Chapter 2: Entity-Relationship Model

- Entity Sets
- Relationship Sets
- Design Issues
- Mapping Constraints
- Keys
- E-R Diagram
- Extended E-R Features
- Design of an E-R Database Schema
- Reduction of an E-R Schema to Tables

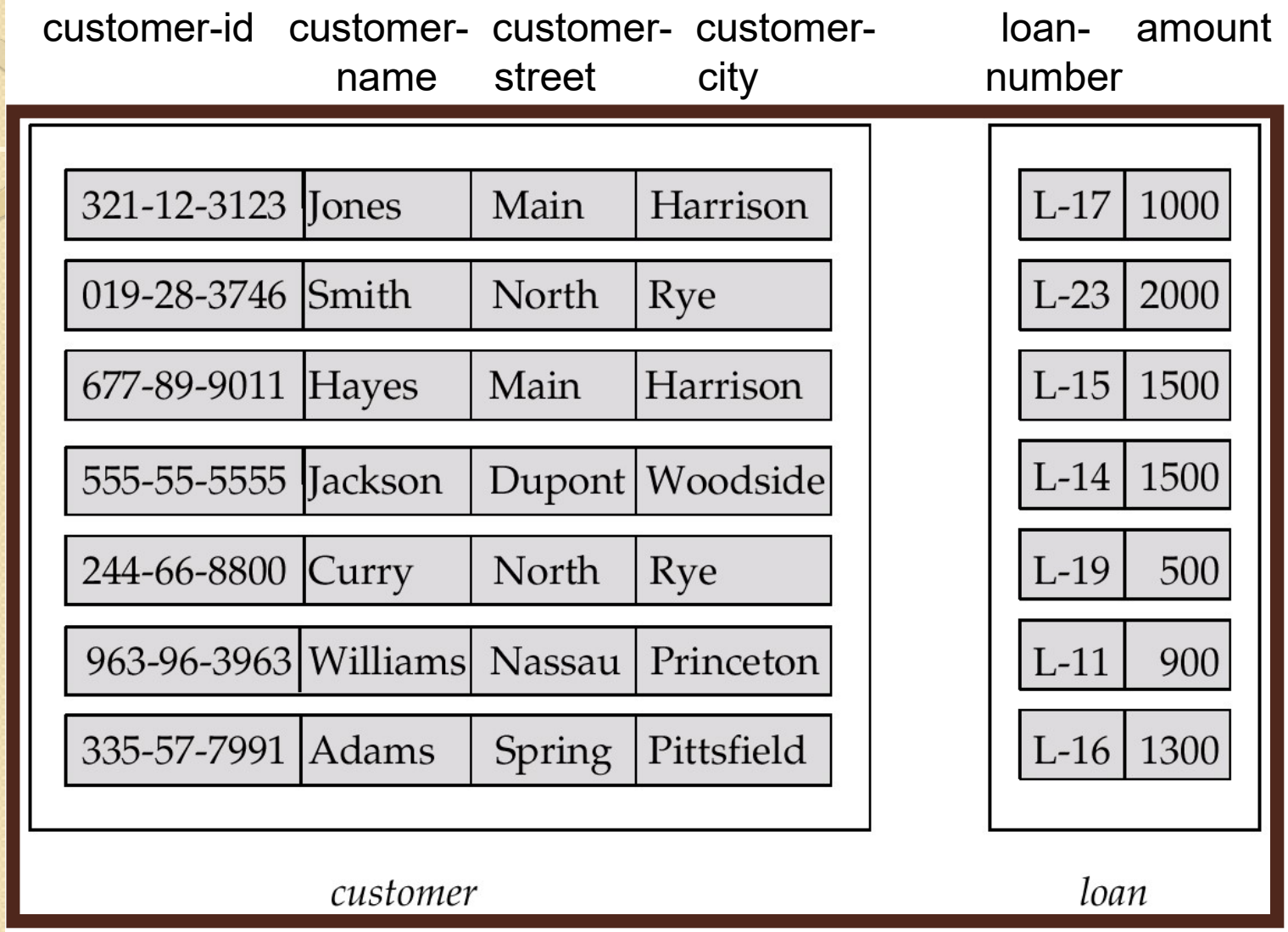
Introduction

- The entity-relationship (E-R) data model perceives the real world.
  - Objects: entities
  - Relationships among these objects
- It was developed to facilitate database design (schema) of an enterprise
- E-R model is then translated into the relational model
- It is one of the semantic data models

Entity Sets

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An *entity* is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have *attributes*
  - Example: people have *names* and *addresses*
- An *entity set* is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
- Entity sets do not need to be disjoint.
  - E.g. person entity may be employee entity or customer entity or both or neither.

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

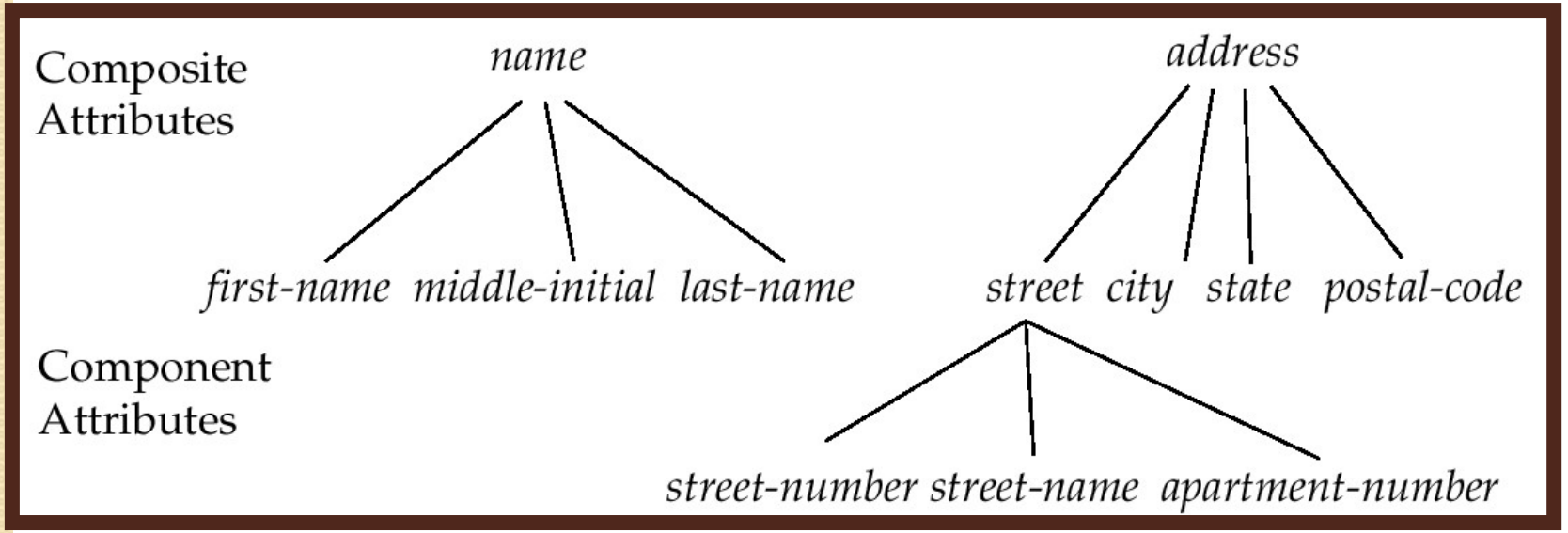
- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

  Example:

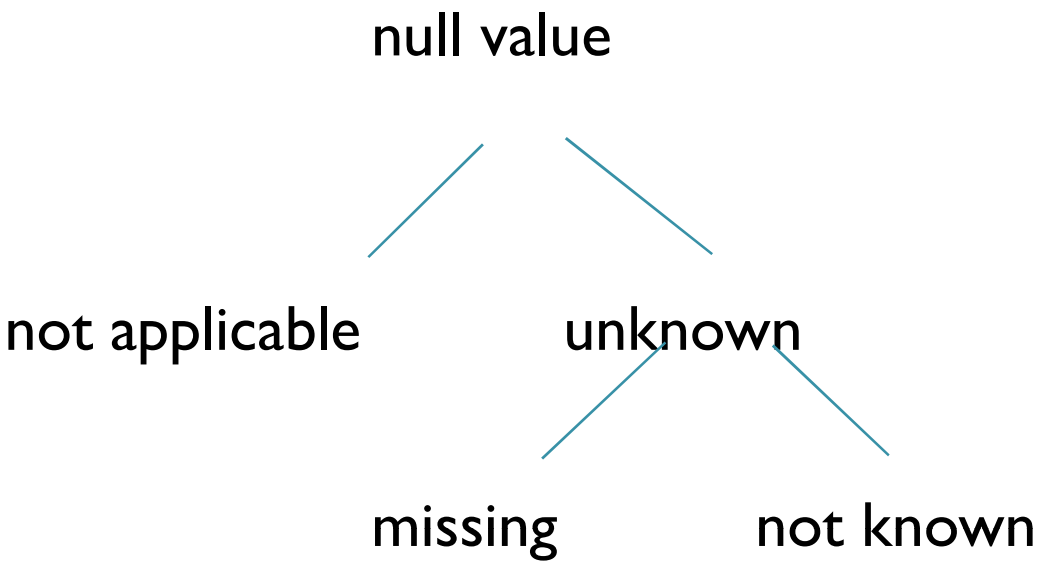
  customer = (customer-id, customer-name, customer-street, customer-city)
  loan = (loan-number, amount)

- *Domain* – the set of permitted values for each attribute
  - Domain of loan-number: set of all strings of the form L-n, where n is a positive integer
- The customer-id uniquely identify customers; since there can be more than one customer with same name, street and city.
- An entity can be described by a set of (attributes, data value) pairs.
  - E.g. {*(customer-id, C-1), (customer-name, abc) , (customer-street, xyz), (customer-city, Ahm)*}

- Attribute types:
  - *Simple* and *composite* attributes.
    - E.g. simple attribute : Amount (cannot be divided into parts)
      composite attribute : name, address
  - *Single-valued* and *multi-valued* attributes
    - E.g. single valued attribute : loan-number
      multivalued attribute: *phone-numbers*
  - *Derived* attributes
    - Can be computed from other attributes
    - E.g. *age*, given date of birth



- An attribute may have null value (when entity does not have value for it)



- Not applicable: the value does not exist for the entity
  - E.g. middle name
- Missing: the value exist, but don't have information
  - e.g. apartment number
- Unknown : don't know whether apt number is part of address or not

Relationship Sets

- A relationship is an association among several entities

  Example:

  Hayes *borrower* 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

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

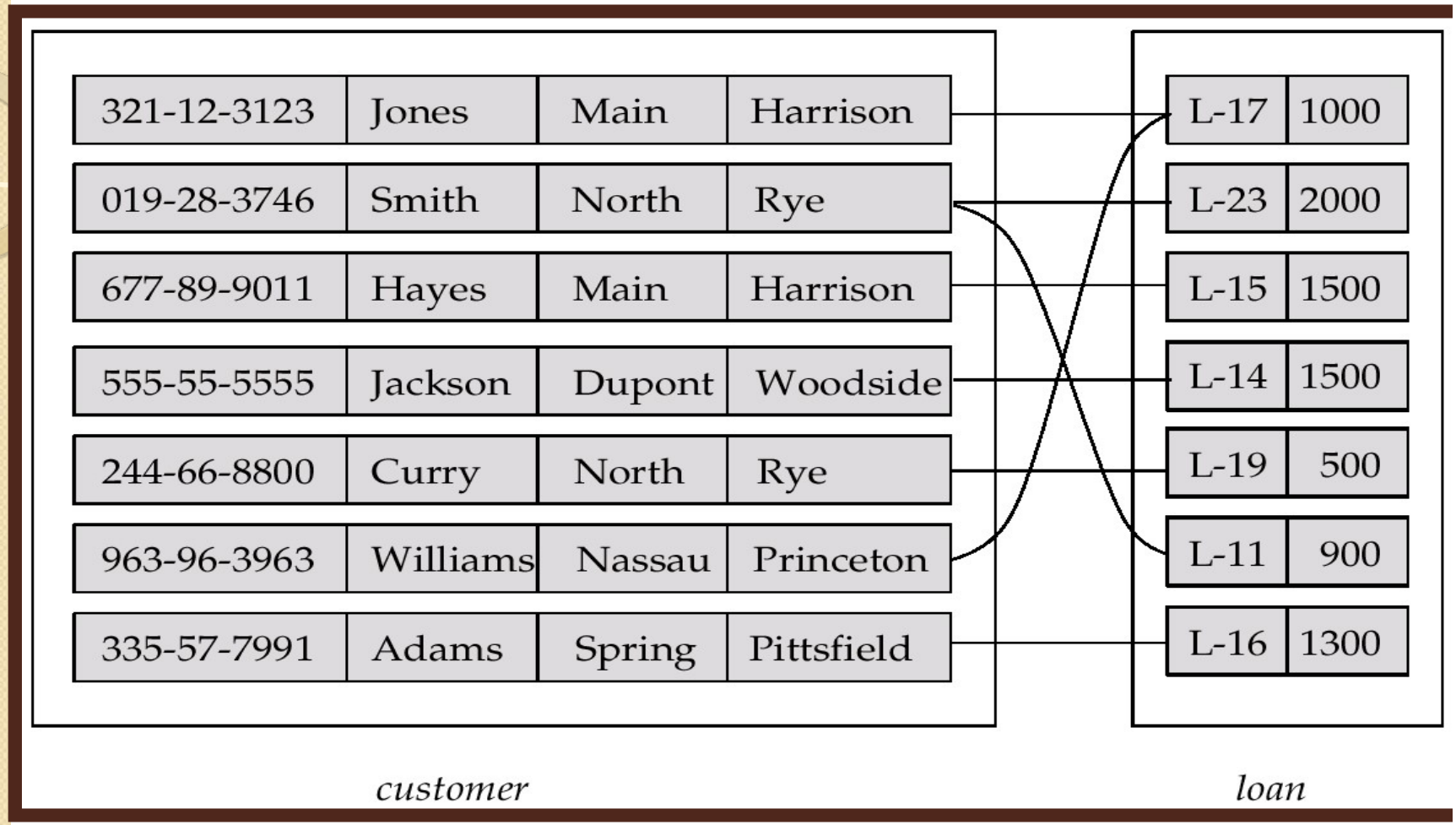
  where $(e_1, e_2, \ldots, e_n)$ is a relationship

  - Example:

$$(\text{Hayes, L-15}) \in \text{borrower}$$

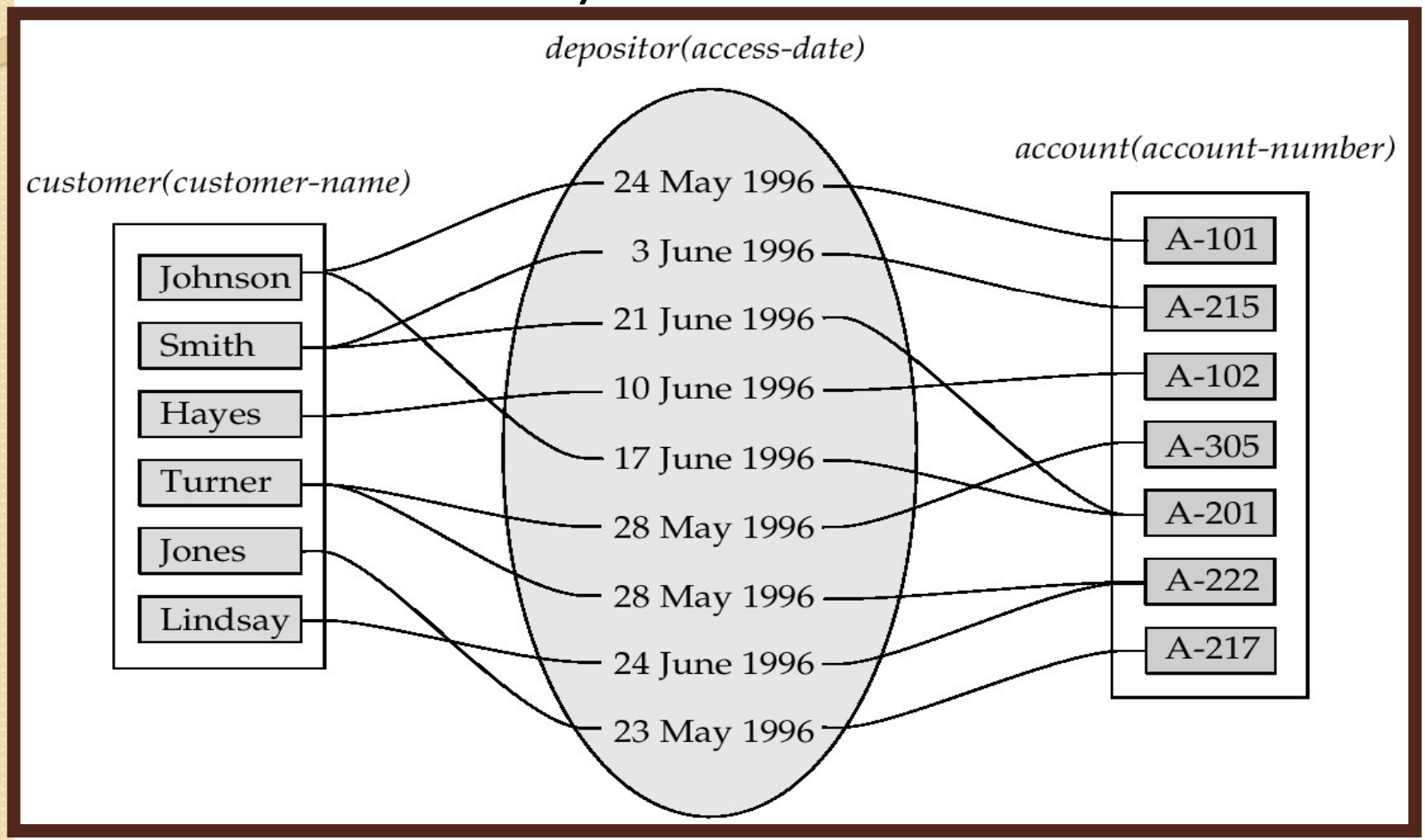
- *The entity sets* $E_1, E_2, \ldots, E_n$ *participate in relationship set R.*

Relationship Set *borrower*

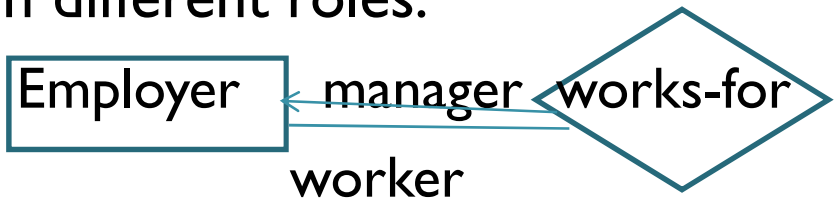


Relationship Sets (Cont.)

- An *attribute* can also be property of a relationship set.
- For example, the *depositor* relationship set between entity sets *customer* and *account* may have the attribute *access-date*



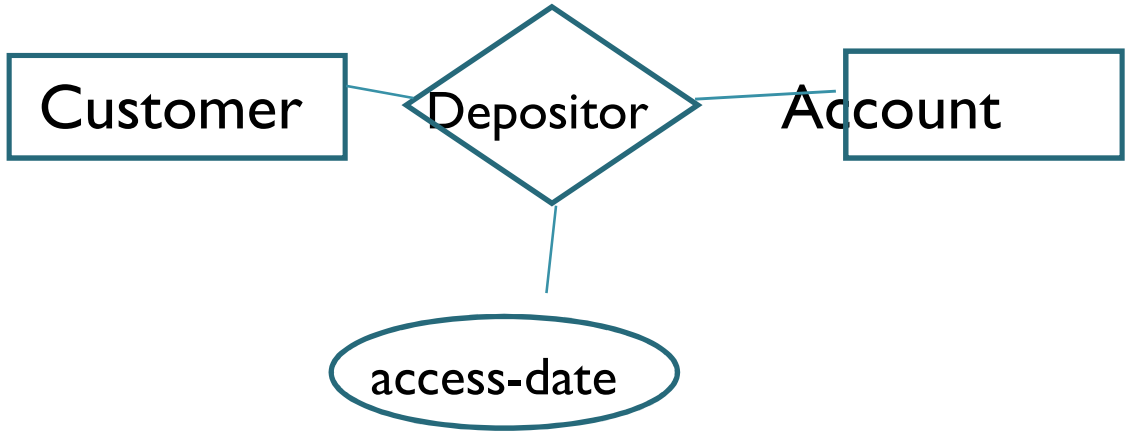
- Same entity set participates in relationship set more than once, in different roles.
  - E.g.



Called as Recursive relationship set.

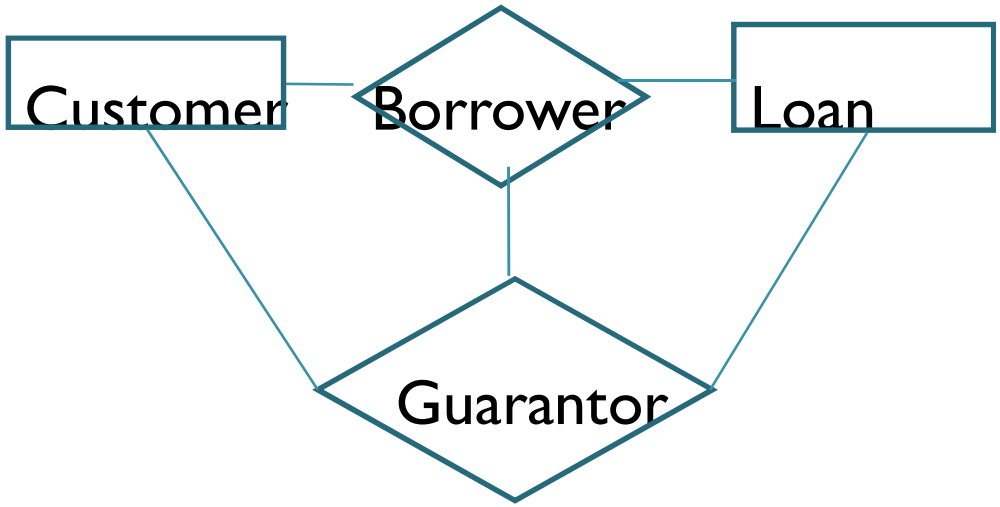
  - (Worker, manager) is valid and (manager, worker) pairs are excluded
- A Relationship may have descriptive attributes.

e.g.



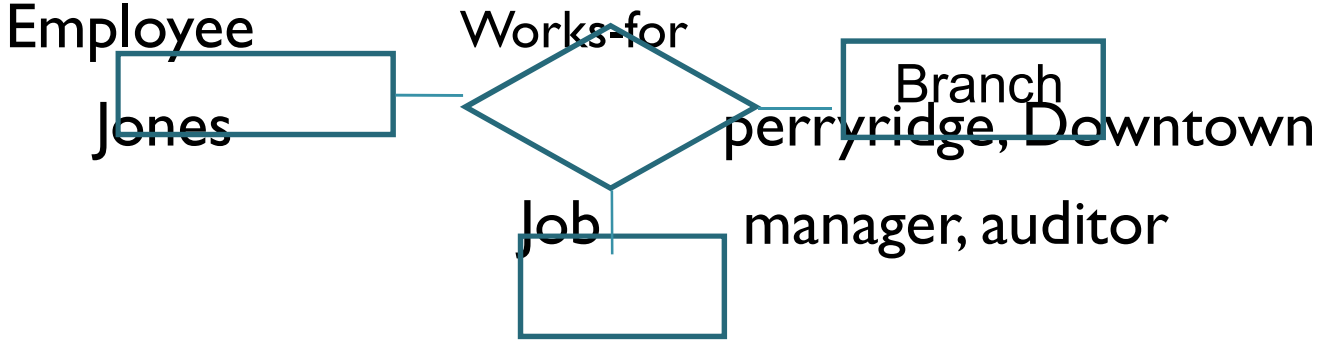
  - Jones customer has recently accessed account A-217 on access date 1-1-13.

- There can be more than one relationship set involving same entity sets.



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.
  - E.g. borrower : binary relationship set(degree = 2)
- Relationship sets may involve more than two entity sets.
- 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(n=3) between entity sets *employee, job and branch*



Constraints

- Constraints — Mapping Cardinalities
  \ participation constraints

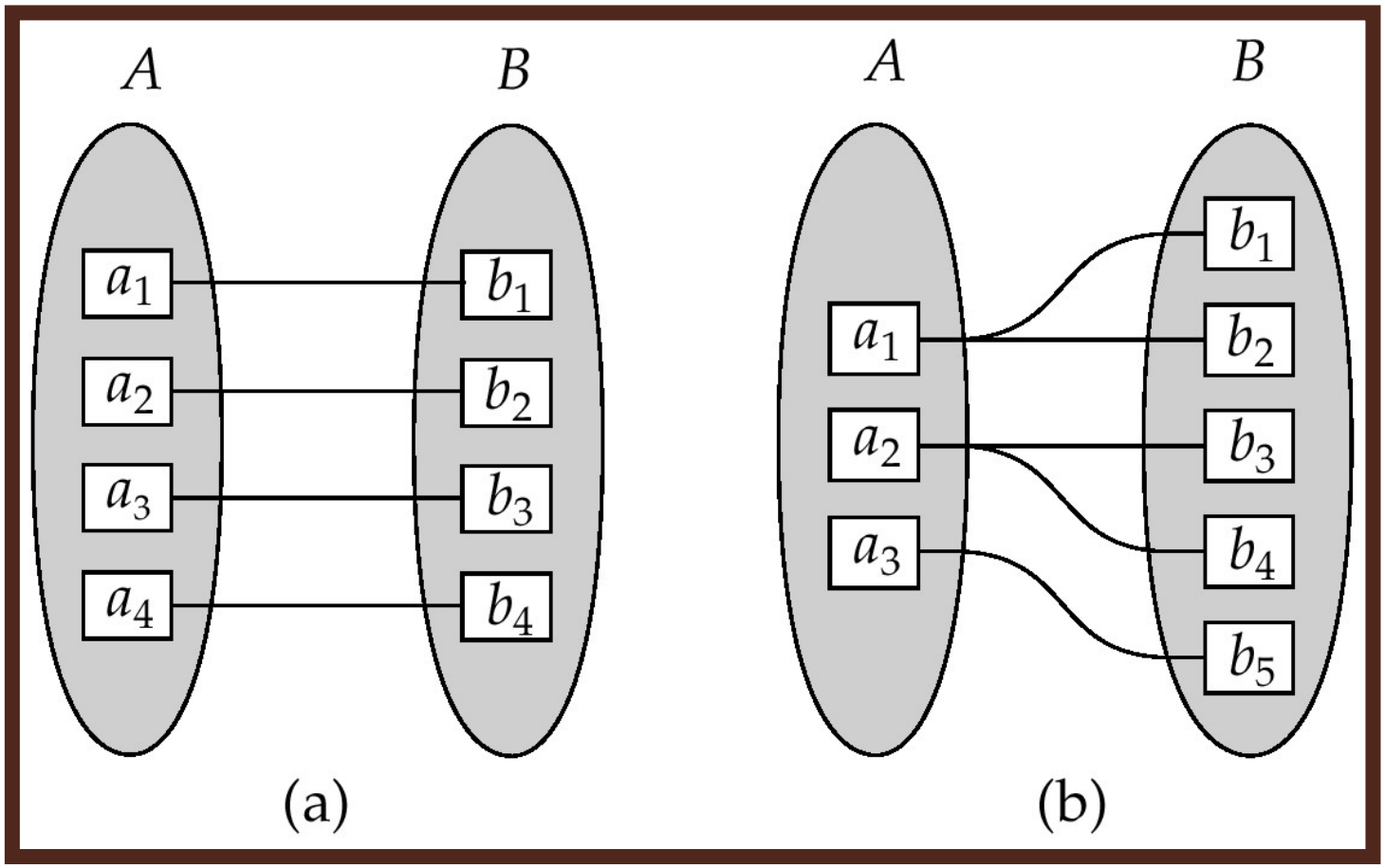
Mapping Cardinalities

- Express the number of entities to which another entity can be associated via a relationship set.
- For a binary relationship set R between entity sets A and B, the mapping cardinality must:
  - One to one
    - An entity in A is associated with atmost one Entity in B &

      An entity in B is associated with atmost one Entity in A
  - One to many
    - An entity in A is associated with any number(zero or more) of Entity in B &

      An entity in B is associated with atmost one Entity in A

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

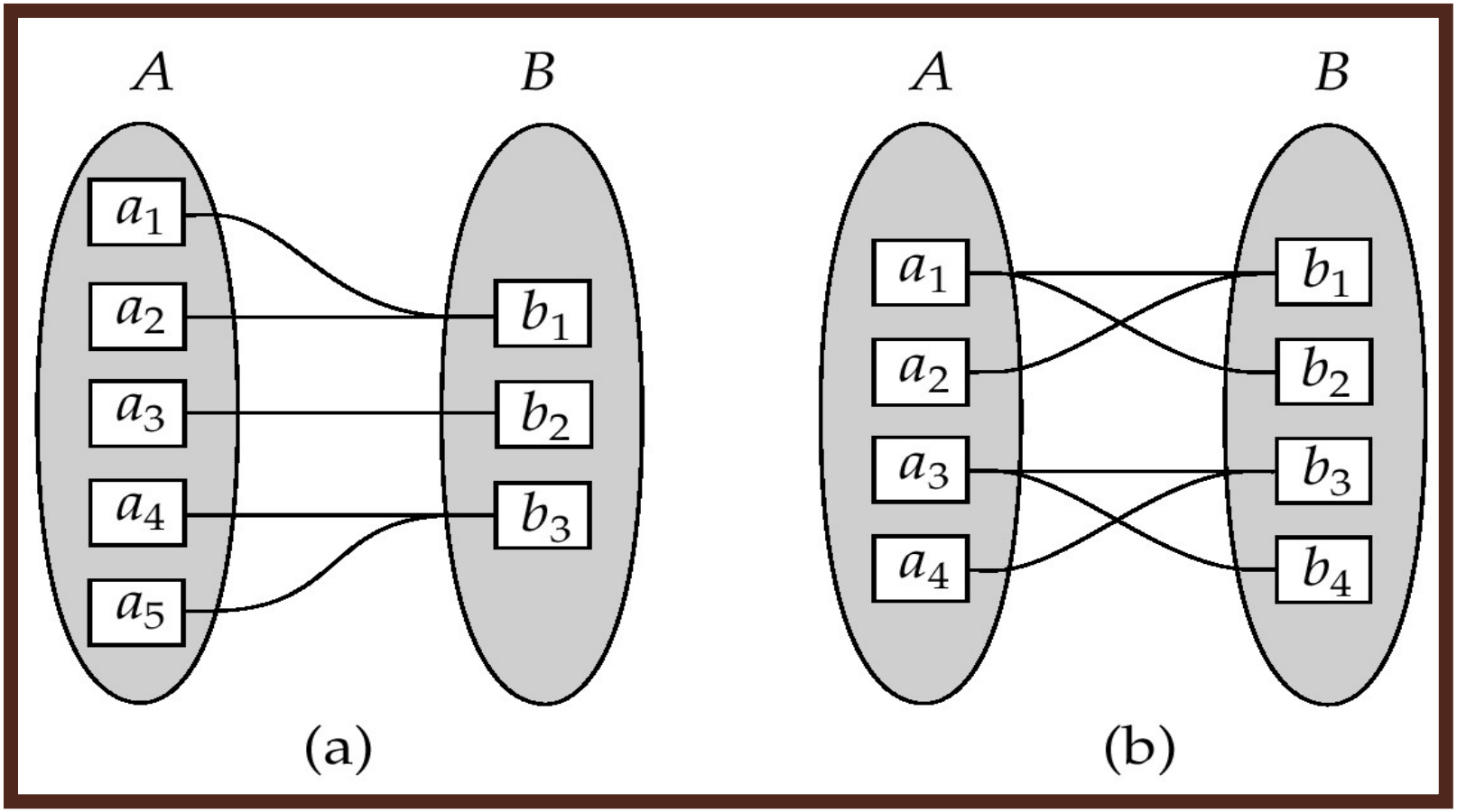
- Many to one
  - An entity in A is associated with atmost one Entity in B &

    An entity in B is associated with any number of Entity in A
- Many to many
  - An entity in A is associated with any number of Entity in B &

    An entity in B is associated with any number of Entity in A

- E.g. R = borrower
  - Customer to loan is one to many (A customer can have several loans)
  - Customer to loan is many to many (if loans taken jointly by several business partners)

Mapping Cardinalities



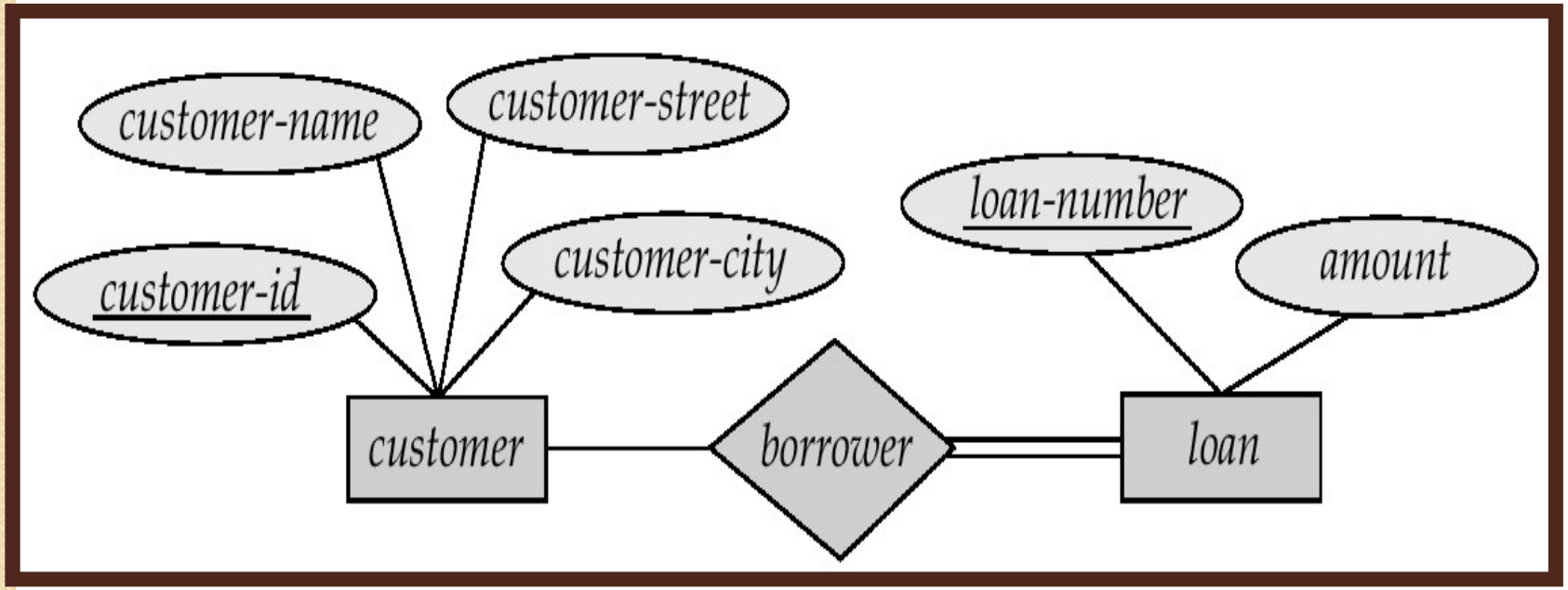
Many to one

Many to many

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

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

- A *super key* of an entity set is a set of one or more attributes whose values uniquely determine each entity.
  - E.g. customer-id, {customer-name, customer-id}
  - Customer-name is not super key
  - Super key may contain extraneous attributes
- A *candidate key*
  - Superkeys for which no proper subset exists. Such minimal superkeys are called as candidate keys.
  - *Customer-id, (customer-name, customer-street)* is candidate key of *customer*
  - *(Customer-id, customer-name) is not candidate key*
  - *account-number* is candidate key of *account*
- A primary key: Although several candidate keys may exist, one of the candidate keys is selected by database designer to be the *primary key*.
  - It should be chosen such that its attributes are never or rarely changed.

Keys for Relationship Sets

- To distinguish various relationship set
- If R has no attributes associated with it, then 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.*
- If R has attributes a1, a2, …am then

  pk(E1) U pk(E2) U… Upk(En) U {a1, a2, …am }

  forms the superkey
- If name of primary key not unique ; then attributes are renamed entityset. attribute name

- Must consider the mapping cardinality of the relationship set when deciding 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.

1) M-M R from Customer to Account (R = depositor)

primary key of R = pk{C} U pk{A}

2) 1-M R from Customer to Account

primary key of R = pk{A}

3) M-1 R from Customer to Account

primary key of R = pk{C}

4) 1-1 R from Customer to Account

primary key of R = pk{C} or pk{A} can be used

- For non-binary R, no cardinality constraints

superkey discussed is only candidate key and it is chosen as the primary key.

- If cardinality constraints exists, choice of PK is difficult.

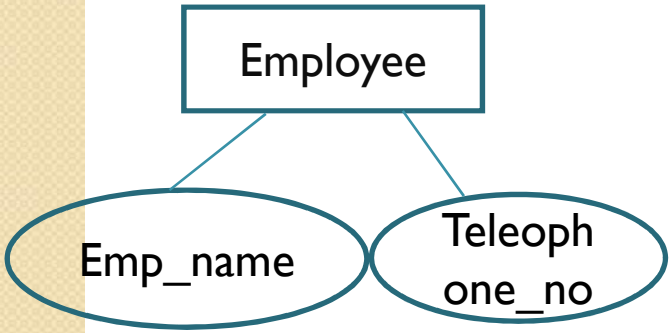
Design Issues

Use of entity sets vs. attributes

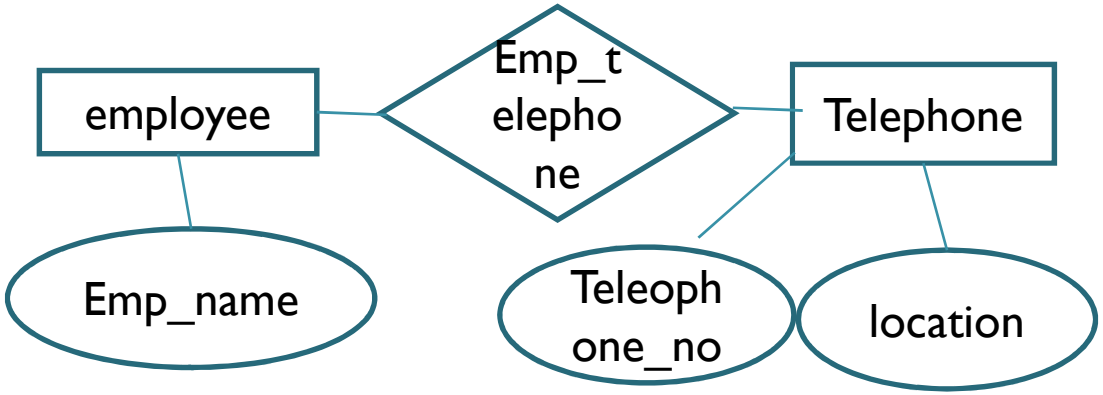
Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.

E.g.

Telephone as attribute



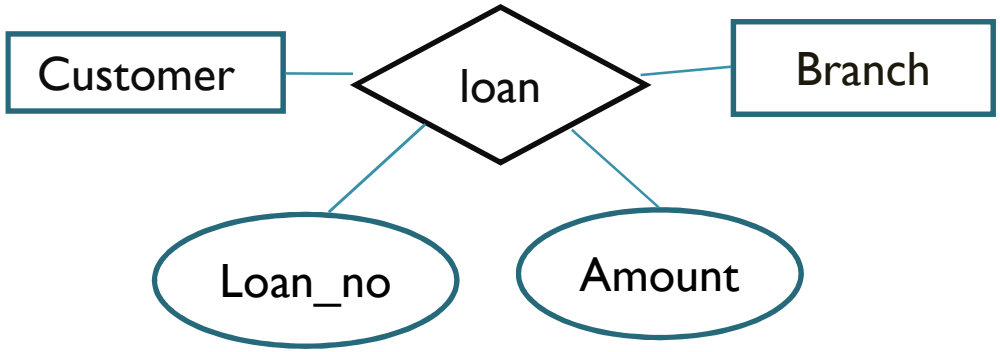
Telephone as entity



Use of entity sets vs. relationship sets

Possible guideline is to designate a relationship set to describe an action that occurs between entities.

- Loan as Relationship set R



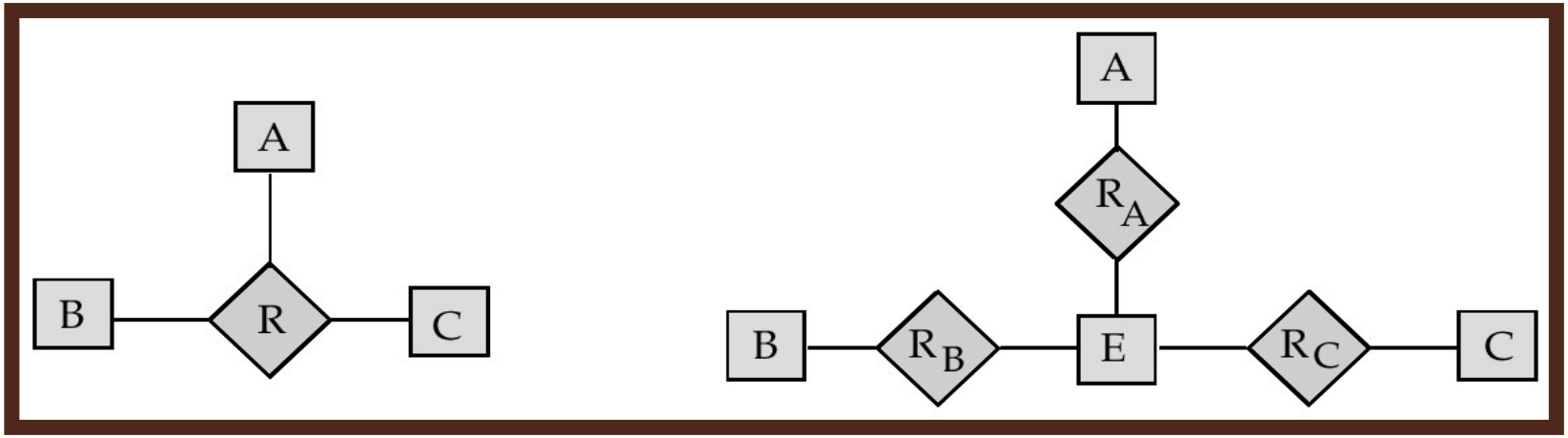
- 1-1R from customer to branch is ok
- M-1R from customer to branch (i.e. several customers hold loan jointly)
  - Need to define separate R joint loan. So, duplication of loan no. and amount.
- Two problem with replication
  - Data stored multiple times, wastage of space
  - Updates leave data in an inconsistent state //values are diff
- Loan as entity : No problem of replication

Binary versus *n*-ary relationship sets

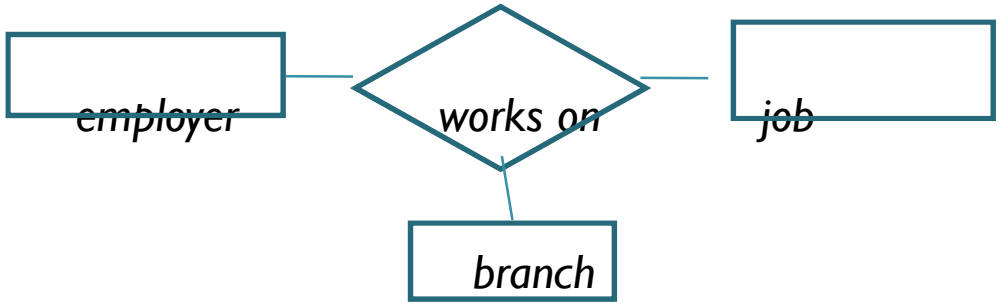
- Although it is possible to replace any non-binary (*n*-ary, for *n* > 2) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g. A ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g. only mother being know)

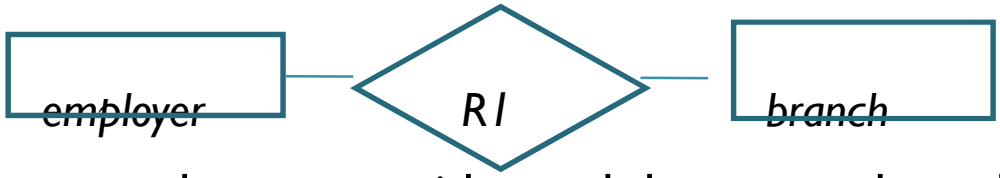
Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:
    1. R_A, relating *E* and *A*
    2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*
  - Create a special identifying attribute for *E*
  - Add any attributes of *R* to *E*
  - For each relationship (a_i , b_i , c_i) in *R*, create
    1. a new entity e_i in the entity set *E*
    2. add (e_i , a_i) to R_A
    3. add (e_i , b_i) to R_B
    4. add (e_i , c_i) to R_C



- But there are some relationships that are naturally non-binary and Translating all constraints may not be possible
  - E.g. *works-on (ternary Relationship set)*
  - *R is 1-M from C to (A,B)*
  - *Works-on is R between employee and branch and employee and job*



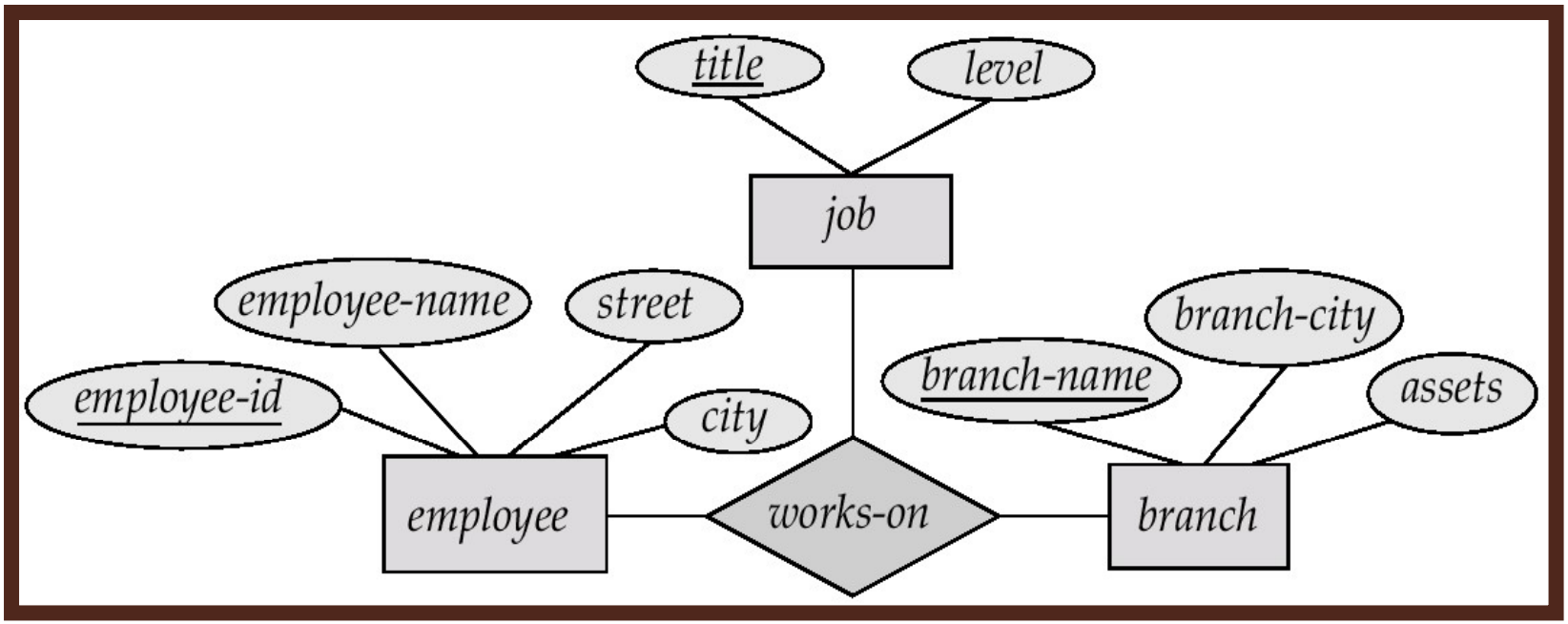


Jones works at perryridge and downtown branch

Jones is manager and auditor

Jones is manager at perryridge and an auditor at downtown can't be shown using binary relationship.

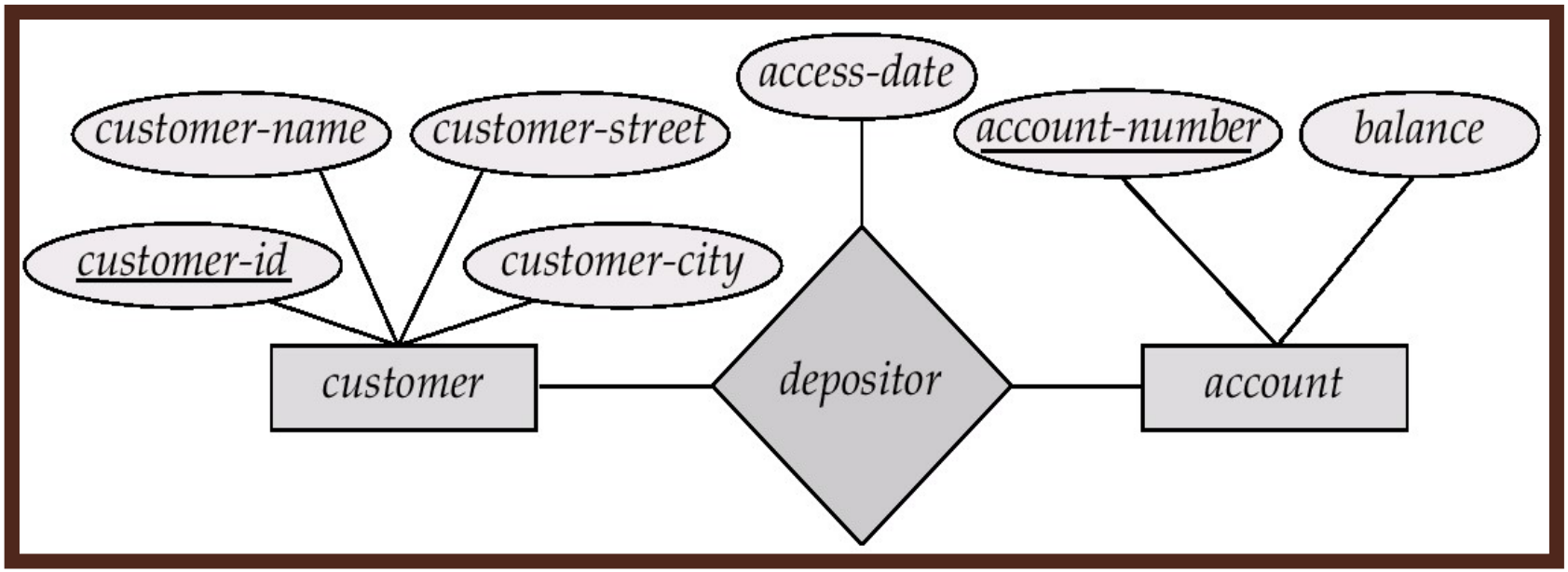
E-R Diagram with a Ternary Relationship



Placement of relationship attributes

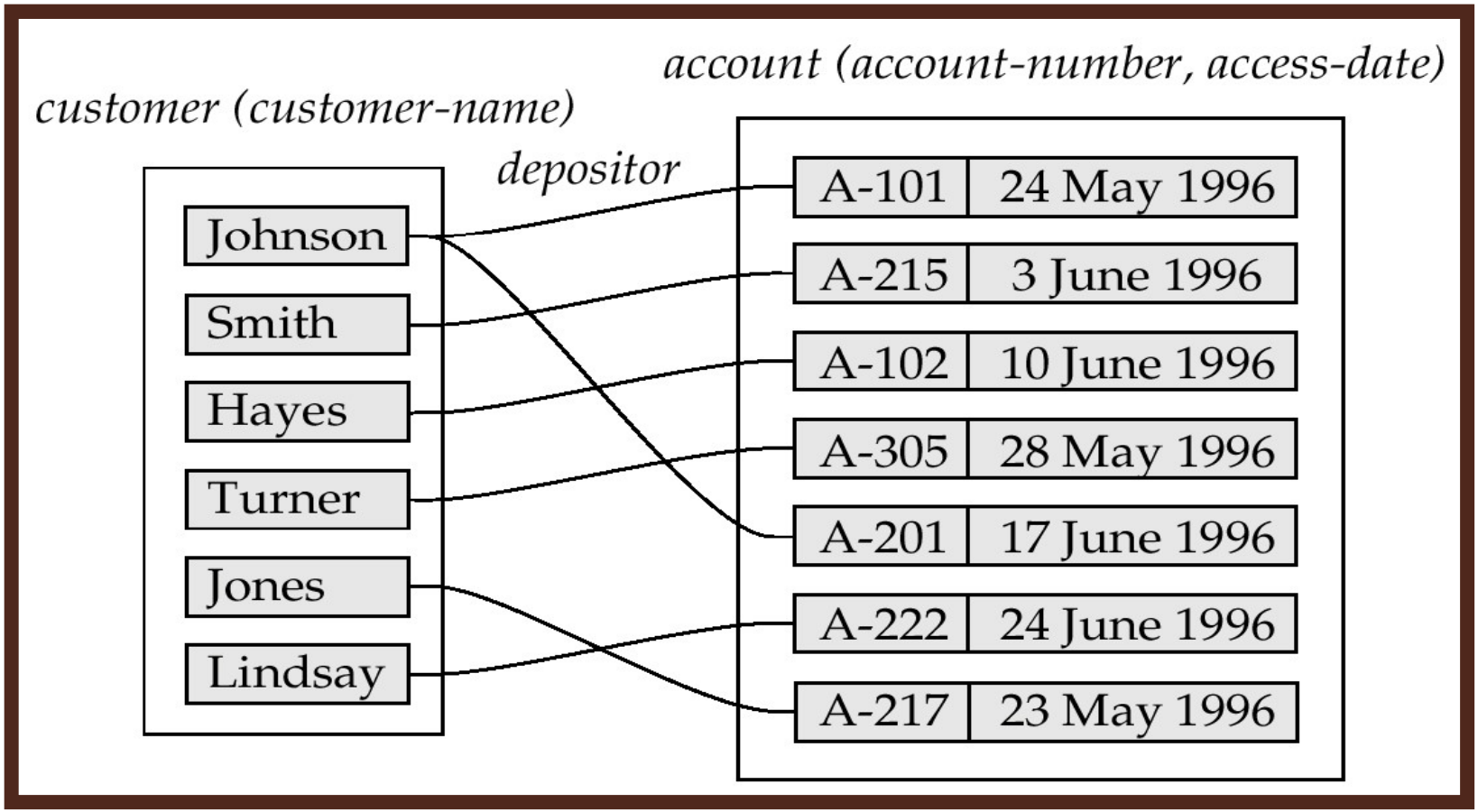
- Cardinality ratio of Relationship affects placement of relationship attributes.
  - M-1/1-M R : attribute of R must be placed at "many" side of entity
  - 1-1 R: attribute of R can be placed on either one of the entity set
  - M-M R : attribute of R must be associated with R.

Relationship Sets with Attributes

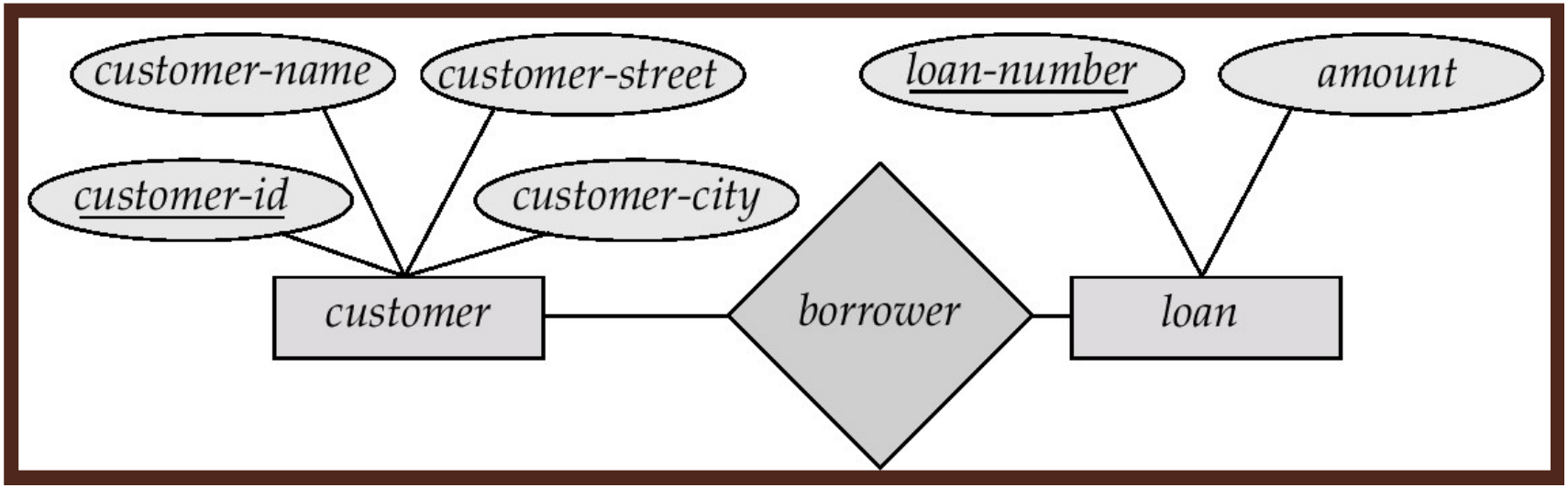


Mapping Cardinalities affect ER Design

- Can make *access-date* an attribute of account, instead of a relationship attribute, if each account can have only one customer
  - I.e., the relationship from account to customer is many to one, or equivalently, customer to account is one to many



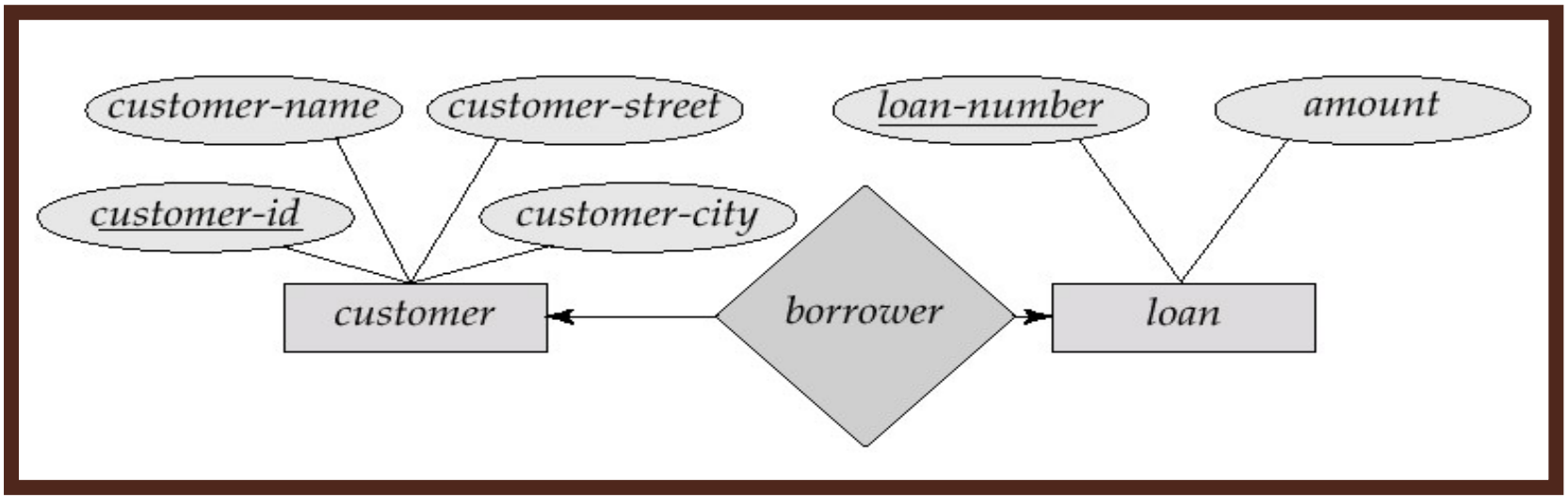
E-R Diagrams



- **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

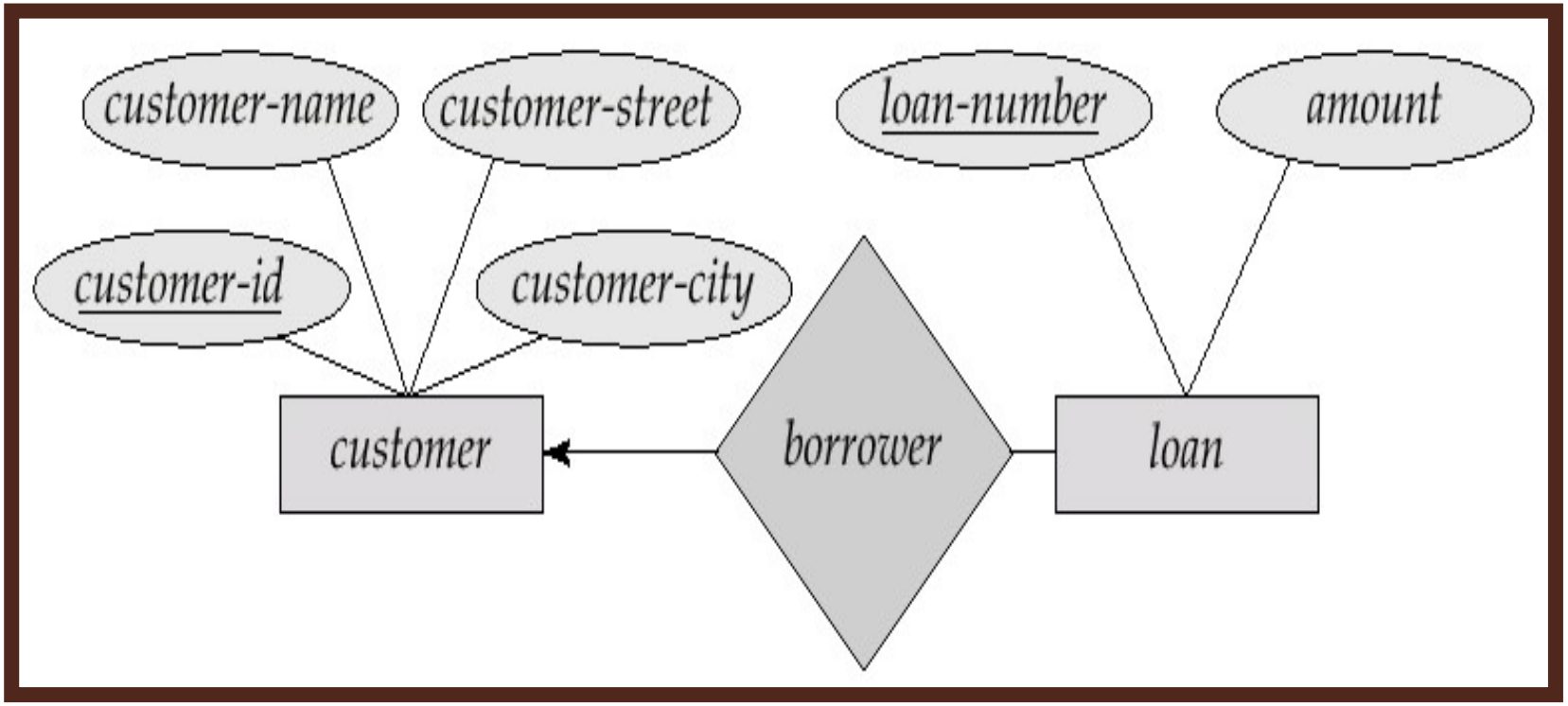
Cardinality Constraints

- 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*



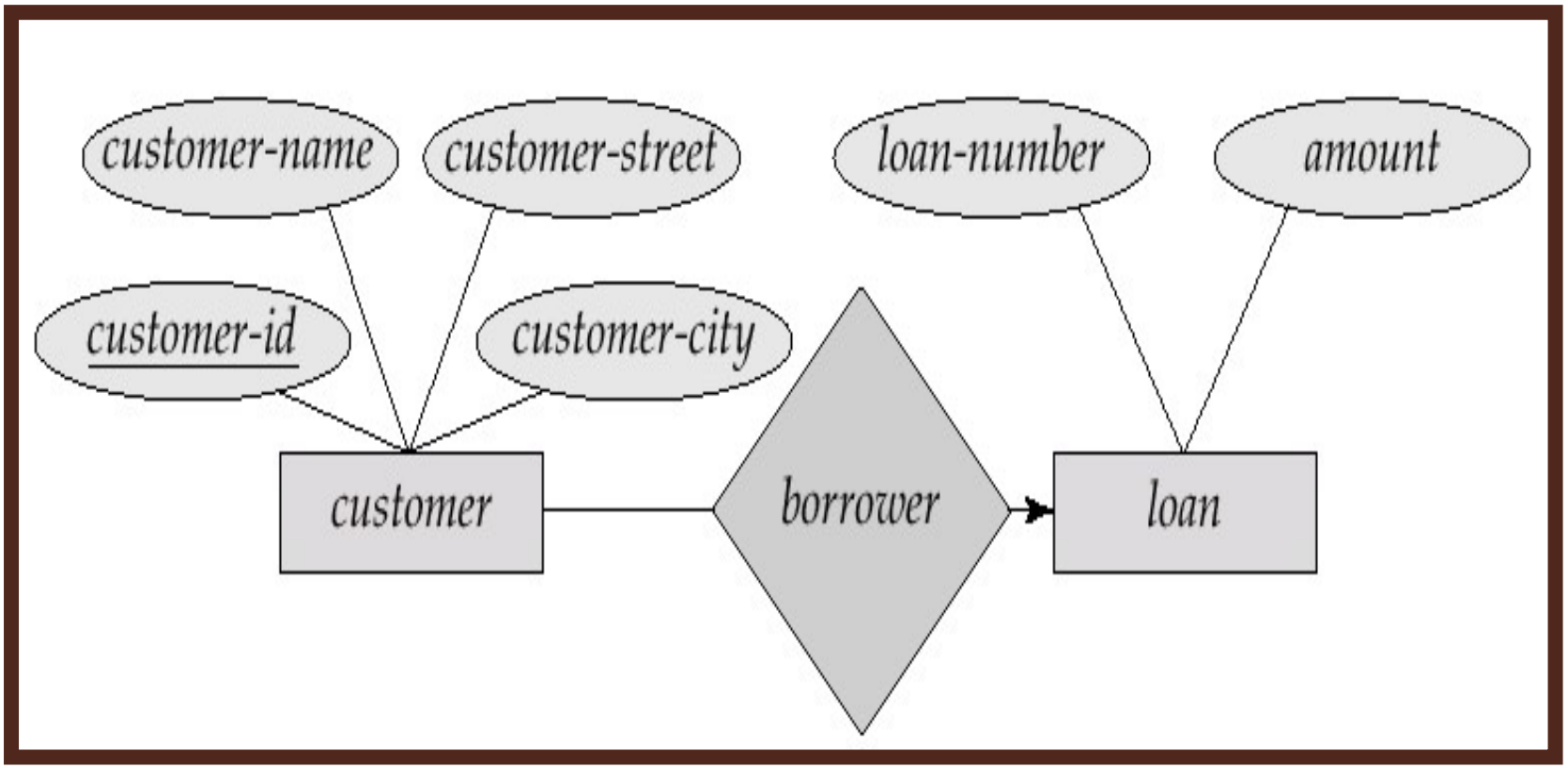
One-To-Many Relationship

- 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*

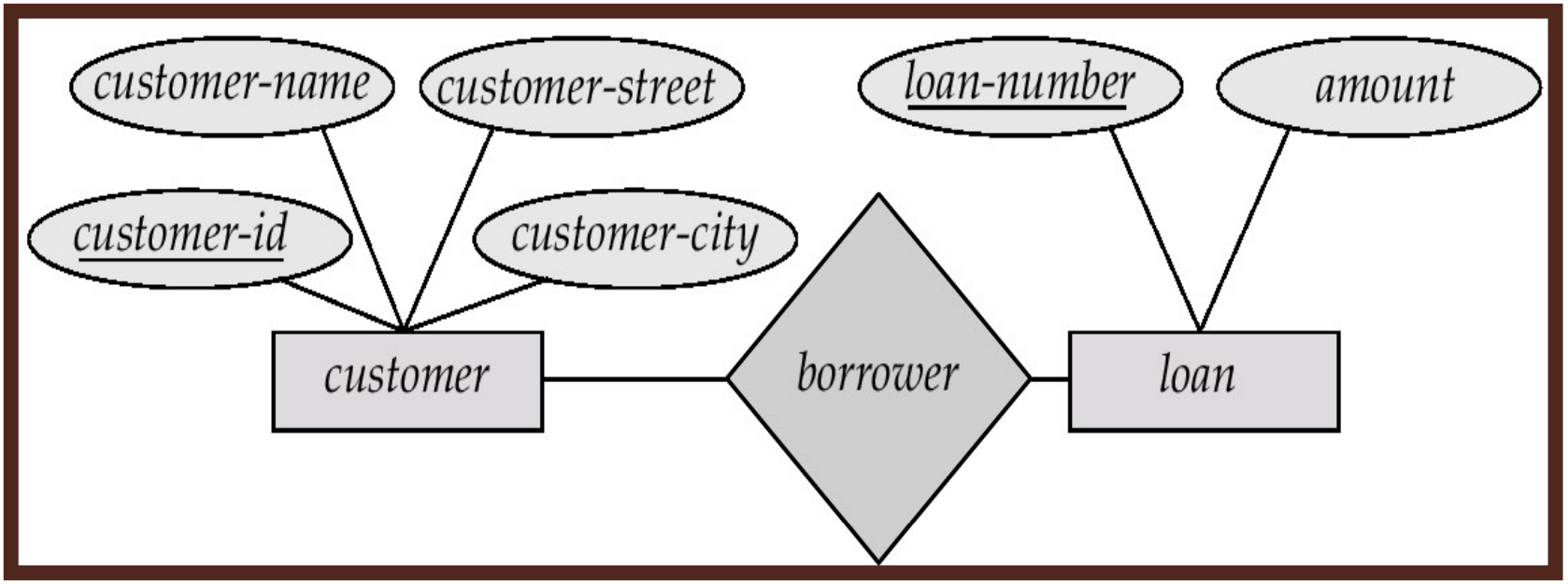


Many-To-One Relationships

- 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*

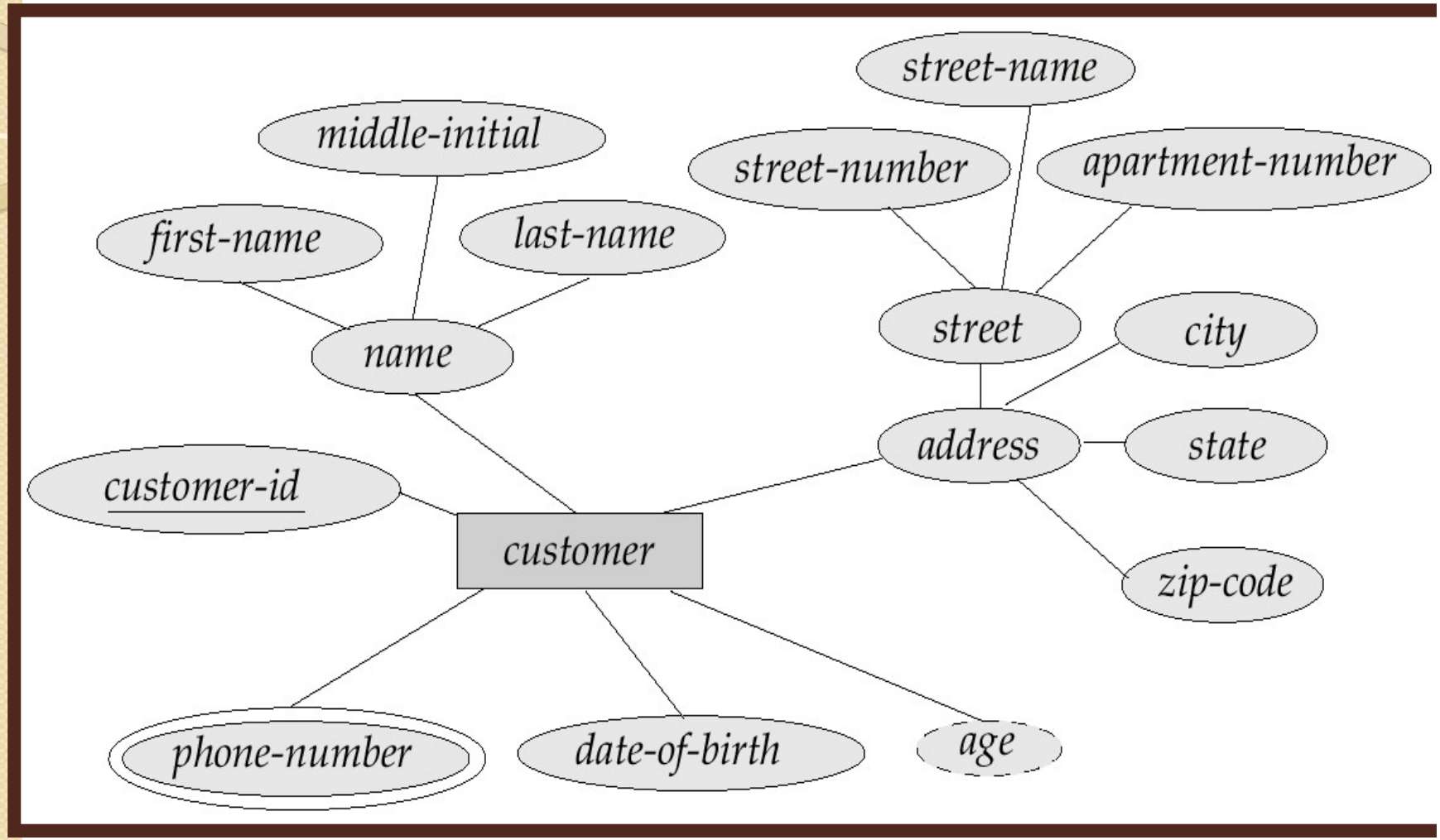


Many-To-Many Relationship



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

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



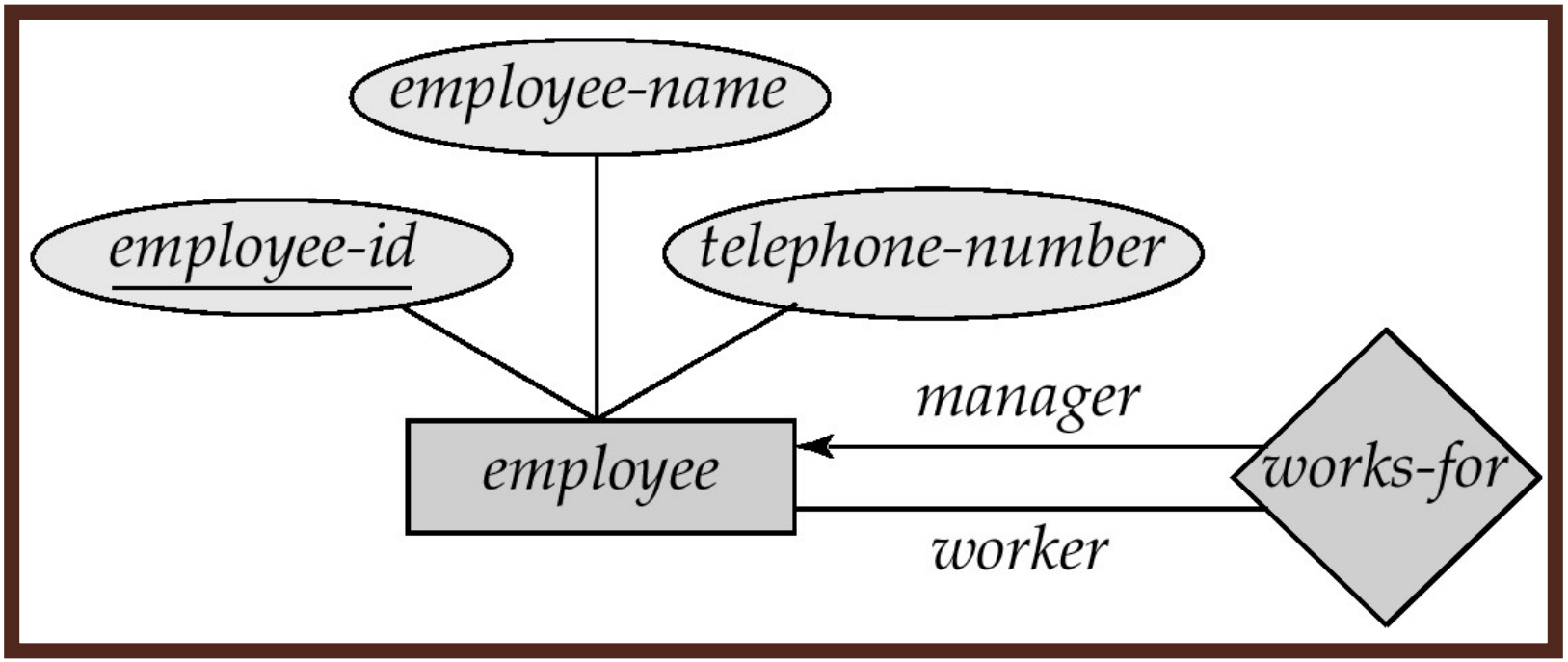
Name, address, street : composite attribute

Phone-number : Multivalued attribute

Age : Derived attribute

Roles

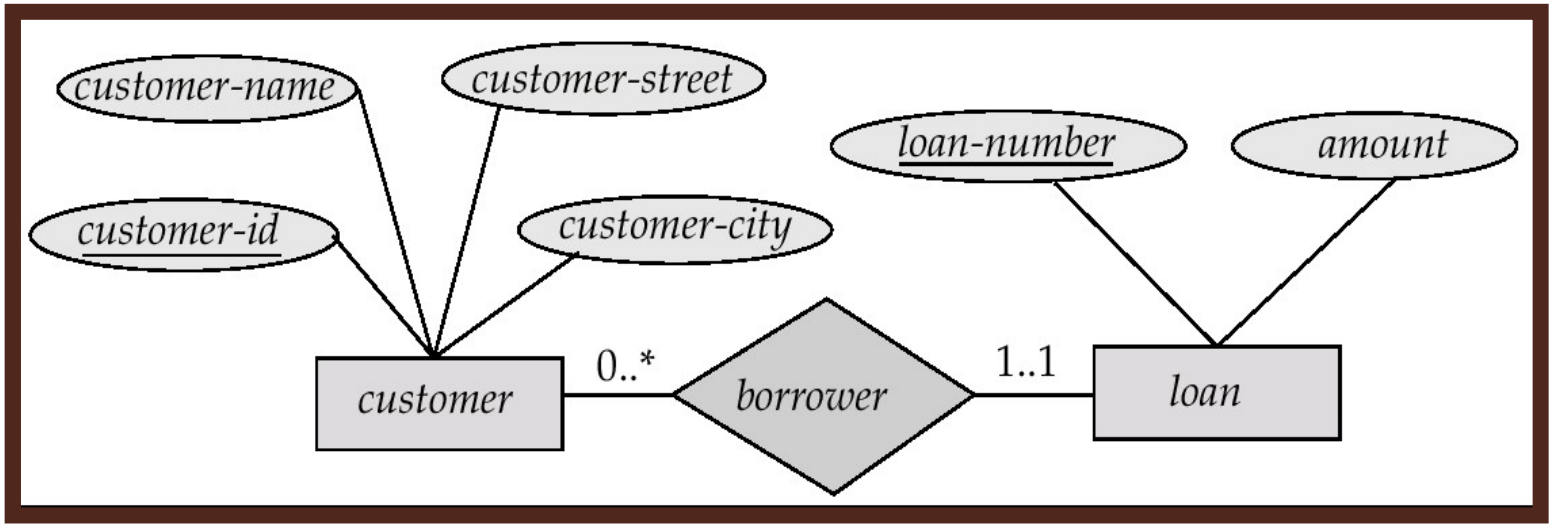
- Entity sets of a relationship need not be distinct
  - The labels "manager" and "worker" are called roles; they specify how employee entities interact via the works-for relationship set.
  - Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
  - Role labels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



Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints.
- Shown in the form l..h, where l = minimum cardinality

  h = maximum cardinality



A minimum value of 1 indicates total participation of the entity set in the relationship set.

- The relationship borrower is one to many from customer to loan and the participation of loan in borrower is total.
- Maximum value of 1 on both side = one to one R
- 1..* one edge customer and borrower = each customer must have at least one loan

Weak Entity Sets

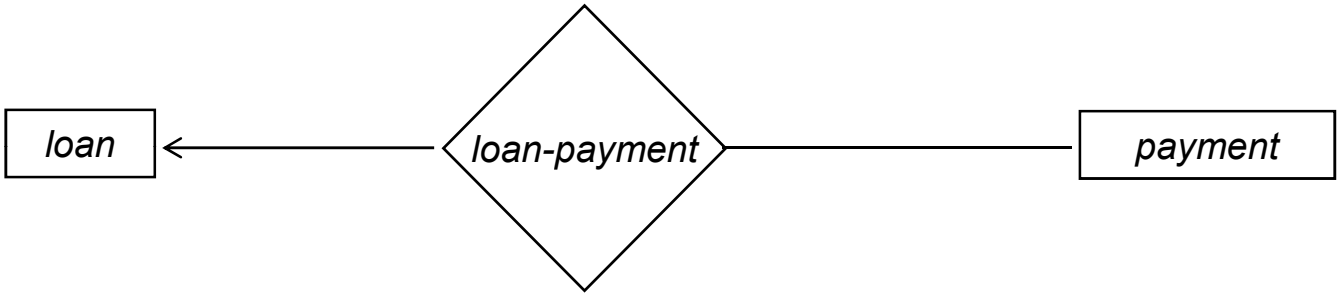
- An entity set that does not have a primary key is referred to as a *weak entity set*.
  - e.g. payment (payments for different loans may share the same payment number)
- An entity set that has a primary key is termed as a strong entity set.
- The existence of a weak entity set depends on the existence of an *identifying or owner 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.

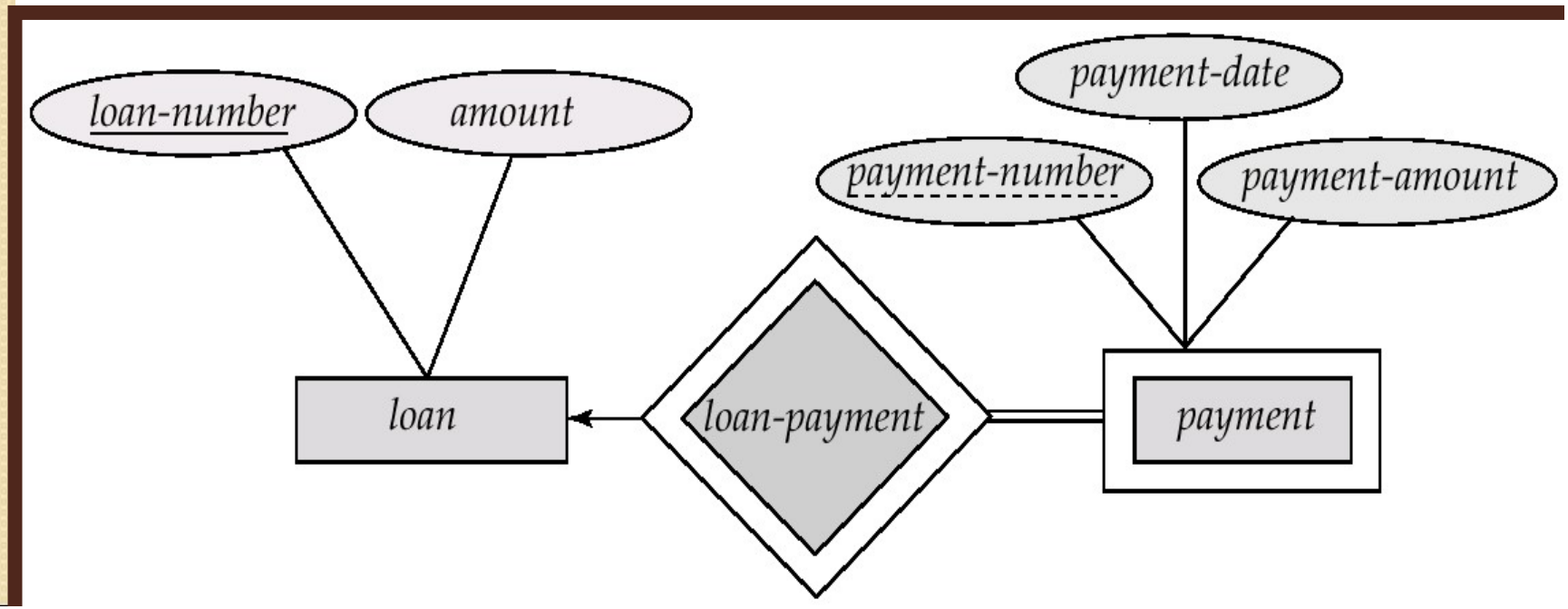
Existence Dependencies

- If the existence of entity *x* depends on the existence of entity *y*, then *x* is said to be *existence dependent* on *y*.
  - *y* is a *dominant entity* (in example below, *loan*)
  - *x* is a *subordinate entity* (in example below, *payment*)



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

- 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*)
- Double lines – indicate total participation of an entity in a relationship set
- Double diamond – identifying relationship

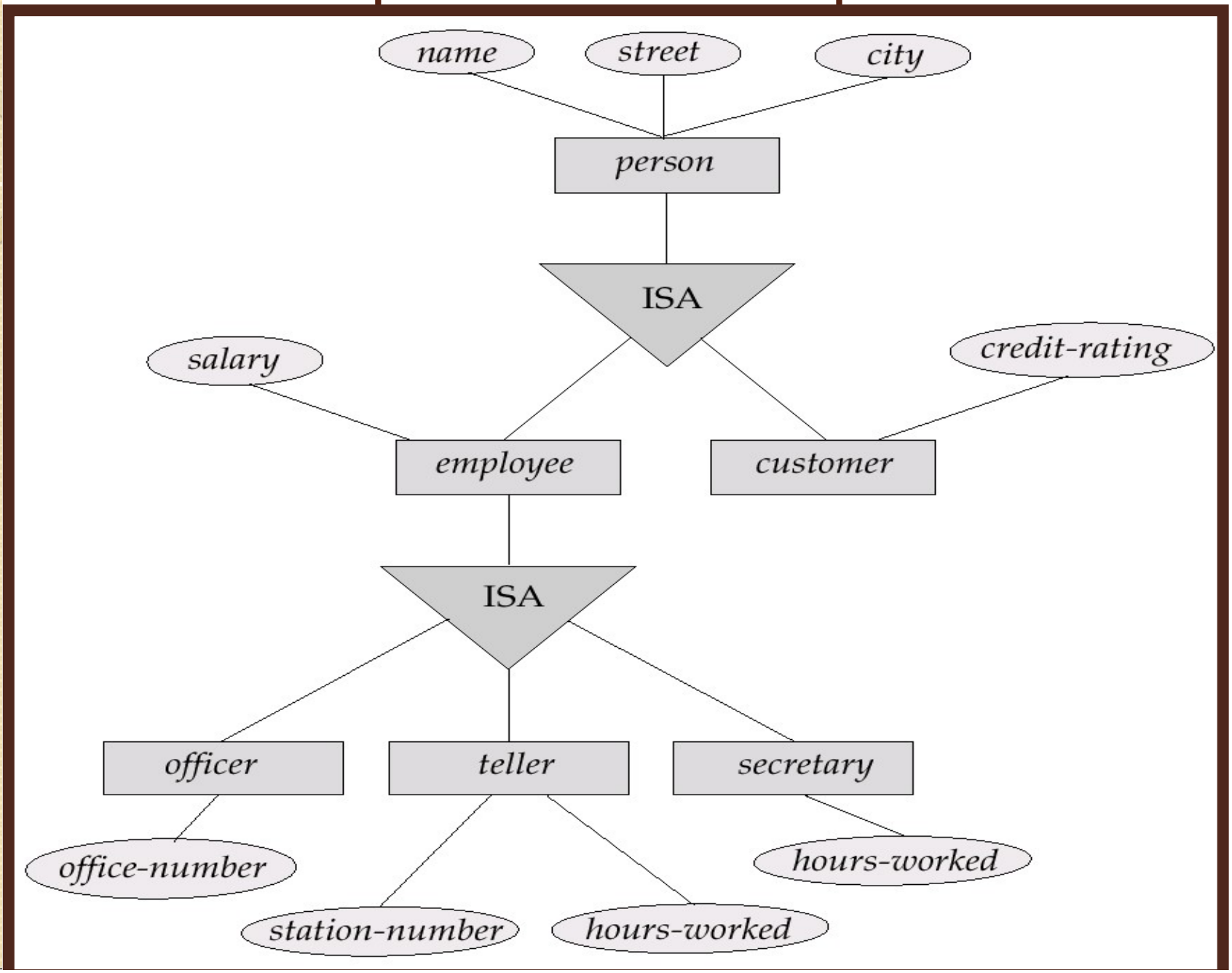


- The identifying relationship set should have no descriptive attributes.
- A weak entity set can participate in relationships other than the identifying relationship.
  - E.g. payment entity can participate in relationship with account entity, to show the account from which payment was made.
- A weak entity set may participate as owner in an identifying relationship with another weak entity set.
- Database designer may choose to express a weak entity set as a multivalued composite attribute of the owner entity set.

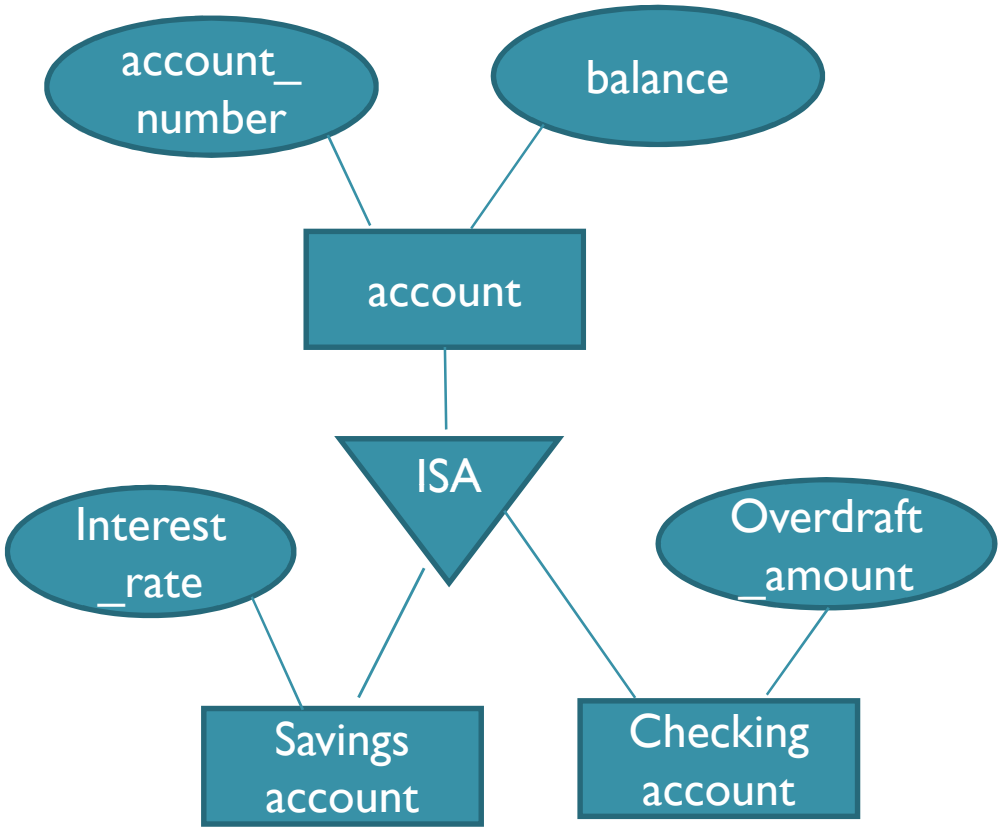
Specialization

- Top-down design process; we designate sub groupings within an entity set that are classifiable from other entities in the set.
- These sub groupings 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*).

Specialization Example



- Another 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.
- Difference in the two approaches may be characterized by their starting point and overall goal.
  - i.e. specialization stems from a single entity set; emphasized differences by creating distinct lower-level entity sets.

Specialization and Generalization

- 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

Attribute inheritance

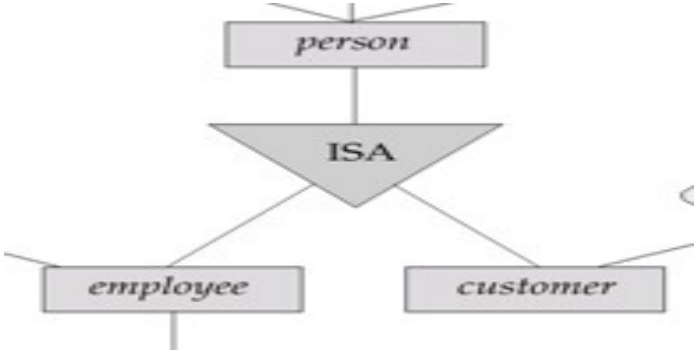
- Property of higher- and lower-level entities created by specialization or generalization is attribute inheritance.
- a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.
  - E.g. customer and emp inherit the attributes of person
  - E.g. officer, teller and secretary participate in works-for R, since employee participates in work-for R.
- E-R model follows Specialization / Generalization if
  - A higher-level entity set with attributes and relationships that apply to all its lower-level entity sets.
  - Lower level entity sets with distinctive features that apply only within particular lower-level entity set.

- In a hierarchy, a given entity set may be involved as lower-level entity set in only one ISA R; then entity set has only single inheritance.
- A given entity set may be involved as lower-level entity set in more ISA R; then entity set has only multiple inheritance and the resulting structure is said to be lattice.

Design Constraints on a Specialization/Generalization

- Constraint determines which entities can be members of a given lower-level entity set.
  - Condition-defined – membership is evaluated on basis of whether or not an entity satisfies an explicit condition or predicate
    - E.g. all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
    - All entities are evaluated on basis of same attribute i.e. age is said to be attribute-defined.
  - User-defined – not constrained by membership condition; rather db user assigns entities to given entity set.
    - E.g. after 3 months of employment , bank emp are assigned to one of four work teams

- 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
    - E.g. account_type can be either savings or checking account but cannot be both.
  - Overlapping
    - an entity can belong to more than one lower-level entity set
    - E.g. managers may participate in more than one of team entity set that are lower-level entity sets of employee.



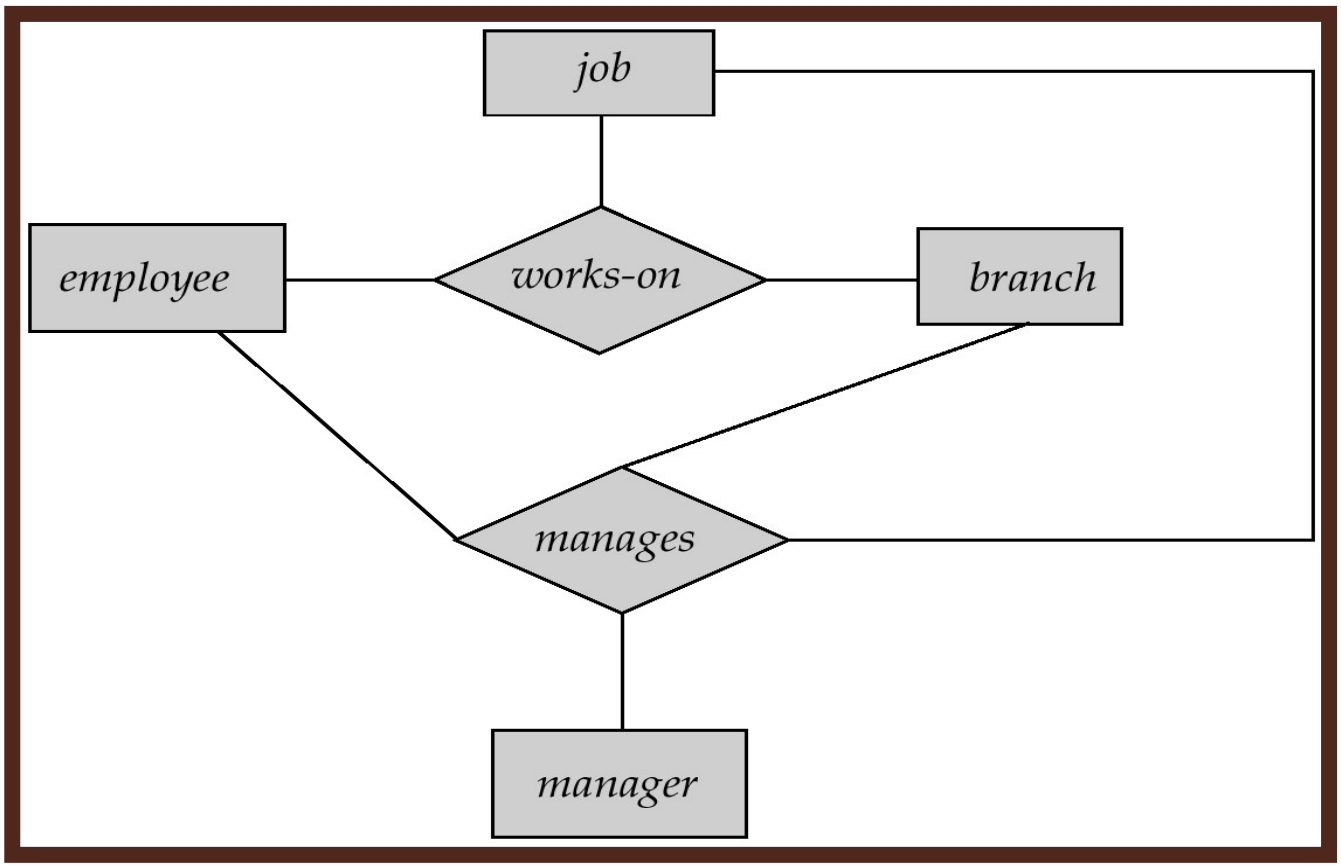
An employer can also be customer -overlapping.
Overlap is default case.

- 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** : Each entity must belong to one of the lower-level entity sets.
  - **partial**: Some higher-level entities may not belong to any of the lower-level entity sets.
    - Default case is partial
- Account generalization is total and disjoint
  - Coz all account entities must be either saving or checking account.
- Work team entity sets –partial and overlapping
  - Coz employees are assigned to team after 3 months of job; hence entity may not be member of any lower level entity set.
- Completeness and disjointness constraint don't depend on each other; therefore partial-disjoint and total-overlapping is possible.

- When total constraint is placed, an entity inserted into higher-level entity set must be inserted into at least one of lower-level entity set depending on condition.
- An entity deleted from higher-level entity set must be deleted from all associated lower-level entity set.

Aggregation

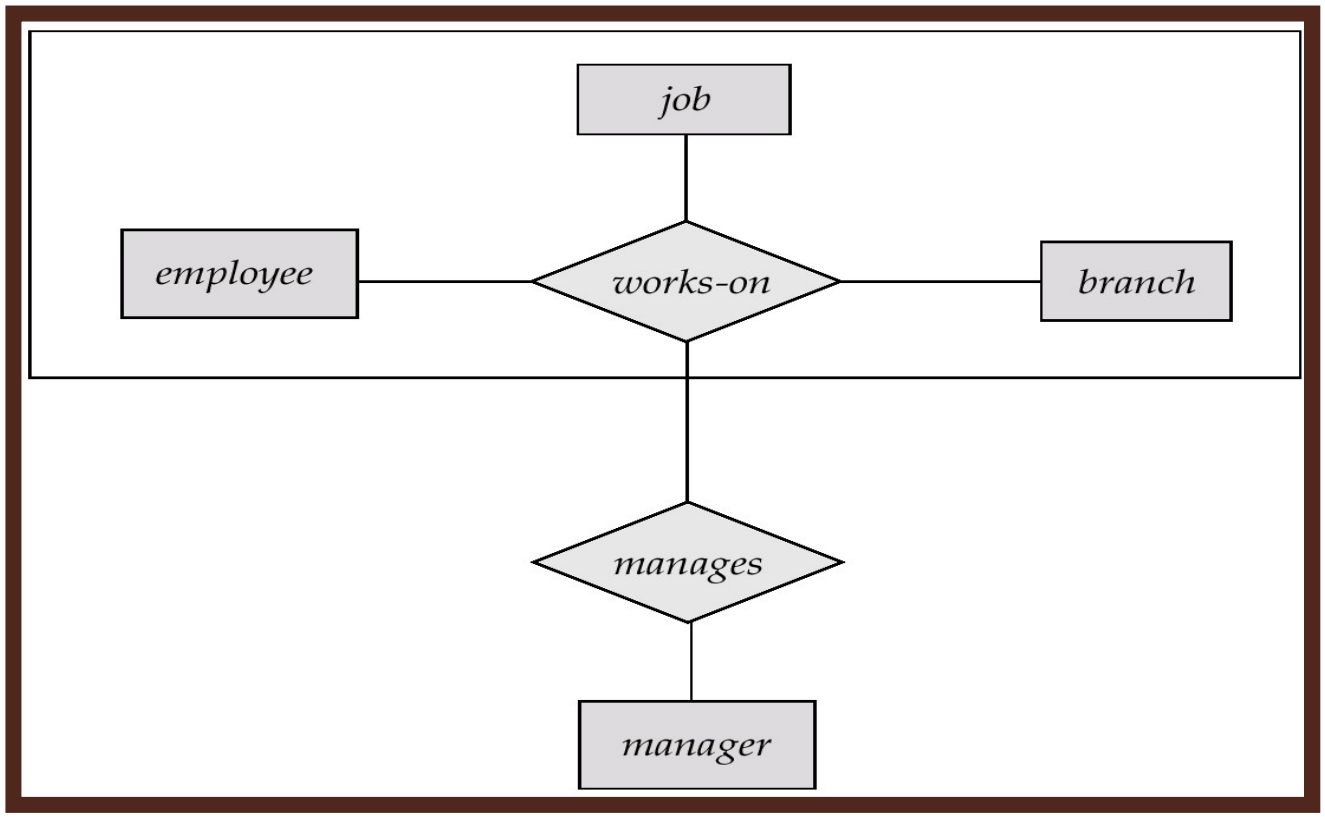
- Limitation of E-R model cannot express relationships among relationships.
- Consider the ternary relationship *works-on*
- Suppose we want to record managers for tasks performed by an employee at a branch



- 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
- If manager is attribute of works-on R
  - We can make multivalued attribute manager
  - Difficult logically as well as in execution cost, as to find emp-branch-job for which manager is responsible.
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract higher-level entity
    - E.g. emp, branch and job – higher-level entity set called works-on.
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

E-R Diagram With Aggregation

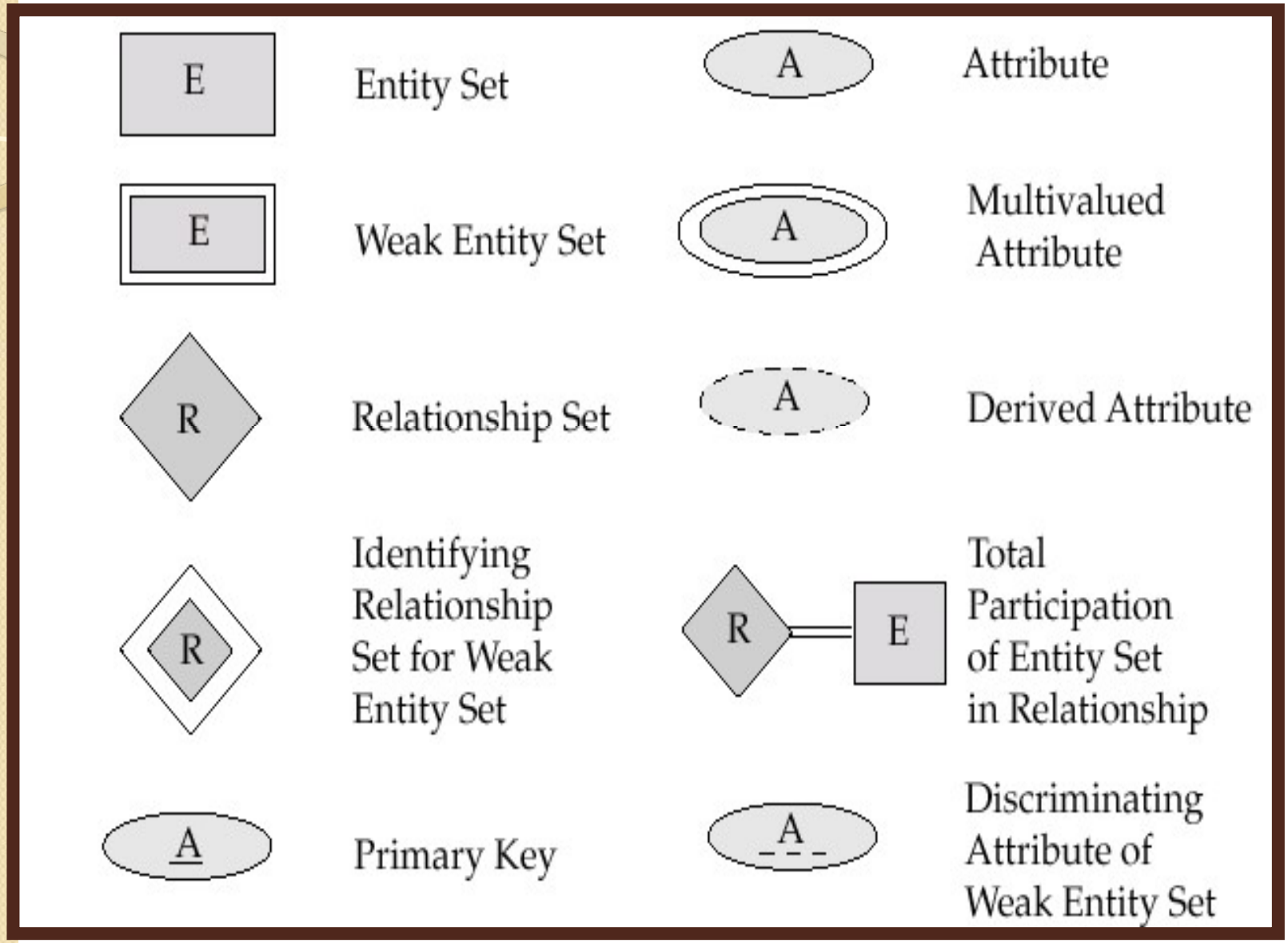
- Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager



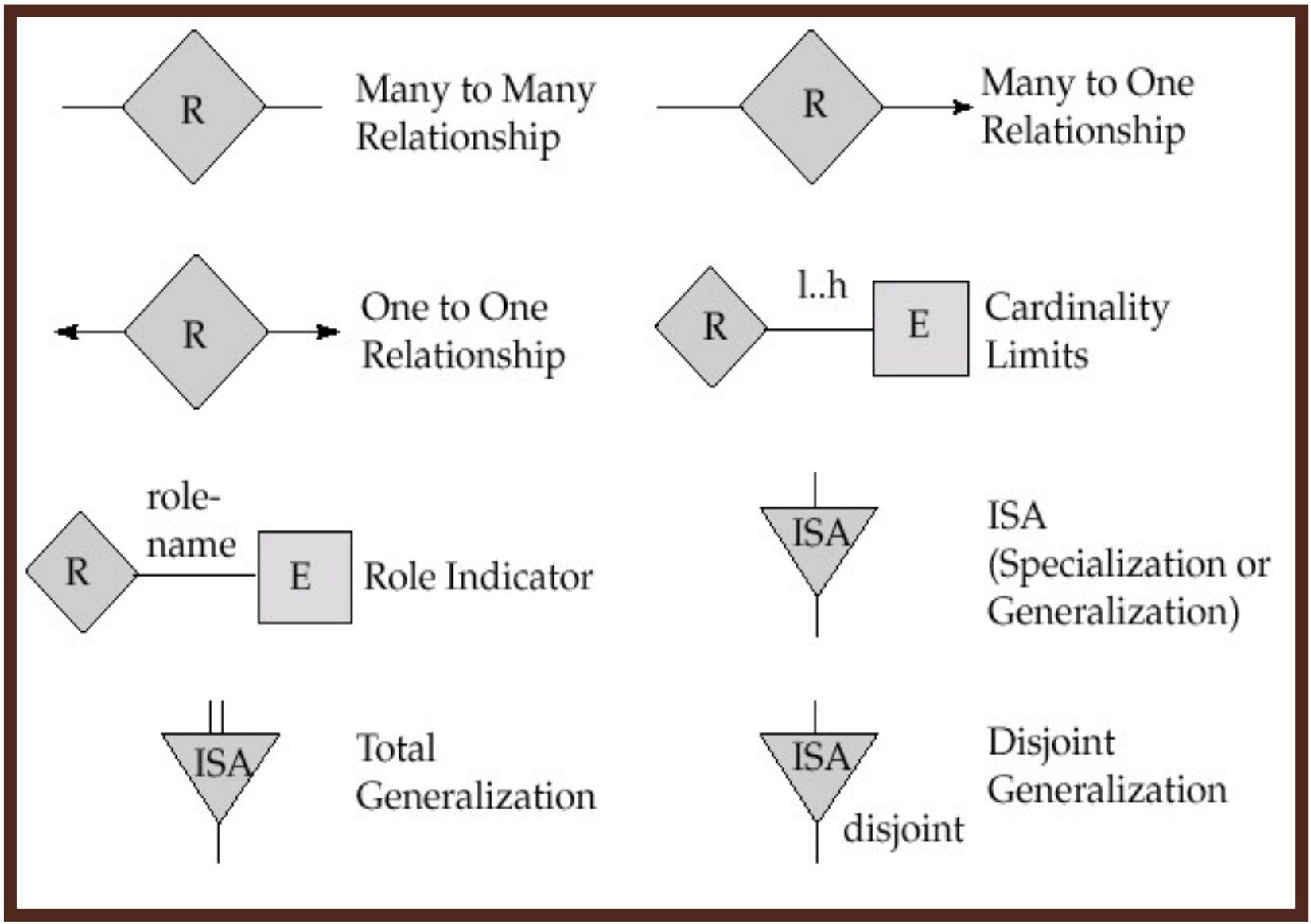
E-R Design Decisions

- 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

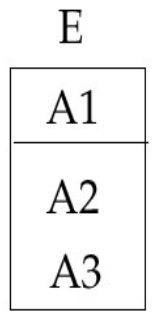


Summary of Symbols

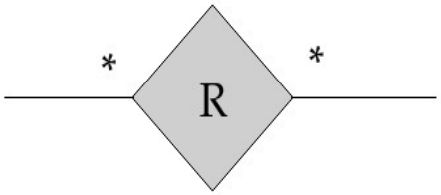


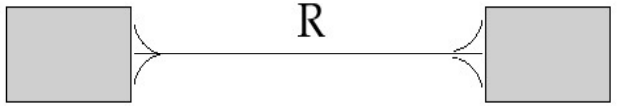
Alternative E-R Notations

Entity set E with attributes A1, A2, A3 and primary key A1

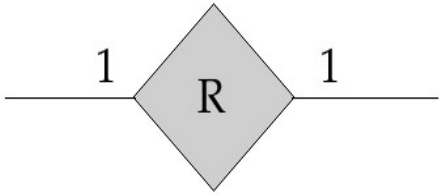


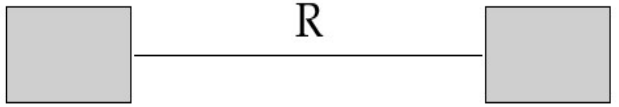
Many to Many Relationship



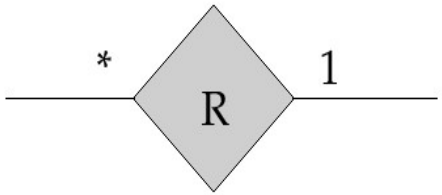


One to One Relationship





Many to One Relationship



Design phases

- Specification of user requirement
  - The initial phase of db design is to characterize fully the data needs of the db users.
  - Db designer needs to interact with domain experts and users to carry out this task.
- Conceptual design phase
  - Provides detail overview of enterprise
  - Designer chooses a data model
  - Translates these requirements into conceptual schema of db
  - E-R model is used to develop conceptual schema
  - High level description of data to be stored (ER model)
  - Schema specifies all Entity sets, relationships set, attributes and mapping constraints

- Designer
  - Examines the design to remove any redundant features
  - Confirms that all data requirements are satisfied.

- Specification of functional requirements
  - User describes the kind of operations that will be performed on data
  - Designer ensures schema meets functional requirements
- Abstract data model is moved to implementation of database
  - Logical-design phase – designer maps conceptual schema onto implementation of data model of db system.
    - Translation of ER diagram to a relational database schema (description of tables)
  - Physical-design phase – includes file organization and internal storage structures.
    - Done by the DB system

Database design for banking system

Data requirements

- Specification of user requirements may be based on interview with db users and on designer's own analysis of the enterprise.
- Characteristics of banking enterprise
  - The bank is organized into branches. Each branch is located in a particular city and is identified by a unique name. The bank monitors the assets of each branch.
  - Bank customers are identified by their customer-id values. The bank stores each customer's name, and the street and city where the customer lives. Customers may have accounts and can take out loans. A customer may be associated with a particular banker, who may act as a loan officer or personal banker for that customer.
  - Bank employees are identified by their employee-id values. The bank administration stores the name and telephone number of each employee, the names of the employee's dependents, and

the employee-id number of the employee's manager. The bank also keeps track of the employee's start date and, thus, length of employment.

- The bank offers two types of accounts – savings and checking accounts. Accounts can be held by more than one customer, and a customer can have more than one account. Each account is assigned a unique account number. The bank maintains a record of each account's balance, and the most recent date on which the account was accessed by each customer holding the account. In addition, each savings account has an interest rate and overdrafts are recorded for each checking account.
- A loan originates at particular branch and can be held by one or more customers. A loan is identified by a unique loan number. For each loan, the bank keeps track of the loan amount and the loan payments. Although a loan-payment number does not uniquely identify a particular payment among those for all the bank's loans, a payment number does identify a particular payment for specific loan. The date and amount are recorded for each payment.

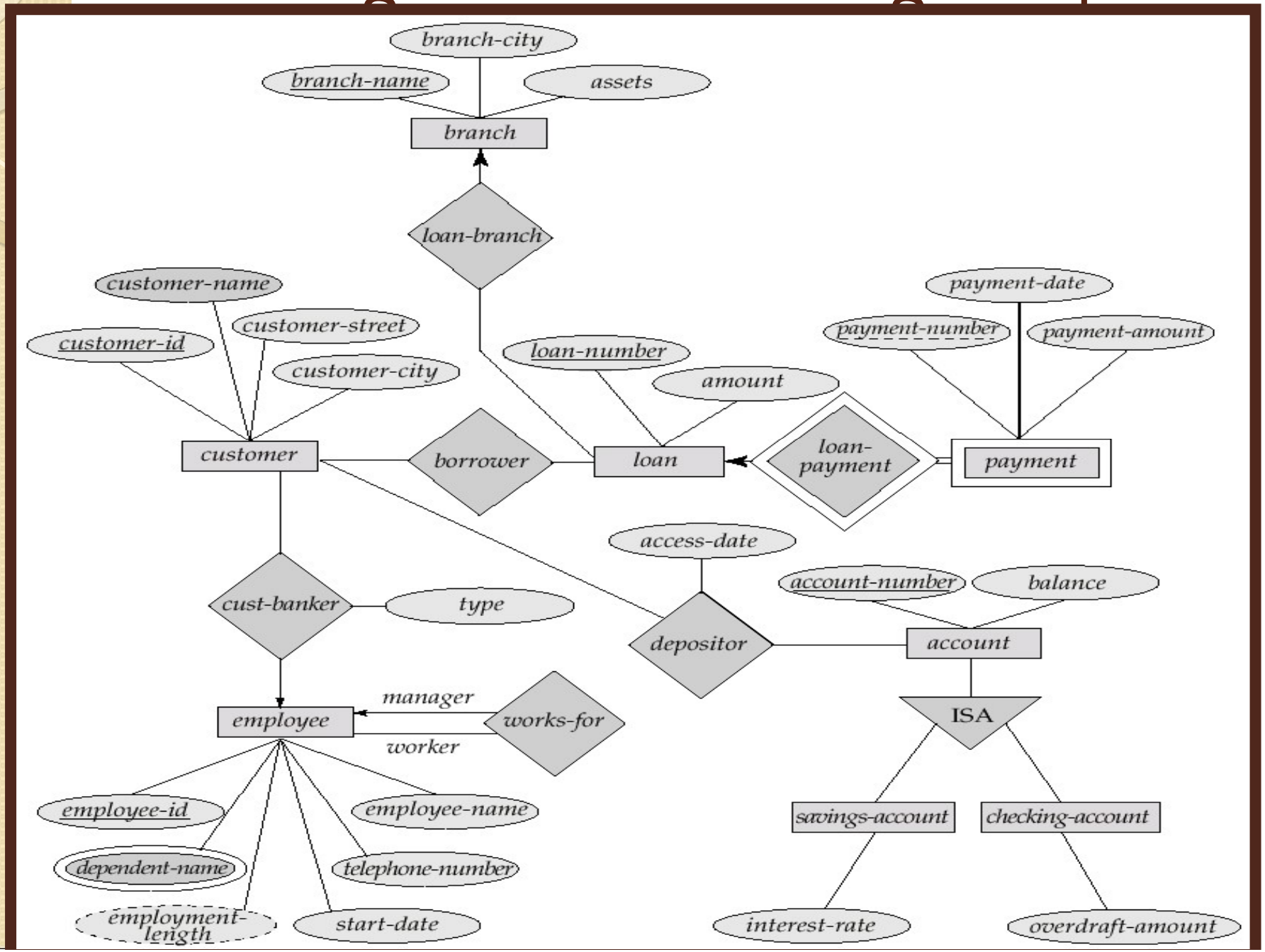
Entity sets Designation

- Branch : branch-name, branch-city and assets
- Customer : customer-id, customer-name, customer-street, customer-city. A possible additional attribute is banker-name
- Employee : employee-id, employee-name, telephone-number, salary and manager.
  - Multivalued attribute dependent-name.
  - Base attribute start-date and derived attribute employement-length
- Two account : savings-account and checking-account
  - Account :account-number and balance
  - Savings-account : interest-rate
  - Checking-account : overdraft-amount
- Loan : loan-number, amount, originating branch
- Weak entity set loan-payment : payment-number, payment-date and payment-amount.

Relationship sets designation

- Borrower : M-M R between customer and loan.
- Loan-branch : M-1 R
- Loan payment : 1-M R from loan to payment
- Depositor : access-date attribute; M-M R between customer and account
- Cust-banker : type attribute; M-1 R from customer to banker. i.e. bank employee can advise one or more customers.
- Works-for : relationship set between employee entities with role indicators manager and worker;
  - Employee works for only one manager and that a manager supervises one or more employees.

E-R Diagram for a Banking Enterprise



Reduction of an E-R Schema to Tables

- 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 Strong Entity Sets as Tables

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

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

E.g. of loan table

- D1 – set of all loan numbers
- D2 – set of all amounts
- Any row of the loan table must consists of (v1, v2); where v1is in set D1 and v2 is in set D2.
- In general, the loan table will contain only a subset of the set of all possible rows.
- Set of all possible rows of loan as Cartesian product of D1 and D2, denoted by

$$D1 * D2$$

In general , for table having n columns, Cartesian product would be

$$D1 * D2 * \ldots * Dn$$

Representing Weak Entity Sets

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

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

Representing Relationship Sets as Tables

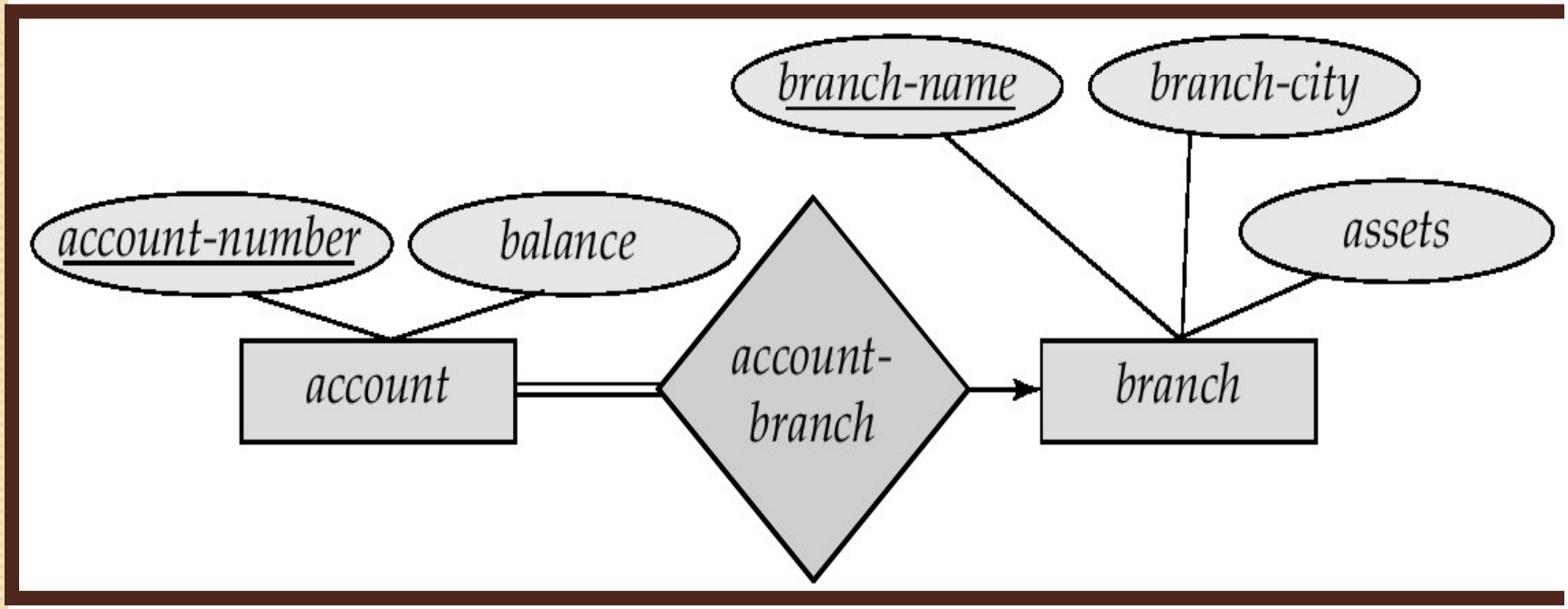
- A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- E.g.: table for relationship set *borrower*.

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

Redundancy of Tables

- 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-name* to the entity set *account*

  Account : account-number, balance, and branch-name
- *Branch : branch-name, branch-city and assets.*



Redundancy of Tables

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is *partial* on the many side, replacing a table by an extra attribute in the relation corresponding to the "many" side could result in null values
- The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - E.g. The *payment* table already contains the information that would appear in the *loan-payment* table (i.e., the columns loan-number and *payment-number*).

Composite and Multivalued Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - E.g. given entity set *customer* with composite attribute *name* with component attributes *first-name* and *last-name* the table corresponding to the entity set has two attributes *name.first-name* and *name.last-name*
- 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
  - E.g. Multivalued attribute *dependent-names* of *employee* is represented by a table *employee-dependent-names*(*employee-id, dname*)
  - Each value of the multivalued attribute maps to a separate row of the table EM
    - E.g., an employee entity with primary key John and dependents Johnson and Johndotir maps to two rows: (John, Johnson) and (John, Johndotir)

Representing Specialization as Tables

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

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

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

Representing Specialization as Tables

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

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

  - If specialization is total, table for generalized entity (*person*) not required to store information
    - Can be defined as a "view" relation containing union of specialization tables
  - Drawback: street and city may be stored redundantly for persons who are both customers and employees
    - If generalization is not complete, some person were neither customer nor employee; such person could not be represented.

Relations Corresponding to Aggregation

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

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