ER Model and Relational Model

Wen-Chih Peng
Dept. of Computer Science
National Chiao Tung University

Introduction

- The **relational model** was first introduced by Ted Codd of IBM Research in 1970,
- The model uses the concept of a **mathematical relation** - which looks like ***a table of values*** –

  as its basic building block.
- The **SQL Query Language** was developed by IBM in the 1970's

Why study the relational model ?

- Dominant data model
- Used in almost DB systems
- Supports simple, powerful **querying** of data

Data Modeling

- Goals:
  - Conceptual representation of the data
  - “Reality” meets “bits and bytes”
  - Must make sense, and be usable by other people
- We will study:
  - Entity-relationship Model
  - Relational Model
    - Note the difference !!

Motivation

- Suppose that you are DBA for online banking web site.
- You are asked to create a database that monitors:
  - customers
  - accounts
  - loans
  - branches
  - transactions

Database Design Steps

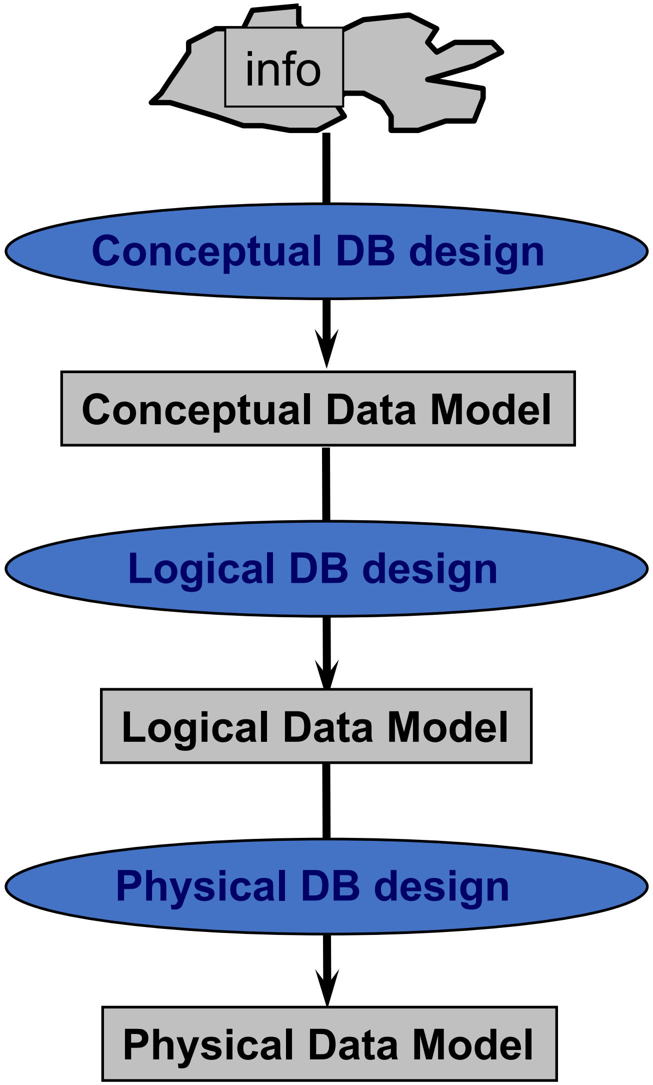
Entity-relationship Model

Typically used for conceptual database design

Three Levels of Modeling

Relational Model

Typically used for logical database design



Entity-Relationship Model

- Two key concepts
  - *Entities*:
    - An object that *exists* and is *distinguishable* from other objects
      - Examples: Bob Smith, BofA, CMSC424
    - Have *attributes* (people have names and addresses)
    - Form *entity sets* with other entities of the same type that share the same properties
      - Set of all people, set of all classes
    - Entity sets may overlap
      - Customers and Employees

Entity-Relationship Model

- Two key concepts
  - *Relationships*:
    - Relate 2 or more entities
      - E.g. Bob Smith *has account at* College Park Branch
    - Form *relationship sets* with other relationships of the same type that share the same properties
      - Customers *have accounts at* Branches
    - Can have attributes:
      - *has account at* may have an attribute *start-date*
    - Can involve more than 2 entities
      - Employee *works at* Branch *at* Job

Relationship Sets

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

  Example:

Hayes	*depositor*	A-102
customer	relationship	*account*

- A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

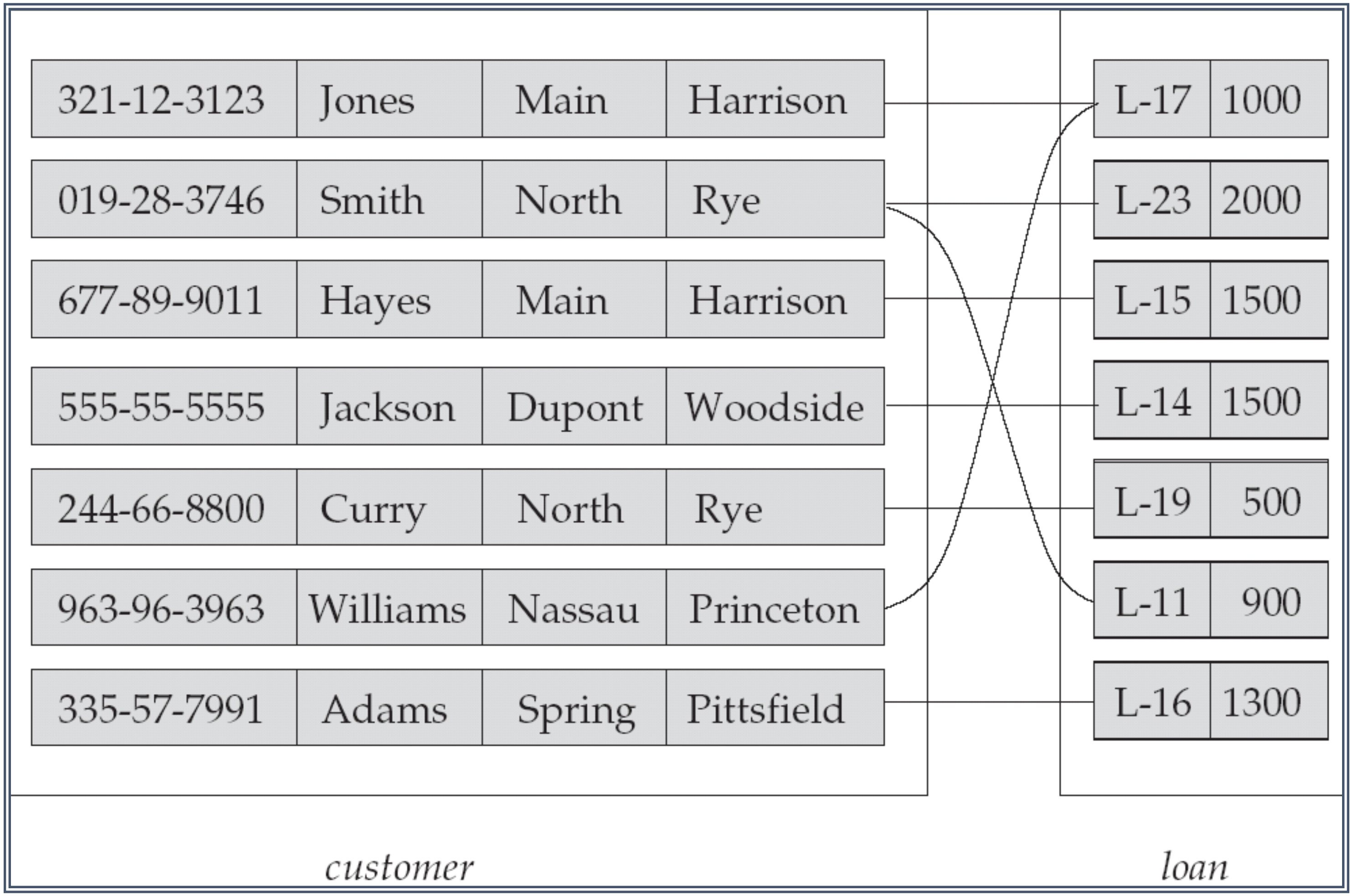
  $\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$

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

  - Example:

    (Hayes, A-102) $\in$ *depositor*

Relationship Set *borrower*



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

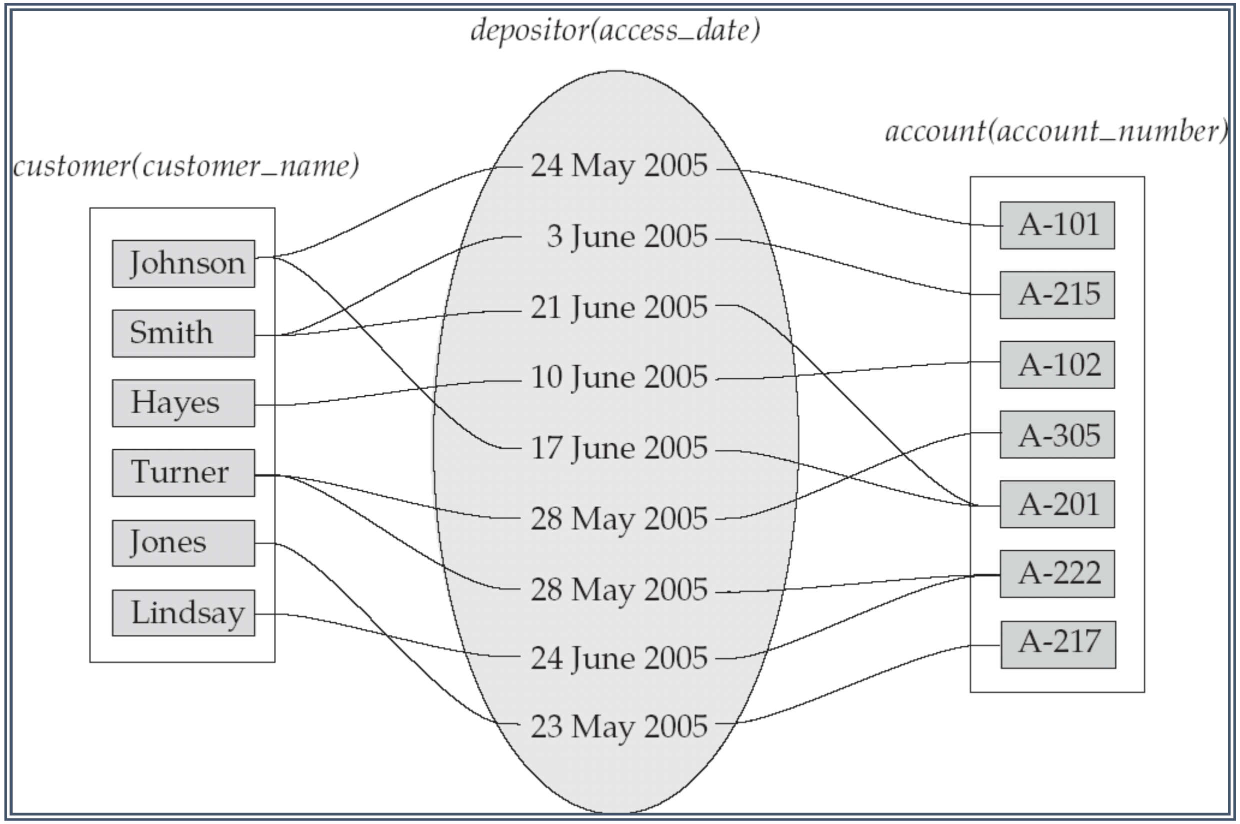
customer

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

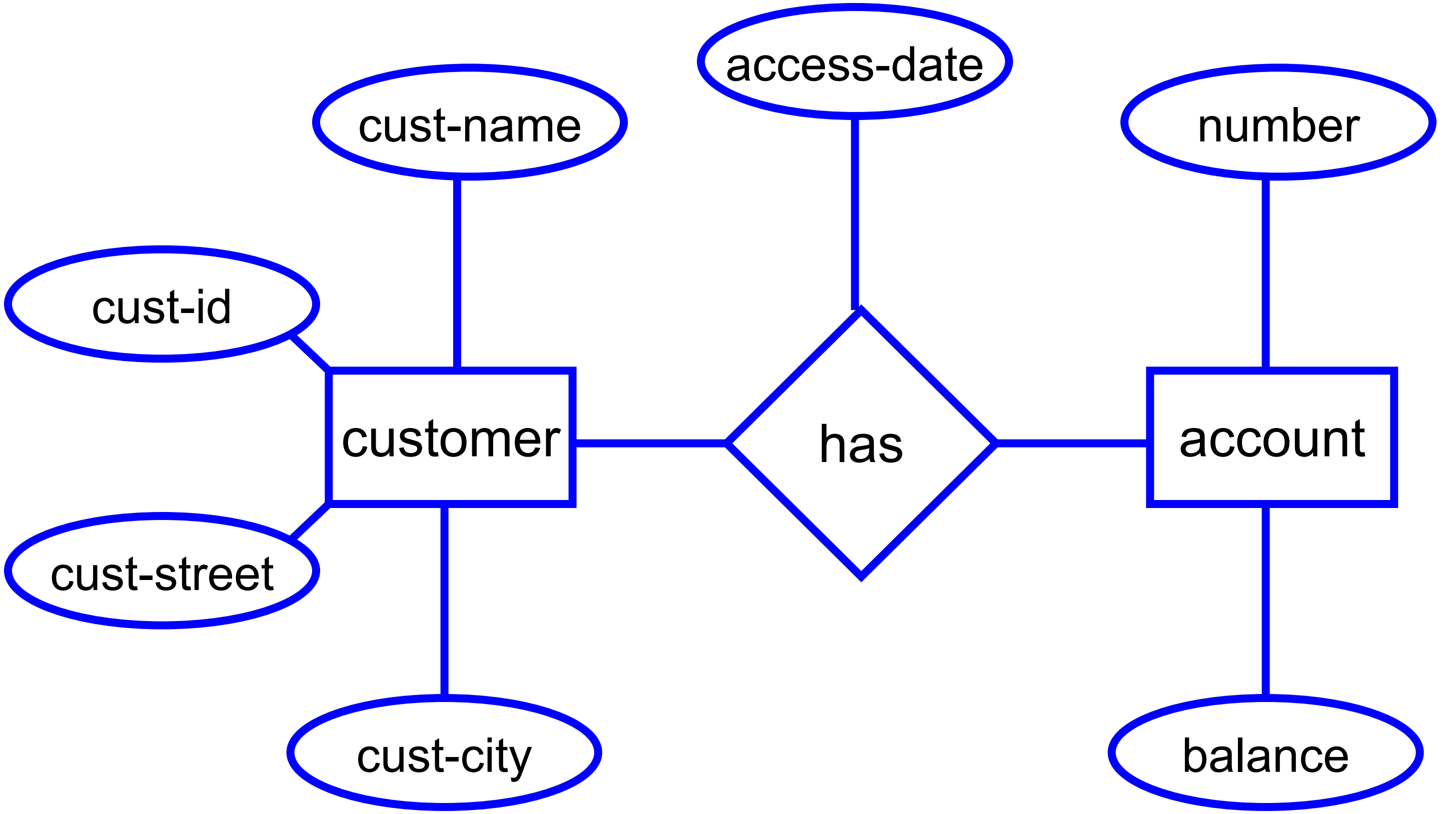
loan

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*



ER Diagram: Starting Example



- Rectangles: entity sets
- Diamonds: relationship sets
- Ellipses: attributes

Rest of the class

- Details of the ER Model
  - How to represent various types of constraints/semantic information
- Design issues
  - A detailed example

Next: Relationship Cardinalities

- We may know:

  One customer can only open one account

  OR

  One customer can open multiple accounts

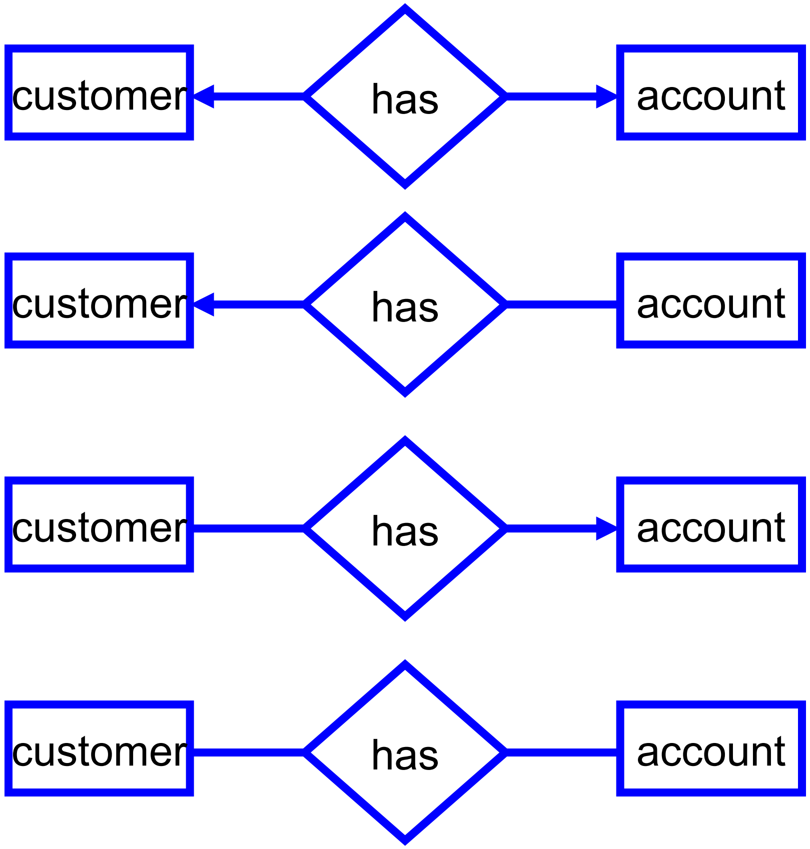
- Representing this is important
- Why ?
  - Better manipulation of data
    - If former, can store the account info in the customer table
  - Can enforce such a constraint
    - Application logic will have to do it; NOT GOOD
  - Remember: If not represented in conceptual model, the domain knowledge may be lost

Mapping Cardinalities

- Express the number of entities to which another entity can be associated via a relationship set
- Most useful in describing binary relationship sets

Mapping Cardinalities

- One-to-One
- One-to-Many
- Many-to-One
- Many-to-Many

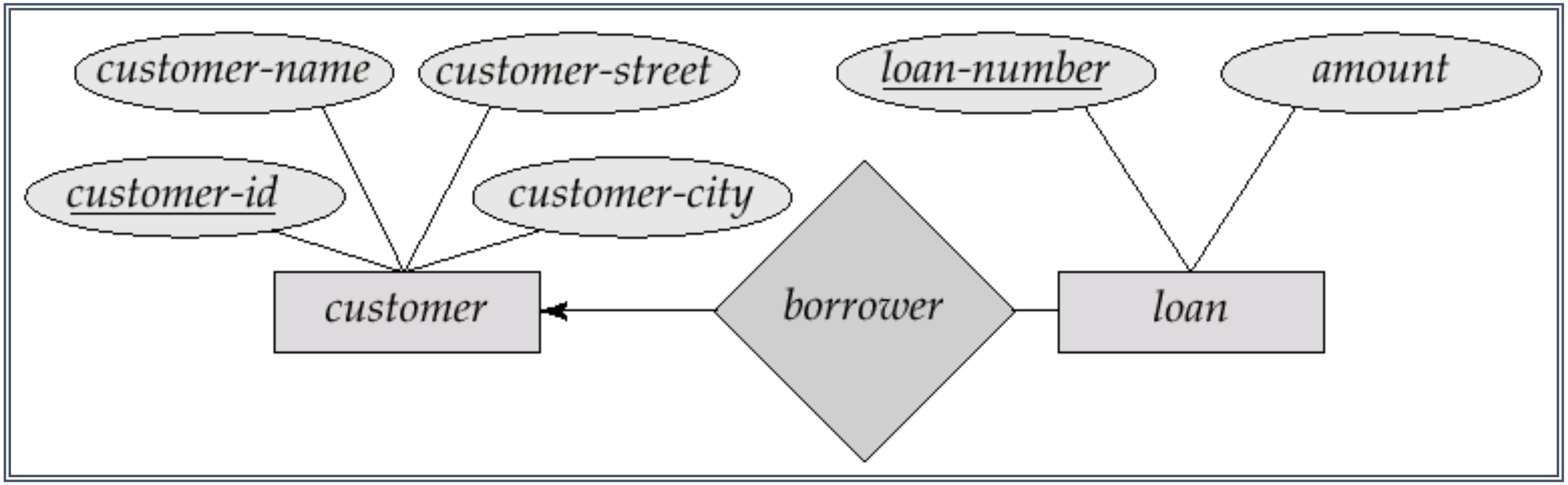


Mapping Cardinalities

- Express the number of entities to which another entity can be associated via a relationship set
- Most useful in describing binary relationship sets
- N-ary relationships ?
  - More complicated
  - Details in the book

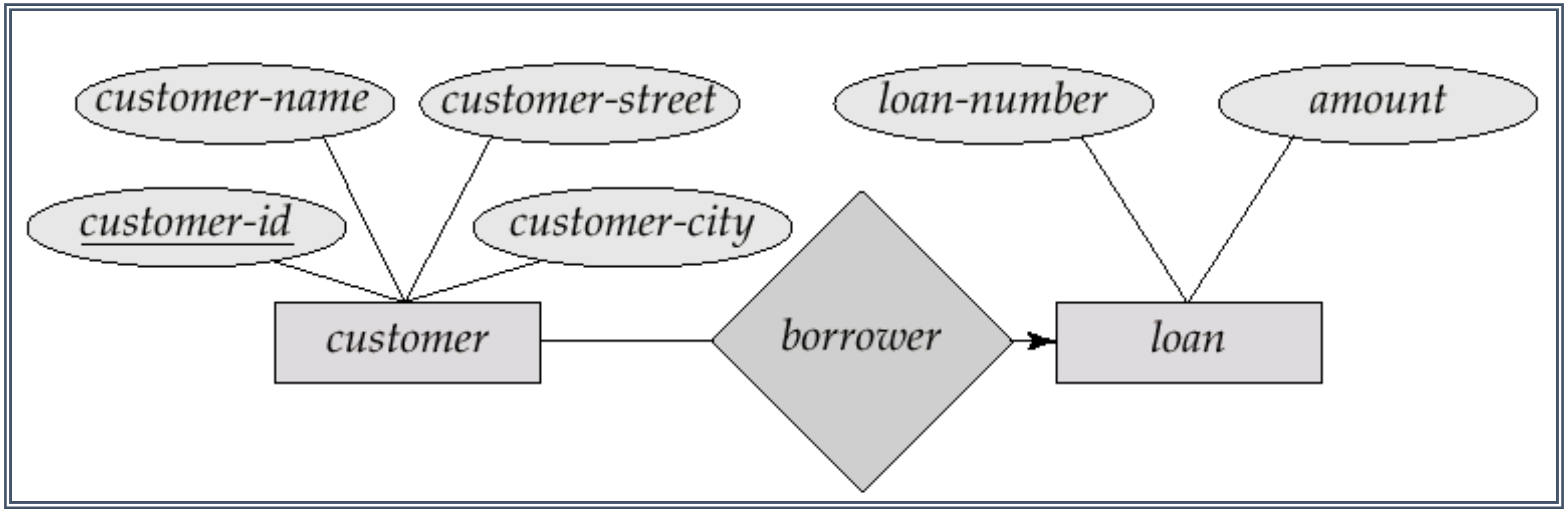
One-To-Many Relationship

- 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*



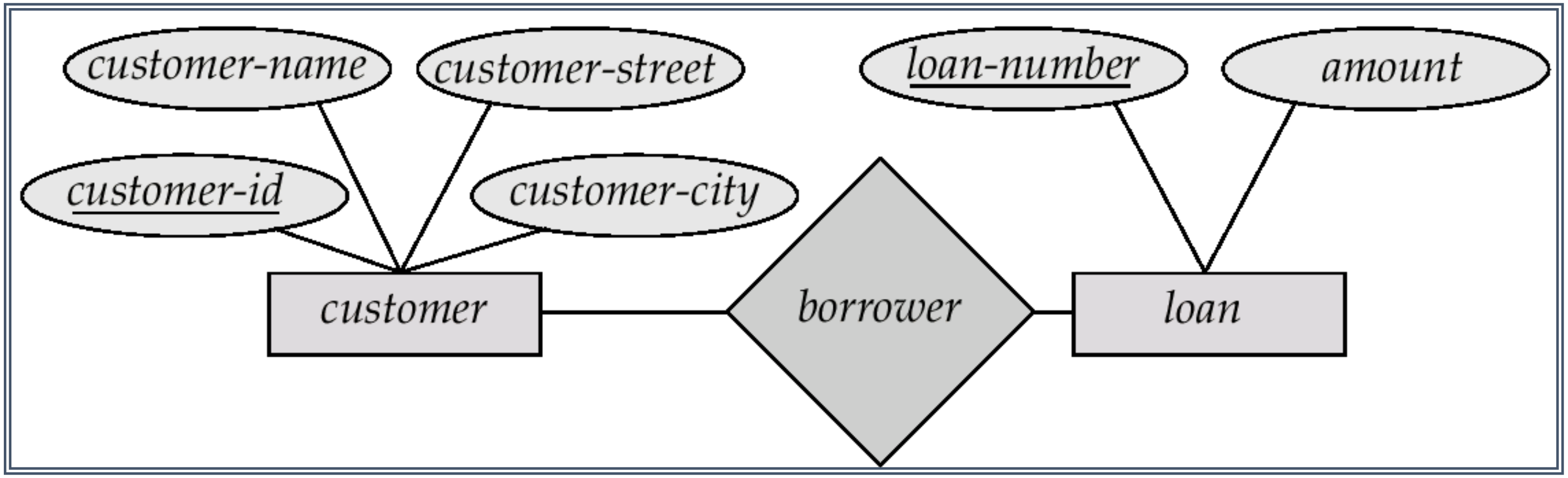
Many-To-One Relationships

- 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*



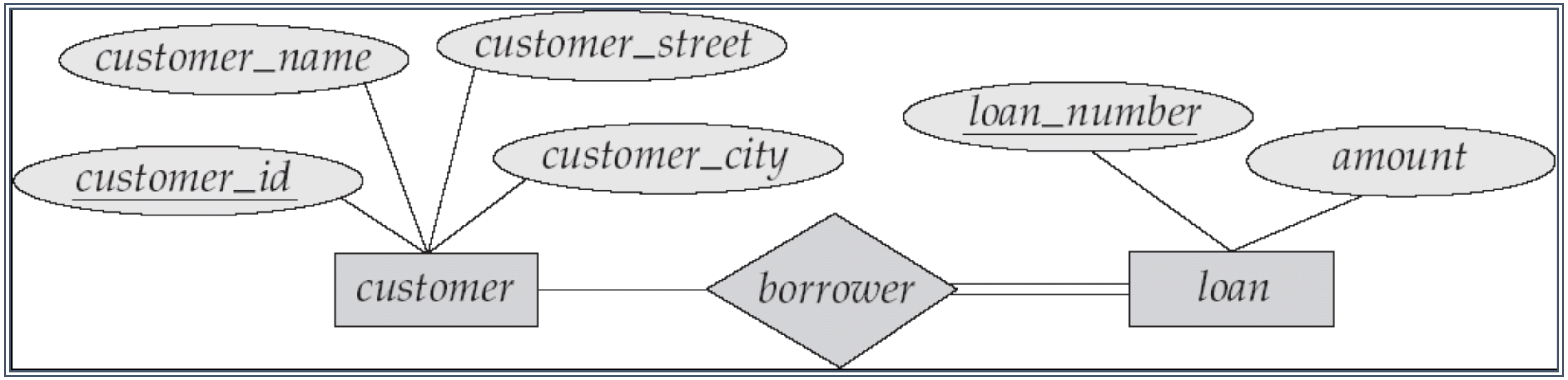
Many-To-Many Relationship

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

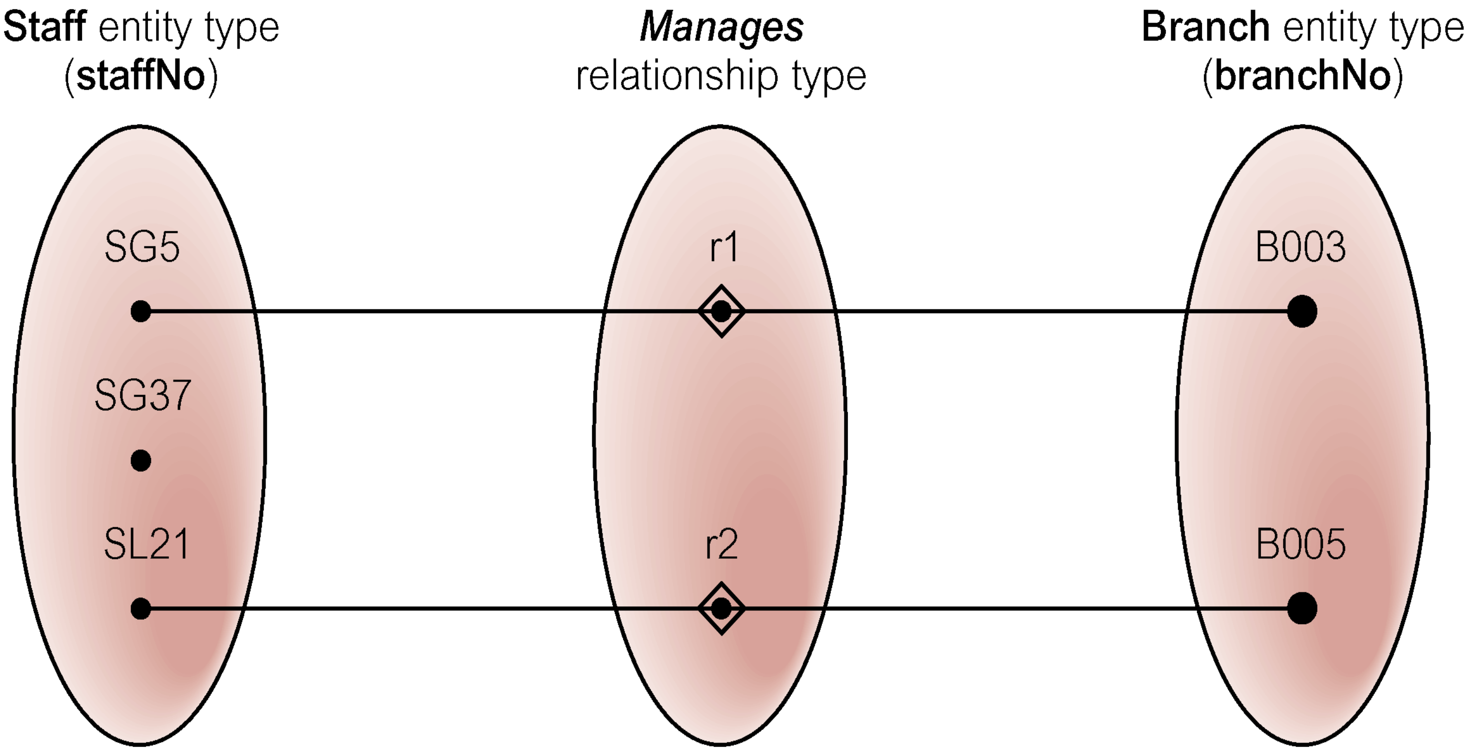


Participation of an Entity Set in a Relationship Set

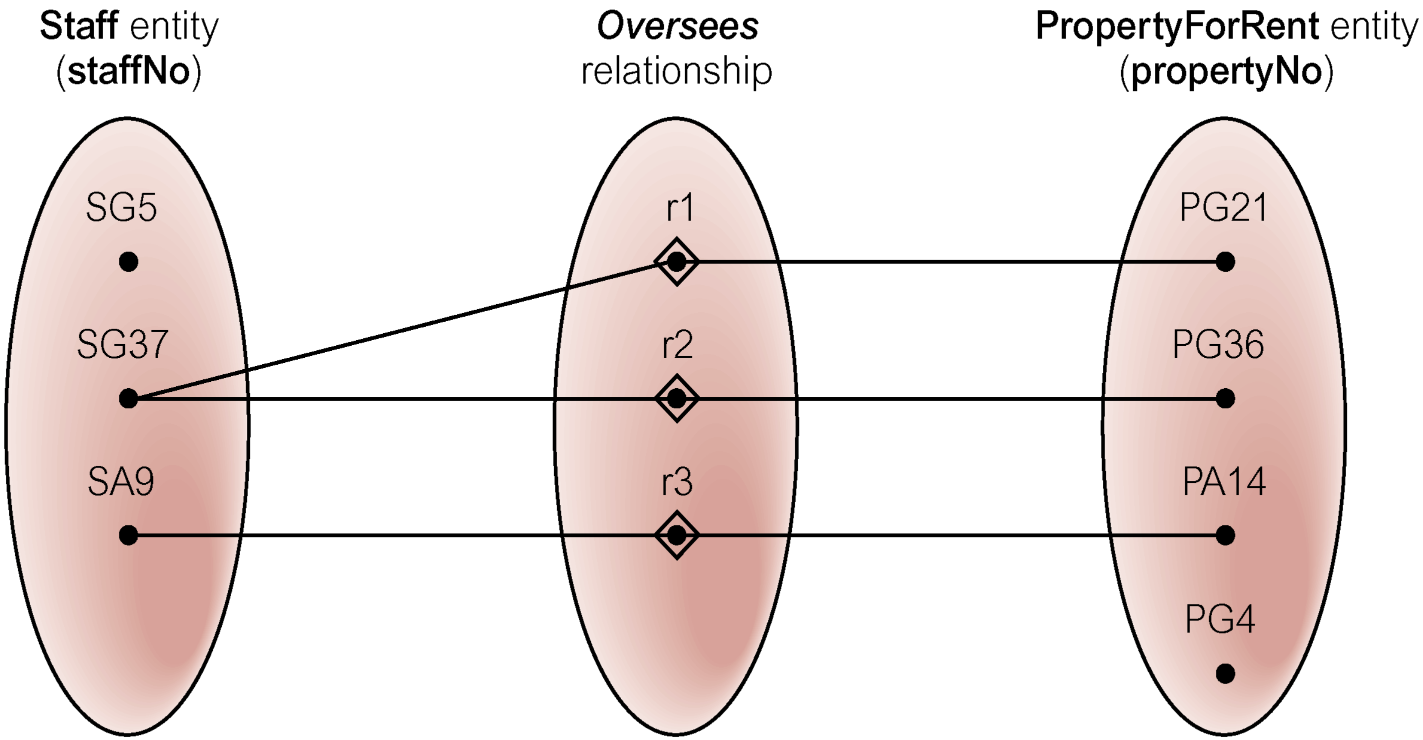
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g. participation of loan in borrower is total
    - every loan must have a customer associated to it via borrower
- Partial participation: some entities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



Semantic net of Staff *Manages* Branch relationship type

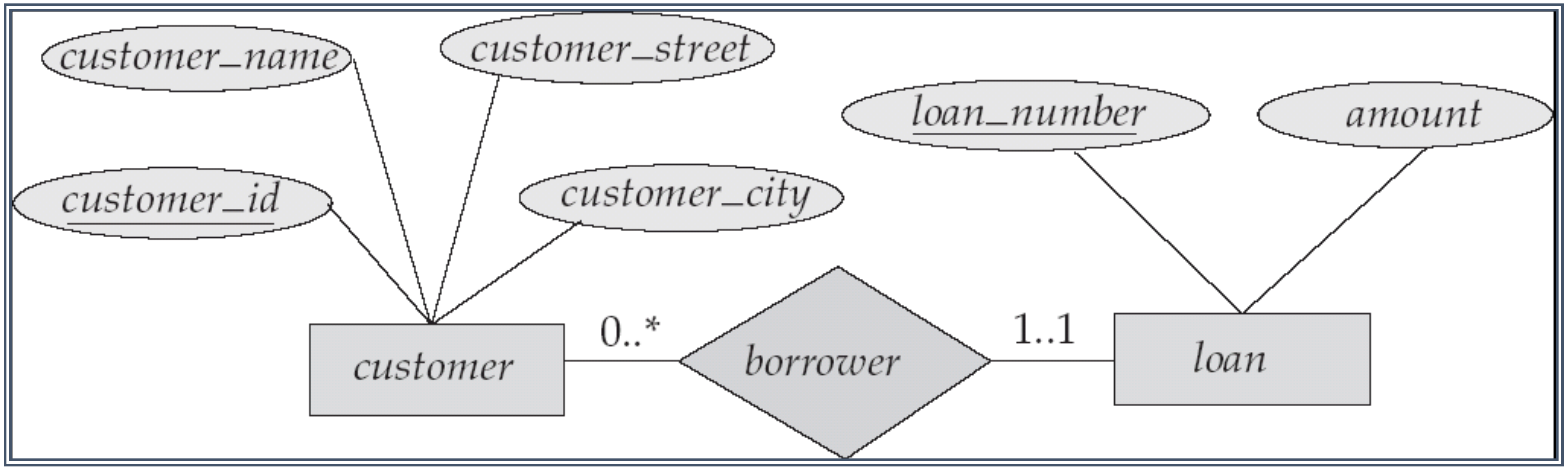


Semantic net of Staff *Oversees* PropertyForRent relationship type



Alternative Notation for Cardinality Limits

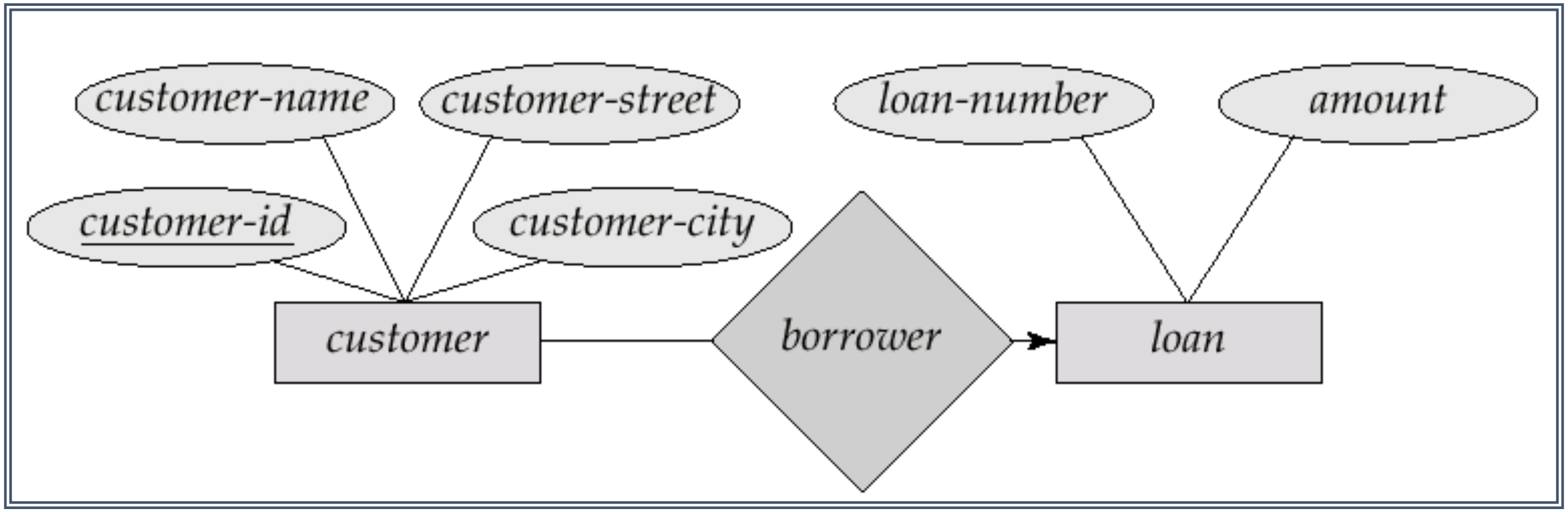
- Cardinality limits can also express participation constraints



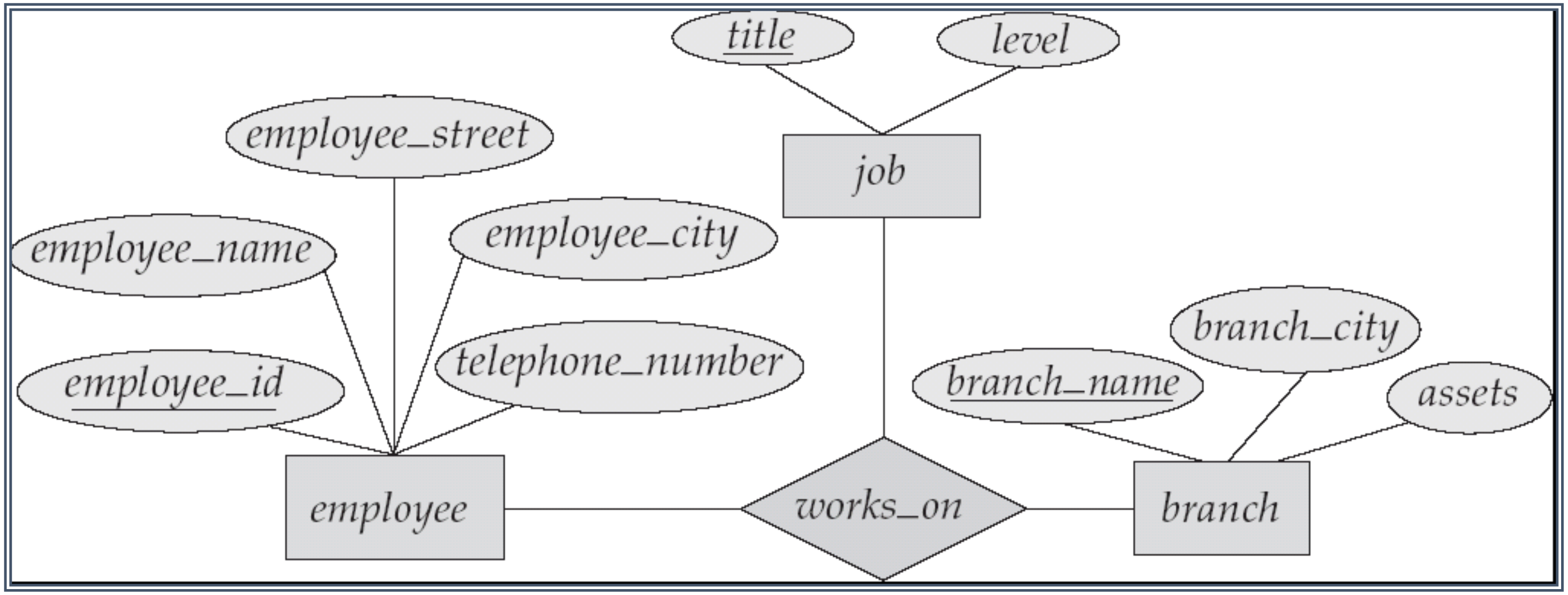
Customer: Loan => *:1

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*



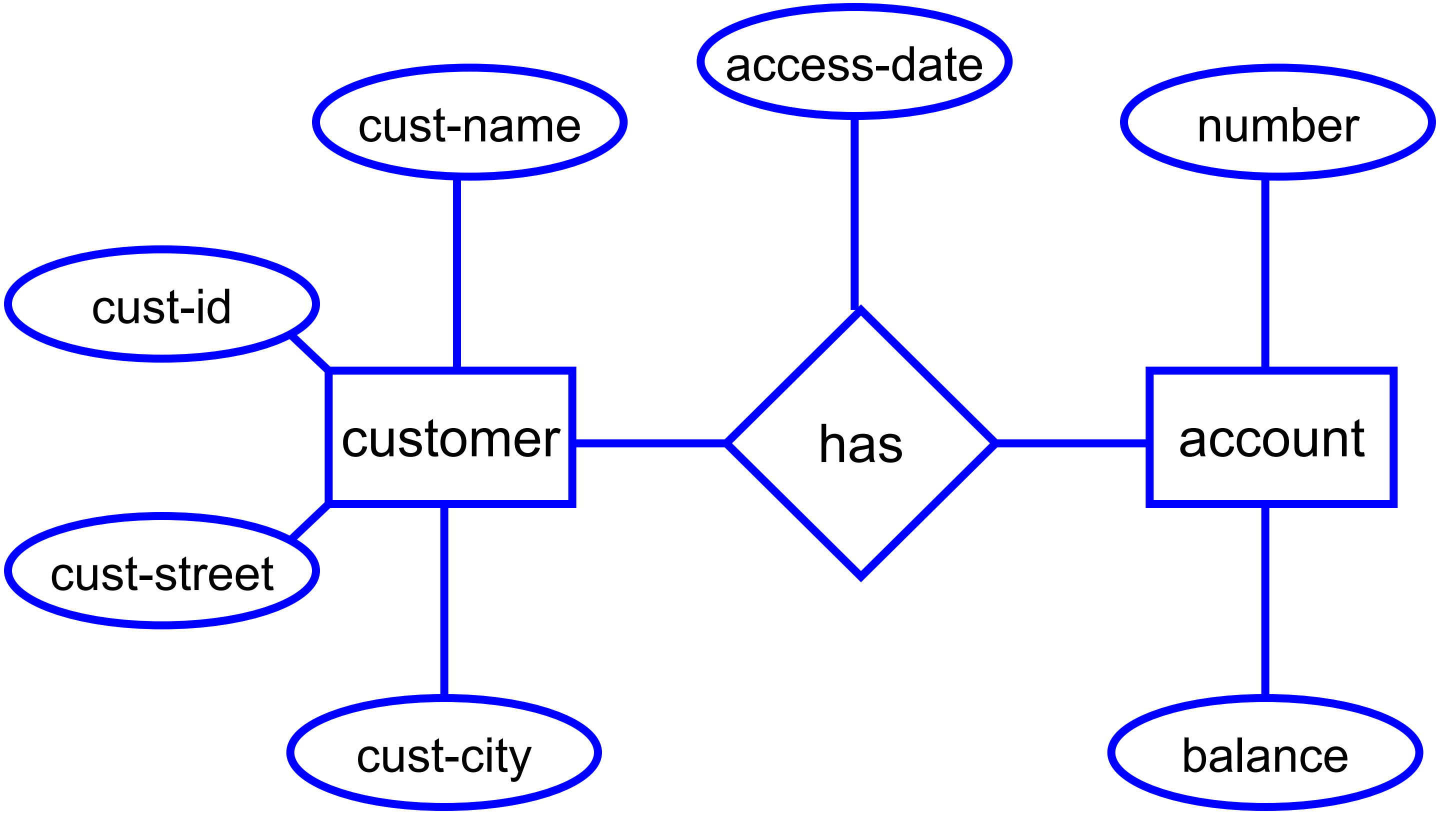
E-R Diagram with a Ternary Relationship



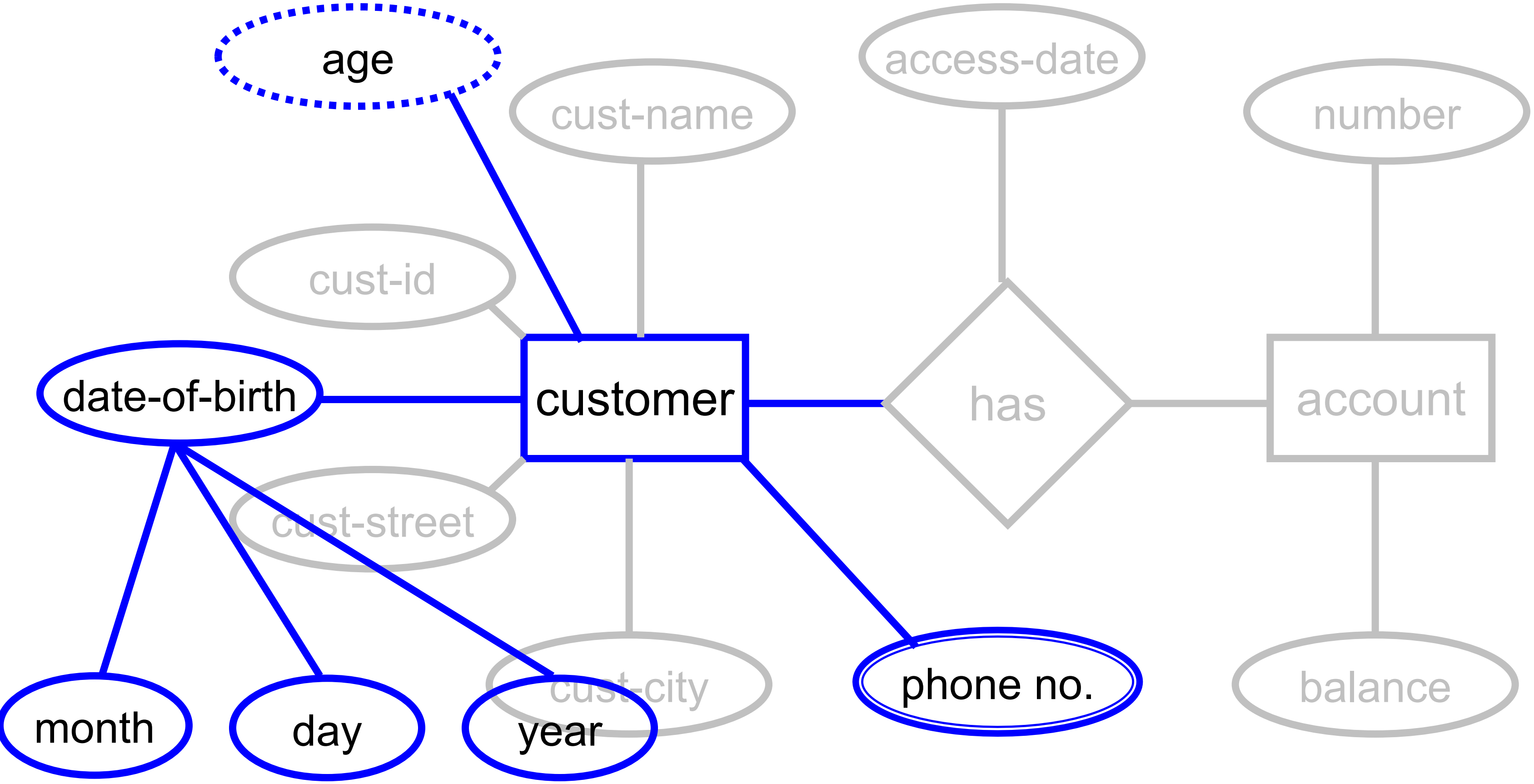
Next: Types of Attributes

- Simple vs Composite
  - Single value per attribute ?
- Single-valued vs Multi-valued
  - E.g. Phone numbers are multi-valued
- Derived
  - If date-of-birth is present, age can be derived
  - Can help in avoiding redundancy, enforcing constraints etc

Types of Attributes



Types of Attributes

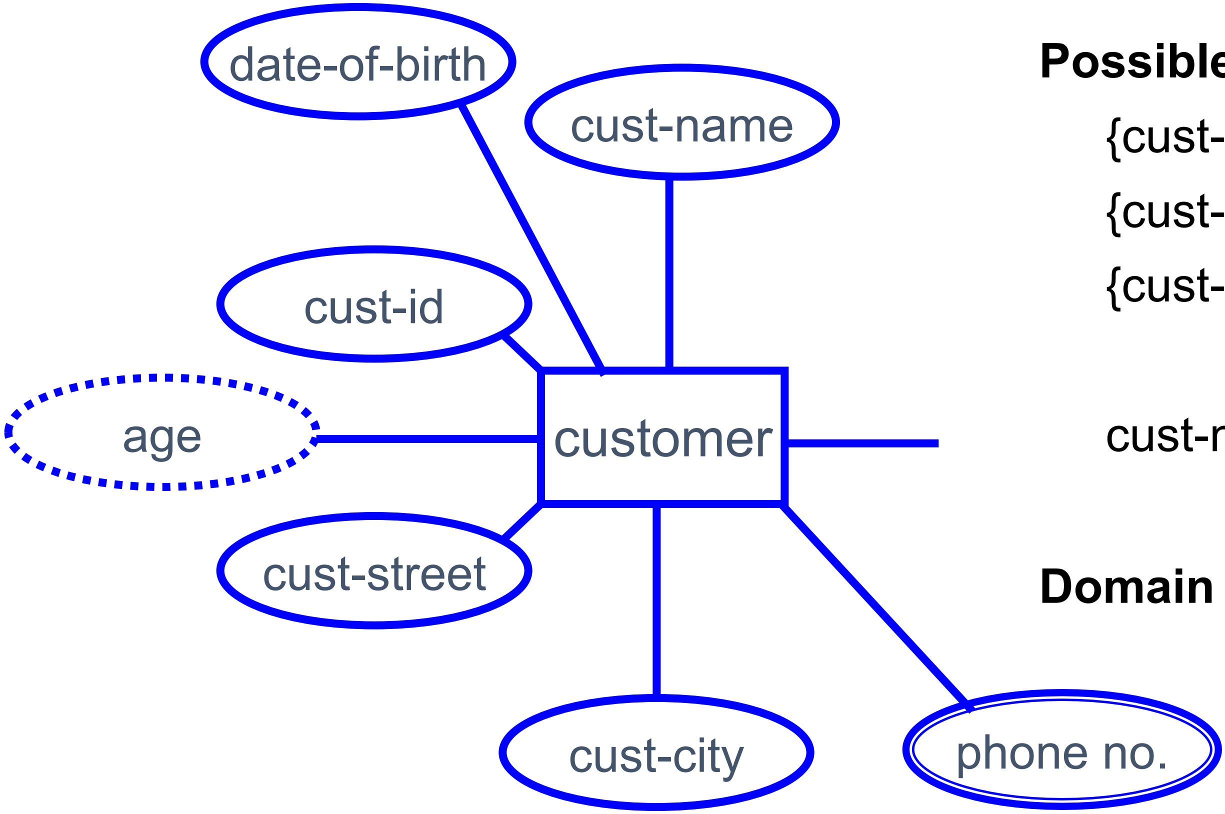


Composite Attribute

Next: Keys

- Key = set of attributes that uniquely identifies an entity or a relationship

Entity Keys



Possible Keys:

{cust-id}

{cust-name, cust-city, cust-street}

{cust-id, age}

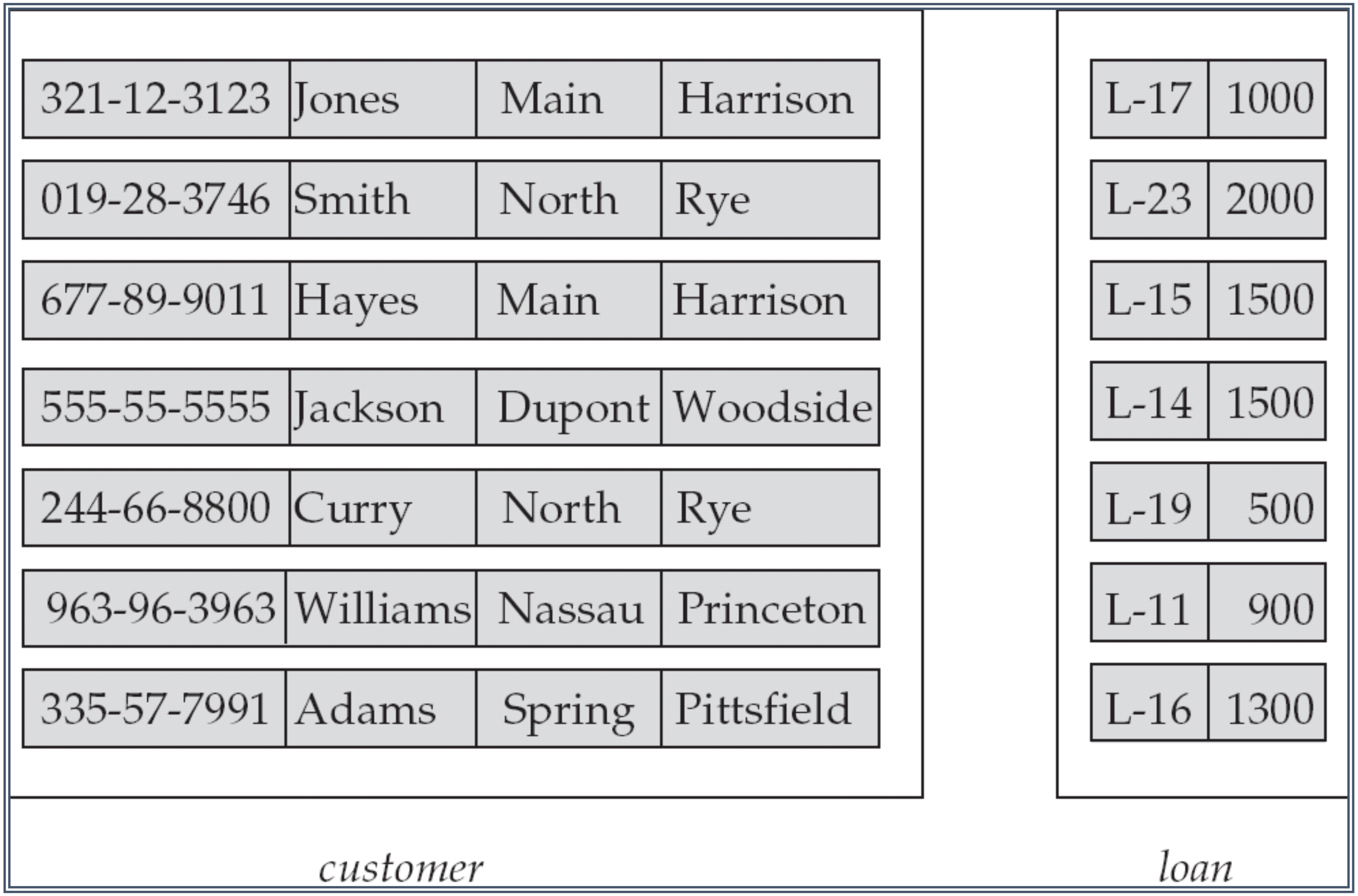
cust-name ?? Probably not.

Domain knowledge dependent !!

Entity Keys

- *Superkey*
  - any attribute set that can distinguish entities
- *Candidate key*
  - a minimal superkey
    - Can’t remove any attribute and preserve key-ness
      - {cust-id, age} not a candidate key since we could have {cust-id} by removing “age” of {cust-id, age}
      - {cust-name, cust-city, cust-street} is
        - assuming cust-name is not unique
- *Primary key*
  - Candidate key chosen as *the* key by DBA
  - Underlined in the ER Diagram

Entity Sets *customer* and *loan*



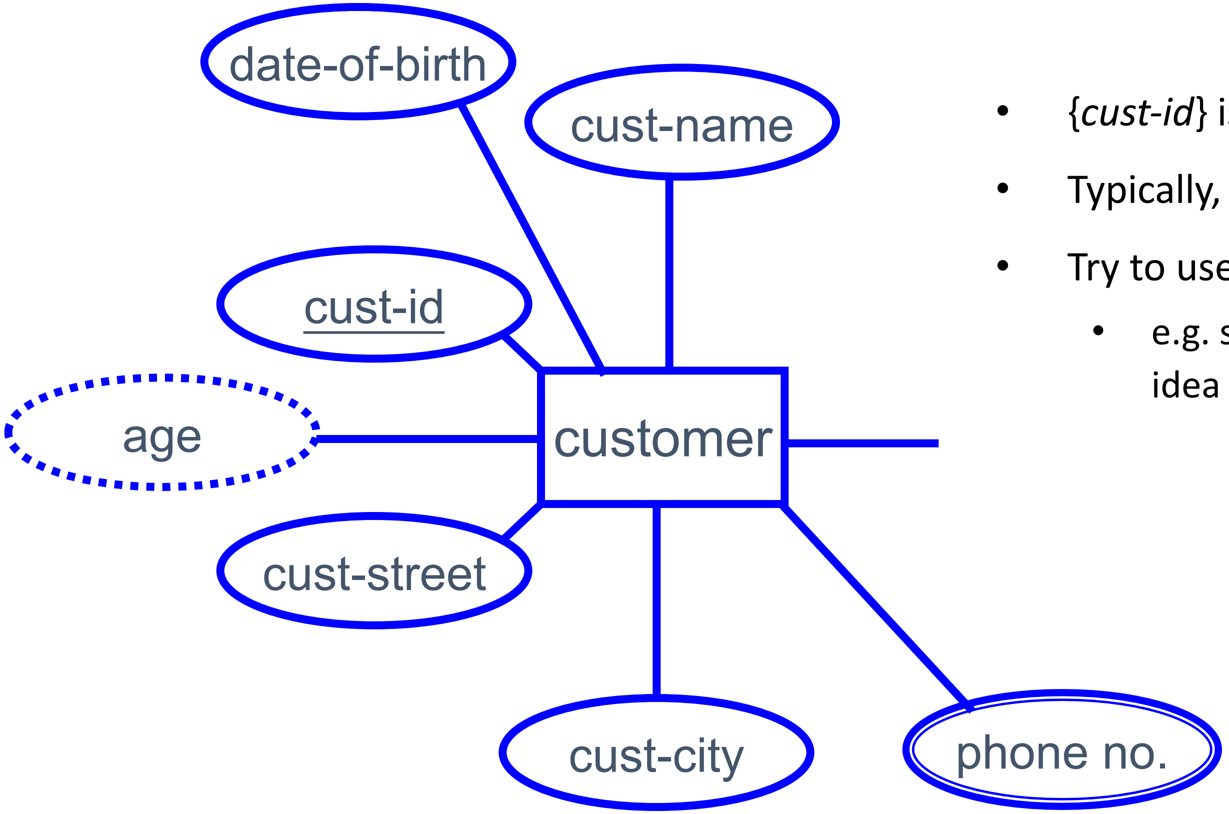
id	Name	street	city
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

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

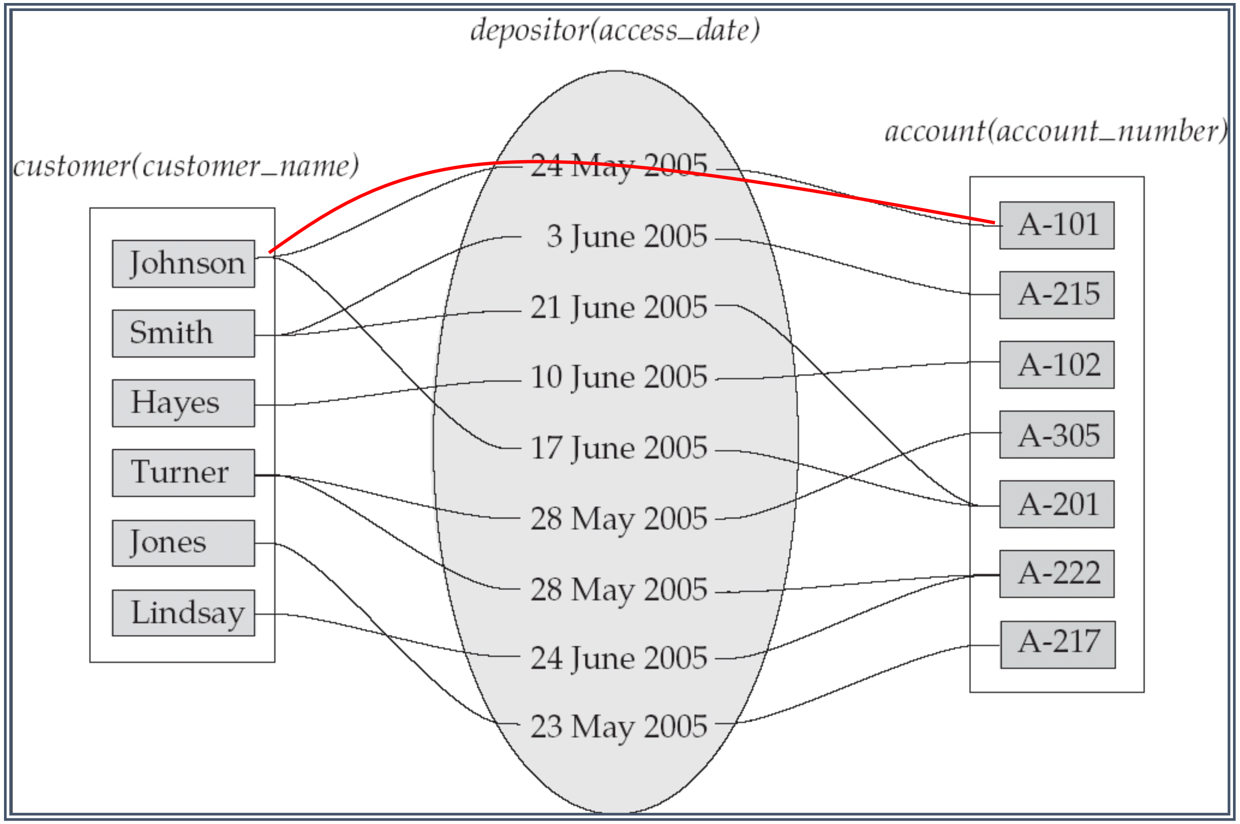
loan

Entity Keys



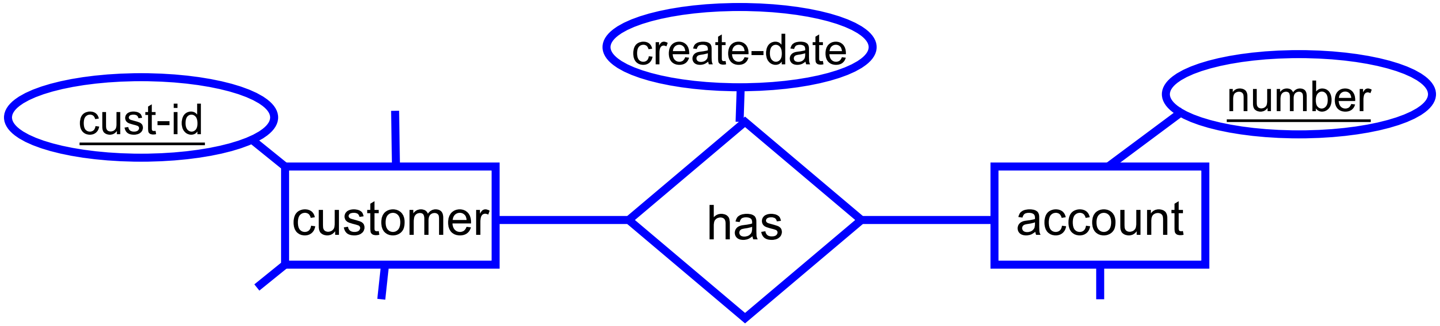
- {*cust-id*} is a natural primary key
- Typically, SSN forms a good primary key
- Try to use a candidate key that rarely changes
  - e.g. something involving address not a great idea

Relationship Set Keys



Relationship Set Keys

- What attributes are needed to represent a relationship completely and uniquely ?
  - Union of primary keys of the entities involved, and relationship attributes



  - *{cust-id, create-date, account number}* describes a relationship completely

Example

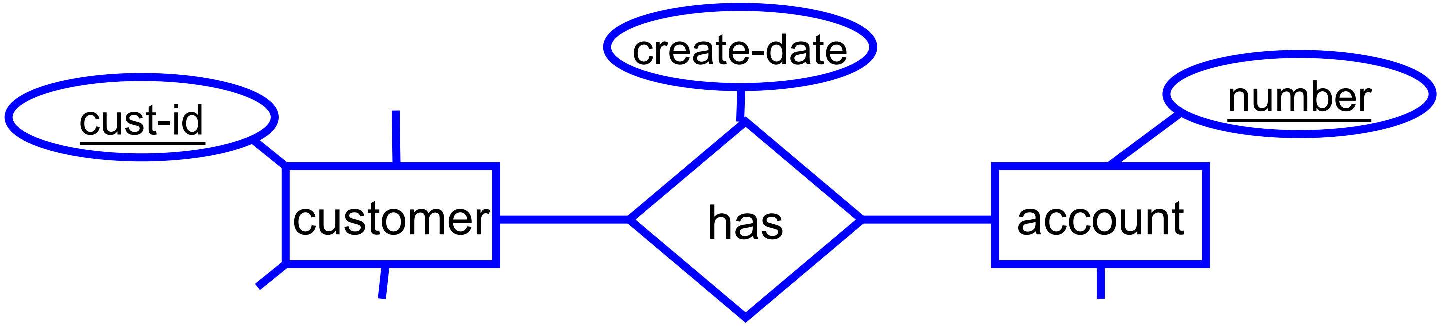
The table below describes the relationship between customers and accounts

Cust-id	Create-date	Account Number
J121454991	2016/03/03	A101
T423323222	2008/05/30	A103
X123234343	2008/01/01	A345
K987655423	2016/03/03	A521

We need to have one “relationship set key” to uniquely identify the above Data. Thus, one possible key is to use “Cust-id, Create-date, account number”.

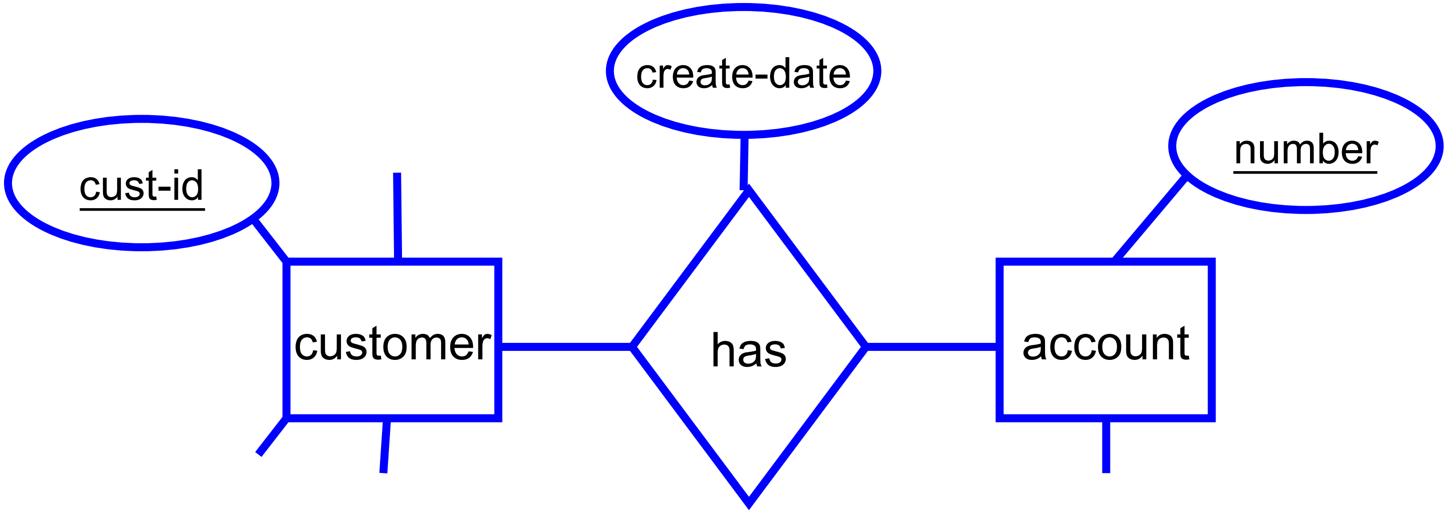
Relationship Set Keys

- Is *{cust-id, create-date, account number}* a candidate key ?
    - No. since we can remove create-date.
    - In fact, union of primary keys of associated entities is always a superkey



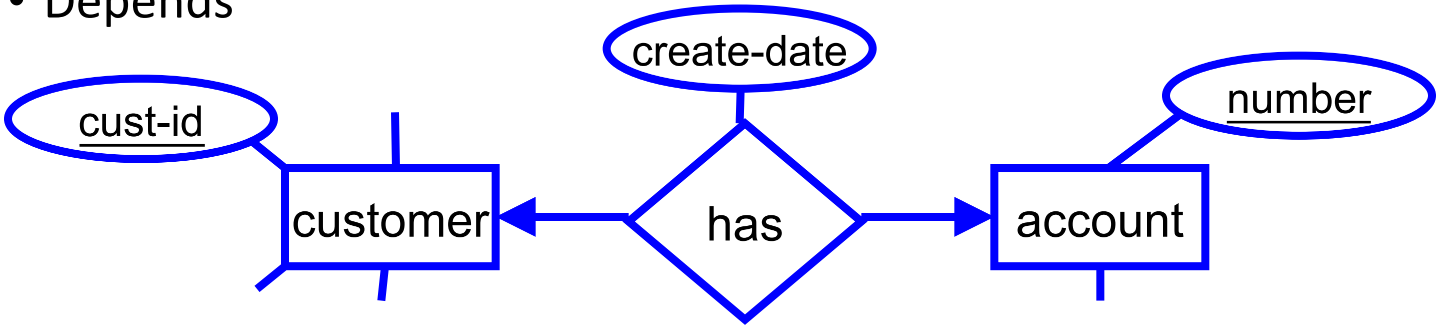
Relationship Set Keys

- Is {cust-id, account-number} a candidate key ?
  - Depends on the cardinality mapping



Relationship Set Keys

- Is {cust-id, account-number} a candidate key ?
  - Depends



- □ If one-to-one relationship, either *{cust-id}* or *{account-number}* sufficient
  - Since a given *customer* can only have one *account*, she can only participate in one relationship

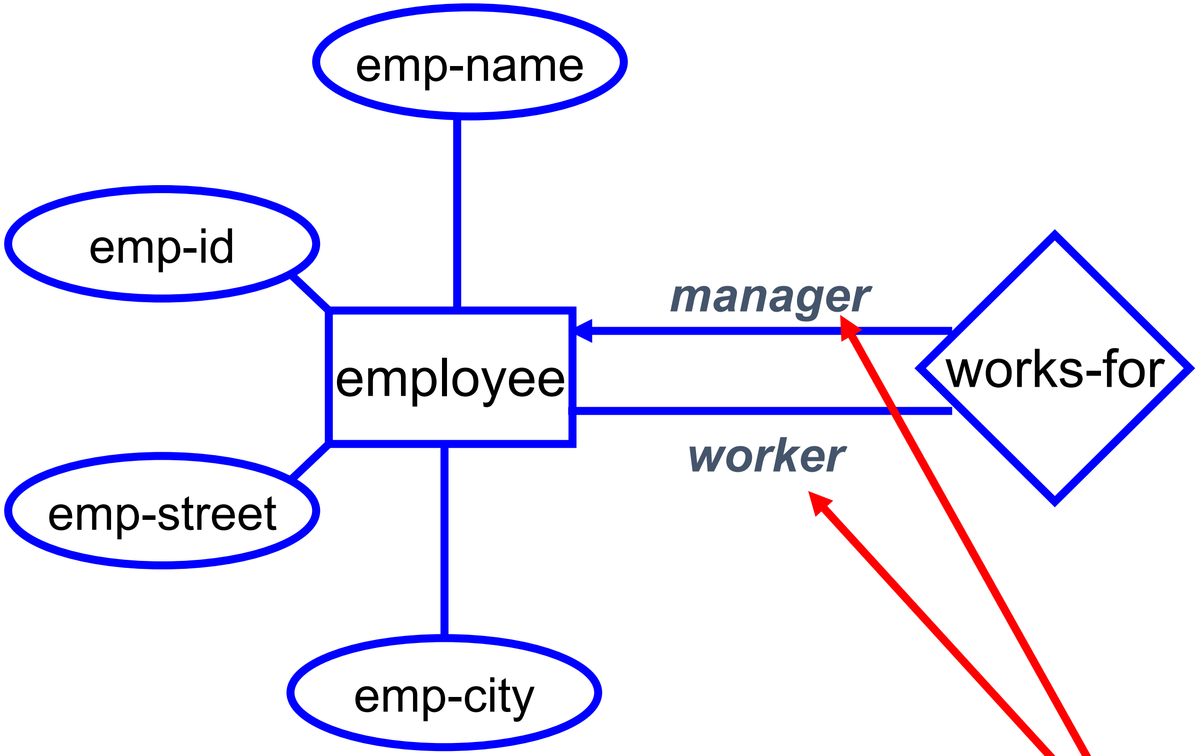
Relationship Set Keys

- General rule for binary relationships
  - one-to-one: primary key of either entity set
  - one-to-many: primary key of the entity set on the many side
  - many-to-many: union of primary keys of the associate entity sets
- n-ary relationships
  - More complicated rules

Next: Recursive Relationships

- Sometimes a relationship associates an entity set to itself

Recursive Relationships



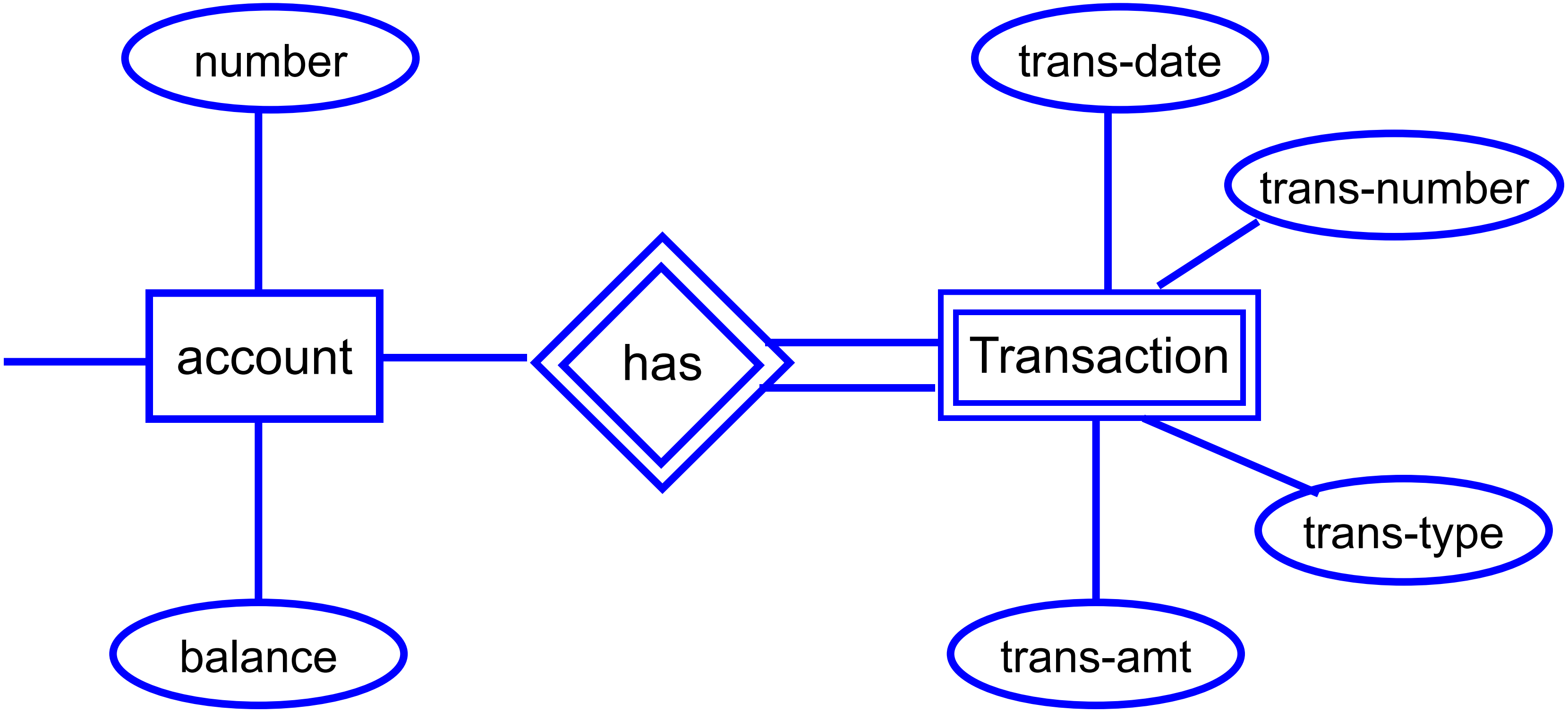
Must be declared with roles

Next: Weak Entity Sets

- An entity set without enough attributes to have a primary key
- E.g. Transaction Entity
  - Attributes:
    - transaction-number, transaction-date, transaction-amount, transaction-type
    - transaction-number: may not be unique across accounts

Weak Entity Sets

Each account has many transactions (e.g., withdraw) and for Each account, trans-number is ranged from 1..*



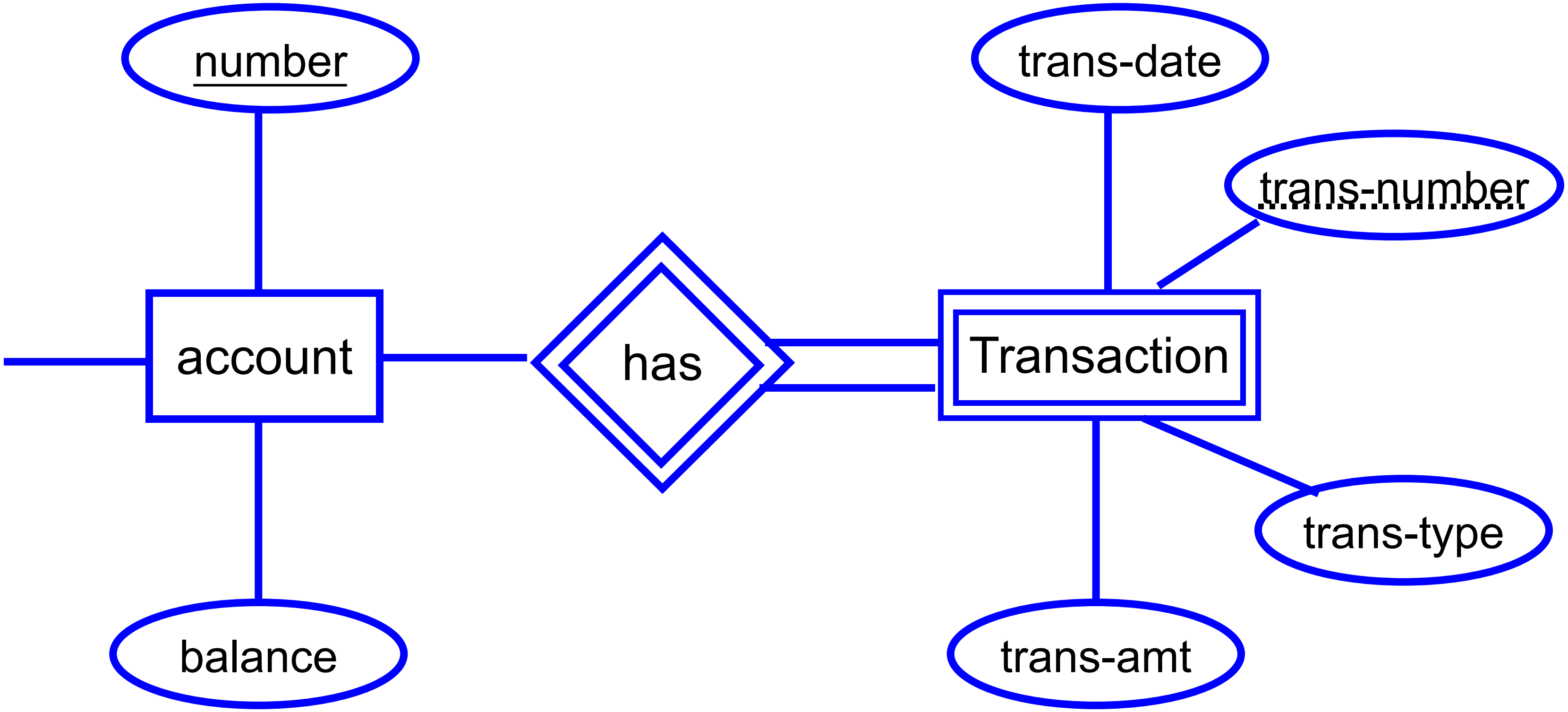
In the above case, different accounts may have the same trans-number

Weak Entity Sets

- A weak entity set must be associated with an identifying or owner entity set
- Account is the owner entity set for Transaction

Weak Entity Sets

Discriminator: A set of attributes that can be used to discriminate in the transaction entity set

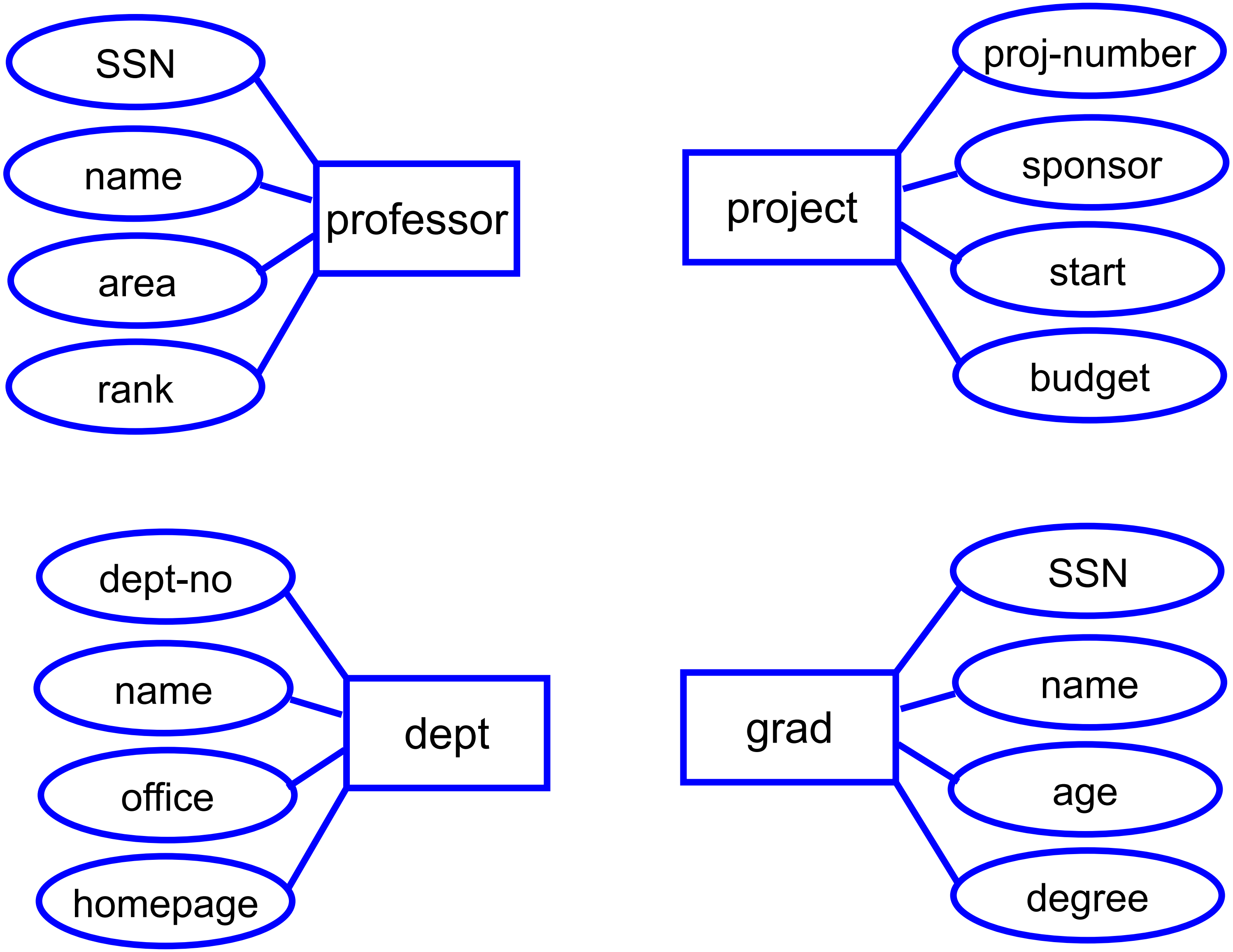


Weak Entity Sets

- Primary key:
    - Primary key of the associated strong entity + discriminator attribute set
    - For Transaction:
        - *{account-number, transaction-number}*

Example Design

- We will model a university database
  - Main entities:
    - Professor
    - Projects
    - Departments
    - Graduate students
    - etc…



professor

SSN

name

area

rank

project

proj-number

sponsor

start

budget

dept

dept-no

name

office

homepage

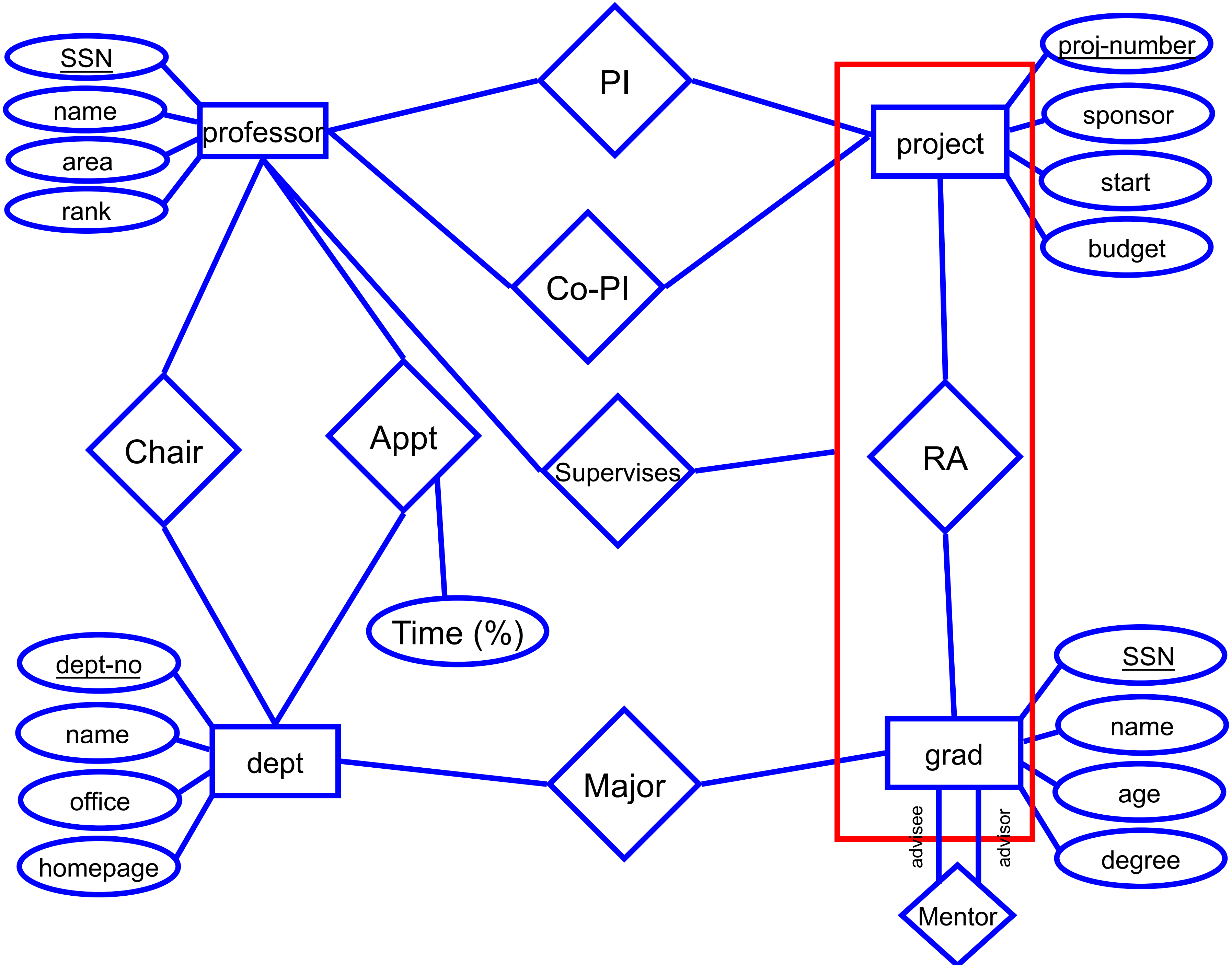
grad

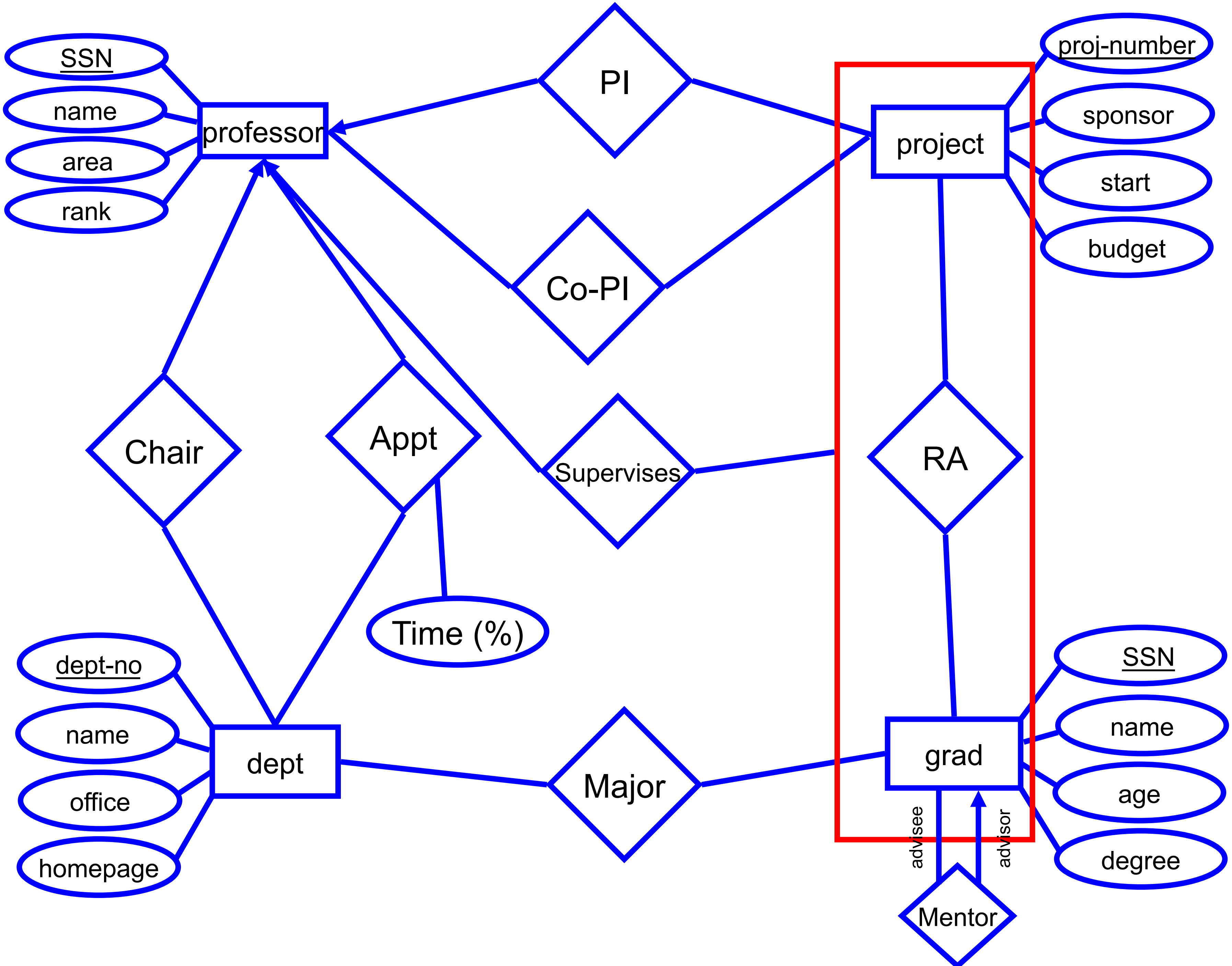
SSN

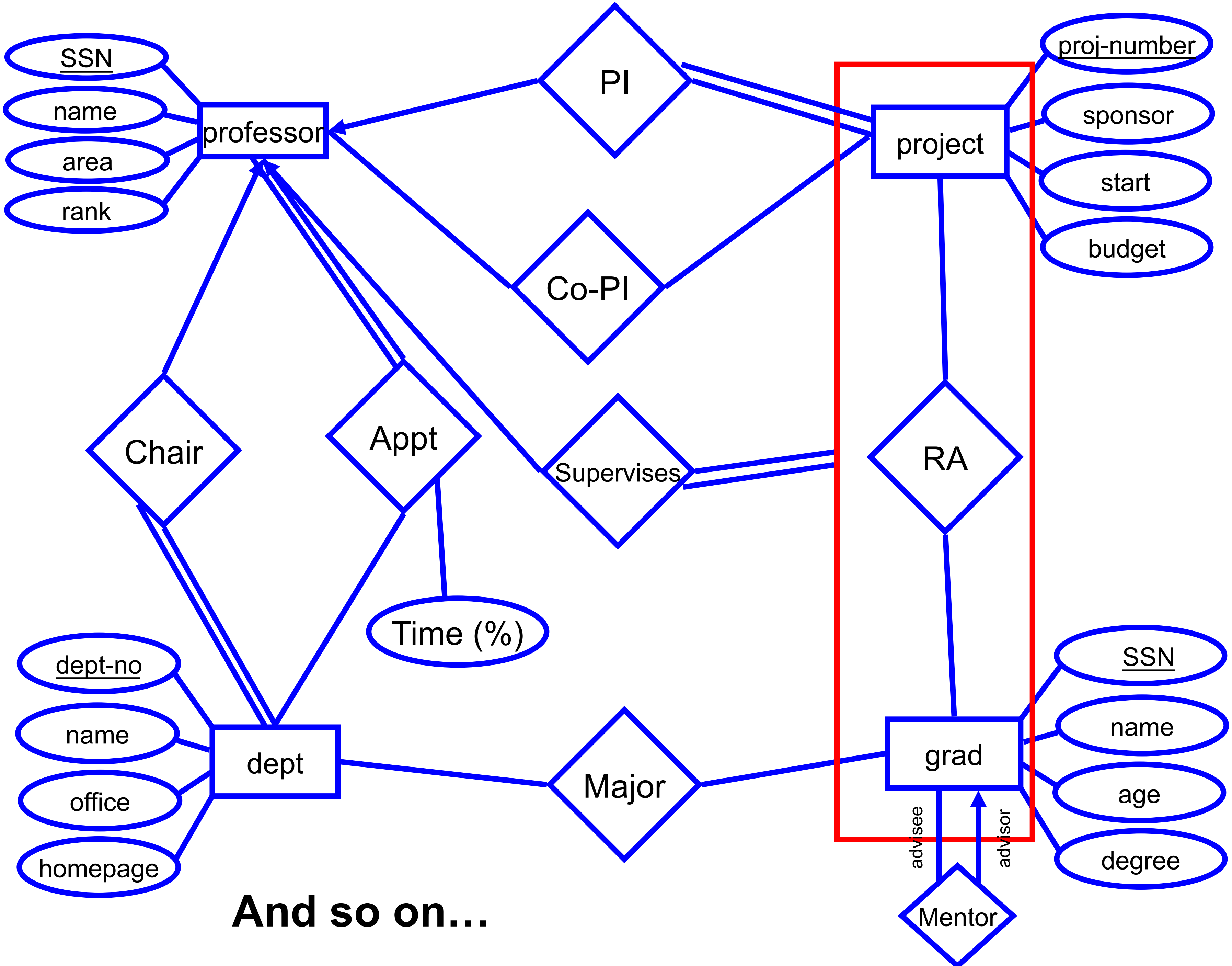
name

age

degree


SSN
name
area
rank
professor
PI
Co-PI
project
proj-number
sponsor
start
budget
Chair
Appt
Supervises
RA
Time (%)
dept-no
name
office
homepage
dept
Major
grad
SSN
name
age
degree
advisee
advisor
Mentor


SSN
name
area
rank
professor
PI
Co-PI
proj-number
sponsor
start
budget
project
Chair
Appt
Supervises
RA
Time (%)
dept-no
name
office
homepage
dept
Major
grad
SSN
name
age
degree
advisee
advisor
Mentor


SSN
name
area
rank
professor
PI
Co-PI
project
proj-number
sponsor
start
budget
Chair
Appt
Supervises
RA
Time (%)
dept-no
name
office
homepage
dept
Major
grad
SSN
name
age
degree
advisee
advisor
Mentor

And so on…

Thoughts…

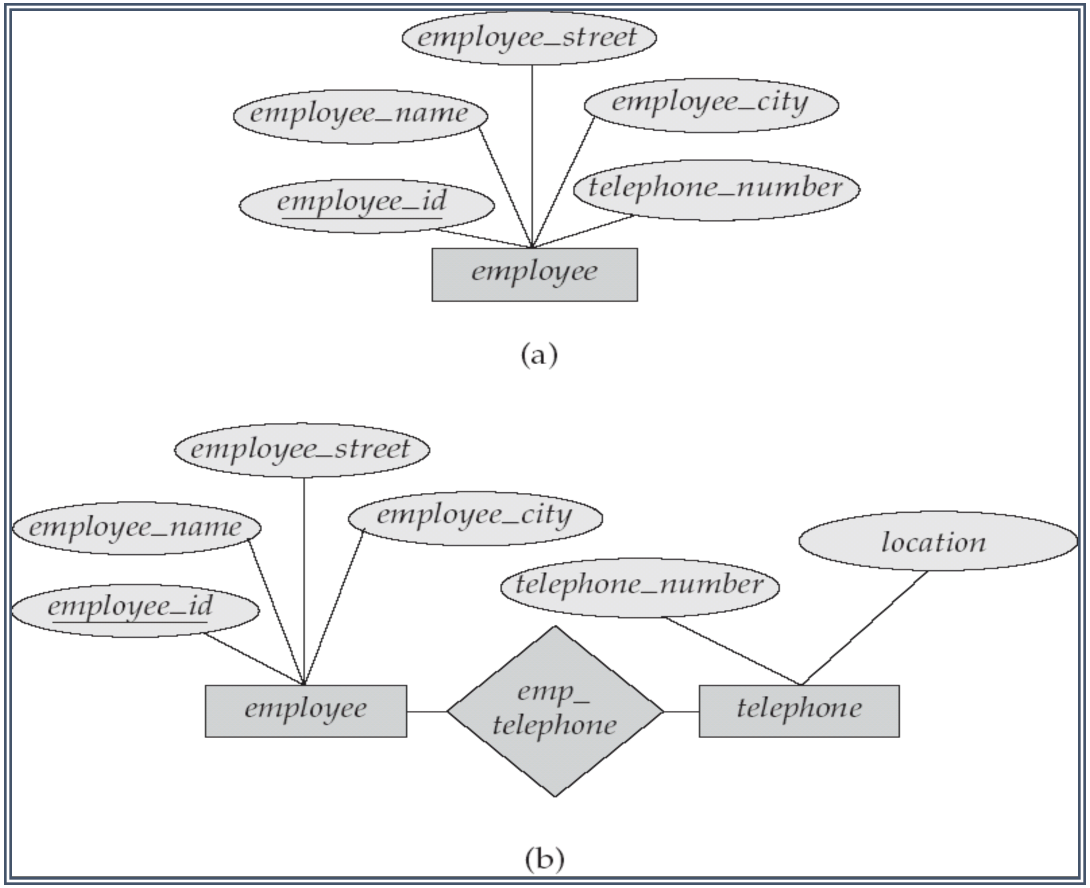
- Nothing about actual data
  - How is it stored ?
- No talk about the query languages
  - How do we access the data ?
- Semantic vs Syntactic Data Models
  - Remember: E/R Model is used for conceptual modeling
  - Many conceptual models have the same properties
- They are much more about representing the knowledge than about database storage/querying

Thoughts…

- Basic design principles
  - Faithful
    - Must make sense
  - Satisfies the application requirements
  - Models the requisite domain knowledge
    - If not modeled, lost afterwards
  - Avoid redundancy
    - Potential for inconsistencies
  - Go for simplicity
- Typically an iterative process that goes back and forth

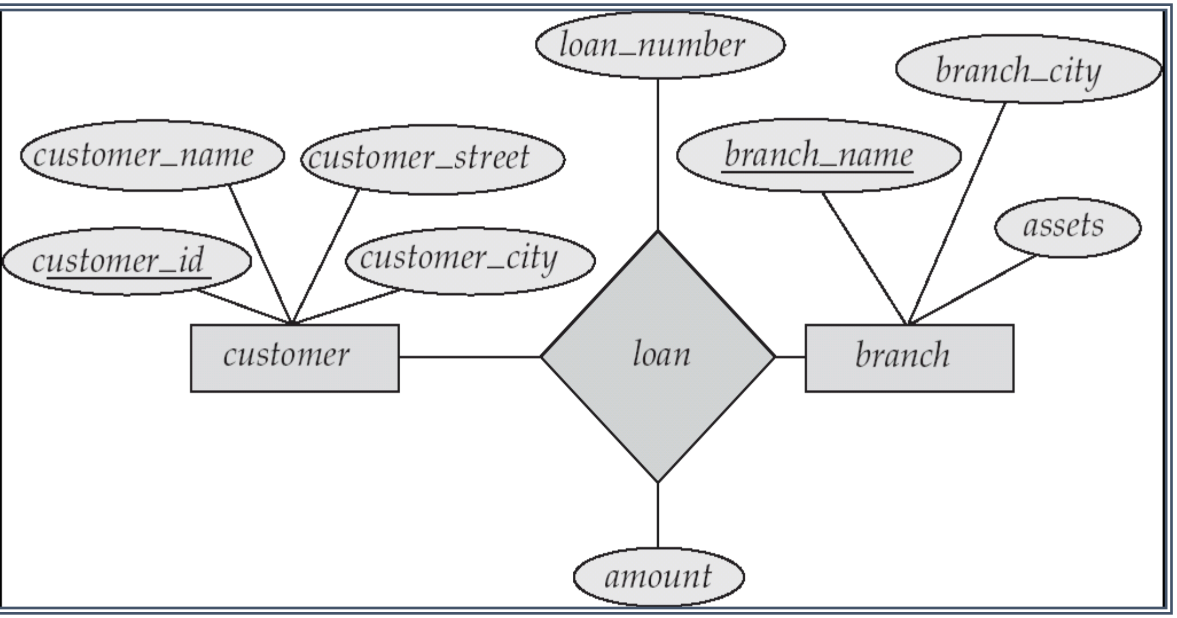
Design Issues

- Entity sets vs attributes
  - Depends on the semantics of the application
  - *If phones have other attributes, it is better to present as an entity)*



Design Issues

- Entity sets vs Relationship sets
  - Consider *loan (Not good to represent many customers hold a loan)*



chris	100	300	CS
Tsao	100	300	CS
John	100	300	CS

Design Issues

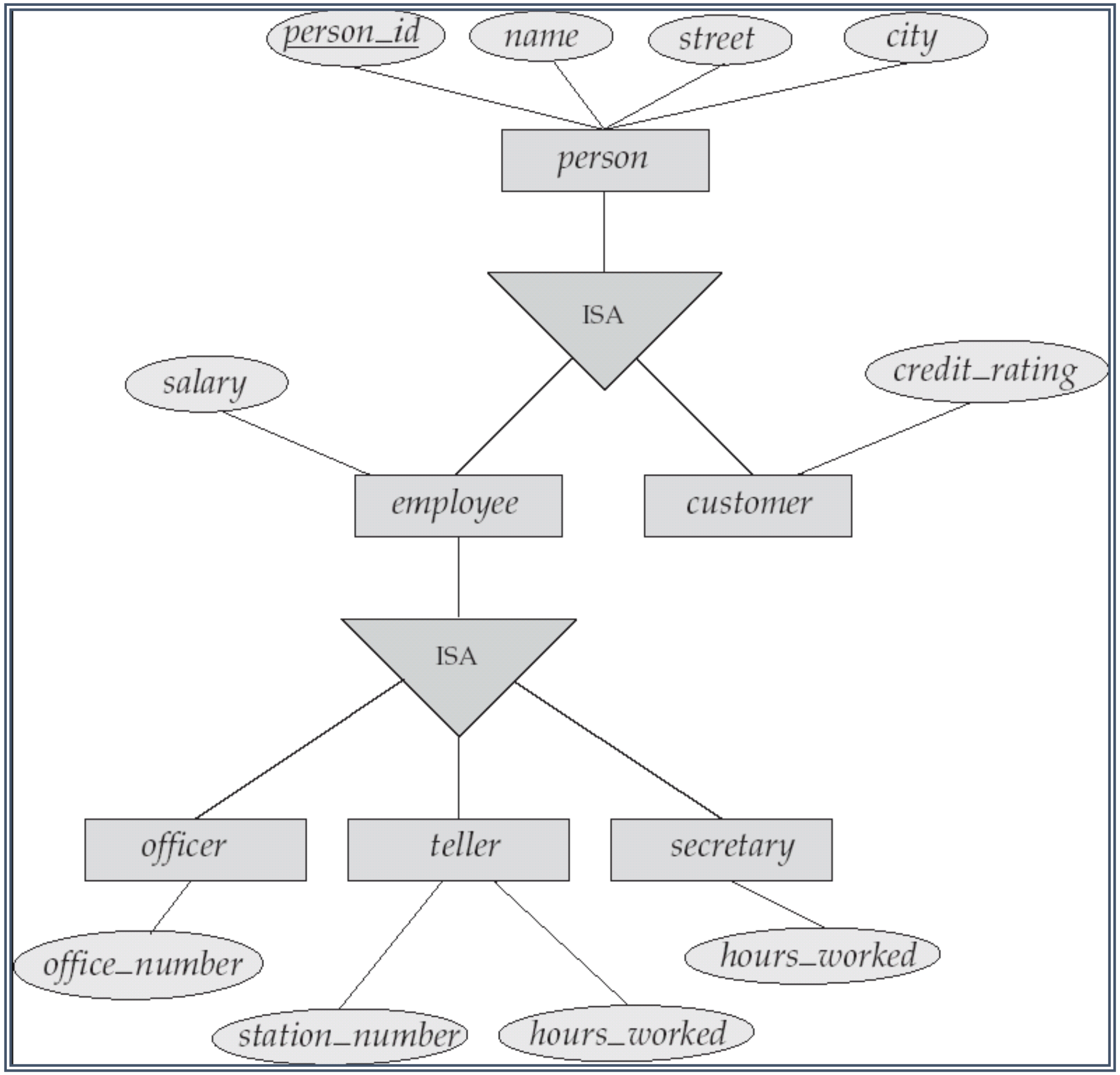
- N-ary vs binary relationships
    - Possible to avoid n-ary relationships, but there are some cases where it is advantageous to use them

Next: Specialization

- Consider entity person:
  - Attributes: name, street, city
- Further classification:
  - customer
    - Additional attributes: customer-id, credit-rating
  - employee
    - Additional attributes: employee-id, salary

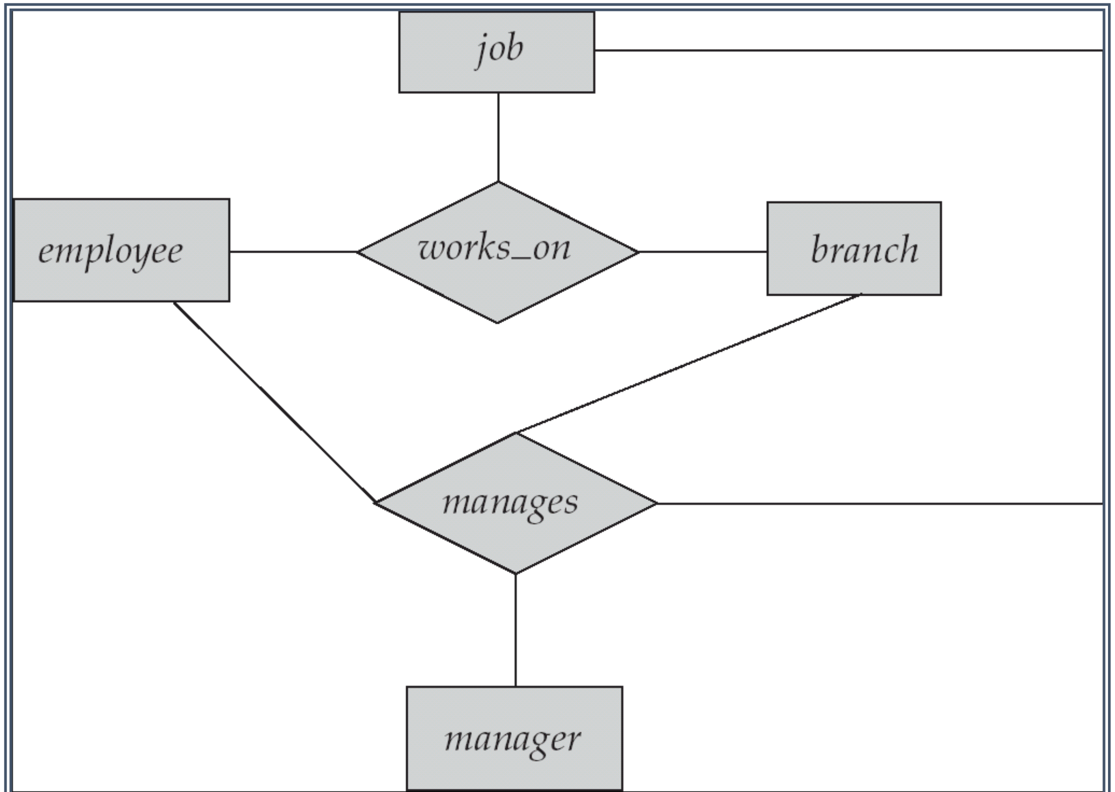
- Note similarities to object-oriented programming

Example

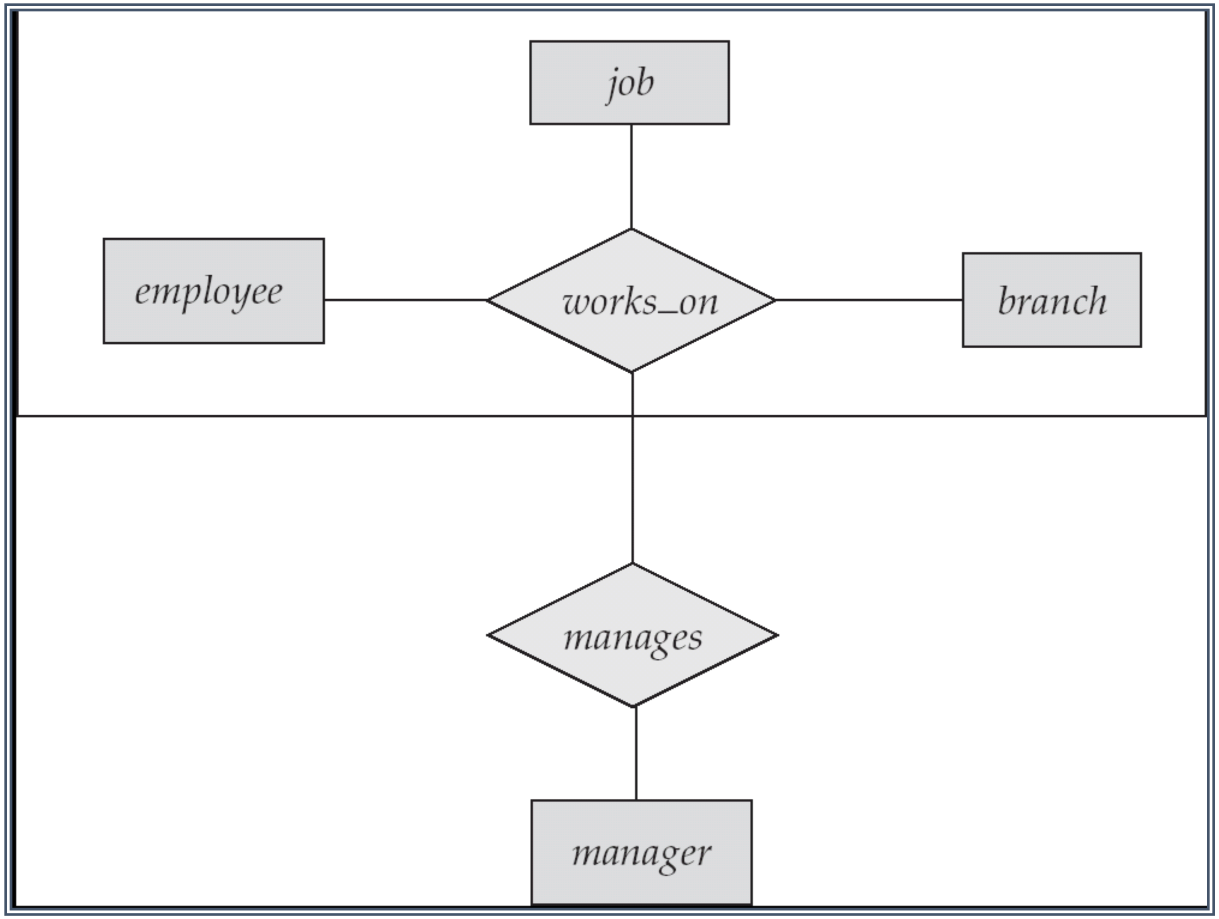

person_id
name
street
city
person
ISA
salary
credit_rating
employee
customer
ISA
officer
teller
secretary
office_number
station_number
hours_worked
hours_worked

Aggregation

- Suppose we want to record managers for tasks performed by an employee at a branch



Aggregation (cont’d)



Break

- Entity-relationship Model
  - Intuitive diagram-based representation of domain knowledge, data properties etc…
  - Two key concepts:
    - Entities
    - Relationships
  - We also looked at:
    - Relationship cardinalities
    - Keys
    - Weak entity sets

Relational Data Model

Introduced by Ted Codd (late 60’s – early 70’s)

- *Before = “Network Data Model” (Cobol as DDL, DML)*
- *Very contentious: Database Wars (Charlie Bachman vs. Mike Stonebraker)*

Relational data model contributes:

1. *Separation of logical, physical data models (data independence)*
2. *Declarative query languages*
3. *Formal semantics*
4. *Query optimization (key to commercial success)*

1st prototypes:

- *Ingres → CA*
- *Postgres → Illustra → Informix → IBM*
- *System R → Oracle, DB2*

Key Abstraction: Relation

Account =

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Terms:

- Tables (aka: Relations)

Why called Relations?

Why Called Relations?

Mathematical relations

Given sets: $R = \{1, 2, 3\}$, $S = \{3, 4\}$

- $R \times S = \{ (1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4) \}$
- *A* ***relation*** *on R, S is any subset* ($\subseteq$) *of* $R \times S$

  (e.g: $\{ (1, 4), (3, 4)\}$)

Database relations

Given attribute domains

Branches = { *Downtown, Brighton, …* }
Accounts = { *A-101, A-201, A-217, …* }
Balances = *R*

Account $\subseteq$ *Branches* × *Accounts* × *Balances*

{ (*Downtown, A-101, 500*),
(*Brighton, A-201, 900*),
(*Brighton, A-217, 500*) }

Relations

Account =

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Considered equivalent to…

{ (Downtown, A-101, 500),
(Brighton, A-201, 900),
(Brighton, A-217, 500) }

Relational database semantics defined in terms of mathematical relations

So…

- That’s the basic relational model
- That’s it ?
    - What about semantic information ?
        - Relationships between entities ?
    - What about the constraints ?
    - How do we represent one-to-one vs many-to-one relationships ?
    - Those constraints are all embedded in the schema

Keys and Relations

- Recall:
  - Keys: Sets of attributes that allow us to identify entities
  - Very loosely speaking, tuples are viewed entities
- Just as in E/R Model:
  - Superkeys, candidate keys, and primary keys

Keys

- Superkey
  - set of attributes of table for which every row has distinct set of values
- Candidate key
  - Minimal such set of attributes
- Primary key
  - DB Chosen Candidate key
  - Plays a very important role
    - E.g. relations typically sorted by this

Keys

- Also act as integrity constraints
  - i.e., guard against illegal/invalid instance of given schema

e.g., Branch = (bname, bcity, assets)

bname	bcity	assets
Brighton	Brooklyn	5M
Brighton	Boston	3M

Invalid

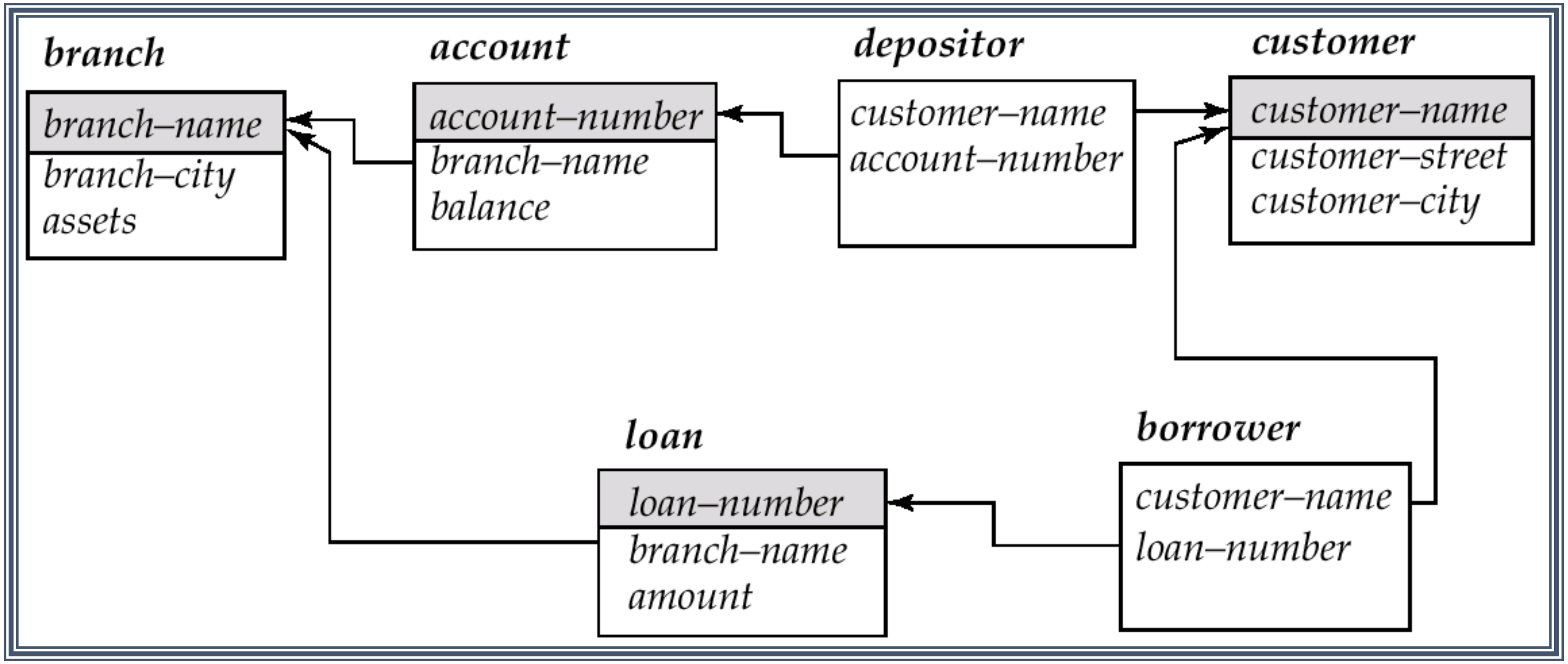
Keys

- In fact, keys are one of the primary ways to enforce constraints/structure
- Consider a one-to-many relationship e.g.
  - Between customers and accounts
  - The relational model will be:
    - Customers(custid, custname,…)
    - Accounts(accountid, custid, balance,…)
  - Allows for multiple accounts per customer, but **not multiple customers per account**
    - Not possible to store such information (Key is accountid in Accounts table)
- In other words, constraints will lead to less representation power
  - Contrast with:
    - Customers(custid, custname,…)
    - Accounts(accountid, balance,…)
    - CustomerHasAccounts(custid, accountid)

More on Keys

- Determining Primary Keys
  - If relation schema derived from E-R diagrams, we can determine the primary keys using the original entity and relationship sets
  - Otherwise, same way we do it for E-R diagrams
    - Find candidate keys (minimal sets of attributes that can uniquely identify a tuple)
    - Designate one of them to be primary key
- Foreign Keys
  - If a relation schema includes the primary key of another relation schema, that attribute is called the *foreign key*

Schema Diagram for the Banking Enterprise



branch

branch–name
branch–city
assets

account

account–number
branch–name
balance

depositor

customer–name
account–number

customer

customer–name
customer–street
customer–city

loan

loan–number
branch–name
amount

borrower

customer–name
loan–number