amount)

Attributes

Types of Attributes:

- **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. Example - a student's phone number is an atomic value of 10 digits.
- **Composite attribute** − Composite attributes are made of more than one simple attribute. Example - a student's complete name may have first_name and last_name.
- **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. Example - average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.
- **Single-value attribute** − Single-value attributes contain single value. Example − Social_Security_Number.
- **Multi-value attribute** − Multi-value attributes may contain more than one values. Example − a person can have more than one phone number, email_address, etc.

Example


Composite Attributes
name
address
first-name
middle-initial
last-name
street
city
state
postal-code
Component Attributes
street-number
street-name
apartment-number

Relationship & Relationship Sets

- **Relationship:**

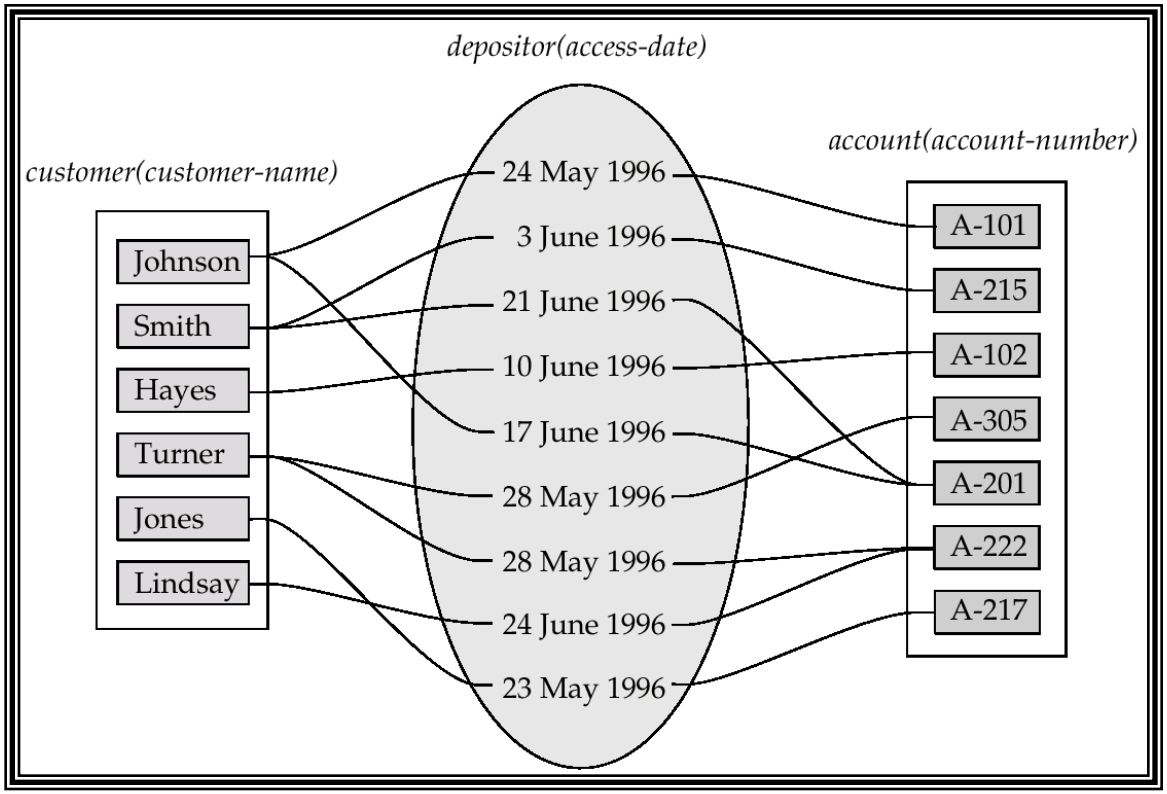
The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

- **Relationship Set:**

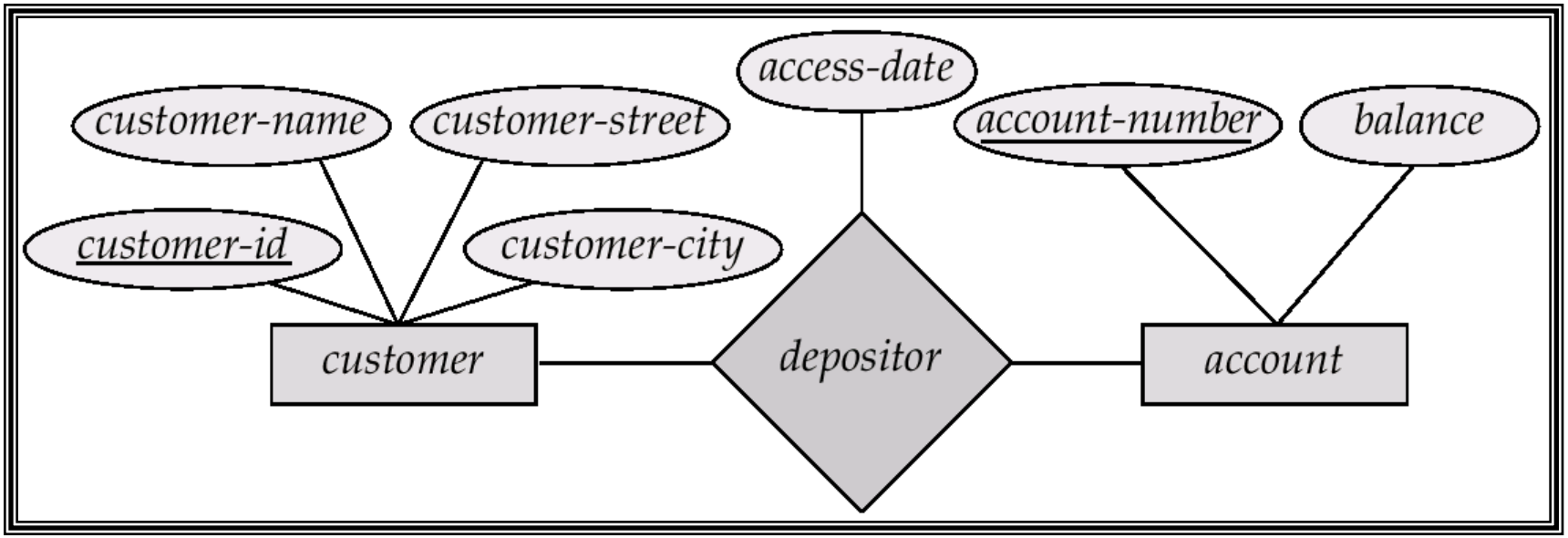
A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

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*

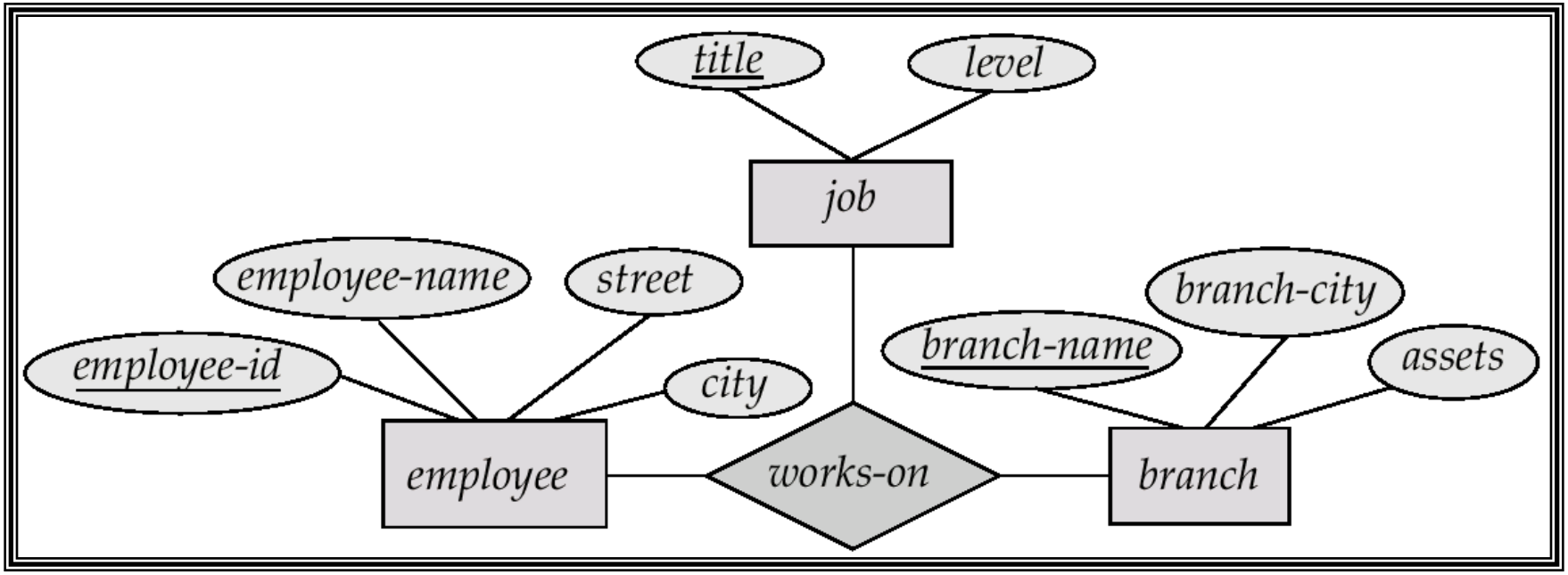


Relationship Sets with Attributes



Relationship Sets

- 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



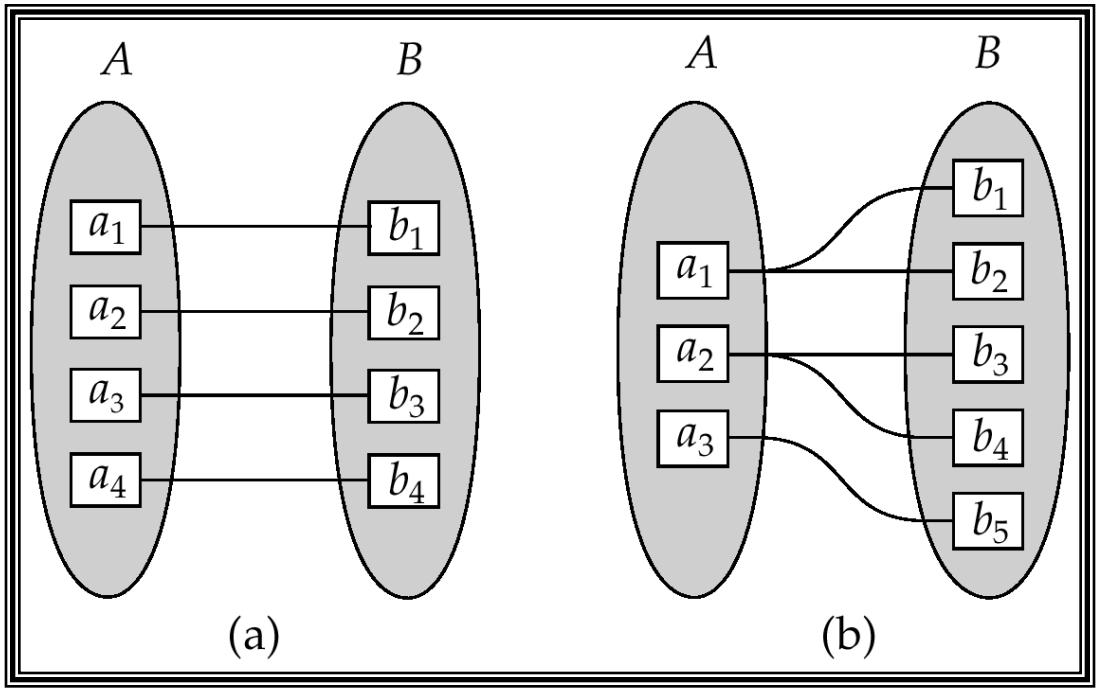
- Relationships between more than two entity sets are rare.

Constraints:

Mapping Cardinalities

- **Refers to number of entity sets that participate in a relationship set.**
- 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

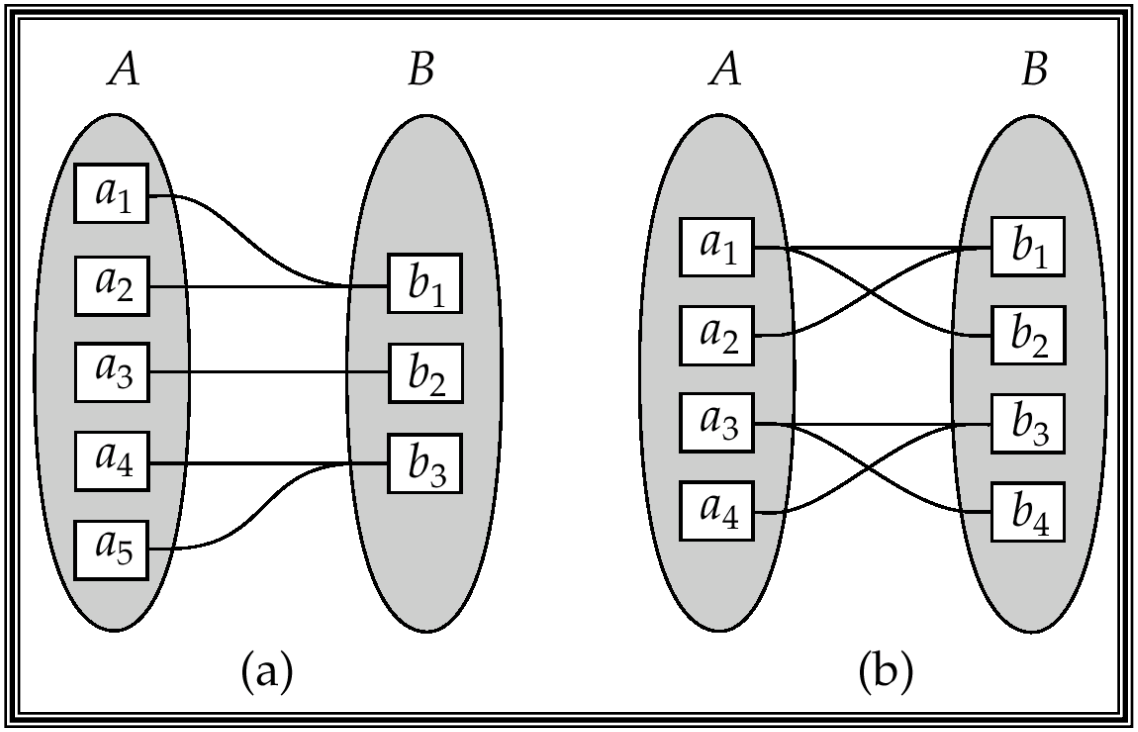


One to one

One to many

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

Mapping Cardinalities



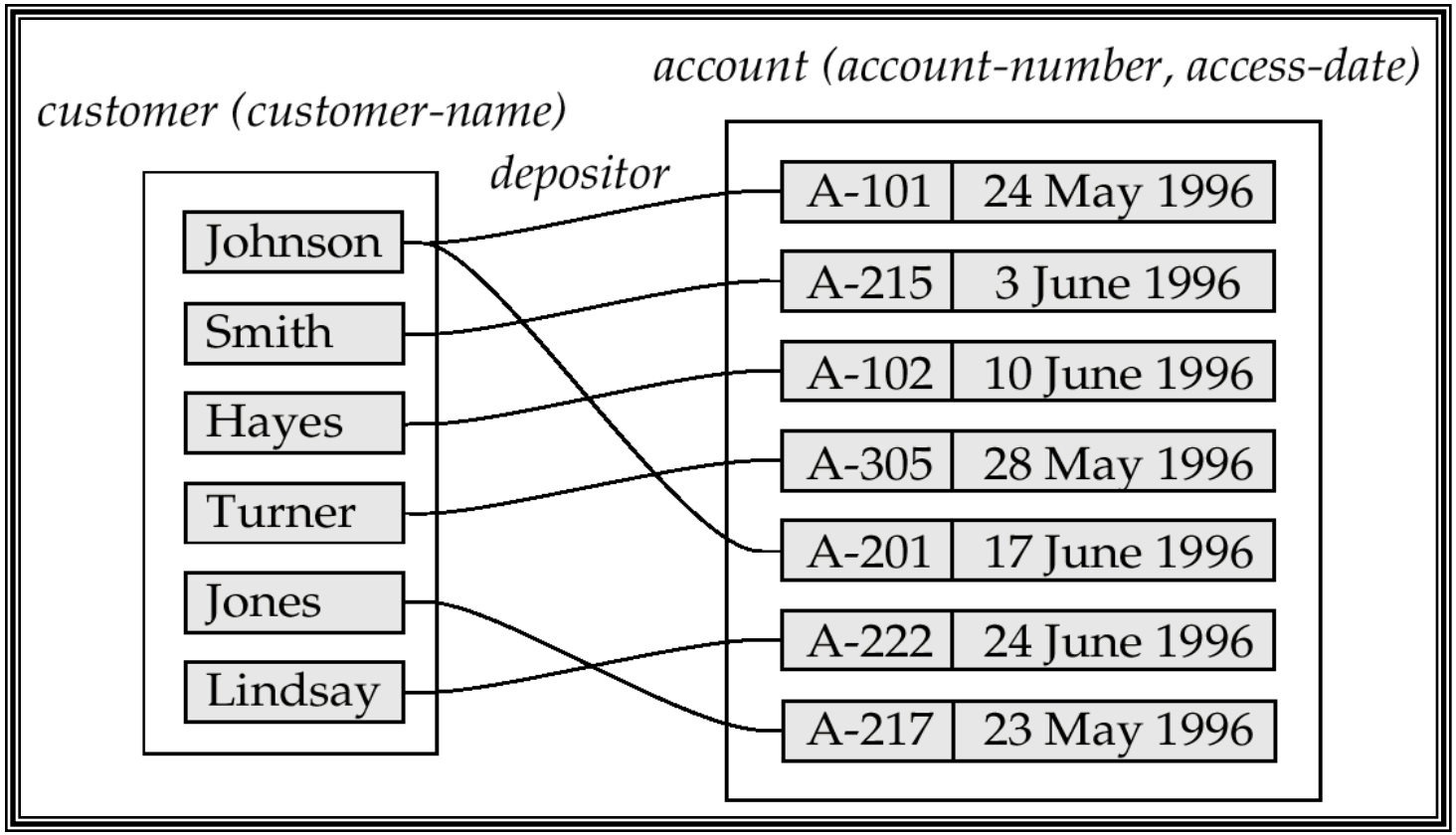
Many to one

Many to many

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

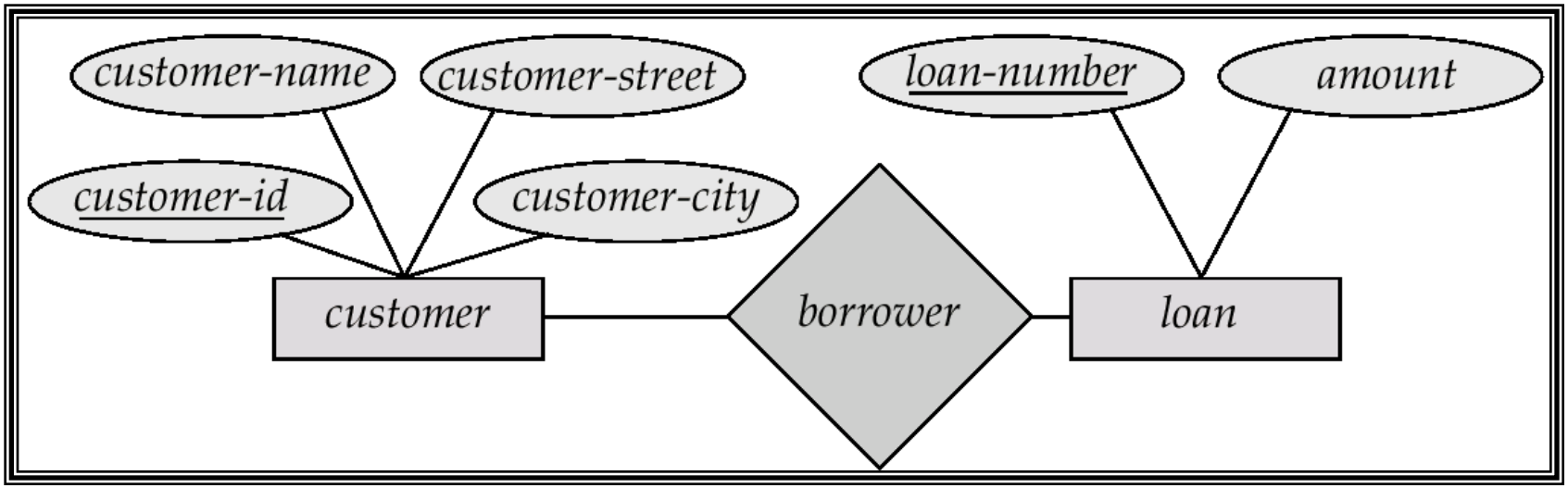
Mapping Cardinalities Example

- Relationship from account to customer is many to one,
- Likewise, customer to account is one to many

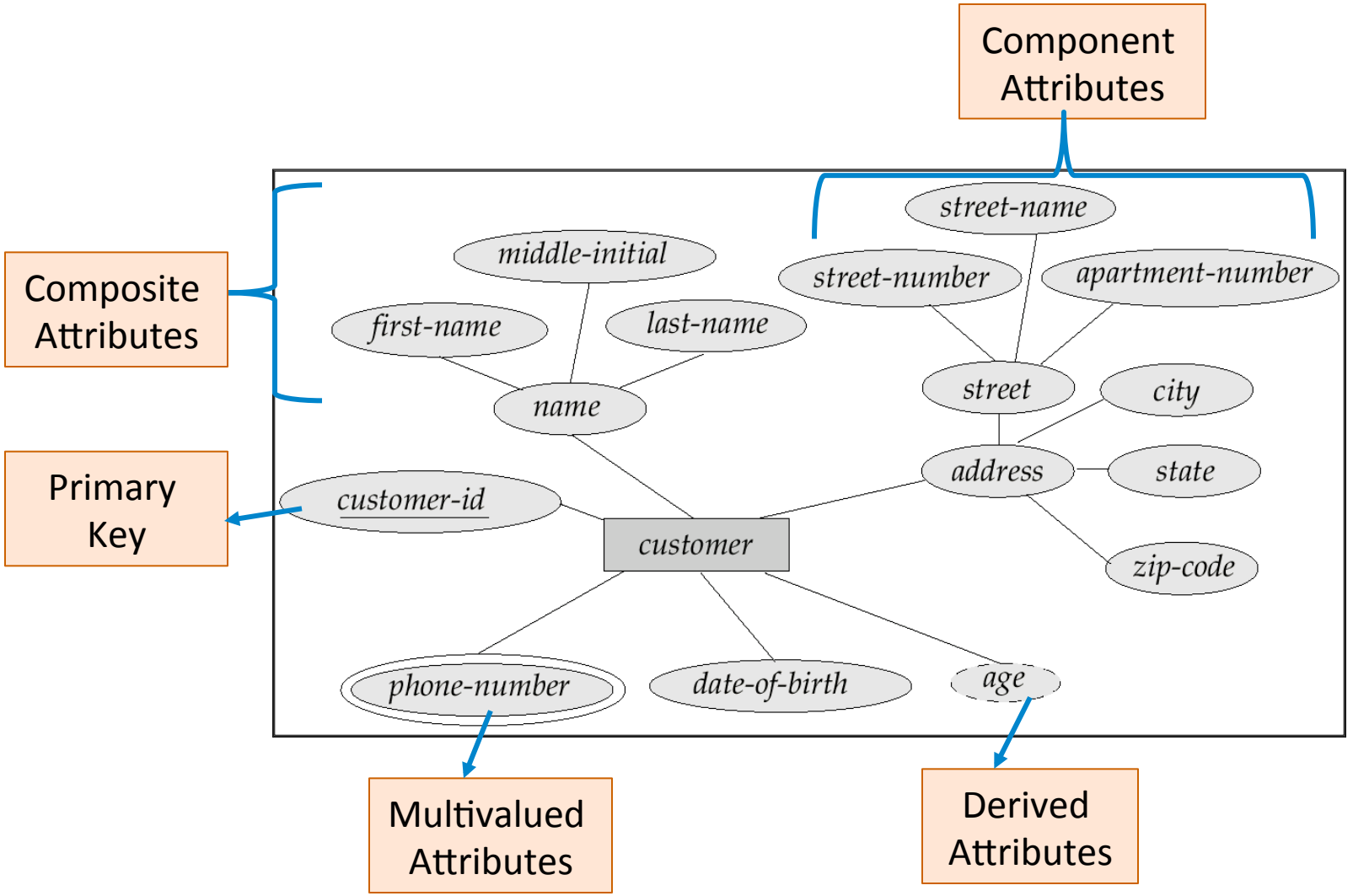


E-R Diagrams

- **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

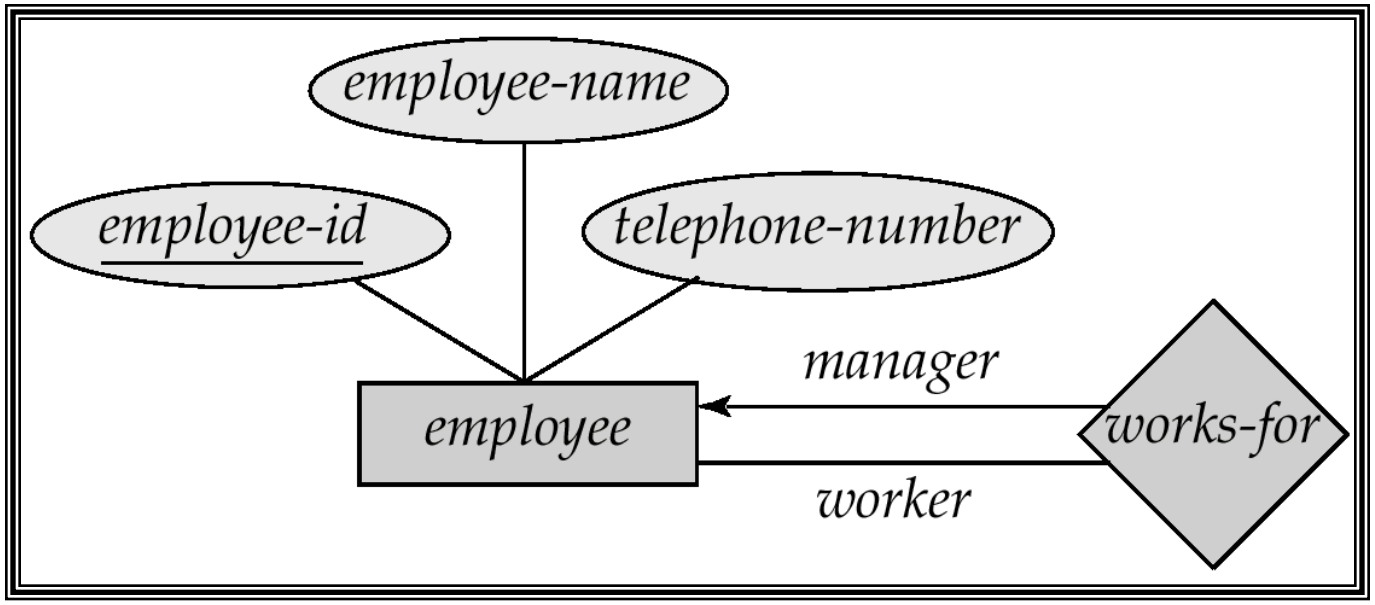


E-R Diagram With Composite, Multivalued, and Derived Attributes



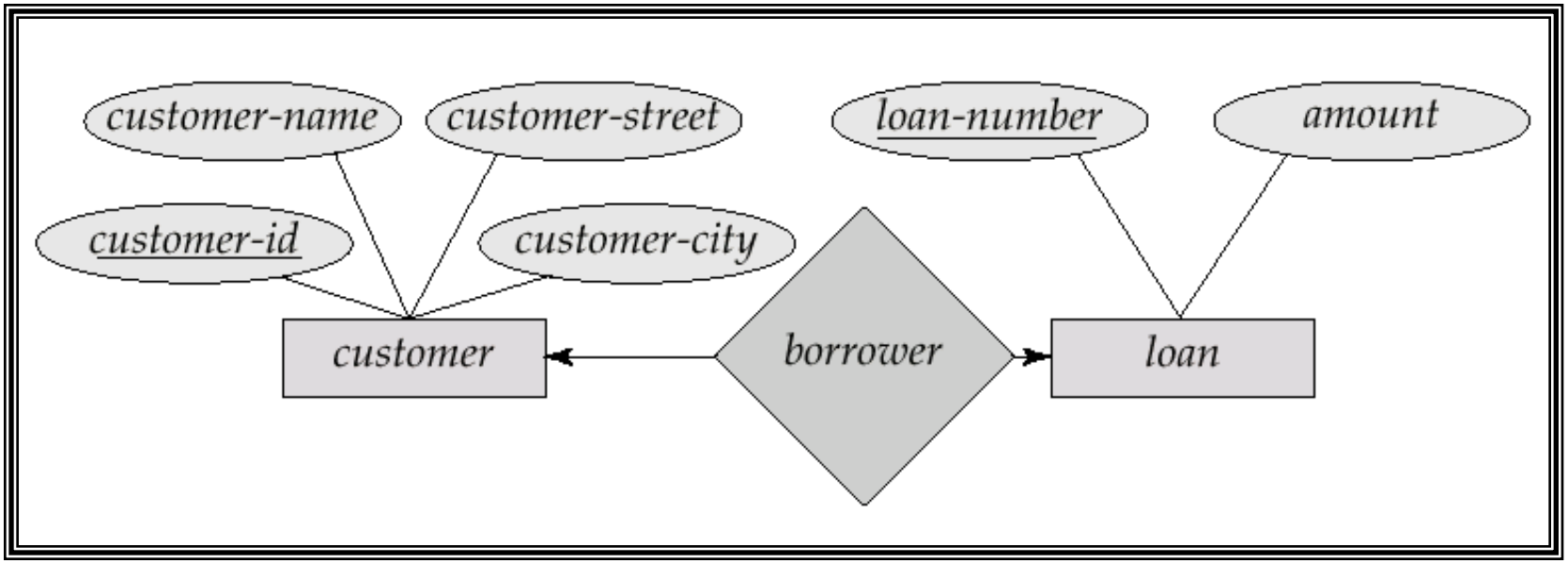
Roles

- 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



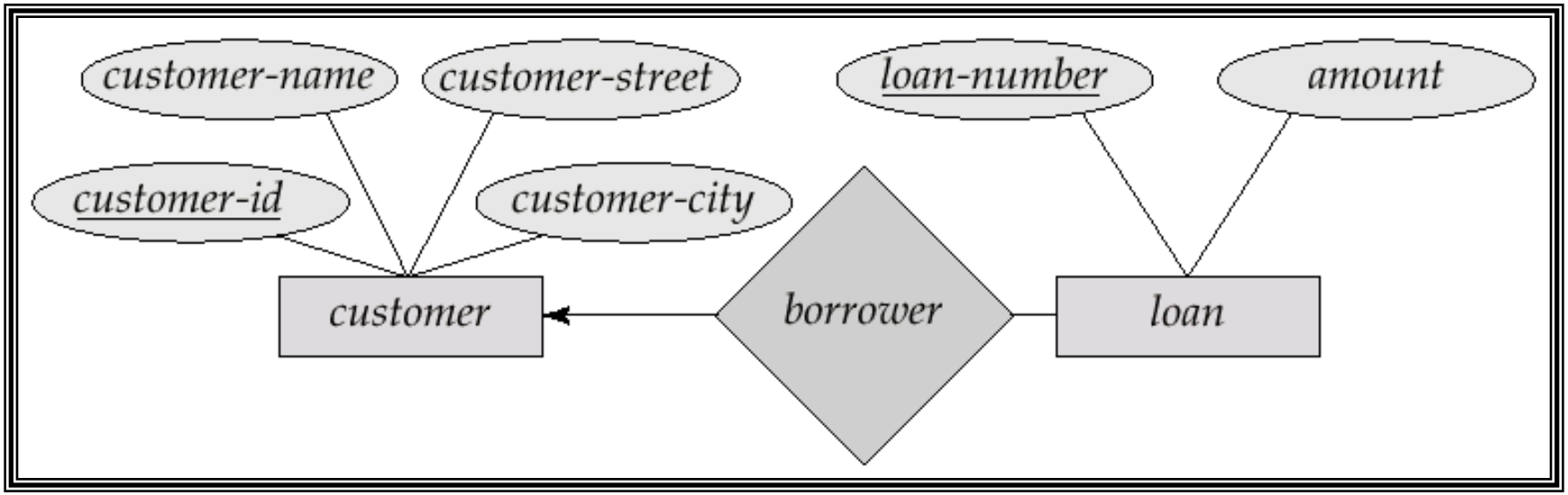
Cardinality Constraints

- 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*



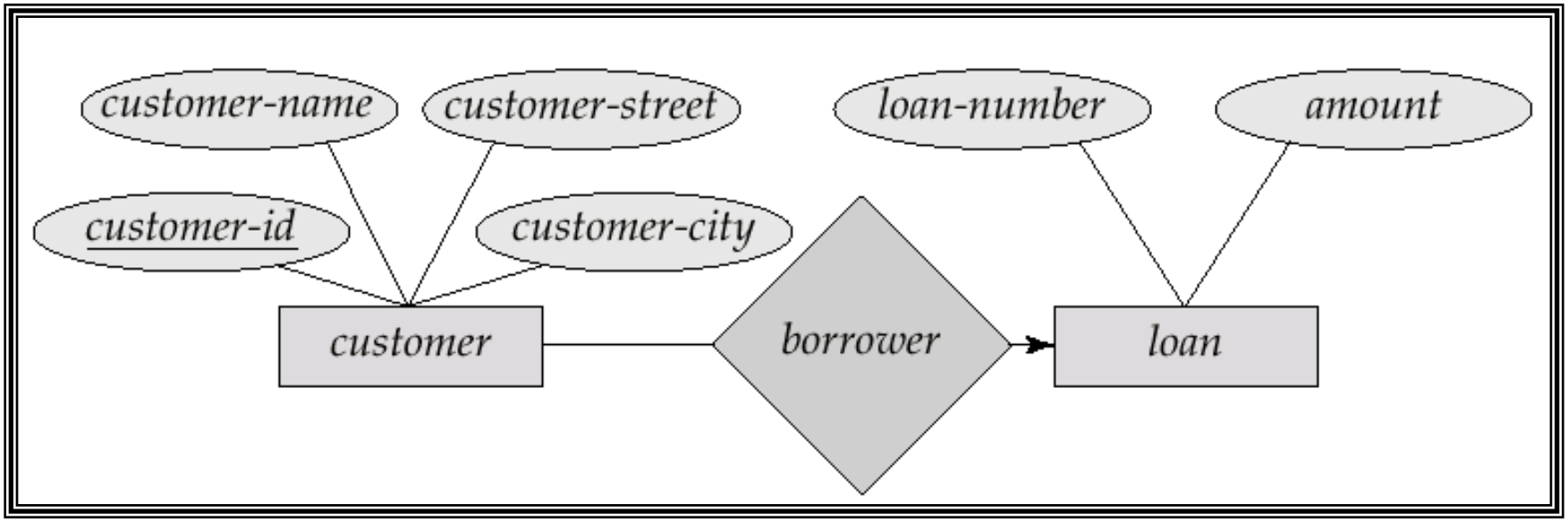
One-To-Many Relationship

- 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*



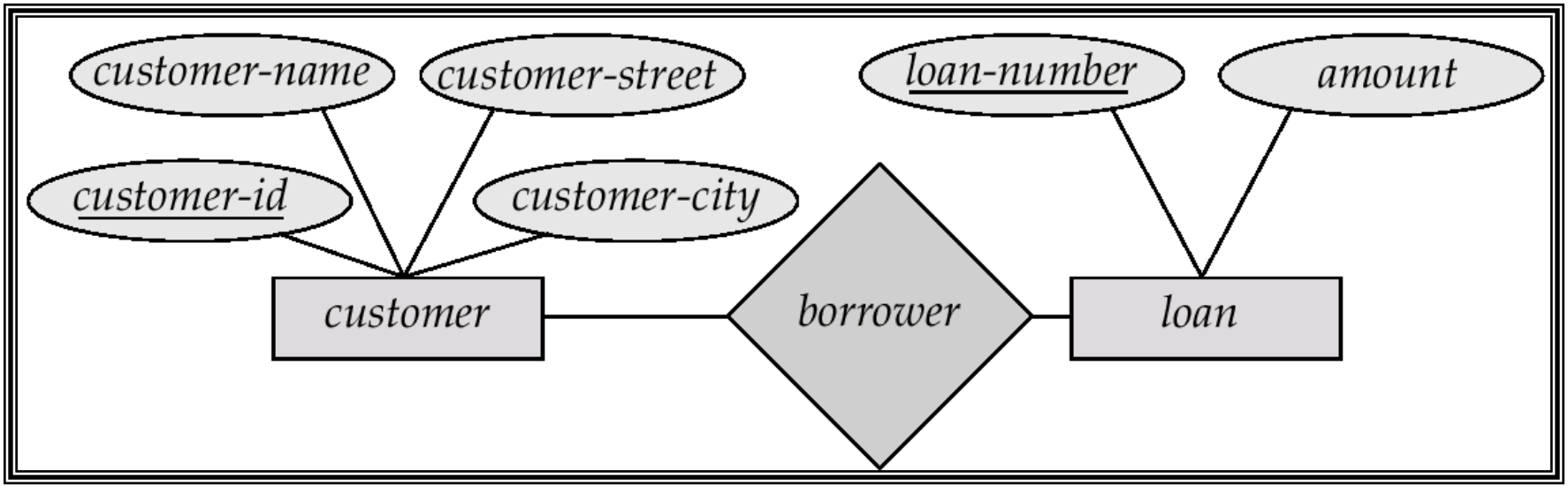
Many-To-One Relationships

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



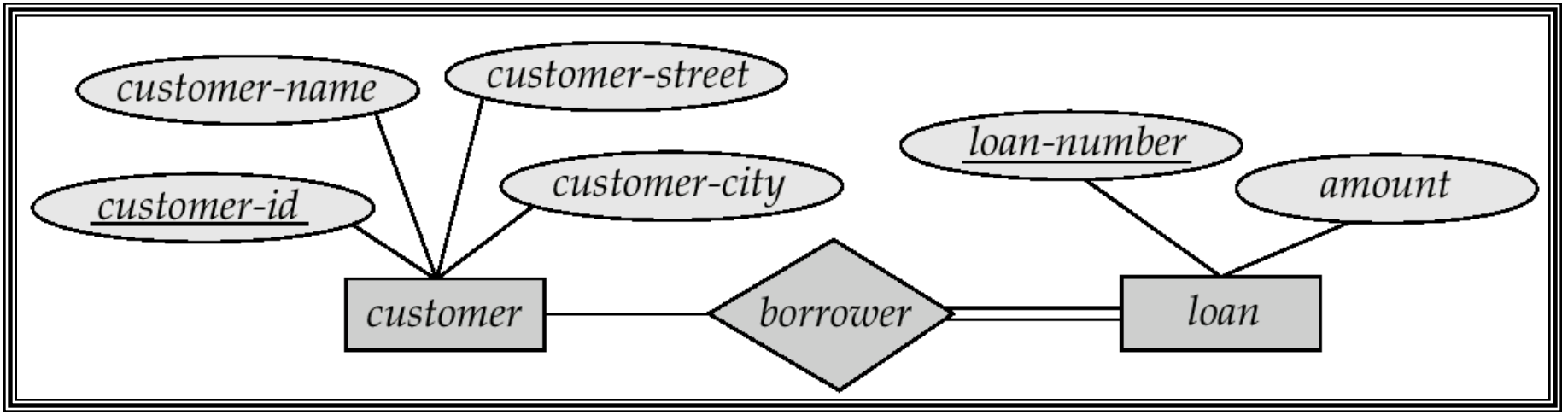
Many-To-Many Relationship

- A customer is associated with several loans via borrower
- A loan is associated with several 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
  - E.g. participation of *customer* in *borrower* is partial



Keys

- **Super Key** – A *super key* of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- **Candidate Key** – A minimal super key is called a candidate key. An entity set may have more than one candidate key. In order to be eligible for a candidate key it must pass certain criteria.
  - It must contain unique values
  - It must not contain null values
  - It contains the minimum number of fields to ensure uniqueness
  - It must uniquely identify each record in the table
- **Primary Key** – A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.
- **Foreign Key –** A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables. It acts as a cross-reference between tables because it references the primary key of another table, thereby establishing a link between them.

Example 1

For Example, We are having table

```
Book (BookId, BookName, Author)
```

So in this table we can have

```
(BookId)
(BookId,BookName)
(BookId, BookName, Author)
(BookId, Author)
(BookName, Author)
```

As our Super key. Each super key is able to uniquely identify each tuple (record).

Candidate Key

```
(BookId)
(BookName,Author)
```

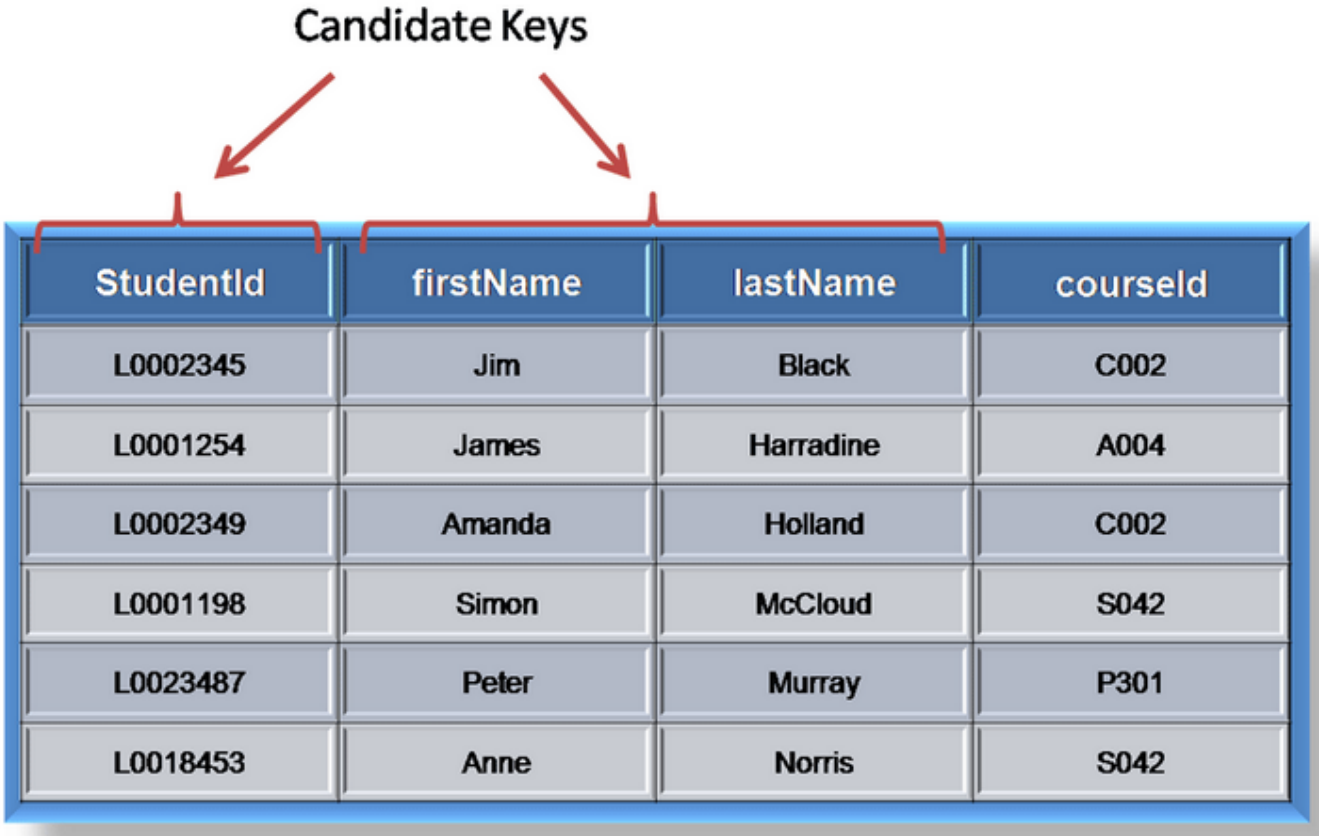
These two keys can be candidate keys, as remaining keys are having redundant attributes. Means in super key (BookId, BookName) record can be uniquely identify by just bookid and therefore Bookname is redundant attribute

Primary Key: It is a candidate key that is chosen by the database designer to identify entities with in an entity set. OR A key which is used to uniquely identify each record is known as primary key.

From above Candidate keys any one can be the primary key. And the another one which is not chosen as primary key will be know as Alternate key

Example 2

- Candidate Key:



StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

- As an example we might have a student_id that uniquely identifies the students in a student table. This would be a candidate key. But in the same table we have the student's first name and last name that also, when combined, uniquely identify the student in a student table. These would both be candidate keys.

Example

- Primary Key:

Primary Keys

StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

Example

- Foreign Key:

TABLE: student

studentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042

Foreign Keys

Relationship

TABLE: course

Primary Keys

courseId	courseName
A004	Accounts
C002	Computing
P301	History
S042	Short Course

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

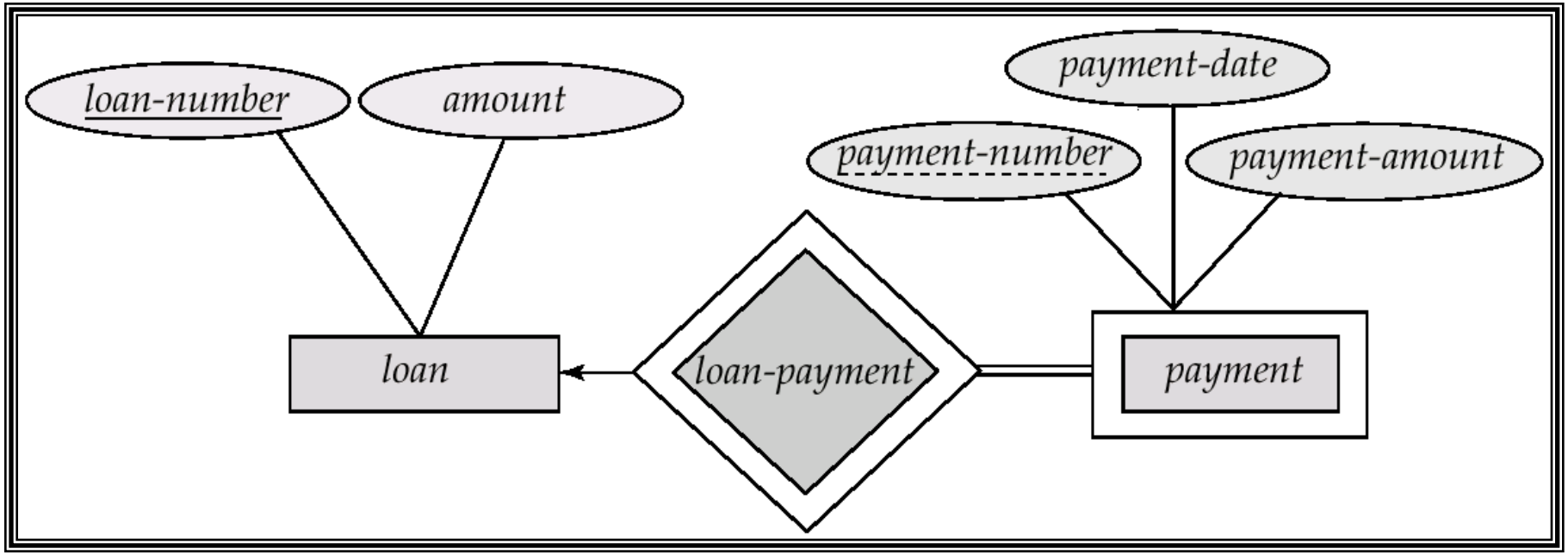
- An entity set that does not have a primary key is referred to as a weak entity set.
- The existence of a weak entity set depends on the existence of a identifying entity set
  - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity 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.

OR

- Primary Key of the weak entity set = Discriminator + Primary key of strong entity set

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*)



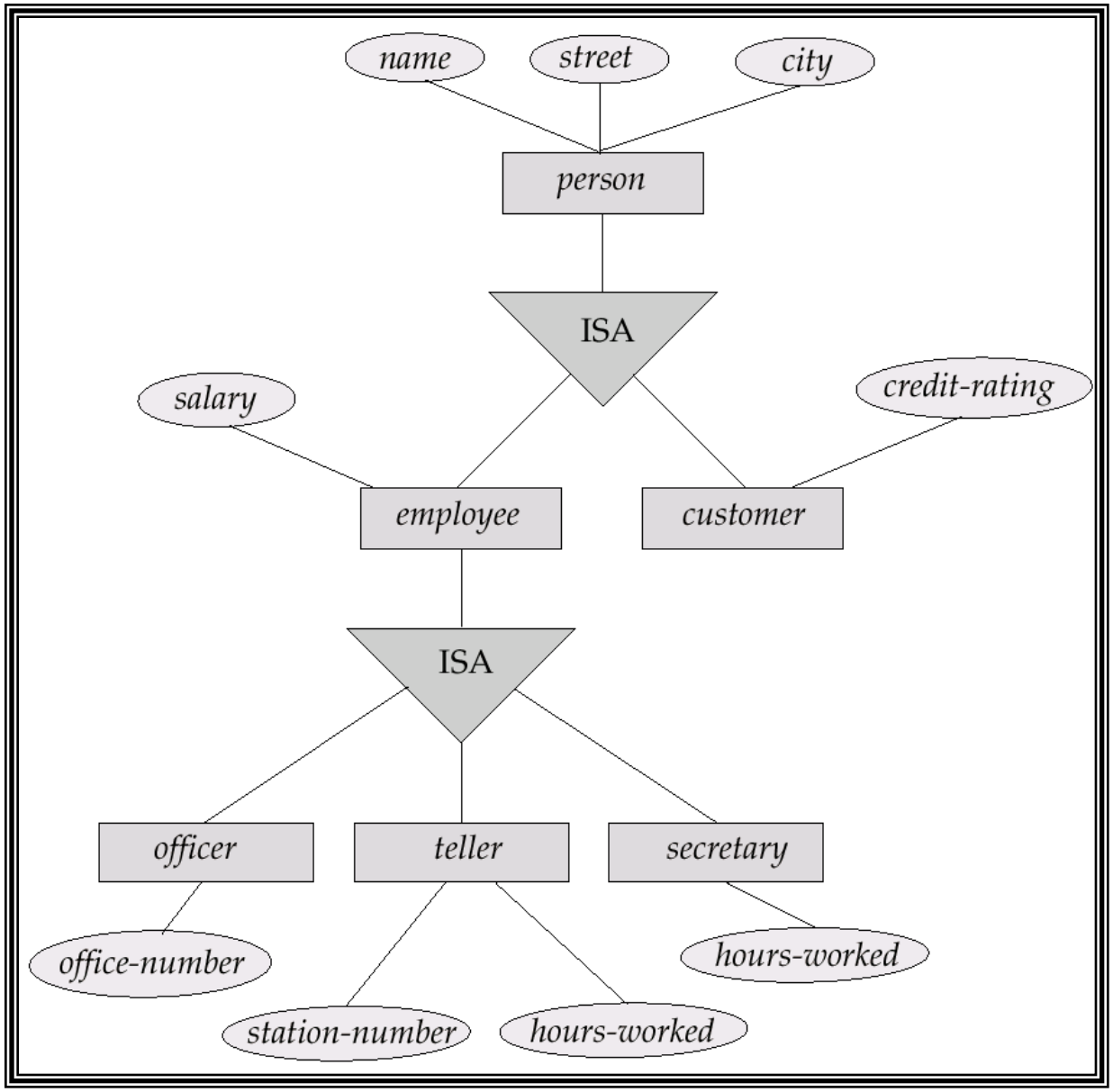
Weak Entity Sets (Cont.)

- 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*

Extended E-R Features:
Specialization

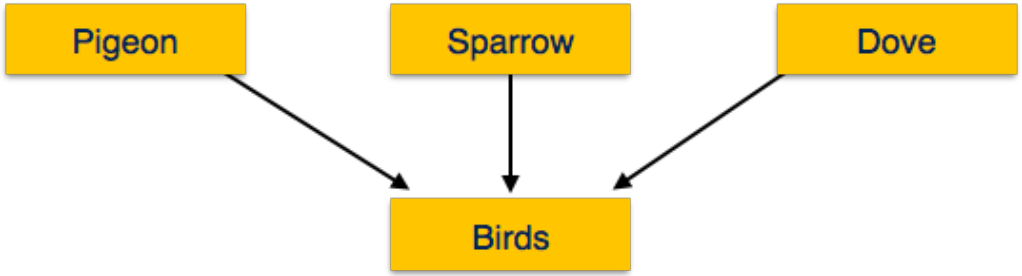
- It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, some higher level entities may not have lower-level entity sets at all.
- Depicted by a *triangle* component labeled ISA (E.g. *customer* "is a" *person*).
- **Attribute inheritance** – 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



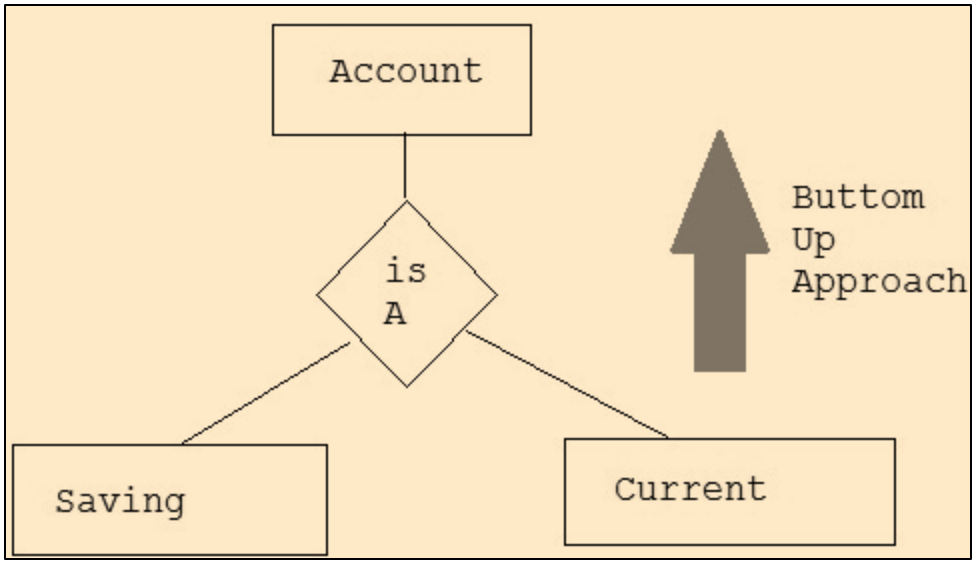
Generalization

- **Generalization** is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entity to make further higher level entity.
- For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.

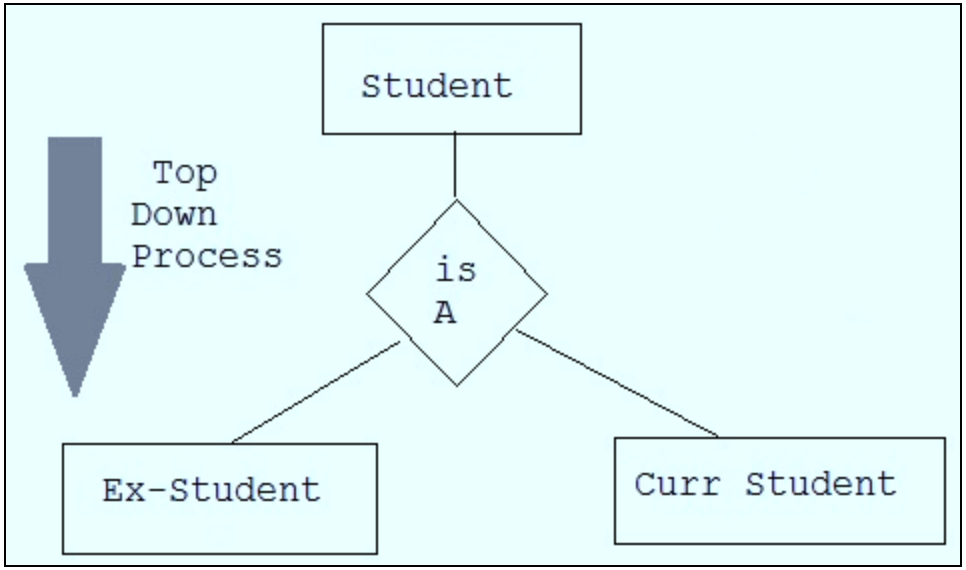


- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.

- GENERALIZATION:

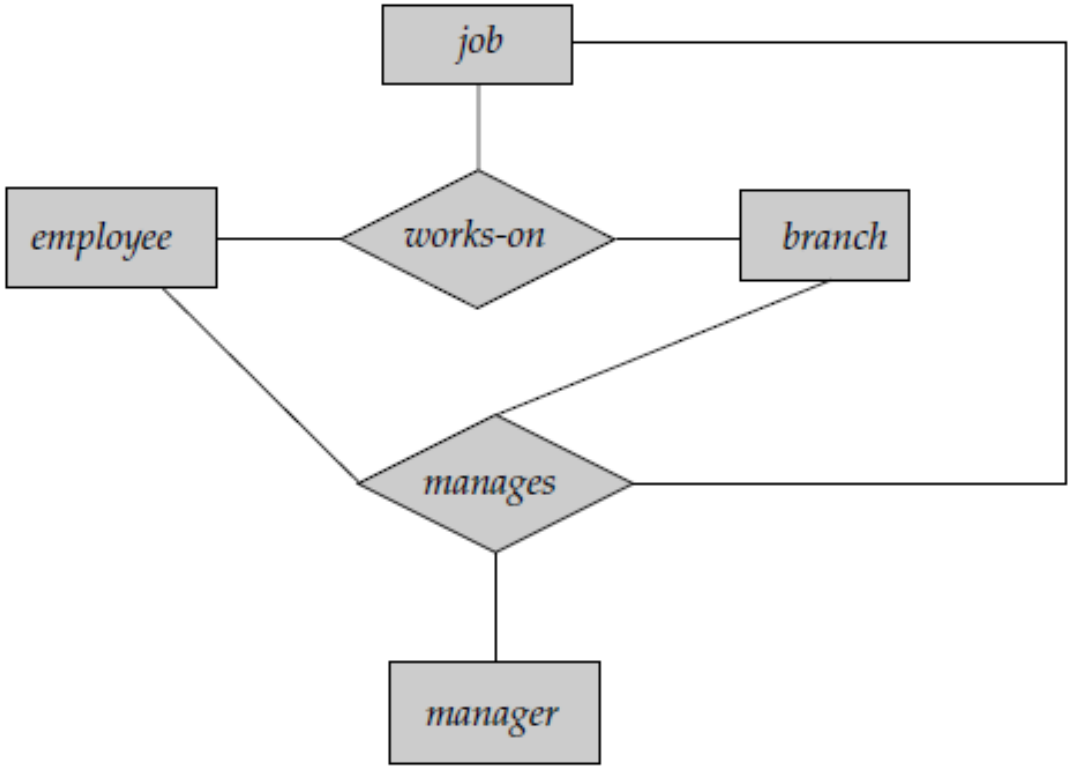


- SPECIALIZATION:



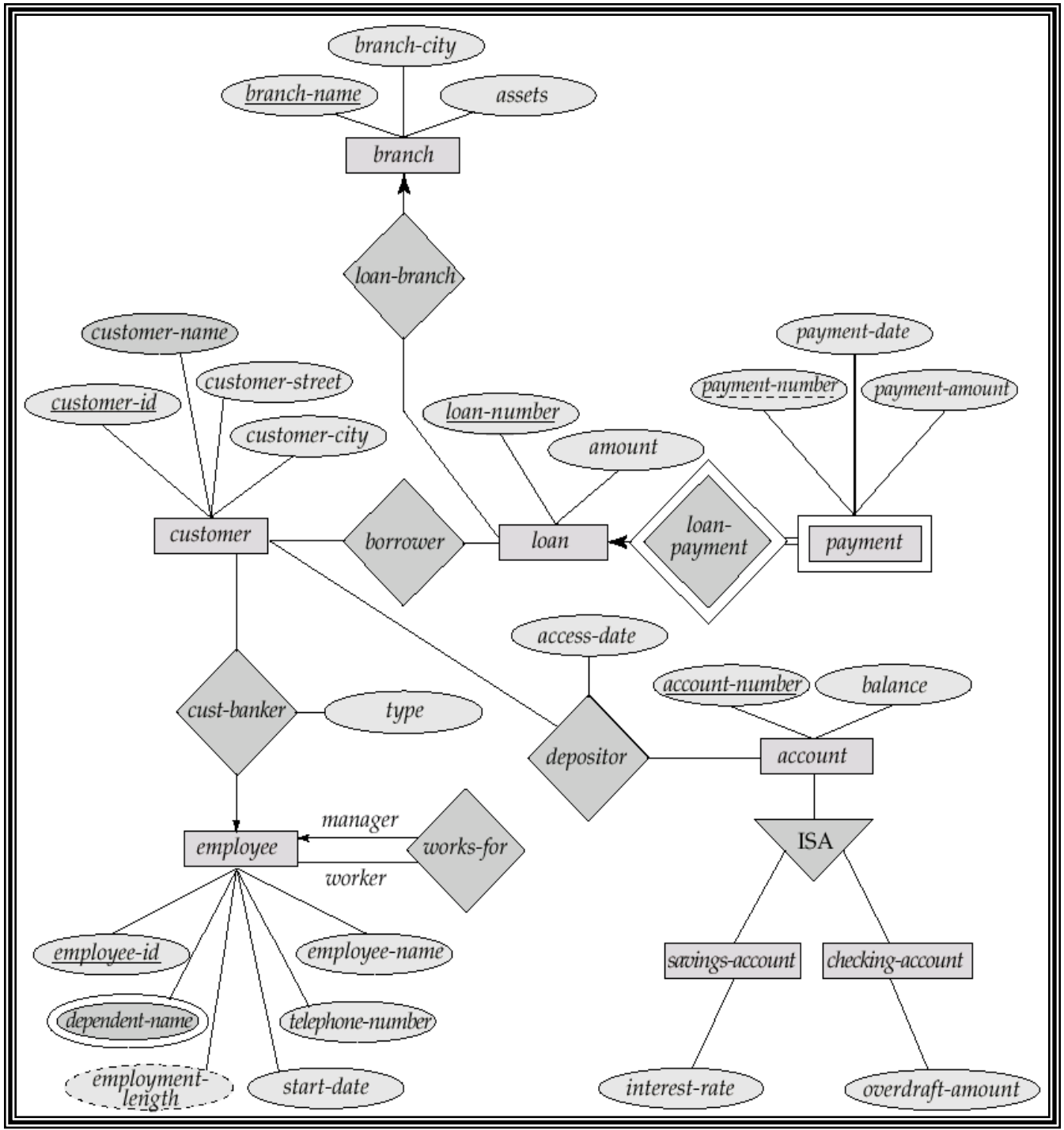
Aggregation

- Aggregation is an abstraction in which relationship sets are treated as higher level entity sets.
- Here, a relationship set is embedded inside an entity set, and these entity sets can participate in relationships.



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.


branch-city
branch-name
assets
branch
loan-branch
customer-name
customer-street
customer-id
customer-city
loan-number
amount
payment-date
payment-number
payment-amount
customer
borrower
loan
loan-payment
payment
access-date
account-number
balance
cust-banker
type
depositor
account
manager
employee
works-for
worker
ISA
employee-id
employee-name
dependent-name
telephone-number
employment-length
start-date
savings-account
checking-account
interest-rate
overdraft-amount

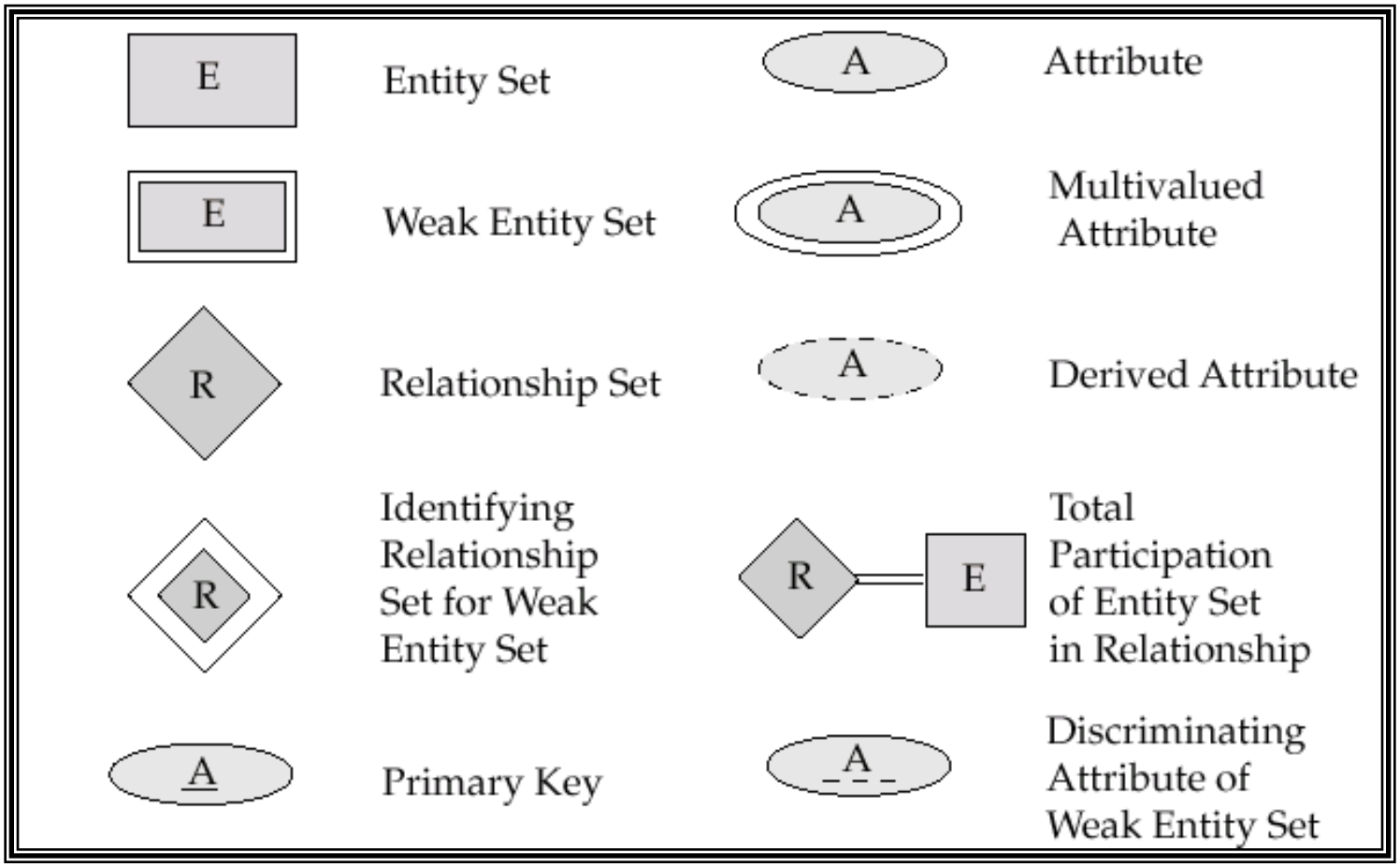
Example – ER Diagram

- The *branch* entity set, with attributes *branch-name*, *branch-city*, and *assets*.
- The *customer* entity set, with attributes *customer-id*, *customer-name*, *customerstreet*, and *customer-city*.
- The *employee* entity set, with attributes *employee-id*, *employee-name*, *telephonenumber*, *salary*, and *manager*. Additional descriptive features are the multivalued attribute *dependent-name*, the base attribute *start-date*, and the derived attribute *employment-length*.
- Two account entity sets—*savings-account* and *checking-account*—with the common attributes of *account-number* and *balance*; in addition, *savings-account* has the attribute *interest-rate* and *checking-account* has the attribute *overdraft-amount*.
- The *loan* entity set, with the attributes *loan-number*, *amount*, and *originatingbranch*.
- The weak entity set *loan-payment*, with attributes *payment-number*, *paymentdate*, and *payment-amount*.

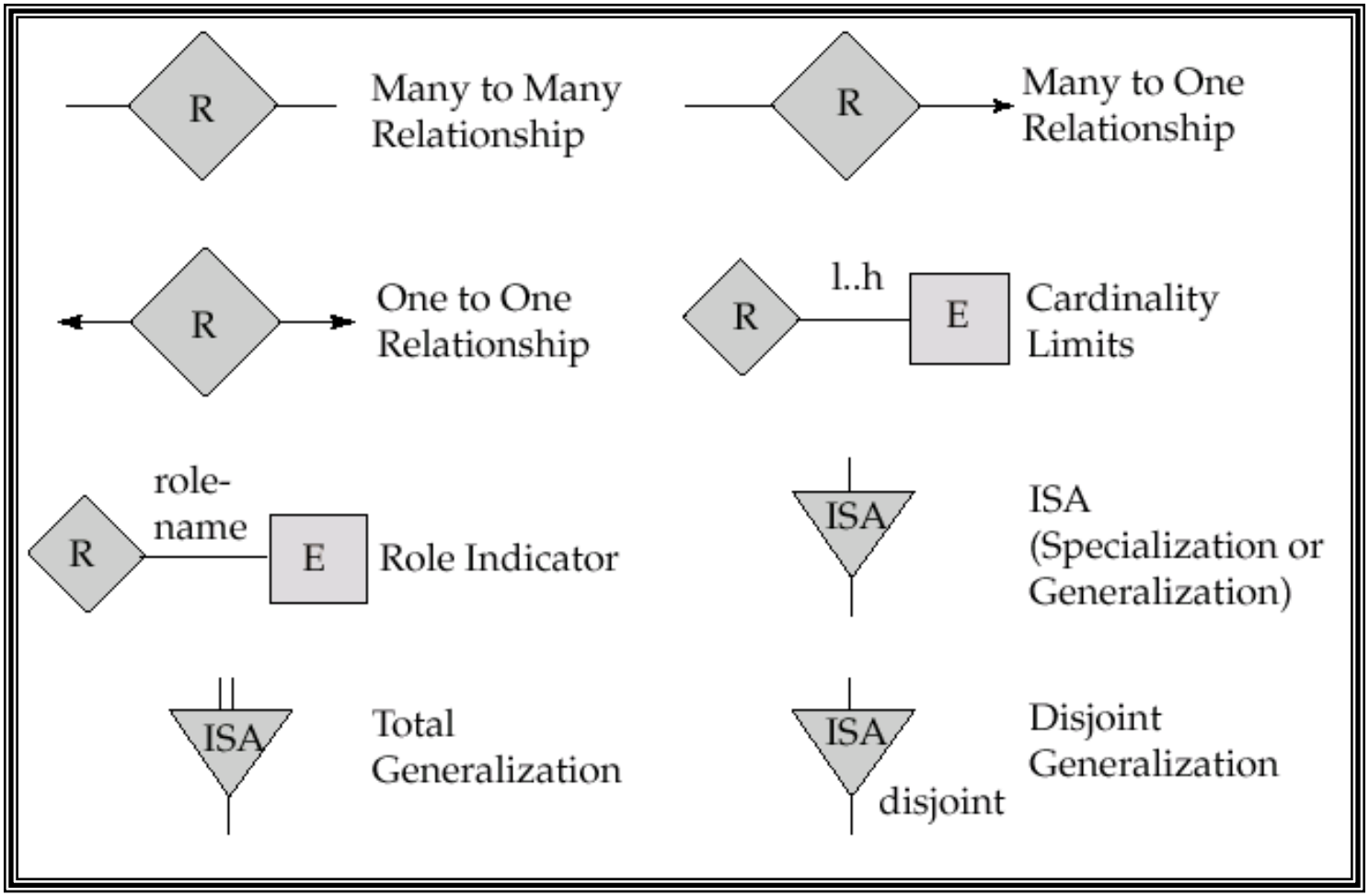
Example – ER Diagram contd..

- *borrower*, a many-to-many relationship set between *customer* and *loan*.
- *loan-branch*, a many-to-one relationship set that indicates in which branch a loan originated. *loan-payment*, a one-to-many relationship from *loan* to *payment*, which documents that a payment is made on a loan.
- *depositor*, with relationship attribute *access-date*, a many-to-many relationship set between *customer* and *account*, indicating that a customer owns an account.
- *cust-banker*, with relationship attribute *type*, a many-to-one relationship set expressing that a customer can be advised by a bank employee, and that a bank employee can advise one or more customers. *works-for*, a relationship set between *employee* entities with role indicators *manager* and *worker*; the mapping cardinalities express that an employee works for only one manager and that a manager supervises one or more employees.

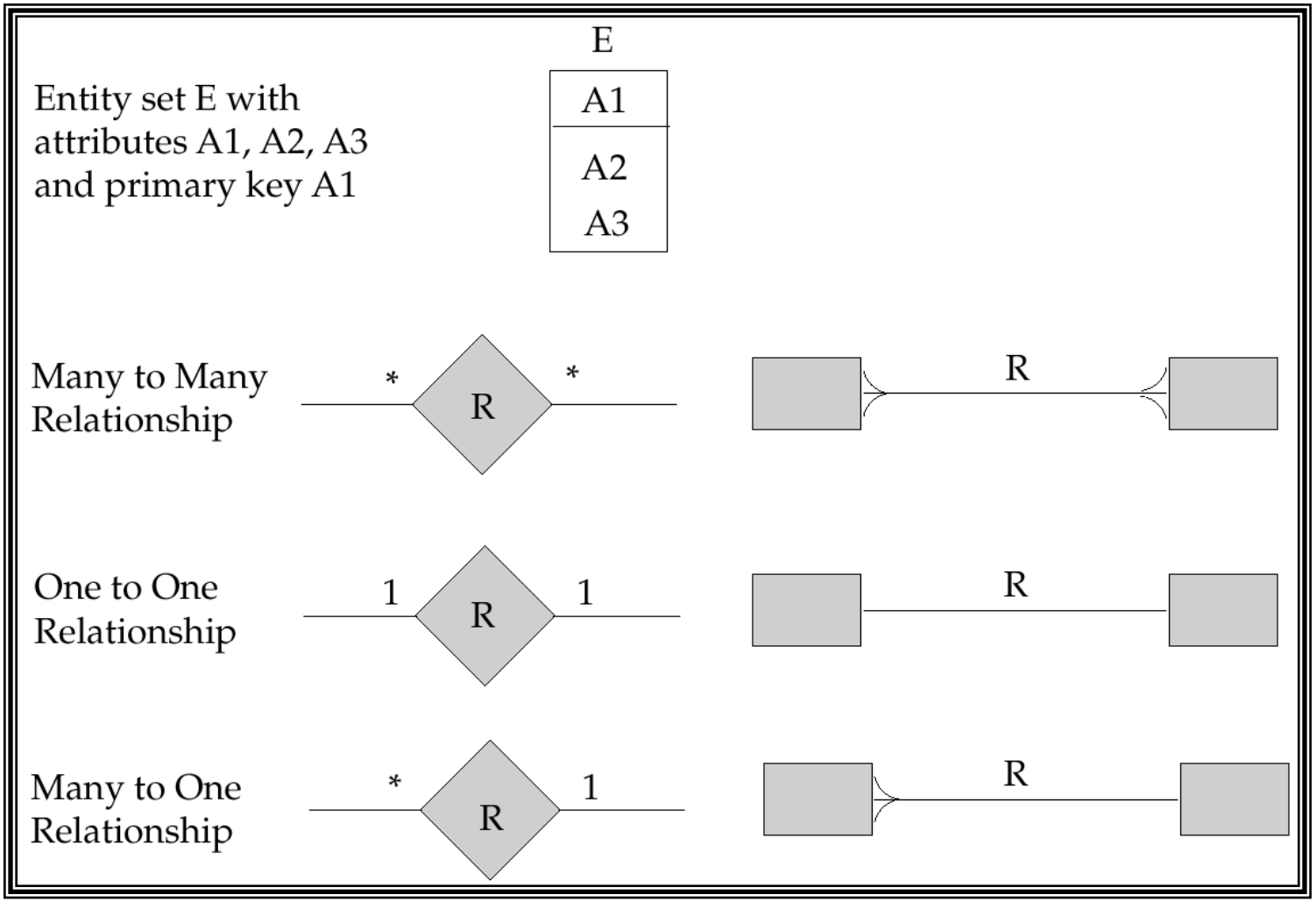
Summary of Symbols Used in E-R Notation



Summary of Symbols (Cont.)



Alternative E-R Notations

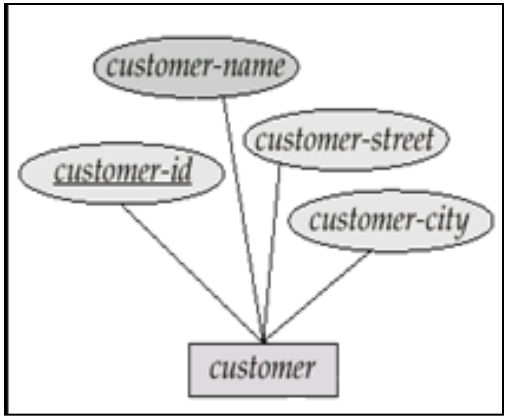


Reduction of an E-R Schema to Tables

- Primary keys allow entity sets and relationship sets to be expressed uniformly as *tables* which represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has a number of columns (generally corresponding to attributes), which have unique names.
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.

Representing Entity Sets as Tables

- A strong entity set (a set having primary key) reduces to a table with the same attributes.

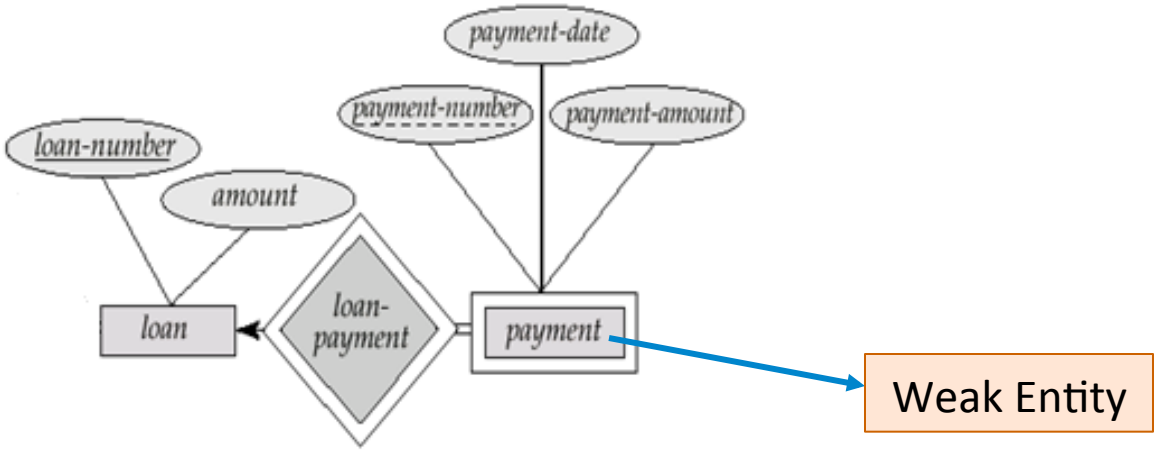


customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011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

Representation of Weak Entity Set

- Weak Entity Set Cannot exists alone
- To build a table/schema for weak entity set
  - Construct a table with one column for each attribute in the weak entity set
  - Remember to include discriminator
  - Augment one extra column on the right side of the table, put in there the primary key of the Strong Entity Set (the entity set that the weak entity set is depending on)
  - Primary Key of the weak entity set = Discriminator + foreign key

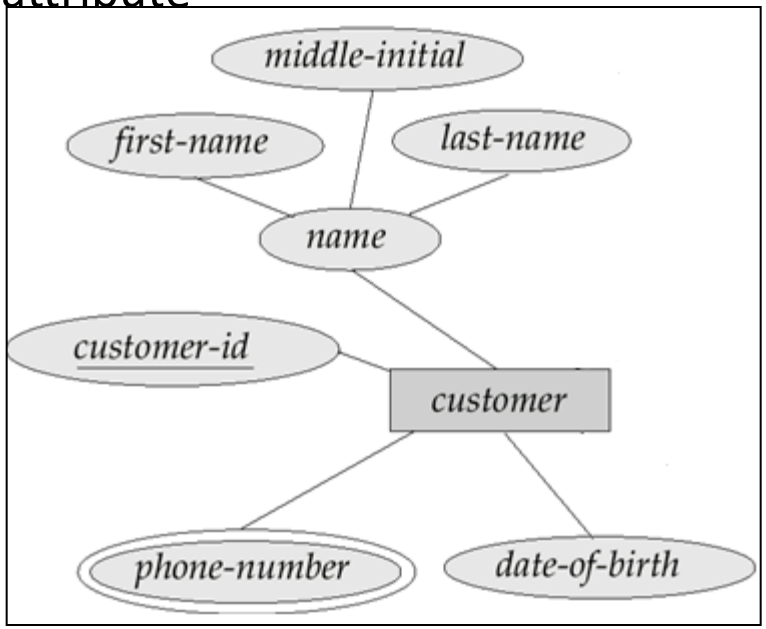
Representing Weak Entity Sets



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 Composite Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute



customer-id	name.first-name	name.middle-initial	name.last-name

Representing Multivalued Attributes

- A multivalued attribute M of an entity E is represented by a separate table EM
  - Table EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M

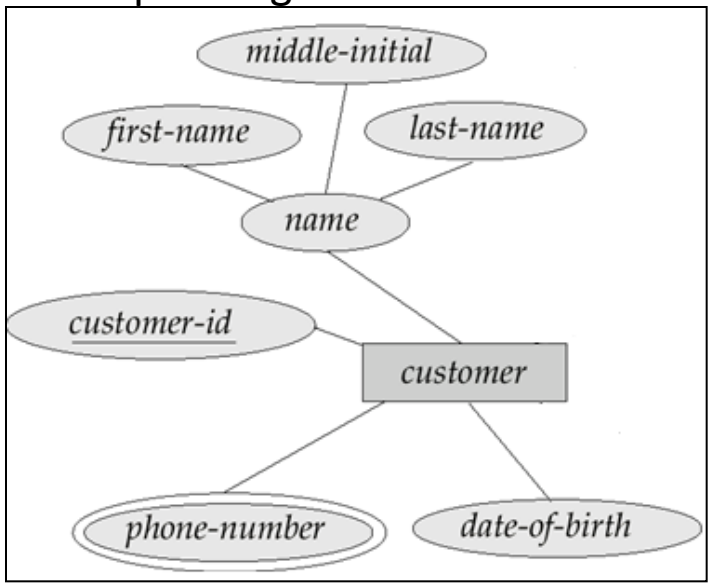


TABLE: Customer-phone-number

customer-id	*phone-number*

Representing Relationship Set

- **One-to-one relationship without Total Participation:**
  - Build a table with two columns, one column for each participating entity set's primary key. Add successive columns, one for each descriptive attributes of the relationship set (if any).

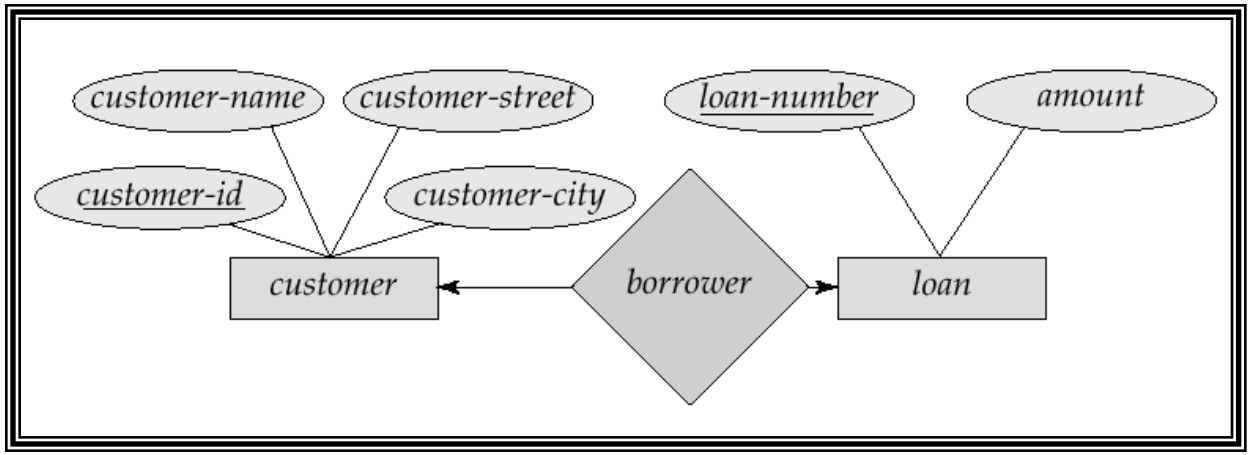


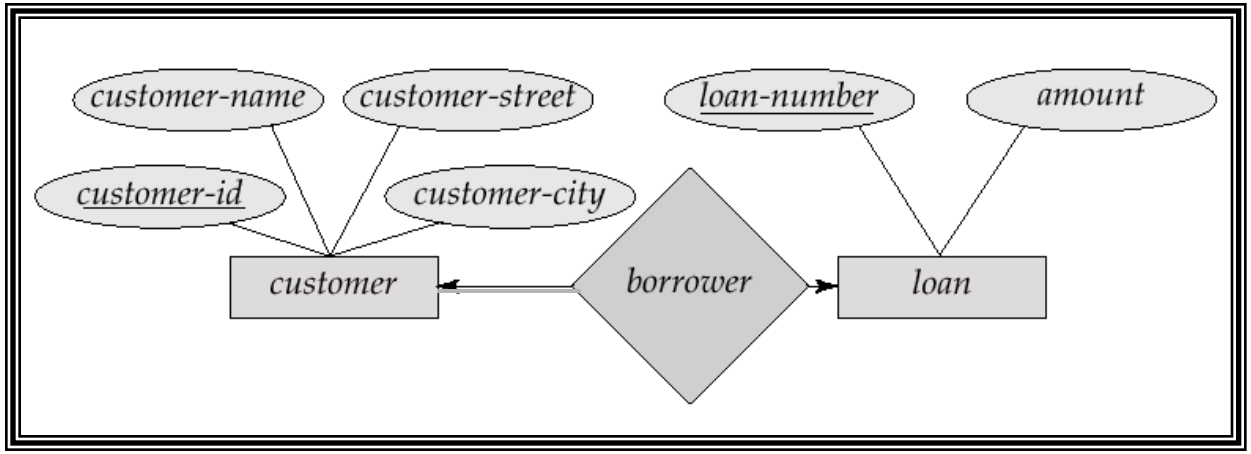
TABLE: borrower

customer-id	loan-number

  - Primary key can be either customer-id or loan-number

Representing Relationship Set

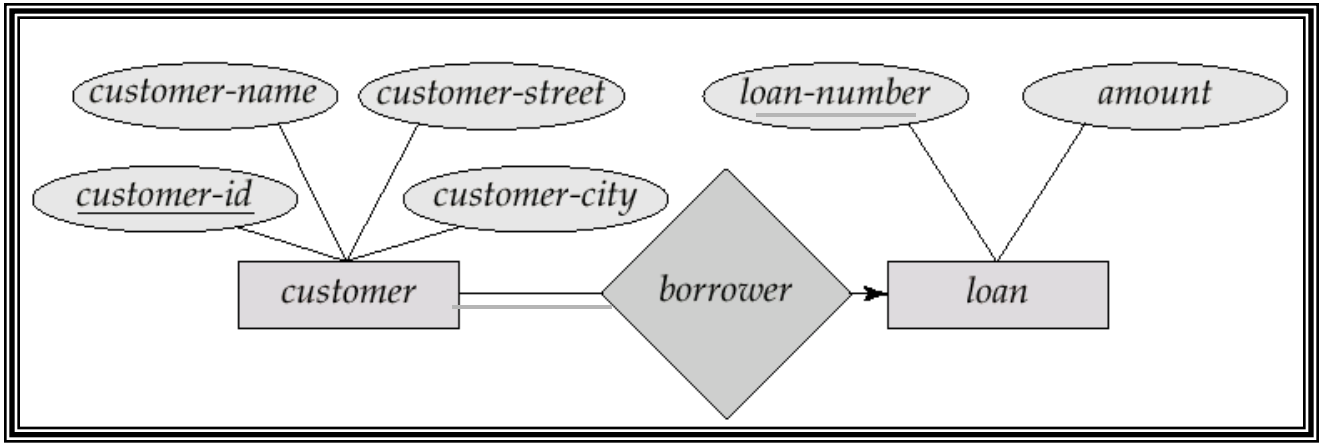
- **One-to-one relationship with one entity set having Total Participation:**
  - Augment one extra column on the right side of the table of the entity set with total participation, put in there the primary key of the entity set without complete participation as per to the relationship.



customer-id	*customer-name*	*customer-street*	*customer-city*	loan-number

Representing Relationship Set

- **For One-to-Many relationship without Total Participation:**
  - Same thing as one-to-one
  - the primary key of the "many" entity set becomes the relation's primary key.
- **For One-to-Many/Many-to-One relationship with one entity set having Total Participation on "many" side:**
  - Augment one extra column on the right side of the table of the entity set on the "many" side, & on the "one" side put in there the primary key of the entity set as per to the relationship.



customer-id	*customer-name*	*customer-street*	*customer-city*	loan-number

Representing Relationship Set

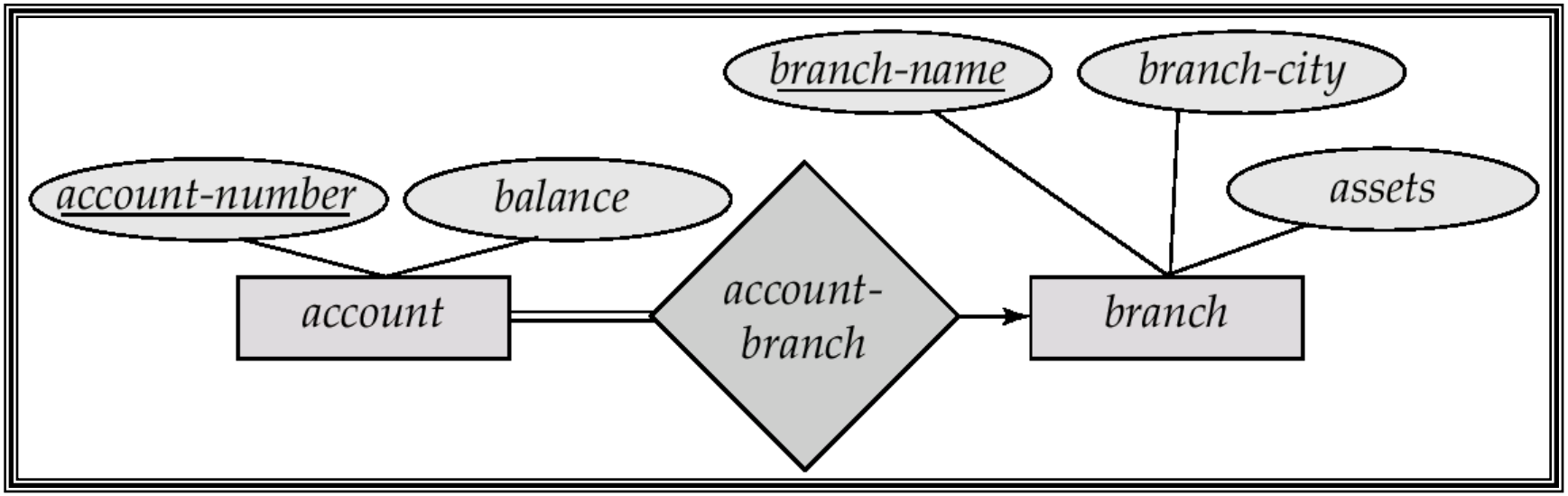
- **For Many-to-Many relationship:**
  - Same thing as one-to-one relationship without total participation.
  - Primary key of this new schema is the union of the foreign keys of both entity sets.

Summary:
Representing Relationship Set

- **Relationship set**. The union of the primary keys of the related entity sets becomes a super key of the relation.
  - For binary many-to-one relationship sets, the primary key of the "many" entity set becomes the relation's primary key.
  - For one-to-one relationship sets, the relation's primary key can be that of either entity set.
  - For many-to-many relationship sets, the union of the primary keys becomes the relation's primary key

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

But, 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*).

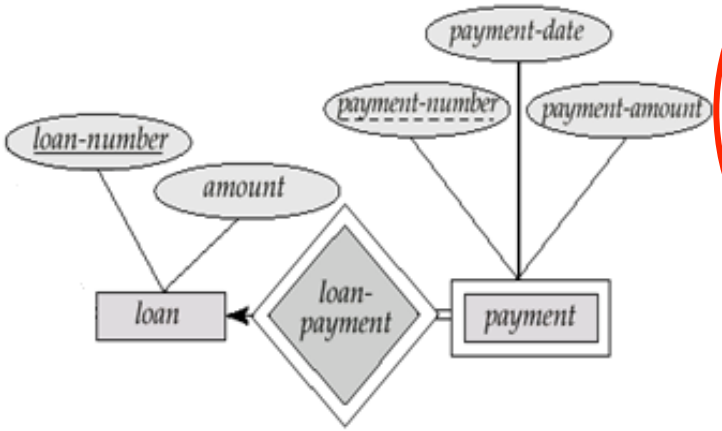


TABLE: payment

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

TABLE: loan-payment

loan-number	*payment-number*
L-11	53
L-14	69
L-15	22
L-16	58

REDUNDANCY

Representing Generalization

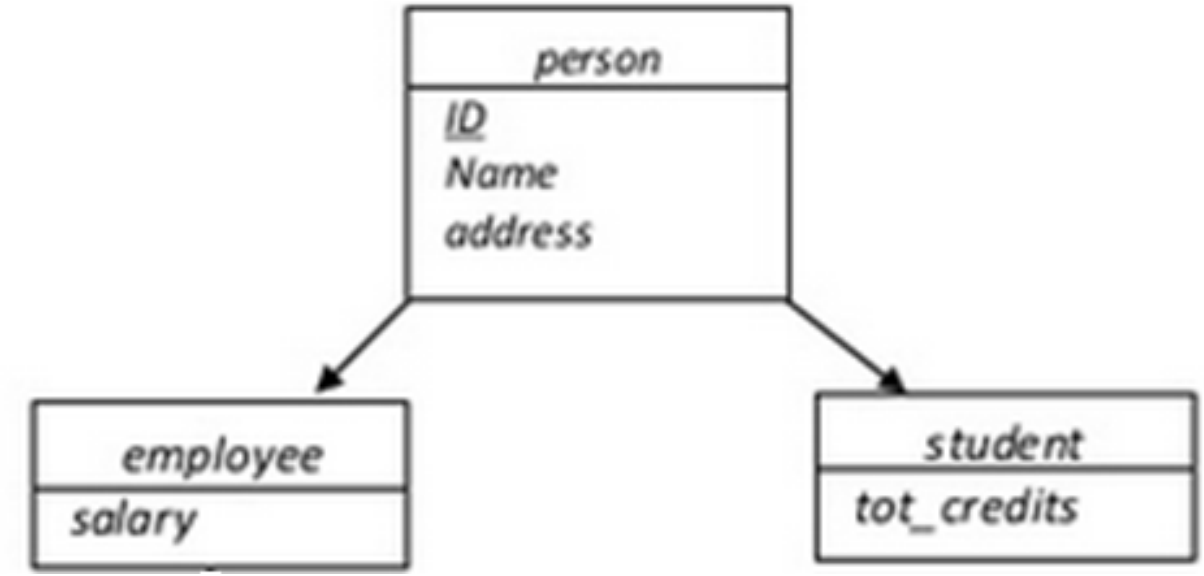
Two general approaches depending on disjointness and completeness

- **Disjoint**: A *disjointness constraint* requires that an entity belong to no more than one lower-level entity set.
- **Complete:** A *completeness constraint* requires that every entity in higher-level entity set is also a member of one of the lower-level entity set.

1. **For non-disjoint and/or non-complete class hierarchy:**
   - create a table for each super class entity set according to normal entity set translation method.
   - Create a table for each subclass entity set with a column for each of the attributes of that entity set plus one for each attributes of the primary key of the super class entity set
   - This primary key from super class entity set is also used as the primary key for this new table

Example 1: For non-disjoint

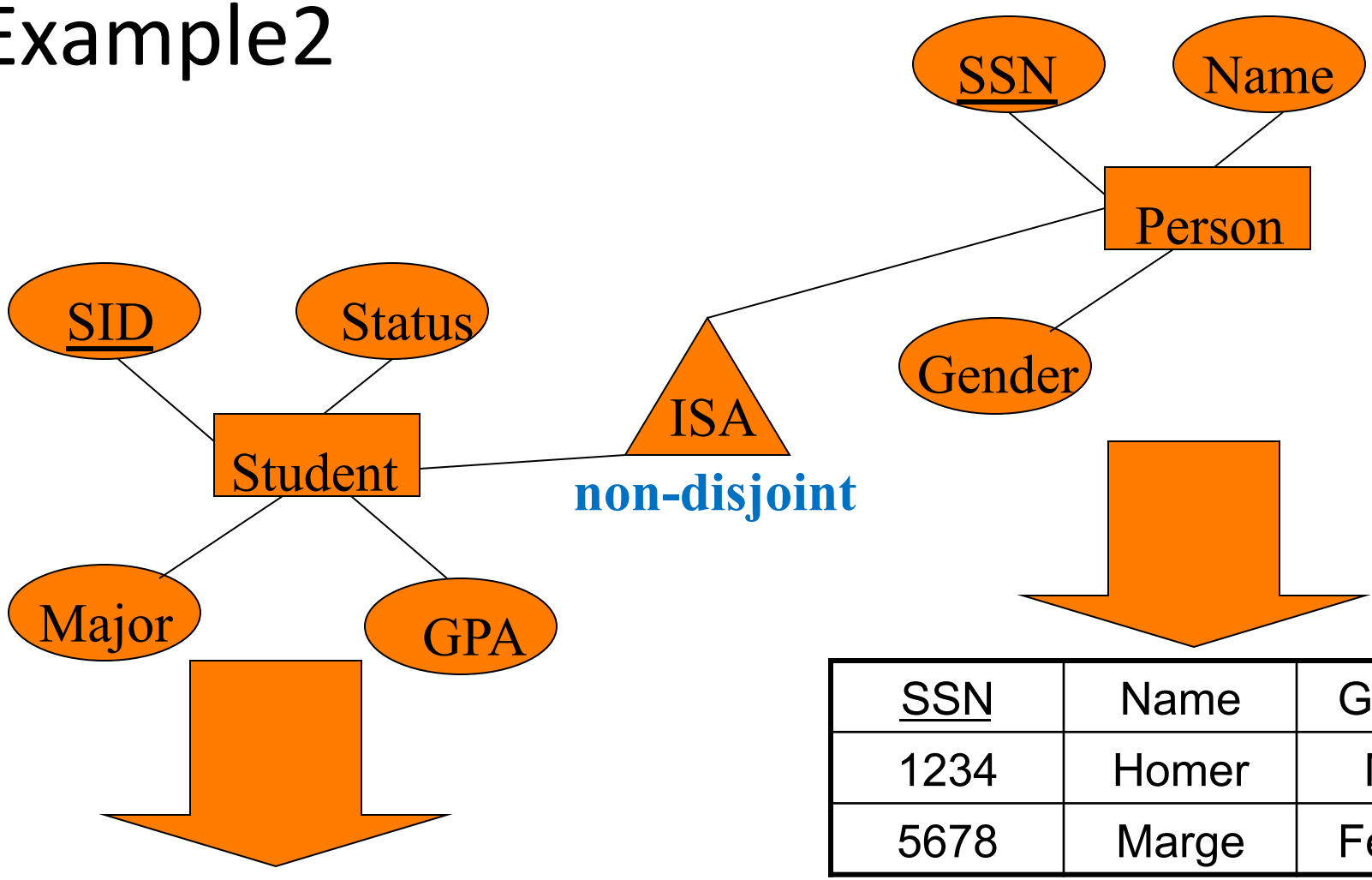
- Show the tables for non-disjoint mapping (employee can be student). Also show the composite attributes of address in person relation.



- person(ID, name, street, city)
- employee(ID, salary)
- student (ID, tot_credits)

Drawback: getting information about *employee* requires accessing two tables

Example2



SSN	Name	Gender
1234	Homer	Male
5678	Marge	Female

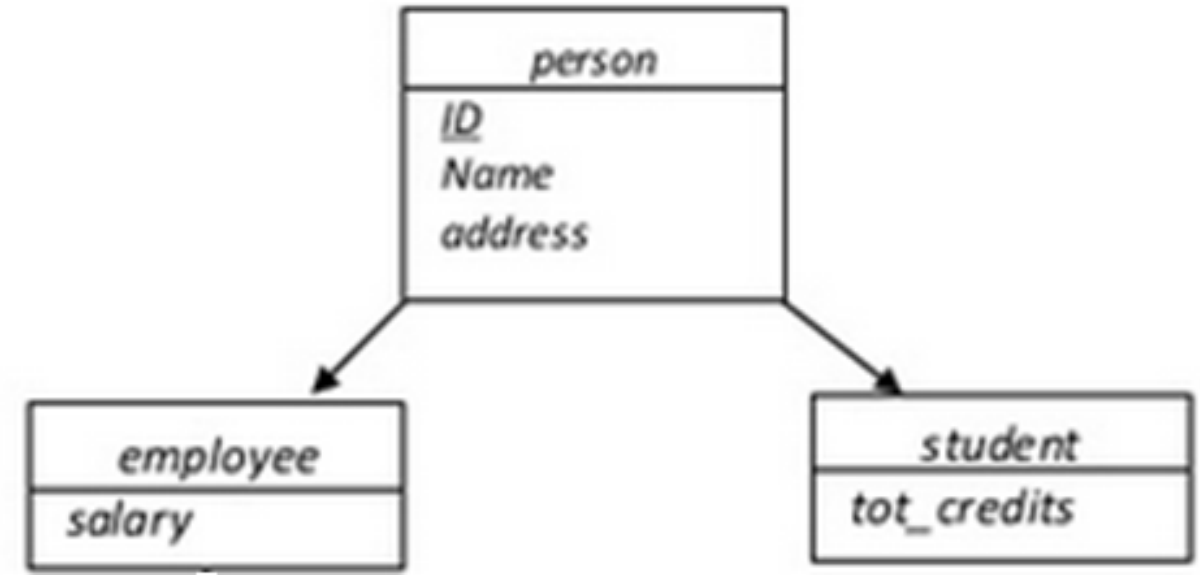
SSN	SID	Status	Major	GPA
1234	9999	Full	CS	2.8
5678	8888	Part	EE	3.6

Drawback: getting information about *employee* requires accessing two tables

Representing Class Hierarchy

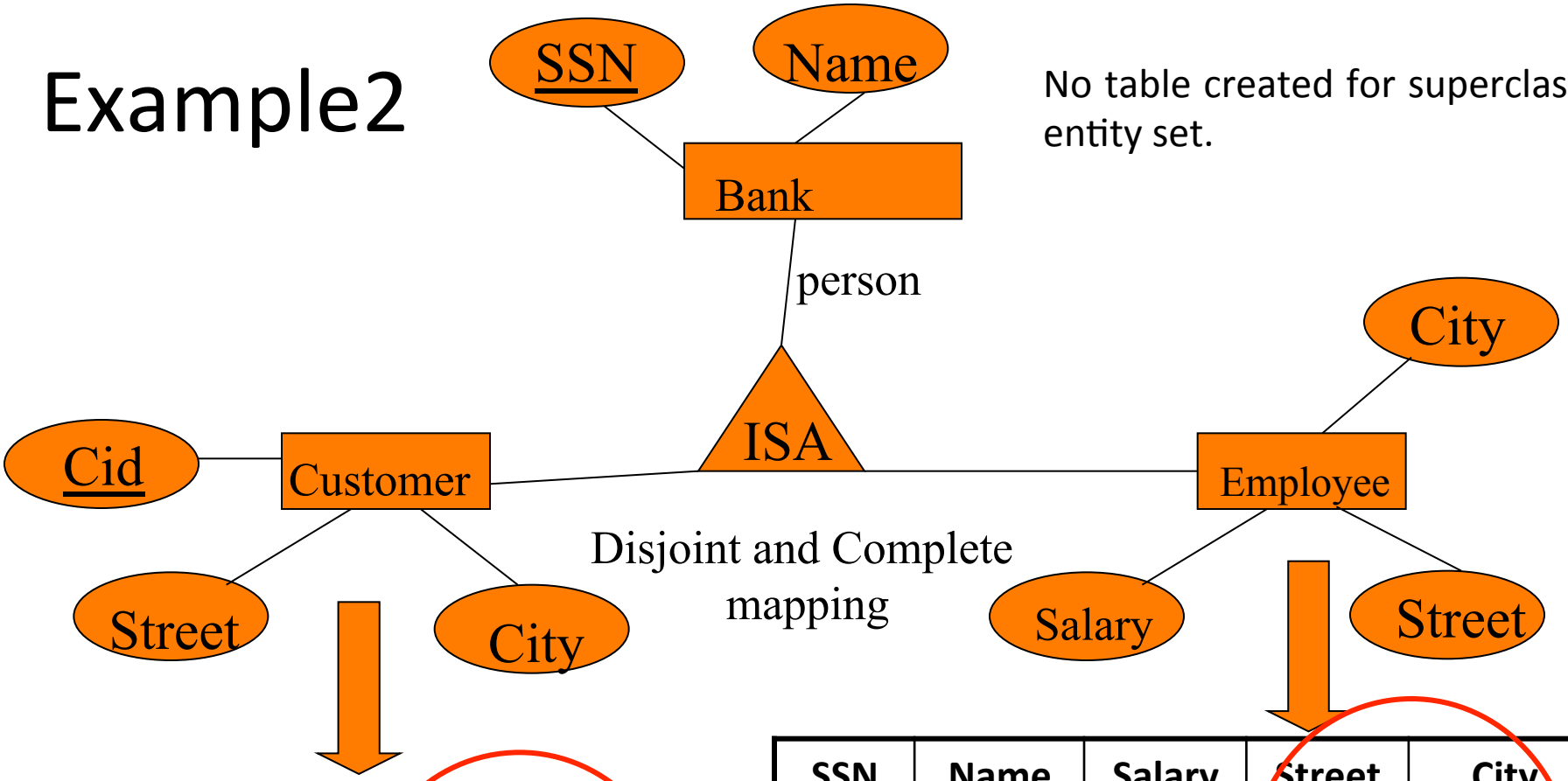
2. **For disjoint AND complete mapping class hierarchy:**
   - DO NOT create a table for the super class entity set
   - Create a table for each subclass entity set include all attributes of that subclass entity set and attributes of the superclass entity set

Example 1: **For disjoint**



- employee(ID, name, street, city, salary)
- student (ID, name, street, city, tot_credits)

Example2



No table created for superclass entity set.

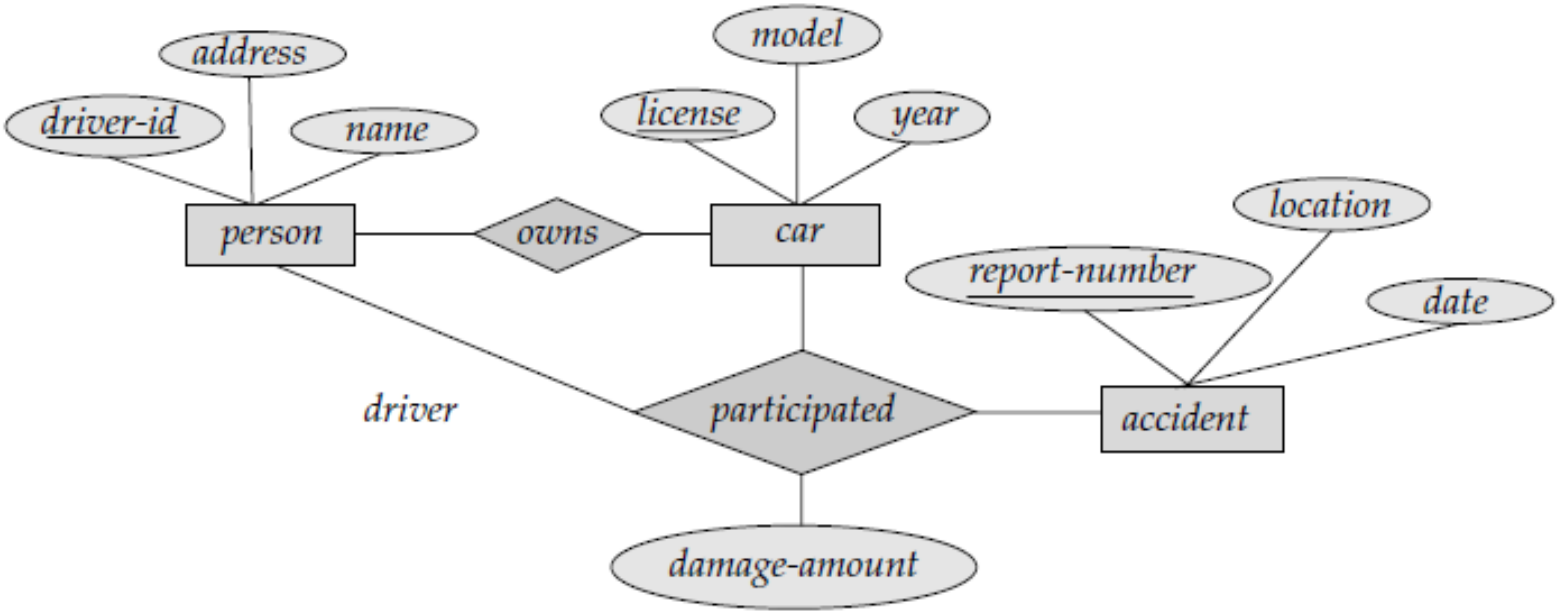
SSN	Name	Cid	Street	City
1234	John	9999	Old Broad	London
5678	Mary	8888	Queen	London

SSN	Name	Salary	Street	City
1234	John	1000$	Old Broad	London
5678	Mary	2000$	Queen	London
9123	Homer	3000$	Polo	New York

Drawback if it would have been Non-Disjoint: Street and City may be stored redundantly for persons who are both customers and employees

Example 1

- Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.



E-R diagram for a Car-insurance company.

Example 2

- Construct appropriate tables for the E-R diagrams of Car Insurance company.

Sol. Car insurance tables:

person (driver-id, name, address)

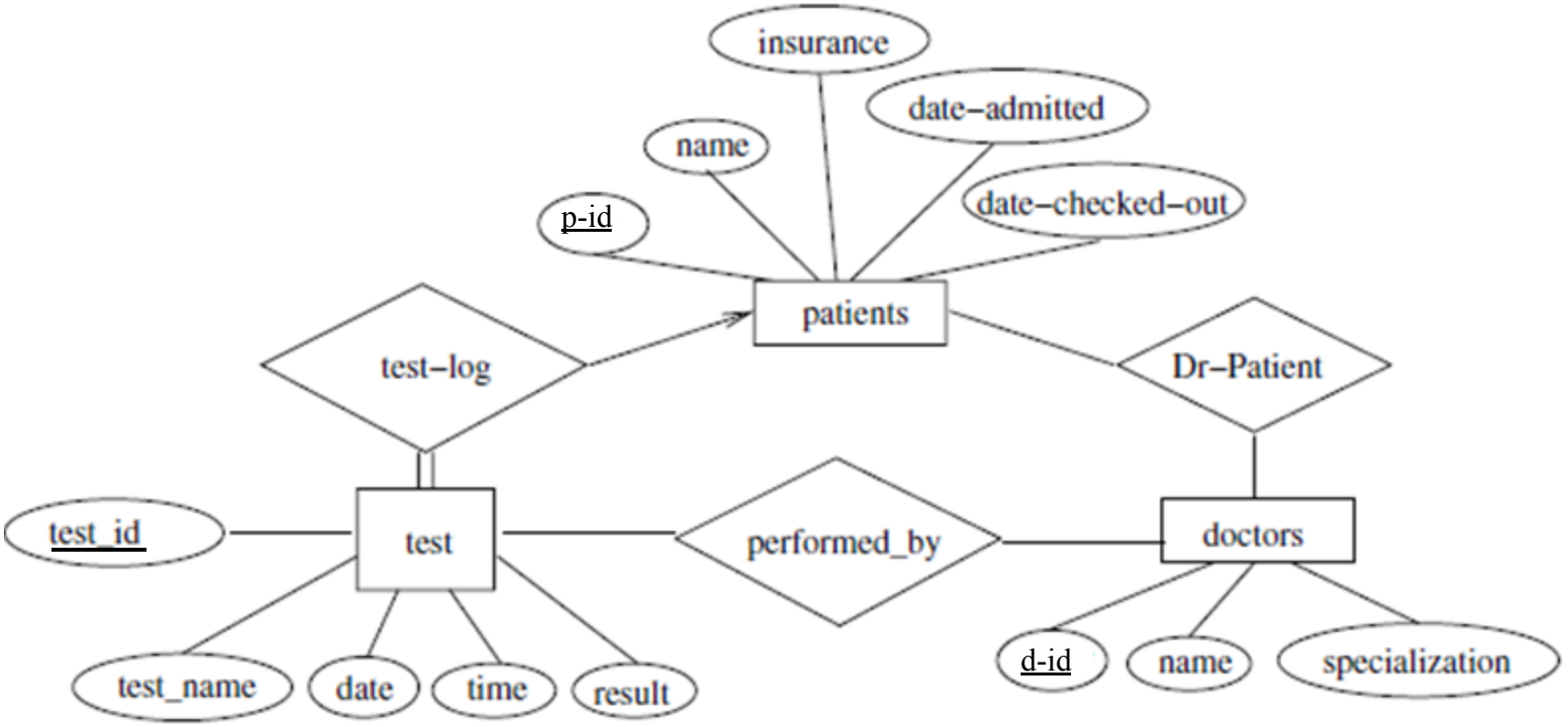
car (license, year, model)

accident (report-number, date, location)

participated(driver-id, license, report-number, damage-amount)

Example 3

- Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.



Example 4

- Construct appropriate tables for the E-R diagram of Hospital.

Sol. Hospital tables:

patients (p-id, name, insurance, date-admitted, date-checked-out)

doctors (d-id, name, specialization)

test (test-id, test_name, date, time, result)

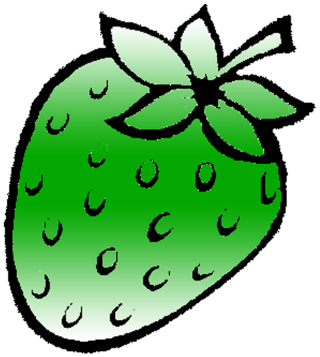
doctor-patient (patient-id, doctor-id)

test-log (test-id, test_name, date, time, result, p-id)

performed-by (test-id, d-id)

dr-patient(p-id, d-id)

STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com