Lecture 10
Entity-relationship diagram

Subject Lecturer: Kevin K.F. YUEN, PhD.

Acknowledgement: Slides were offered from Prof. Ken Yiu.
Some parts might be revised and indicated.

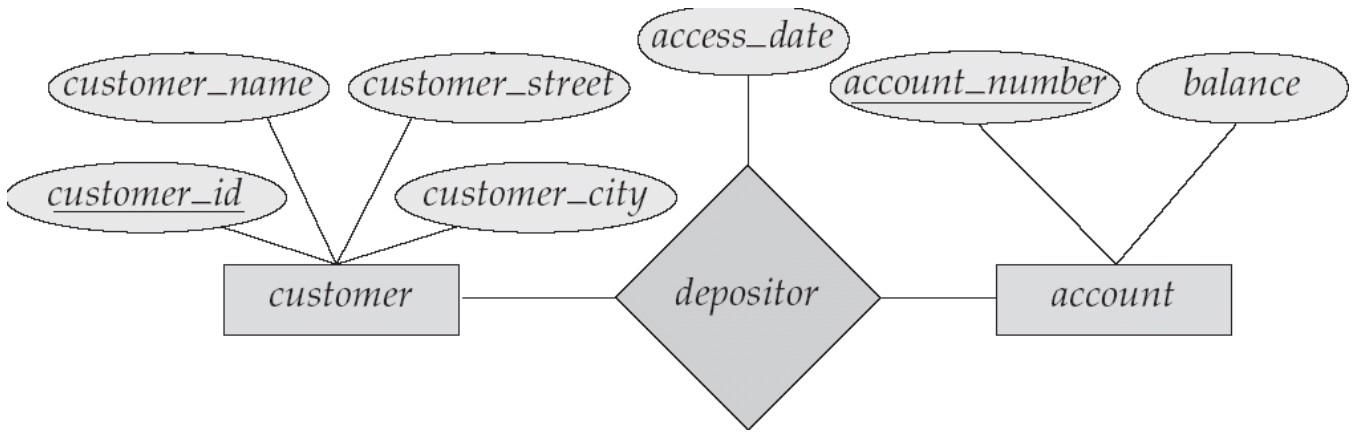
Slides content modified from
Database Systems Concepts

Outline

- Entity relationship (ER) diagram
- Extended features of ER diagram
- Design issues and decisions
- How to convert a ER diagram to relational schemas?

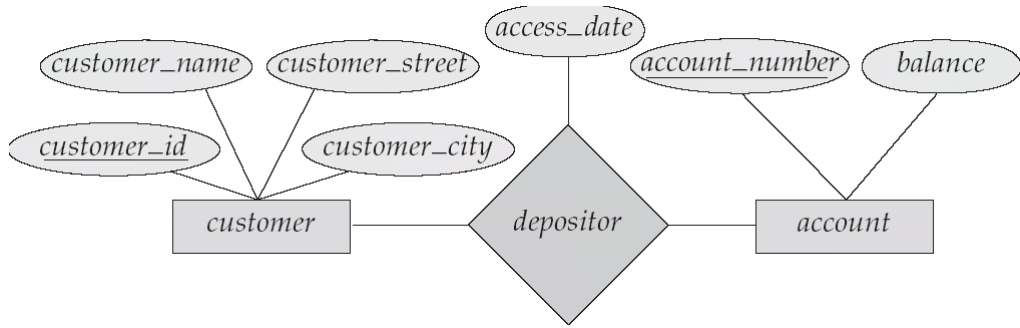
Entity-Relationship (ER) Diagram

- ER diagram is used to design the schema of a *database*
  - A collection of **entities**
  - **Relationship** among entities
  - **Attributes** of entities and relationships
- *Example*: part of the ER diagram for the bank database
  - *customer* and *account* are **entities**
  - *depositor* is a **relationship** between *customer* and *account*
  - *customer_name* is an **attribute** of *customer*

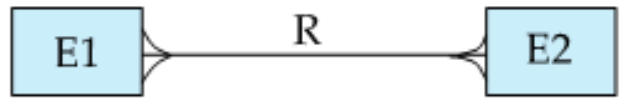


Notations for ER diagrams

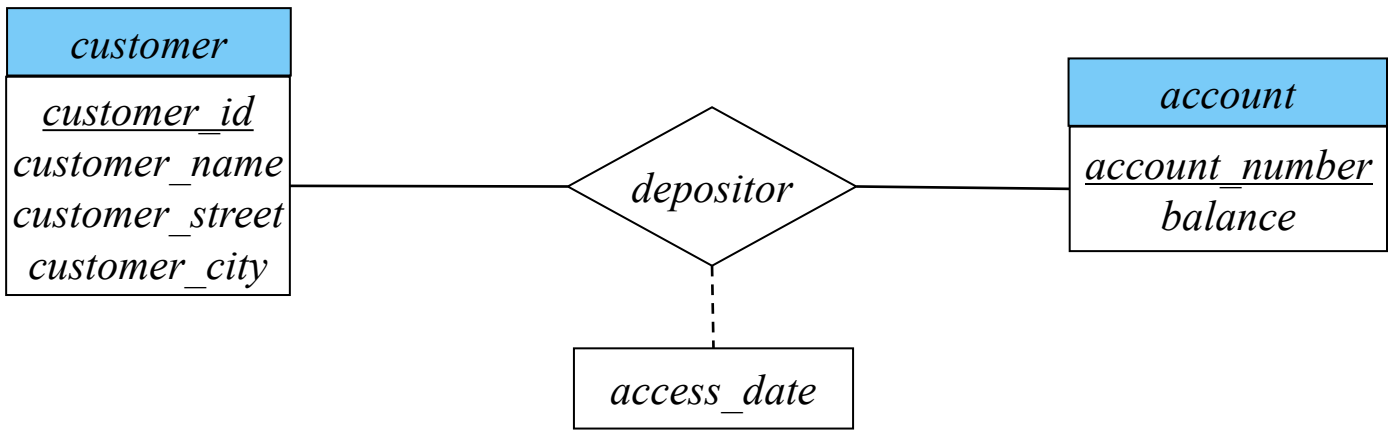
- Chen's notation
  - The traditional notation
  - Used in our lecture slides



- IDEF1X
  - Also called the crow's foot notation

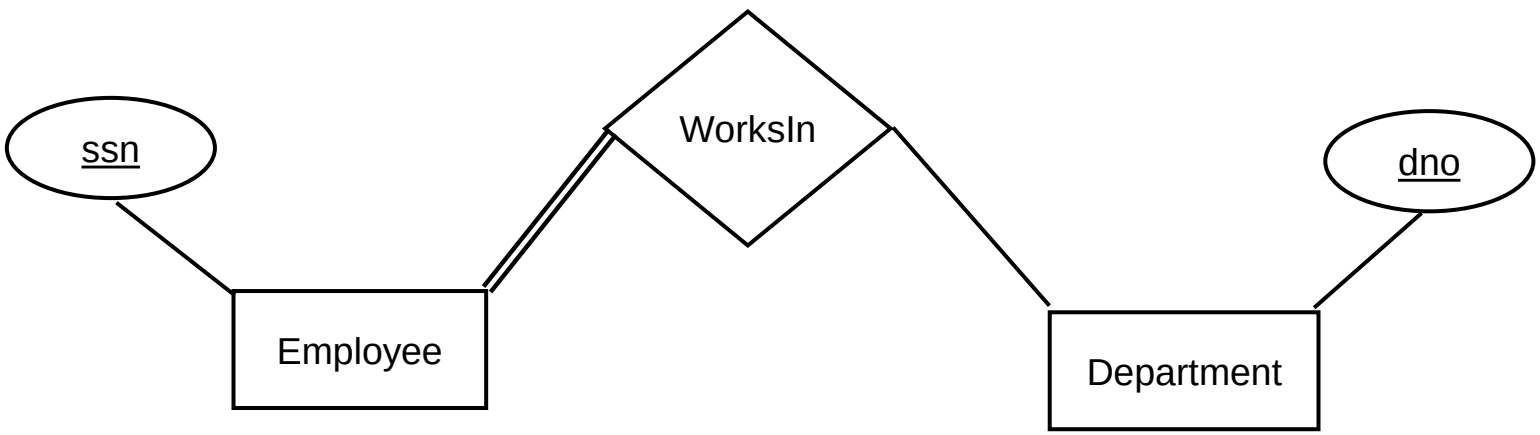


- UML notation
  - Express attributes in a more compact way



Examples of ER diagram drawing tools

- draw.io
  - https://drawio-app.com/entity-relationship-diagrams-with-draw-io/
- Smartdraw
  - https://www.smartdraw.com/entity-relationship-diagram
- DBDiagram.io
  - https://dbdiagram.io/home
- Microsoft Powerpoint



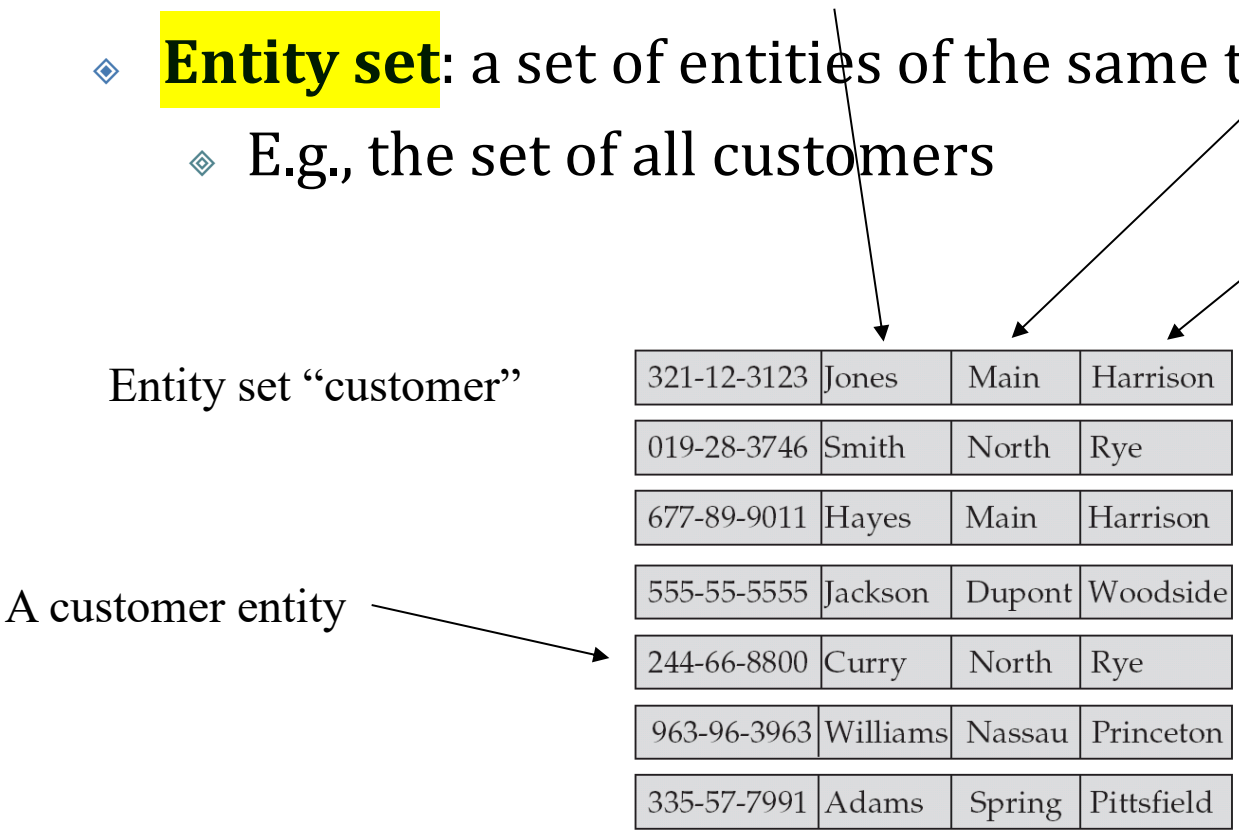
Objectives

- At the end of this lecture, you should be able to:
  - *Draw* a correct ER diagram for a given scenario
    - to capture all the requirements
  - *Explain* to others why your ER diagram is correct
  - *Compare* two different ER diagrams (for the same scenario)

Entity

- **Entity**: an object that is distinguishable from other objects
  - It may have *attributes*
  - E.g., a customer is an entity:

    its attributes are name and address (street & city)
- **Entity set**: a set of entities of the same type
  - E.g., the set of all customers



Keys for Entity Sets

- The designer selects one of the candidate key(s) to be the **primary key**
  - A *candidate key* of an entity set is a **minimal** super key
  - A *super key* of an entity set is a set of attribute(s) whose value(s) **uniquely determine** each entity

- Example: consider the schema
  customer = (*customer_id, customer_name, customer_street, customer_city*)
  - (*customer_id, customer_name*) is a super key
  - *customer_id* is a candidate key
  - We choose *customer_id* as the primary key

customer = (<u>*customer_id*</u>, *customer_name, customer_street, customer_city*)

Relationship Sets

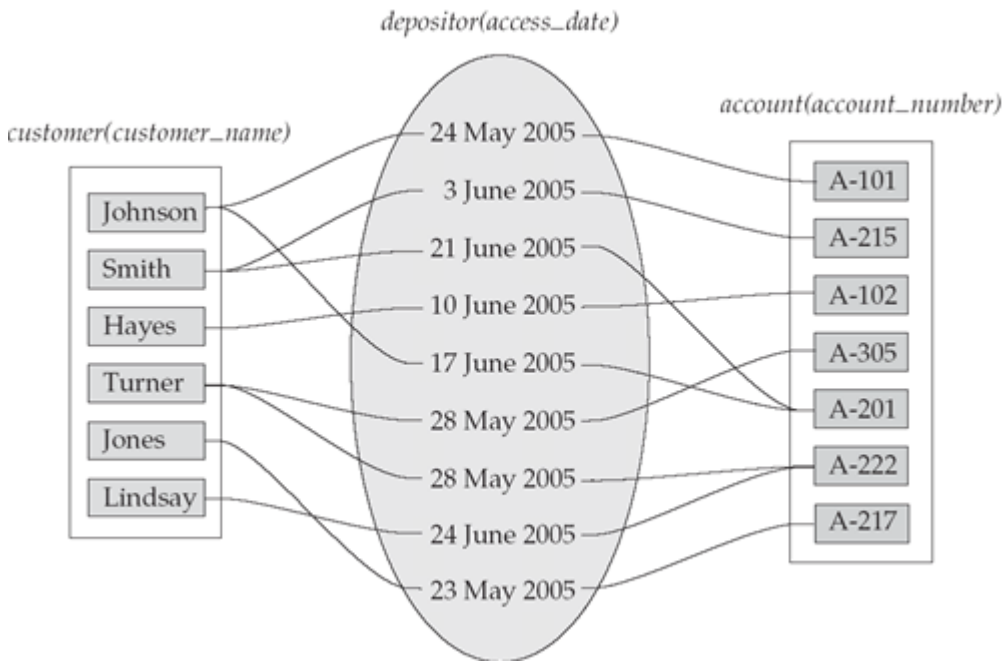
- **Relationship**: an association among entities

Hayes ——————— A-102

customer entity *account* entity

  - It may have attributes (e.g., *access-date)*

- **Relationship set**: a collection of relationships
  - E.g., *depositor*

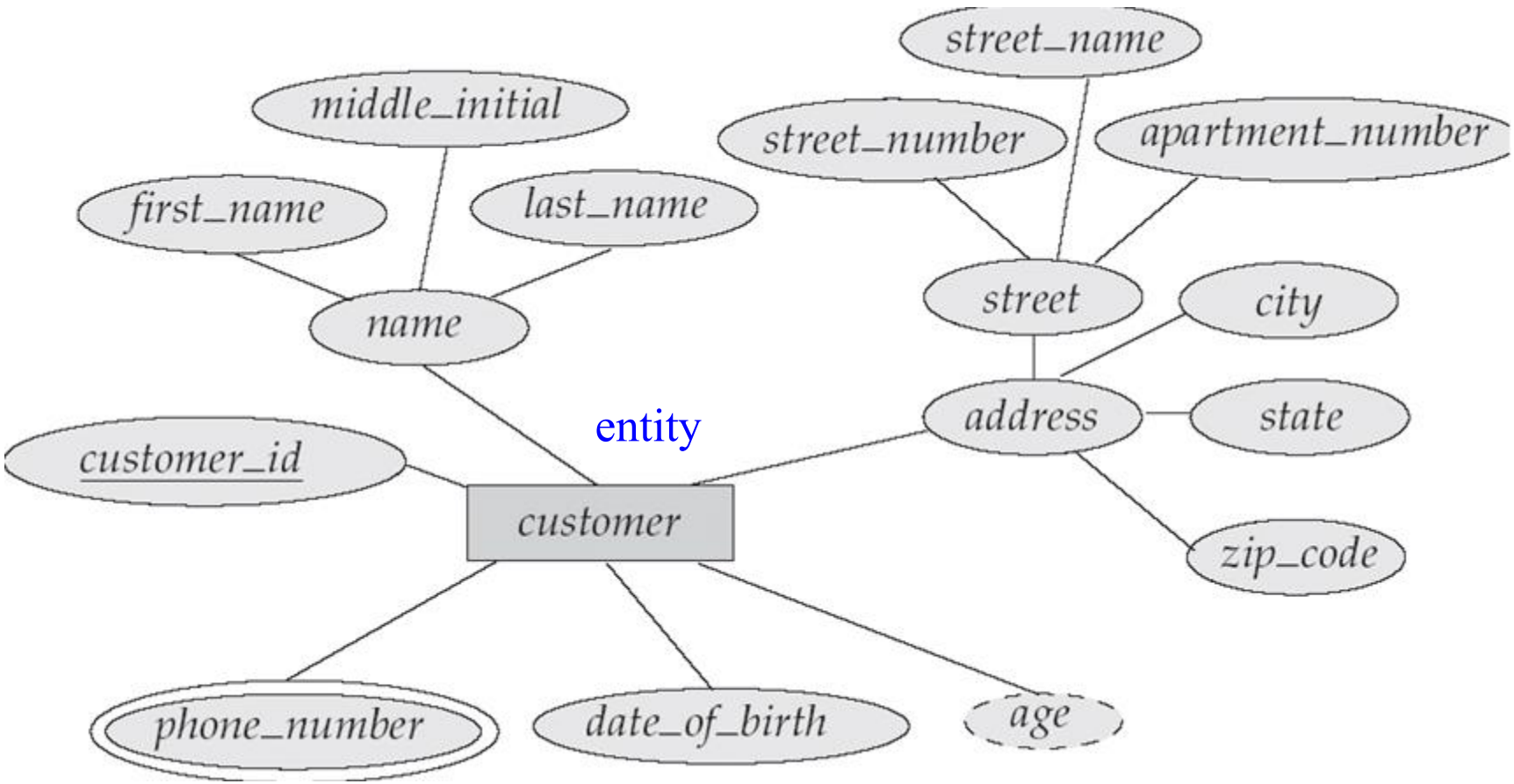


Attributes

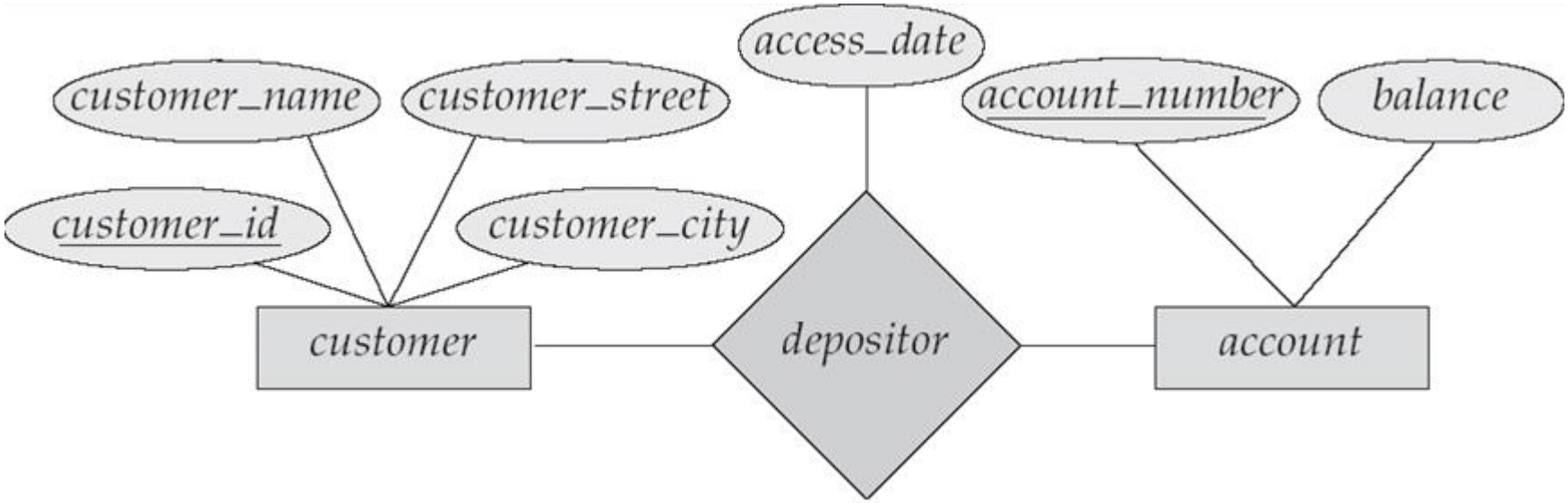
- Attributes: description/property of an entity
  - *customer* = (*customer_id*, *customer_name*, *customer_street*, *customer_city*)
  - *loan* = (*loan_number*, *amount*)

- *Simple* attribute
  - E.g., amount
- *Composite* attribute
  - It contains multiple simple attributes
  - E.g., address can be decomposed into customer_street and customer_city
- *Multi-valued* attributes
  - E.g., phone_numbers (a customer may have several numbers)
- *Derived* attributes
  - It can be computed from other attributes
  - E.g., age (derived from date_of_birth)

ER Diagram with Attributes



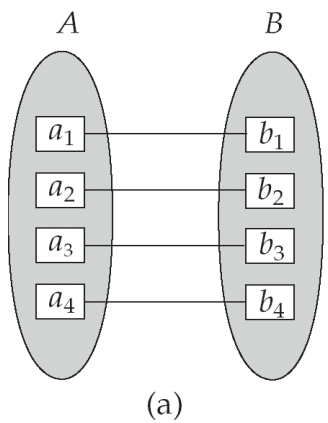
ER Diagrams



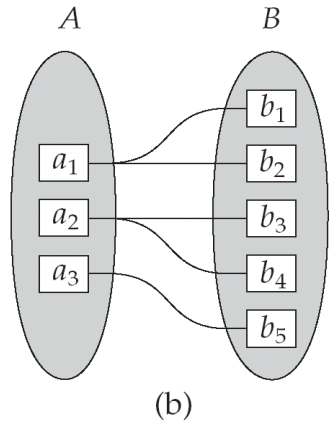
- Rectangles represent entity sets
  - Lines link entity sets to attributes
- Diamonds represent relationship sets
  - Lines link relationship sets to entity sets or attributes
- Ellipses represent attributes
- Underline indicates primary key attributes

Cardinality Constraints

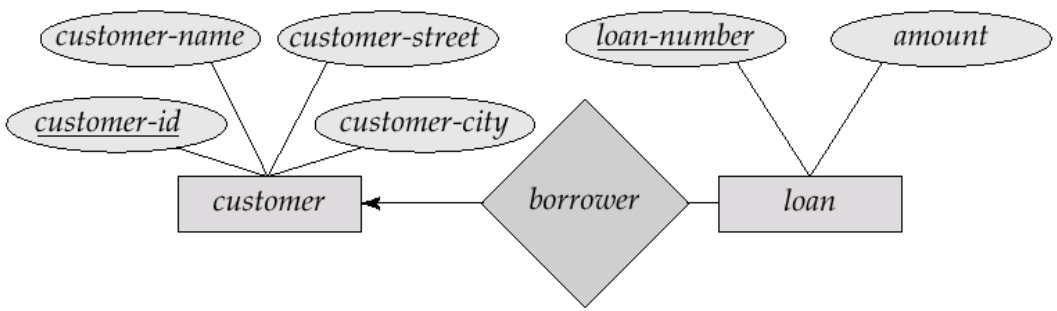
- Express cardinality constraints between the relationship set and the entity set by:
  - a directed line (→): "one" (possibly 0)
  - an undirected line (—): "many" (possibly 0)
- 4 types of cardinality constraints
- (1) One-to-one relationship
- (2) One-to-many relationship
  - E.g., a loan is associated with at most one customer; a customer can have several loans



one-to-one



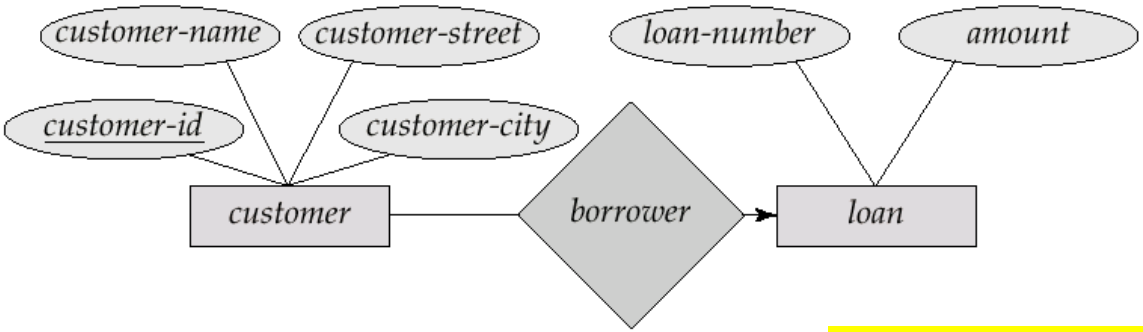
one-to-many



Cardinality Constraints (Cont')

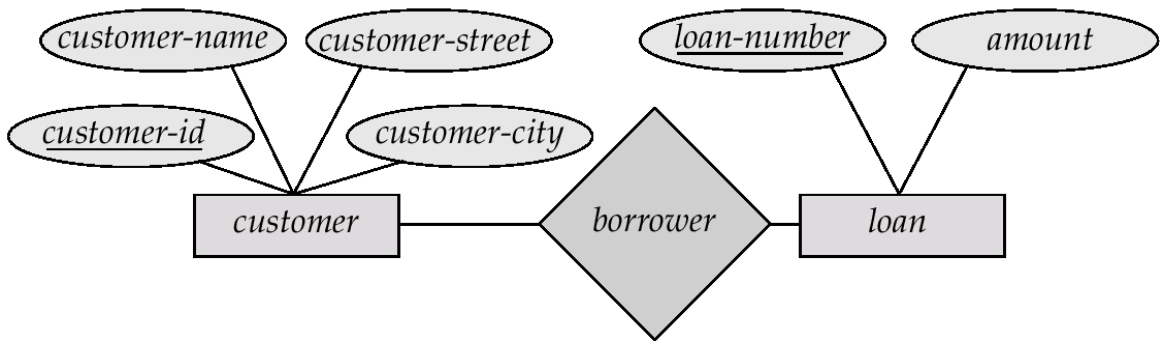
(3) Many-to-one relationship:

E.g., a loan can be associated with several customers;
a customer has at most one loan

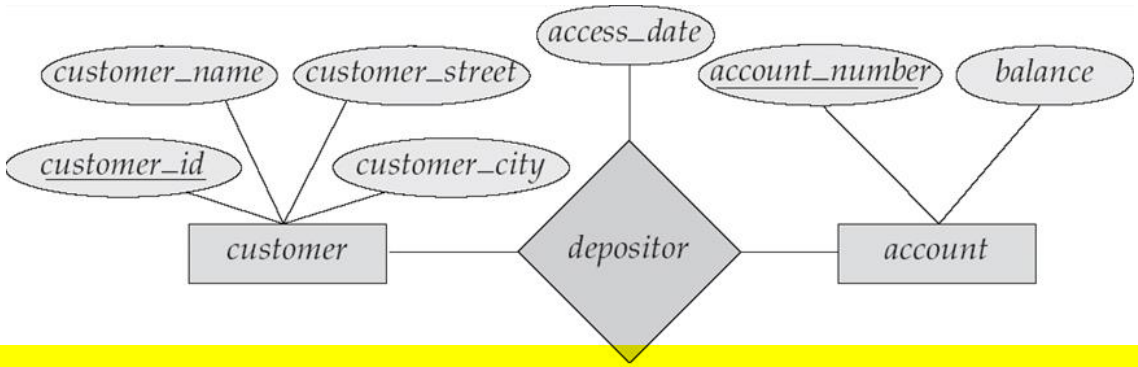


(4) Many-to-many relationship:

E.g., a customer can have several loans;
a loan can be associated with several customers



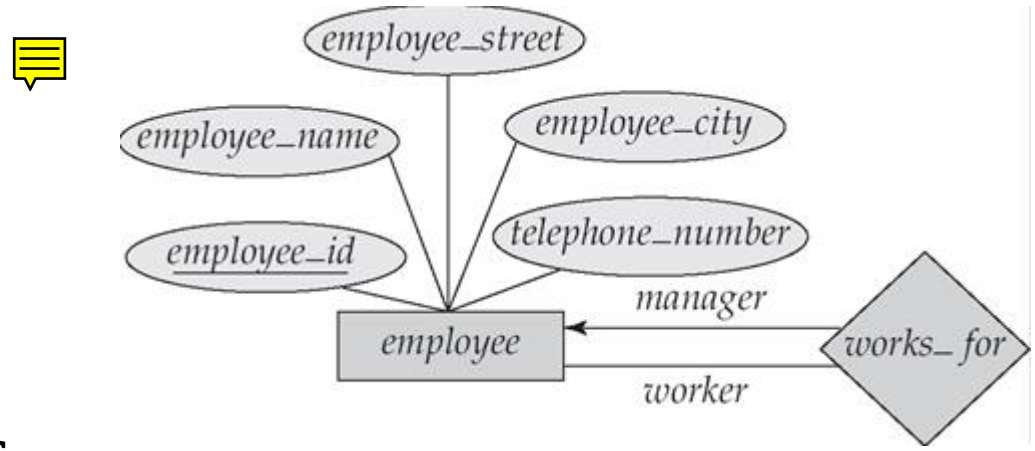
Keys for Relationship Sets



- 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*
    - *customer_id* is the primary key of *customer*
    - *account_number* is the primary key of *account*
- To decide the candidate key, we must consider the mapping cardinality of the relationship set
  - Examples of candidate key
    - One-to-one mapping: (*account_number*)
    - Many-to-many mapping: (*customer_id, account_number*)

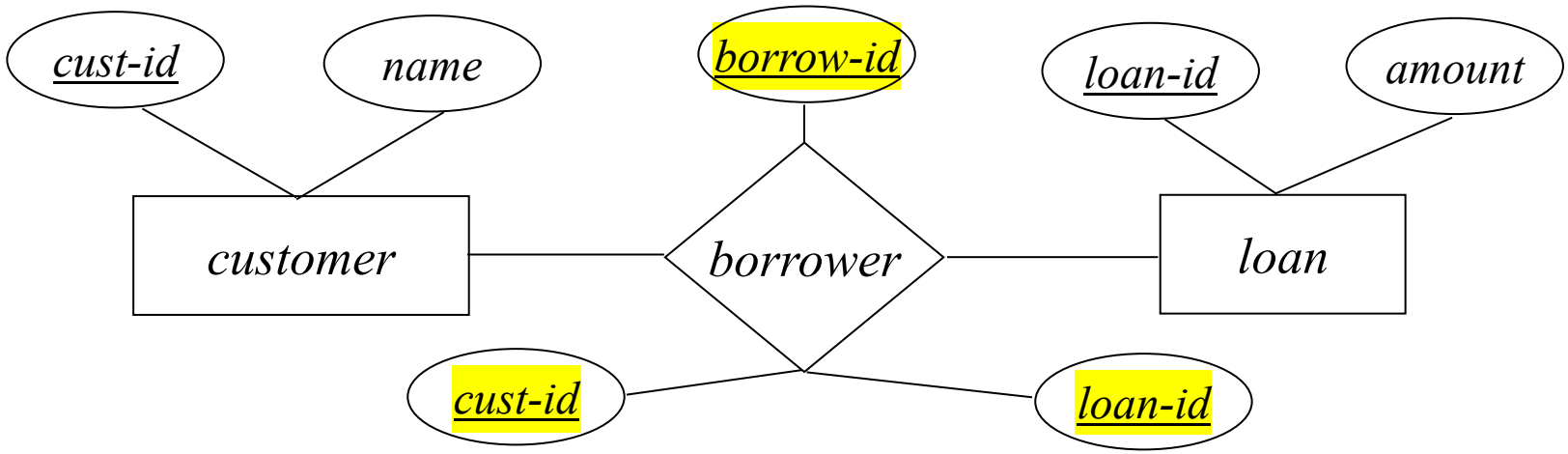
Roles

- Scenario
  - Some employees are workers, some employees are managers
  - How to express the "works_for" relationship?

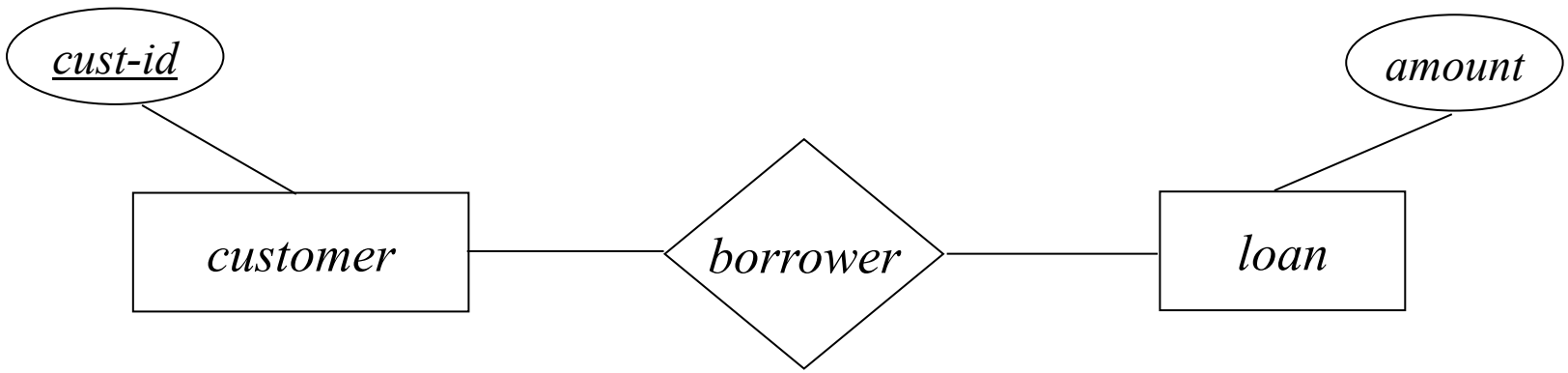


- Role labels
  - Clarify semantics of the relationship
  - Indicated in ER diagrams by labeling the lines that connect diamonds to rectangles
  - E.g., in the above diagram, the role labels are "manager" and "worker"

[Exercise] Find the mistakes here

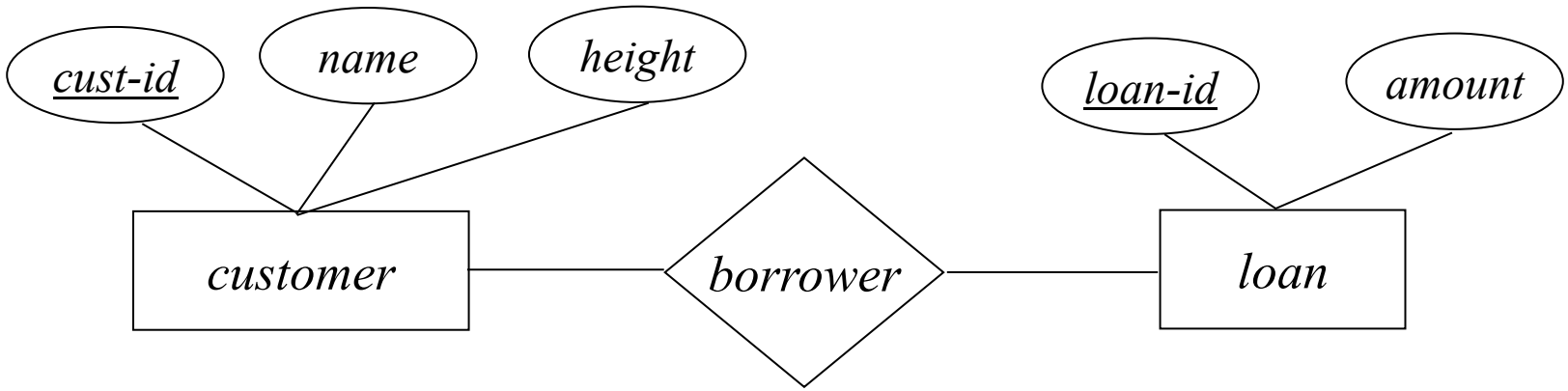


- **IDs from entities**
- **incorrect to "make" a primary key for borrower**

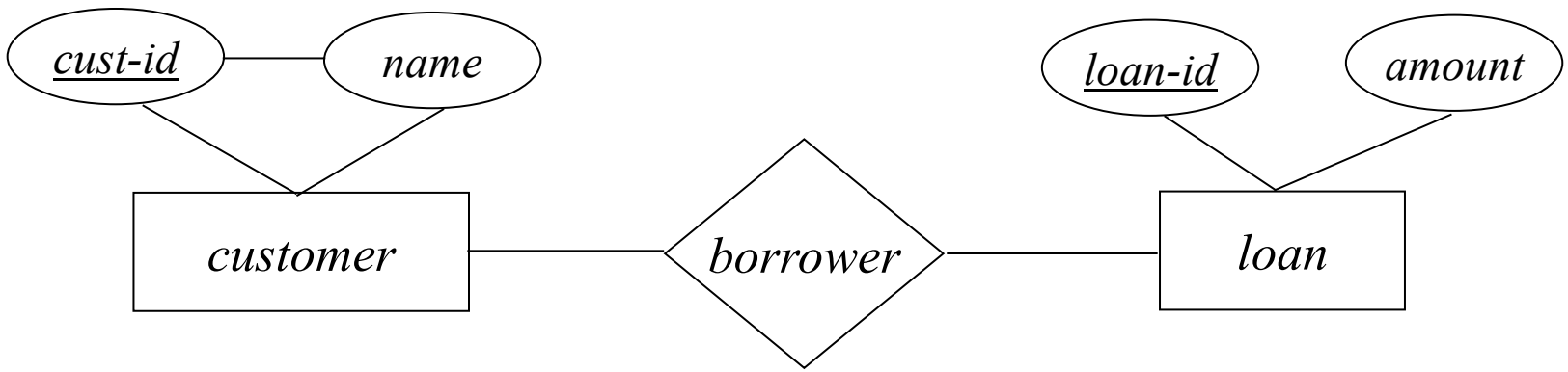


- **customer missing other attributes**
- **loan missing key**

[Exercise] Find the mistakes here



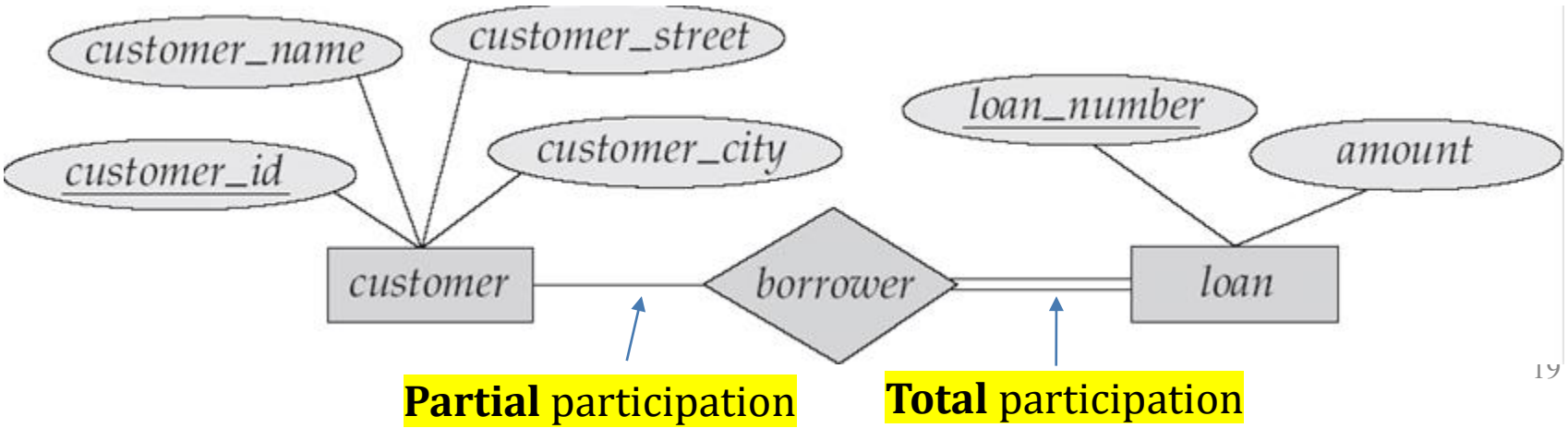
- irrelevant attribute in customer



- incorrect to link attributes

Participation of an Entity Set in a Relationship Set

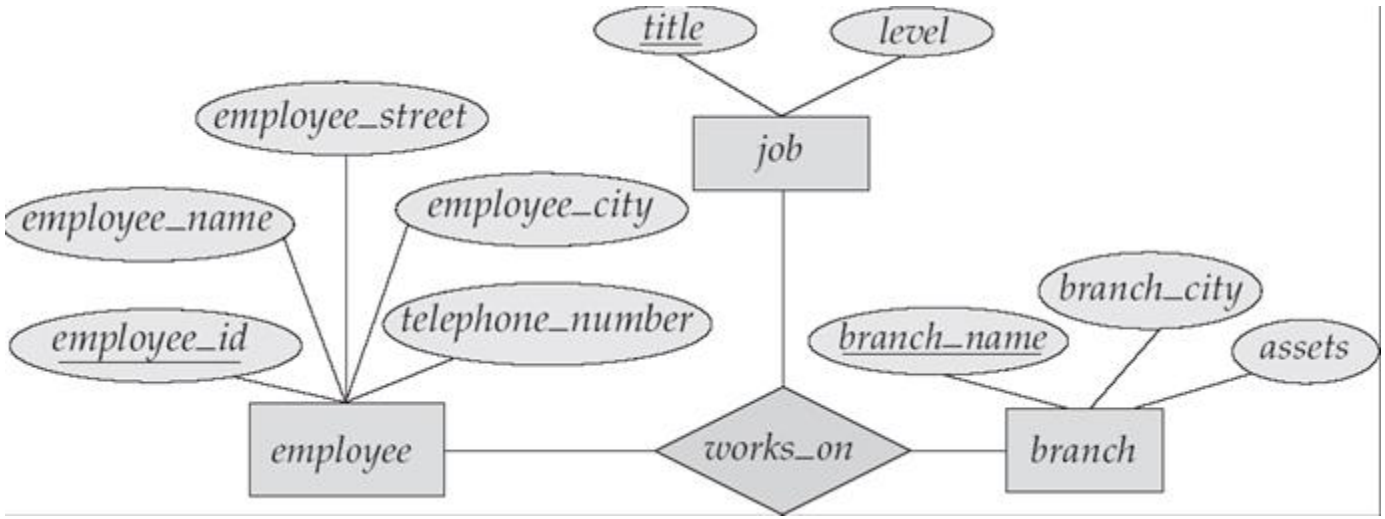
- **Total** participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., every loan must have a customer (via borrower)
- **Partial** participation: some entities may not participate in any relationship in the relationship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



Degree of a Relationship Set

- Degree = the number of entity sets that participate in a relationship set
- [Common] **binary** relationship sets that involve two entity sets
- [Rare] higher-degree relationship sets
  - *Example*: Employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches.

→ ternary relationship set between entity sets *employee, job, and branch*

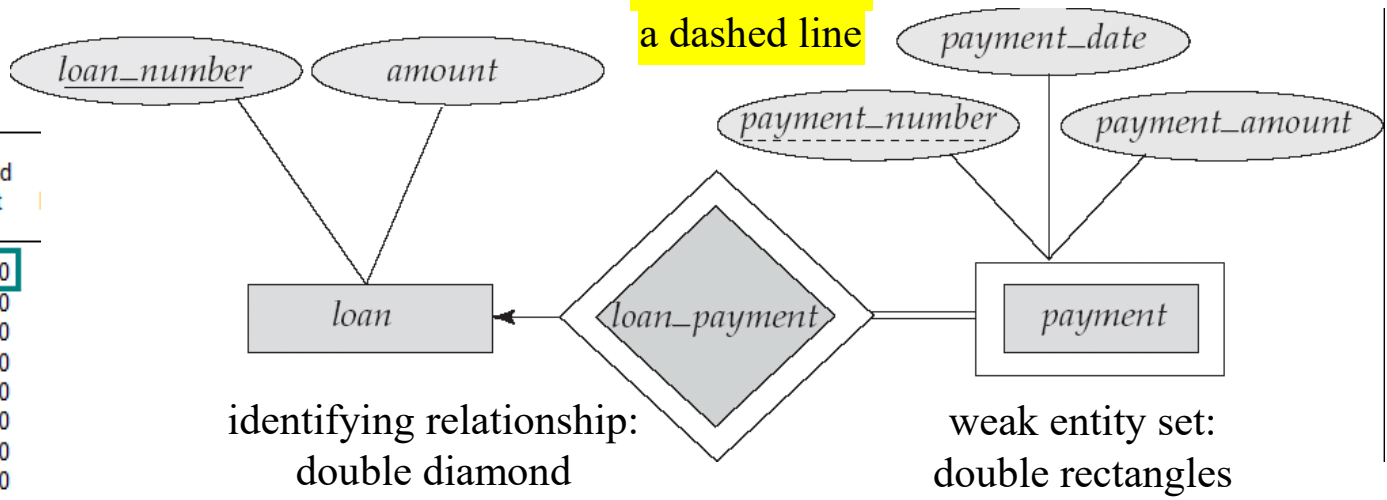


Weak Entity Sets

A **weak entity** is a type of entity that does not have any unique key for the attribute tuples. A weak entity depends on another entity (called also the identifying entity) that is considered its owner. A weak entity is an entity that cannot exist without an entity it depends on. It is depicted as a rectangle with a double border. The entity is connected with an identifying relationship to the identifying entity.

- **Weak entity set**: an entity set without a primary key
- The existence of a weak entity set depends on its **identifying entity set**
  - Identifying relationship: a total, one-to-many relationship set from the identifying entity set to the weak entity set
- The **discriminator** (*i.e., partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set. E.g., *payment_number*
- The primary key of a weak entity set: primary key of its identifying entity set & the weak entity set's discriminator. E.g., (*loan_number, payment_number*)

Payment No.	Payment Date	Beginning Balance	Scheduled Payment
1	1/1/2011	$ 100,000.00	$ 13,586.80
2	1/1/2012	92,413.20	13,586.80
3	1/1/2013	84,371.20	13,586.80
4	1/1/2014	75,846.68	13,586.80
5	1/1/2015	66,810.68	13,586.80
6	1/1/2016	57,232.53	13,586.80
7	1/1/2017	47,079.68	13,586.80
8	1/1/2018	36,317.67	13,586.80
9	1/1/2019	24,909.93	13,586.80
10	1/1/2020	12,817.73	13,586.80



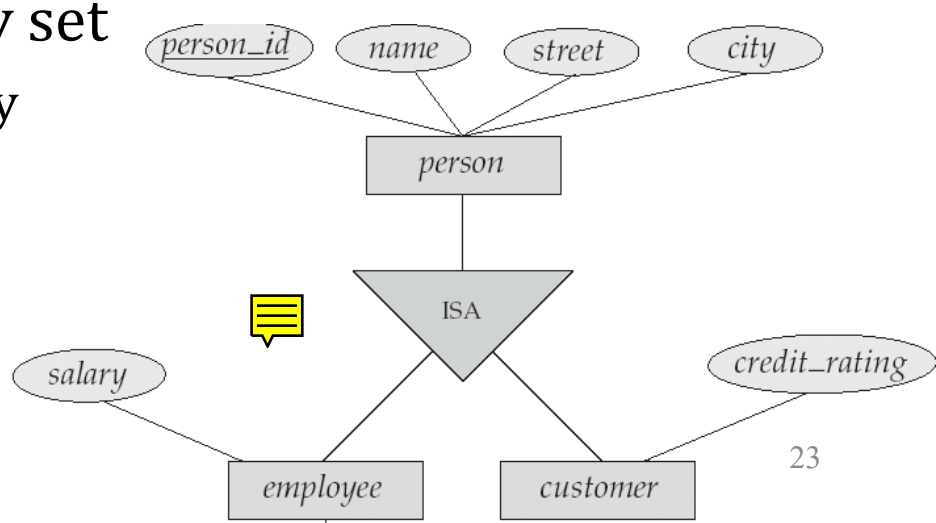
Outline

- Entity relationship (ER) diagram

- Extended features of ER diagram
- Design issues and decisions
- How to convert a ER diagram to relational schemas?

Specialization/Generalization

- A *triangle* labeled ISA
  - Partition an entity set into subgroups
    - E.g., *customer* "is a" *person*
  - **Attribute inheritance**, e.g., employee inherits all the attributes and relationship participation of person
  - Entities from the same subgroup share the same characteristics
    - E.g., employee has salary
  - These subgroups have attributes / relationships that are not used in the higher-level entity set
    - E.g., person does not have salary

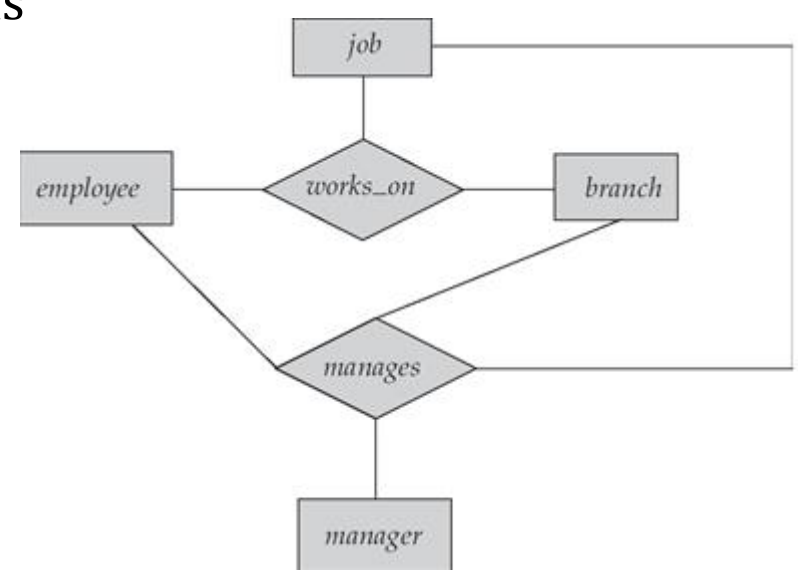


Aggregation

- Consider the ternary relationship *works_on*; suppose we want to record managers for tasks performed by an employee at a branch

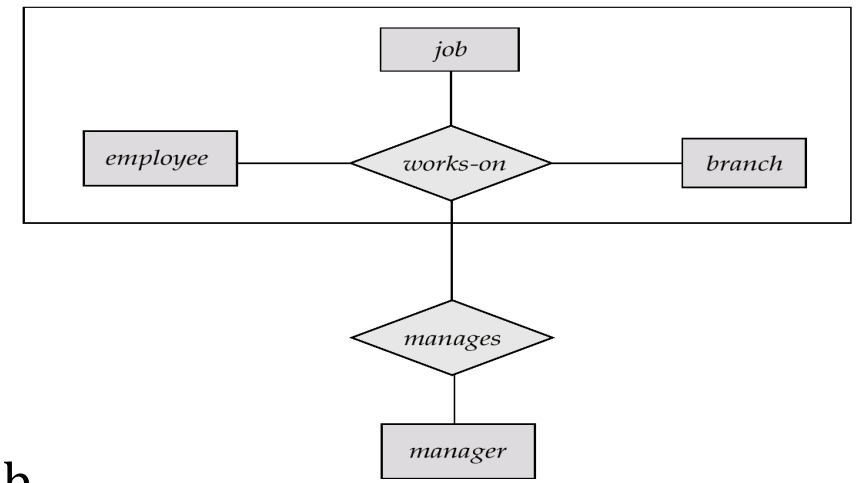
Ternary: consisting of three parts

- 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



Aggregation (Cont.)

- 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



Outline

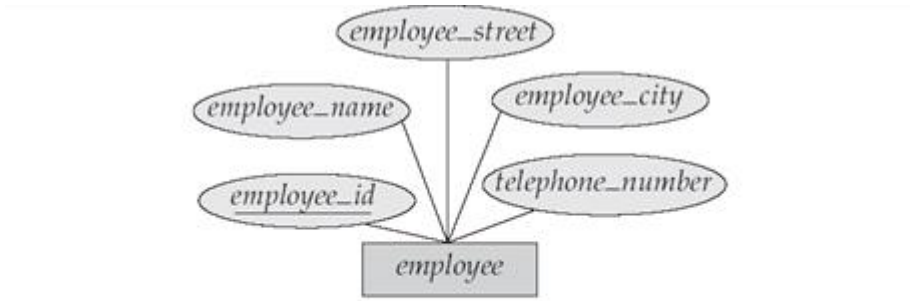
- Entity relationship (ER) diagram
- Extended features of ER diagram
- Design issues and decisions
- How to convert a ER diagram to relational schemas?

Summary of ER Design Decisions

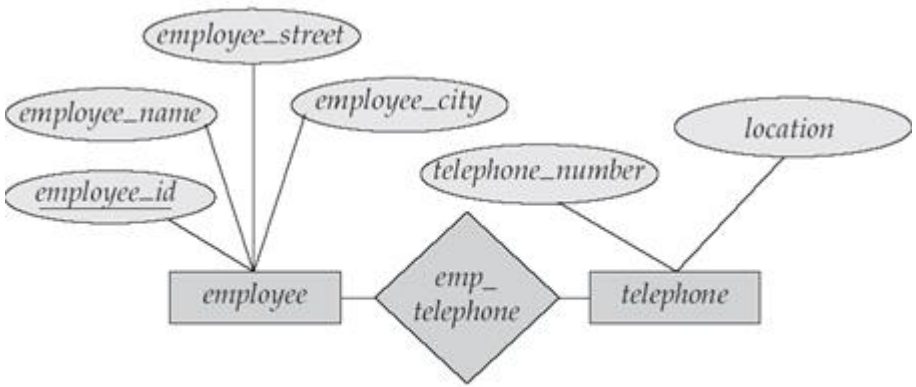
- Entity set vs. attribute
- Entity set vs. relationship set
- A ternary relationship vs. a pair of binary relationships
- Strong entity set vs. weak entity set
- The use of specialization/generalization
  - contributes to modularity in the design
- The use of aggregation
  - treat the aggregate entity set as a single unit without caring its internal structure

Design Issues

- Use of entity sets vs. attributes
  - **Question**: What is the difference between these two models, in terms of the "telephone_number" attribute?



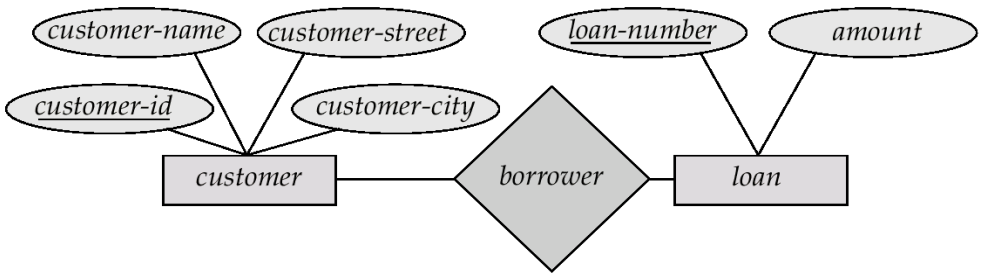
(a)

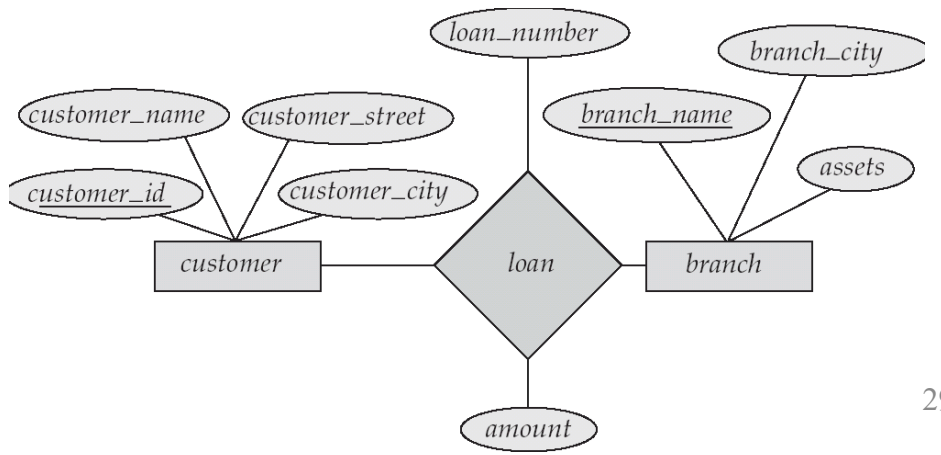


(b)

Design Issues

- Use of entity sets vs. relationship sets
  - Designate a relationship set to describe an action that occurs between entities
  - The second ER-diagram cannot directly model the case that "a loan can be jointly held by several customers"





Design Issues

- Binary versus n-ary relationship sets
  - Some relationships are naturally non-binary
    - Example: *works_on*
  - Some relationships that appear to be non-binary may be better represented using binary relationships
    - E.g., ternary relationship *parents* (between child and his/her father and mother)

      is best replaced by

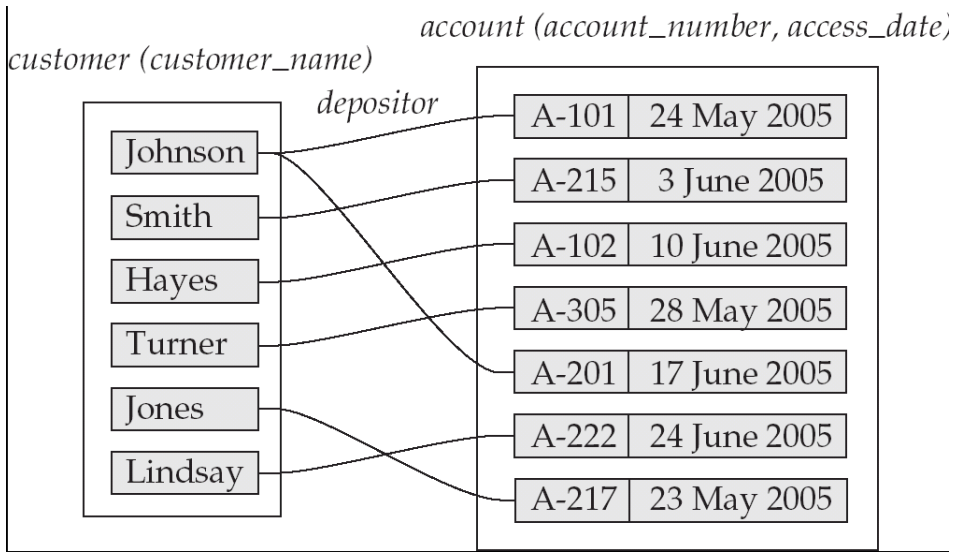
      two binary relationships: *fatherOf* and *motherOf*

Design Issues

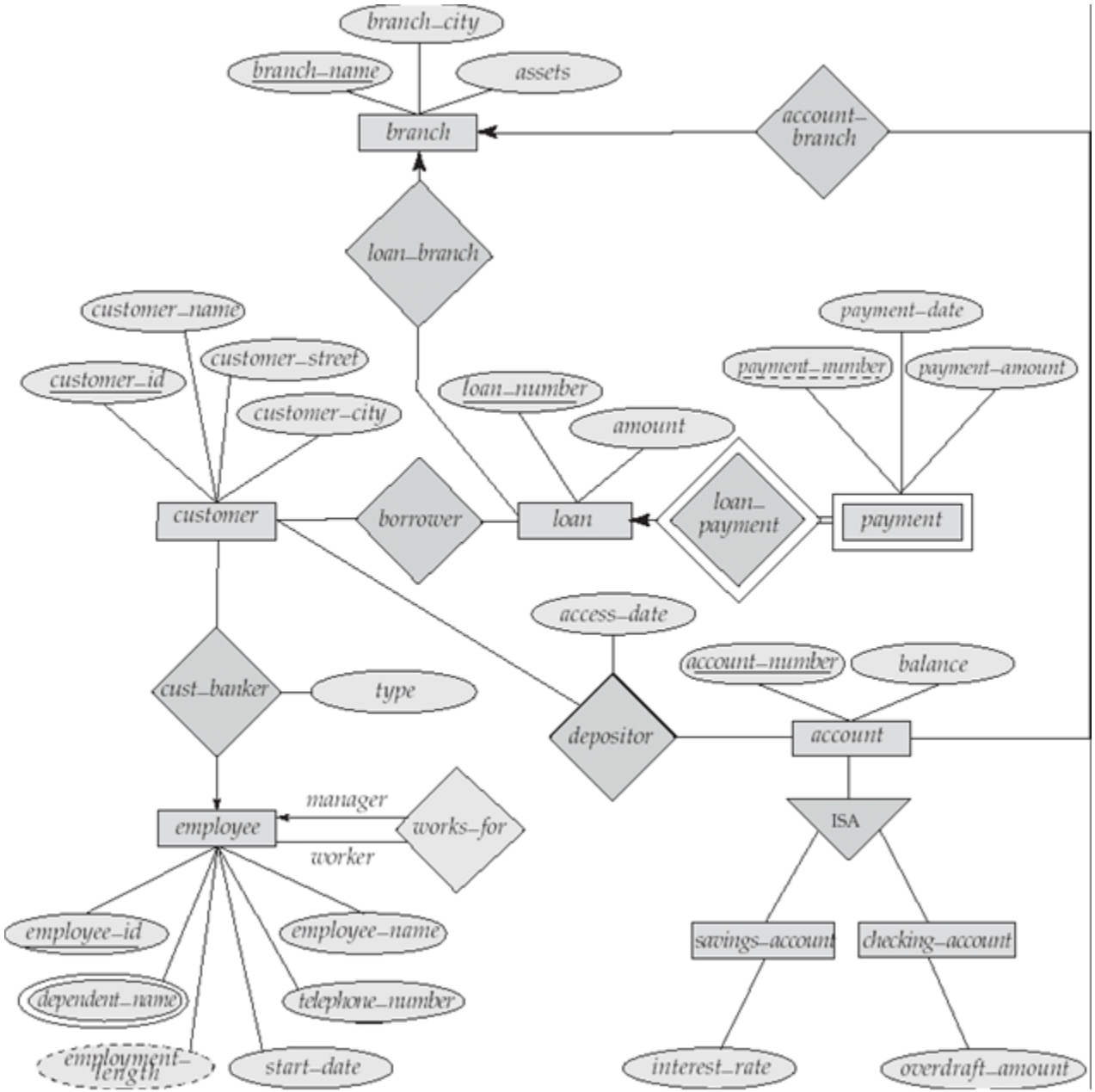
- **Placement of relationship attributes** (depending on the cardinality ratio)

  If each account can have only one customer, use access-date as an attribute of account, instead of a relationship attribute

  That is, the relationship from customer to account is one to many



ER Diagram for the Bank Database



Outline

- Entity relationship (ER) diagram
- Extended features of ER diagram
- Design issues and decisions

- How to convert a ER diagram to relational schemas?

How to convert a ER digram into relational schemas?

- Convert each entity set into a schema
- Convert each relationship set into a schema
- Example: convert the entity set "customer" into:
  - *Customer_schema* = (*customer_id*, *customer_city*)

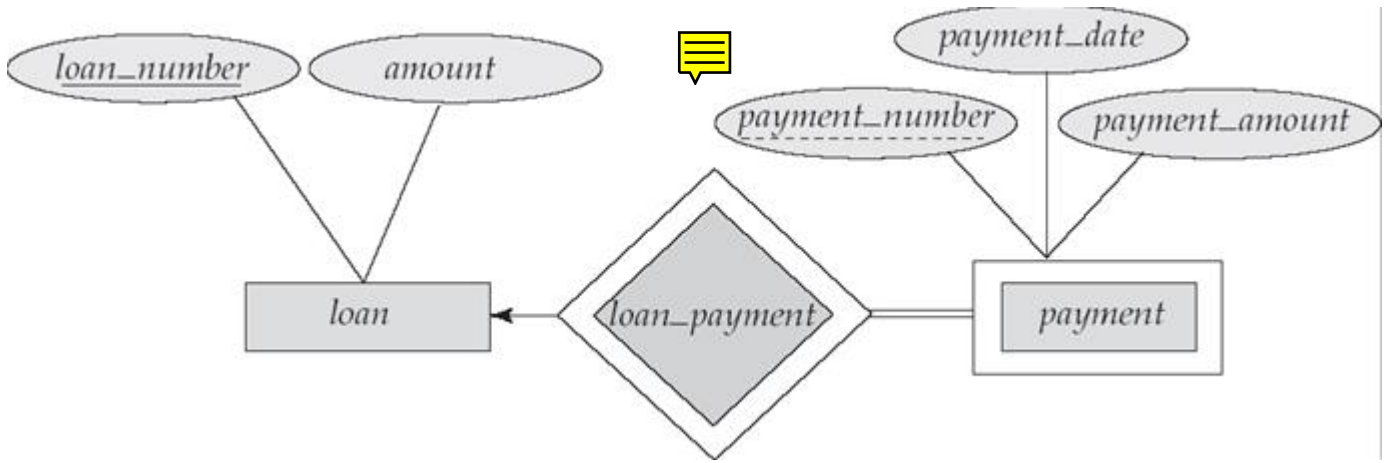
Representation as Schemas

- Representing an entity set
  - A strong entity set → a schema with the same attributes

    loan = (*loan_number*, *amount*)

  - A weak entity set → its attributes + the primary key of the identifying strong entity set

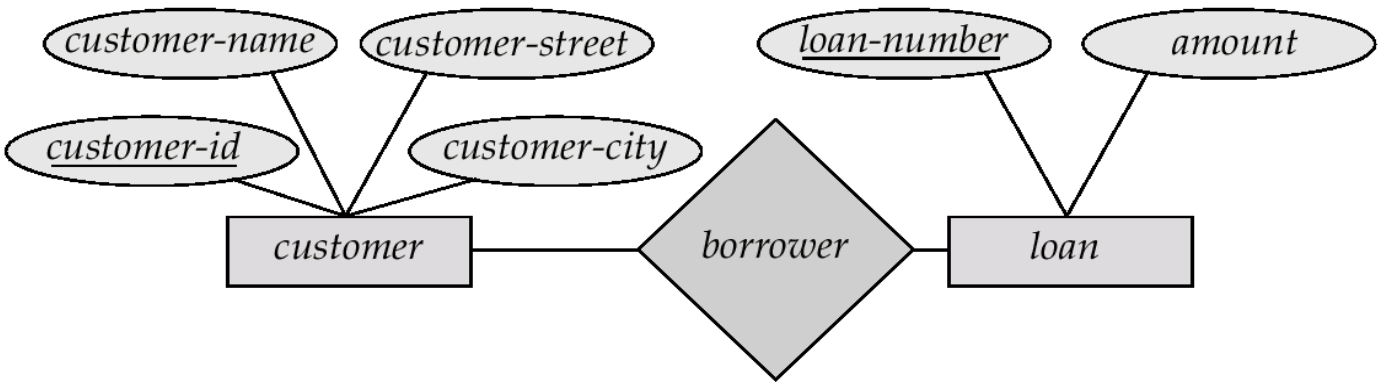
    payment = (*loan_number*, *payment_number*, *payment_date*, *payment_amount*)



Representation as Schemas

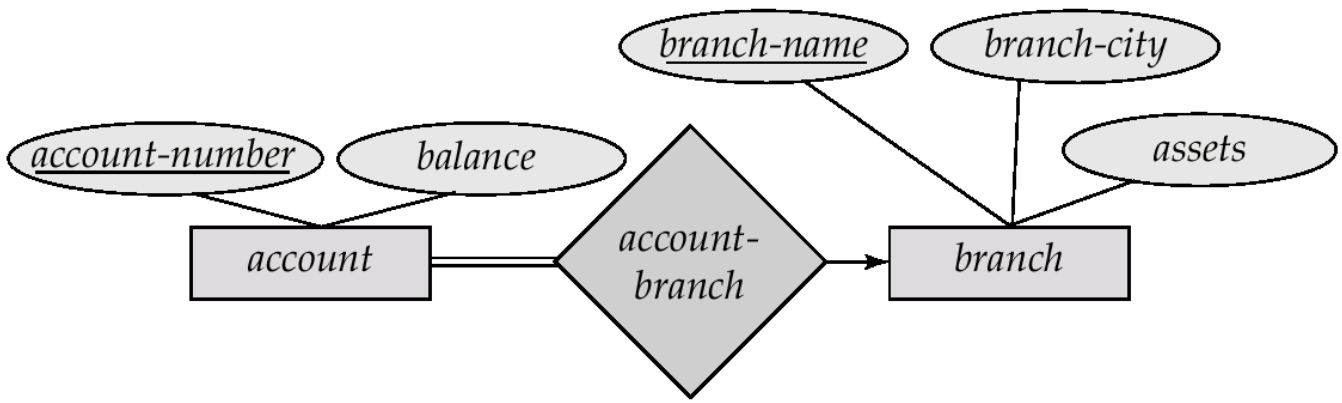
- Representing a relationship set
  - A **many-to-many** relationship set → a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set
  - Example: schema for relationship set borrower

borrower = (*customer_id, loan_number*)



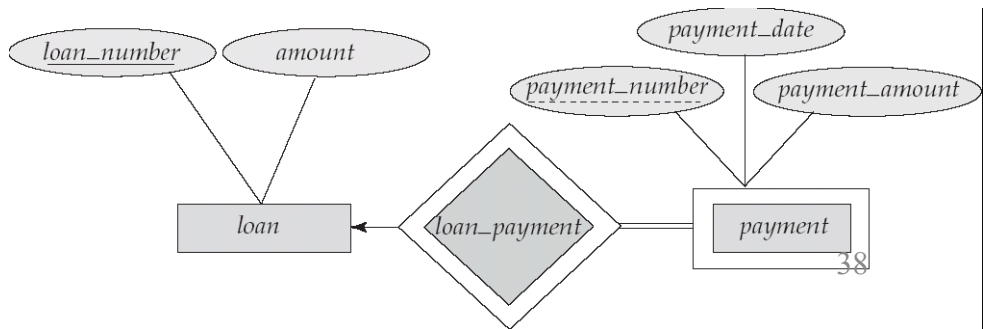
Redundancy of Schemas

- Example on many-to-one relationship:
  - Should we use the schema *account_branch*(*account-number,branch-name*)?
    - It is not necessary!
  - Just add an attribute *branch_name* to the schema of *account* :
    - *account* = (*account-number*, *balance, branch-name*)
- **Many-to-one** and **one-to-many** relationship sets that are total on the many-side: represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side



Redundancy of Schemas (Cont.)

- For one-to-one relationship set, either side can be used as the "many" side
  - The extra attribute can be added to either side
- 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*)

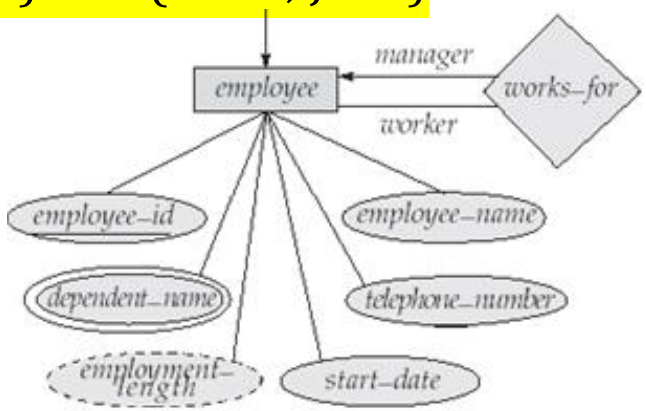


Composite and Multivalued Attributes

- For composite attribute, create an attribute for each component
  - Example: given entity set *customer* 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*
  - Example: multivalued attribute *dependent_name* of *employee* is represented by the following schema

    employee_dependent_names = (*employee_id*, *dname*)
    - E.g., an employee with primary key "C125" and dependents Jack and Jane → two tuples (C125, Jack) and (C125, Jane) in the above table



Representing Specialization via Schemas

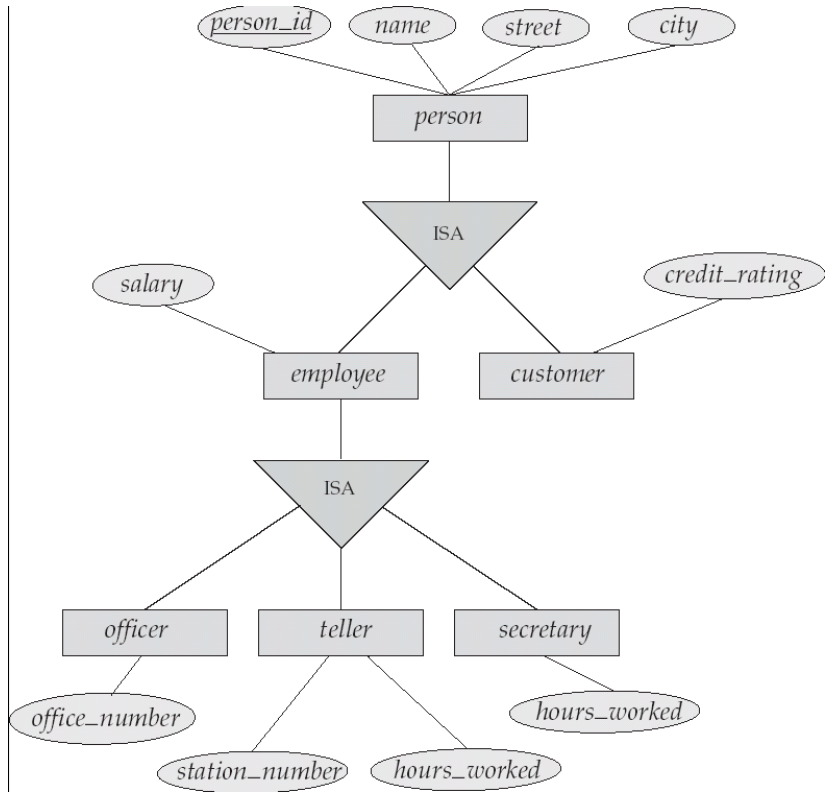
Method 1

- 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

person = (*person_id*, *name*, *street*, *city*)

customer = (*person_id*, *credit_rating*)

employee = (*person_id*, *salary*)



- Drawback: to find all the information of an *employee*, we need to access two relations (*person* and *employee*)

Representing Specialization as Schemas (Cont.)

Method 2

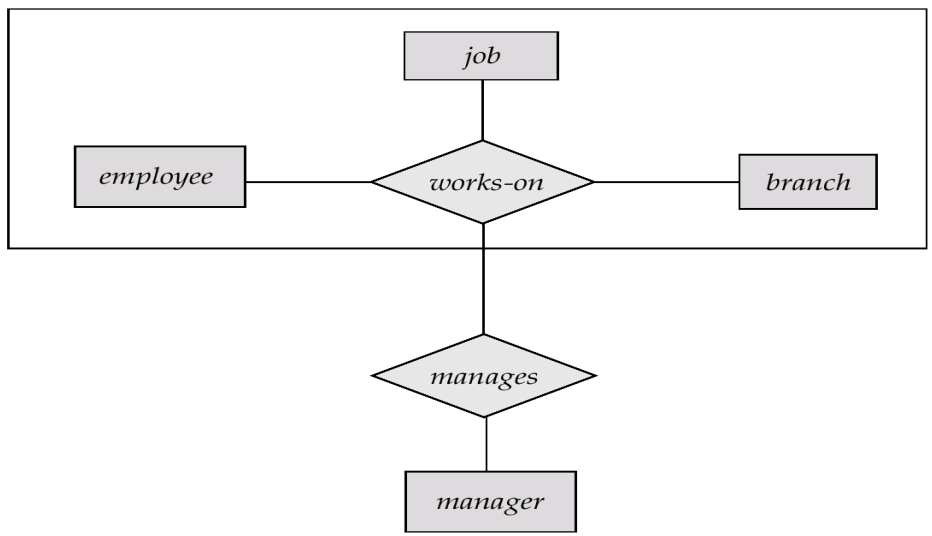
- Form a schema for each entity set with all local and inherited attributes

person =	(*person_id*, *name*, *street*, *city*)
customer =	(*person_id*, *name*, *street*, *city*, *credit_rating*)
employee =	(*person_id*, *name*, *street*, *city*, *salary*)

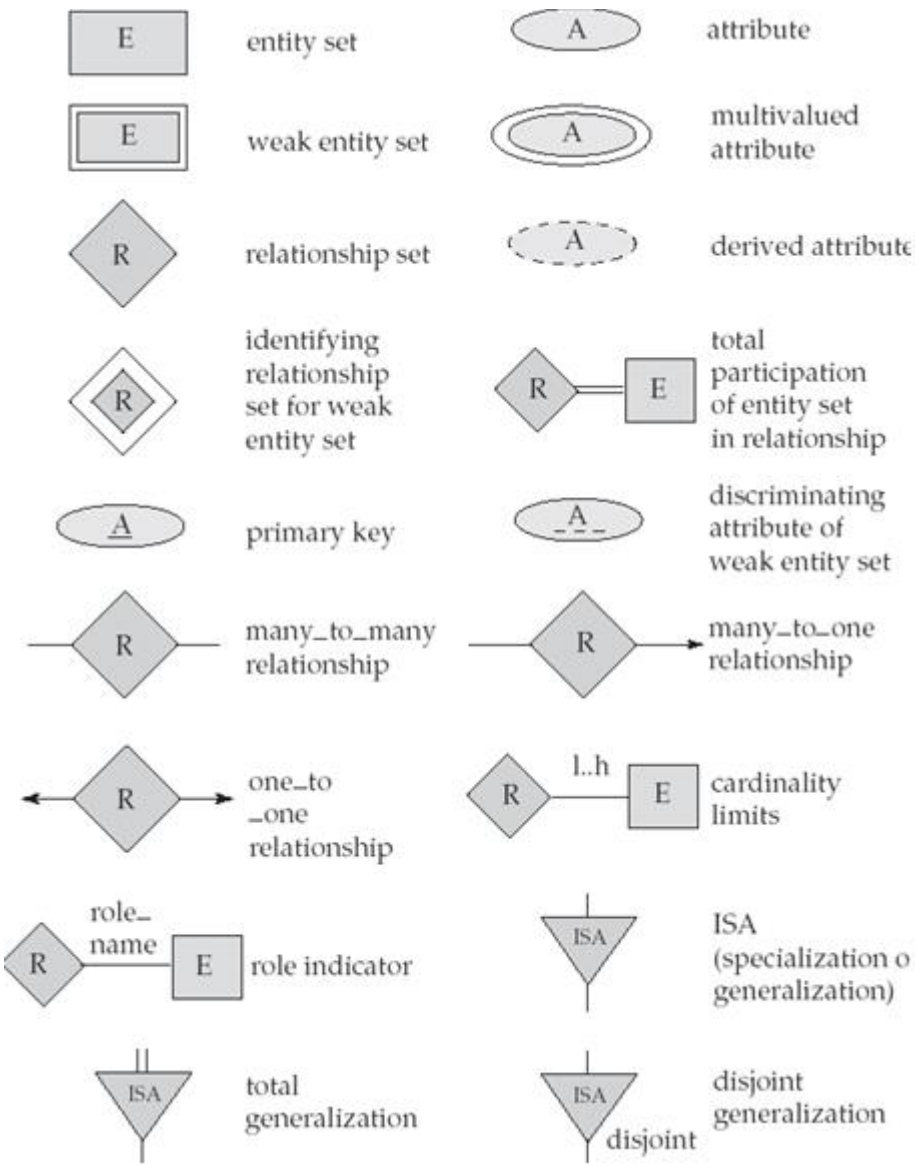
- If specialization is total, the schema for the generalized entity set (*person*) not required to store information
  - Can be defined as a "view" relation containing union of specialization relations
- Drawback: *street* and *city* may be stored redundantly for people who are both customers and employees

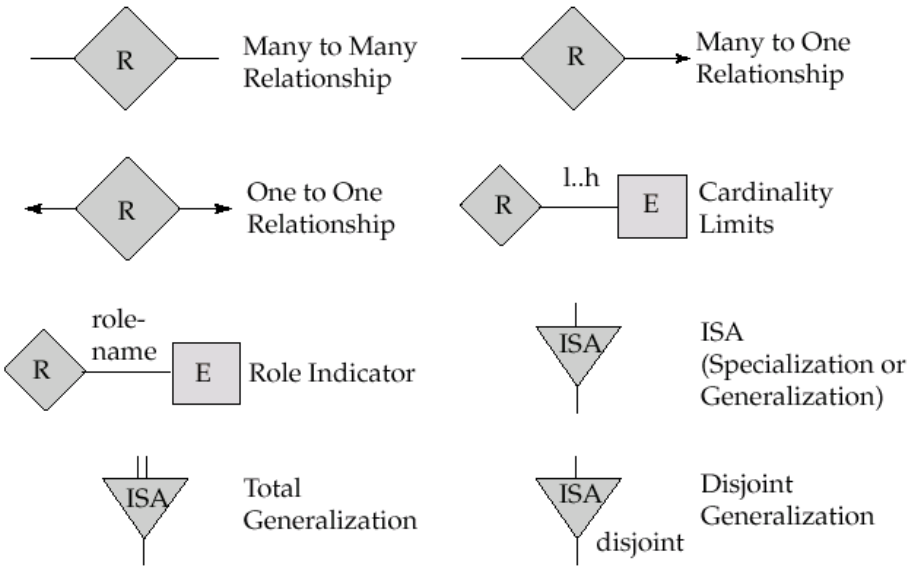
Schemas Corresponding to Aggregation

- 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*)
- We can remove the schema *works_on* provided that we are willing to store *NULL* values for attribute *manager_name* in relation on schema *manages*

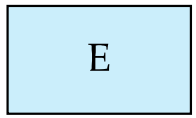


Summary of Chen's Notations

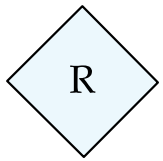




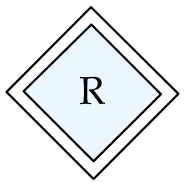
Appendix: UML notations



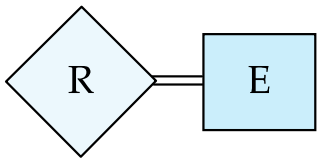
entity set



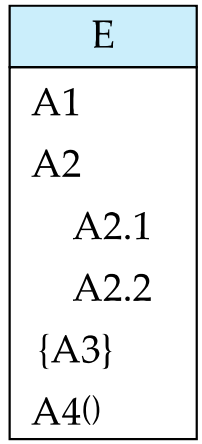
relationship set



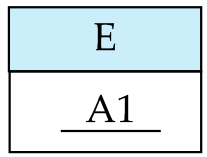
identifying relationship set for weak entity set



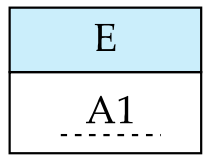
total participation of entity set in relationship



attributes:
simple (A1),
composite (A2) and
multivalued (A3)
derived (A4)

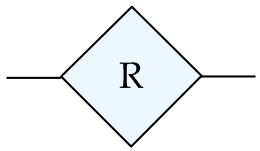


primary key

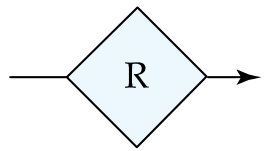


discriminating attribute of weak entity set

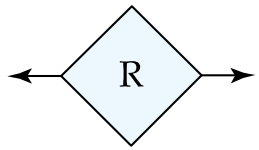
Appendix: UML notations



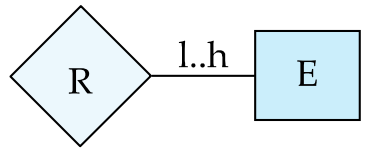
many-to-many relationship



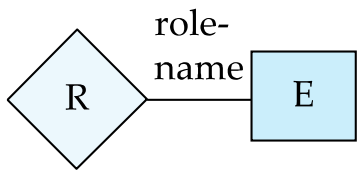
many-to-one relationship



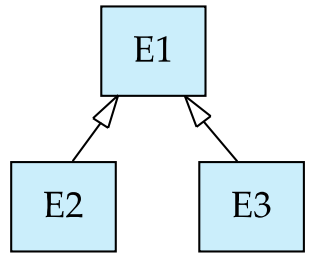
one-to-one relationship



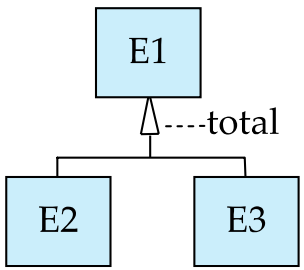
cardinality limits



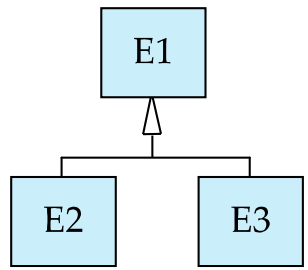
role indicator



ISA: generalization or specialization



total (disjoint) generalization



disjoint generalization