CMSC 424 – Database design
Lecture 3:
Entity-Relationship Model

Book: Chap. 1 and 6

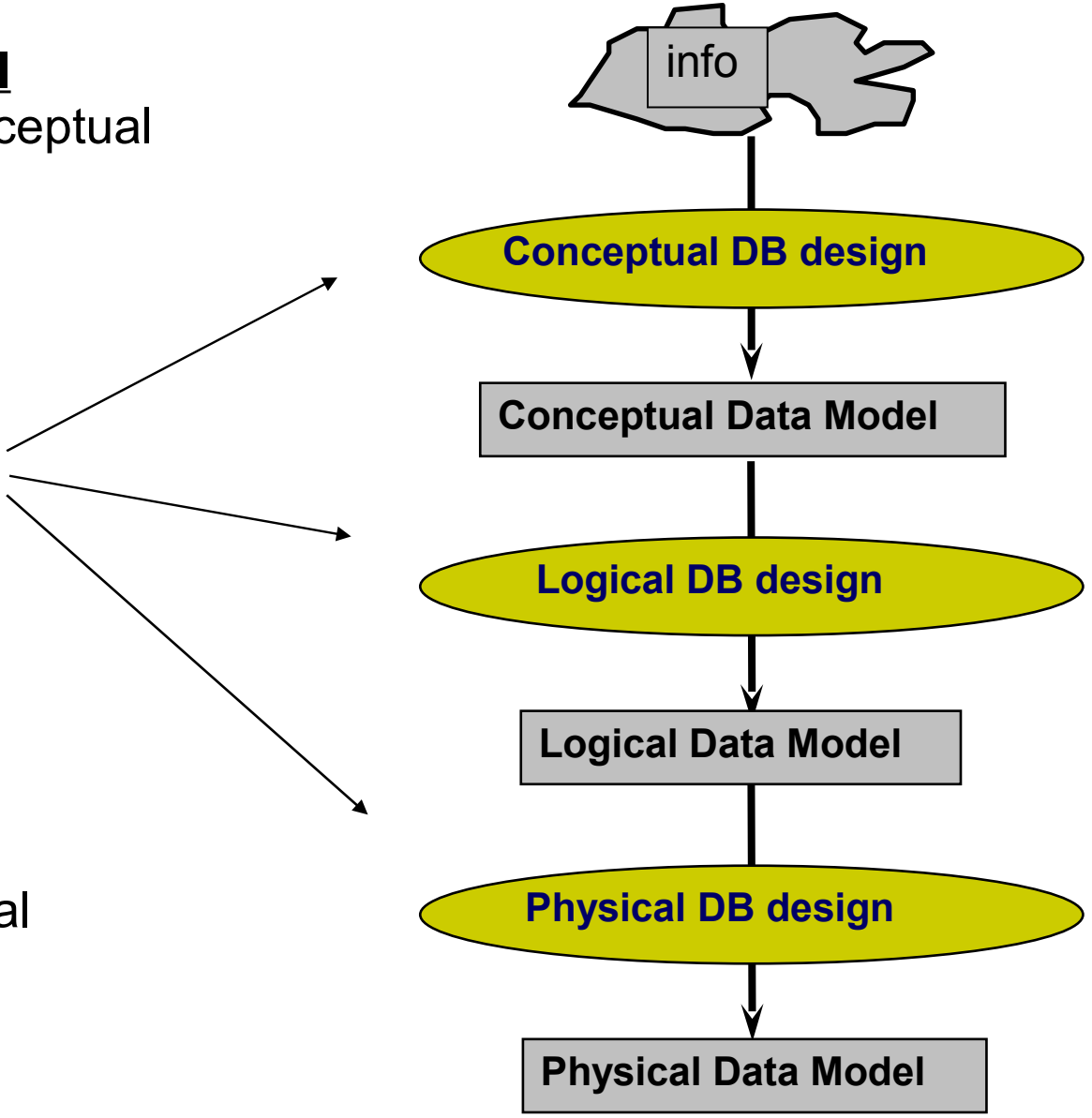
Mihai Pop

Database Design Steps

Entity-relationship Model

Typically used for conceptual database design

Three Levels of Modeling

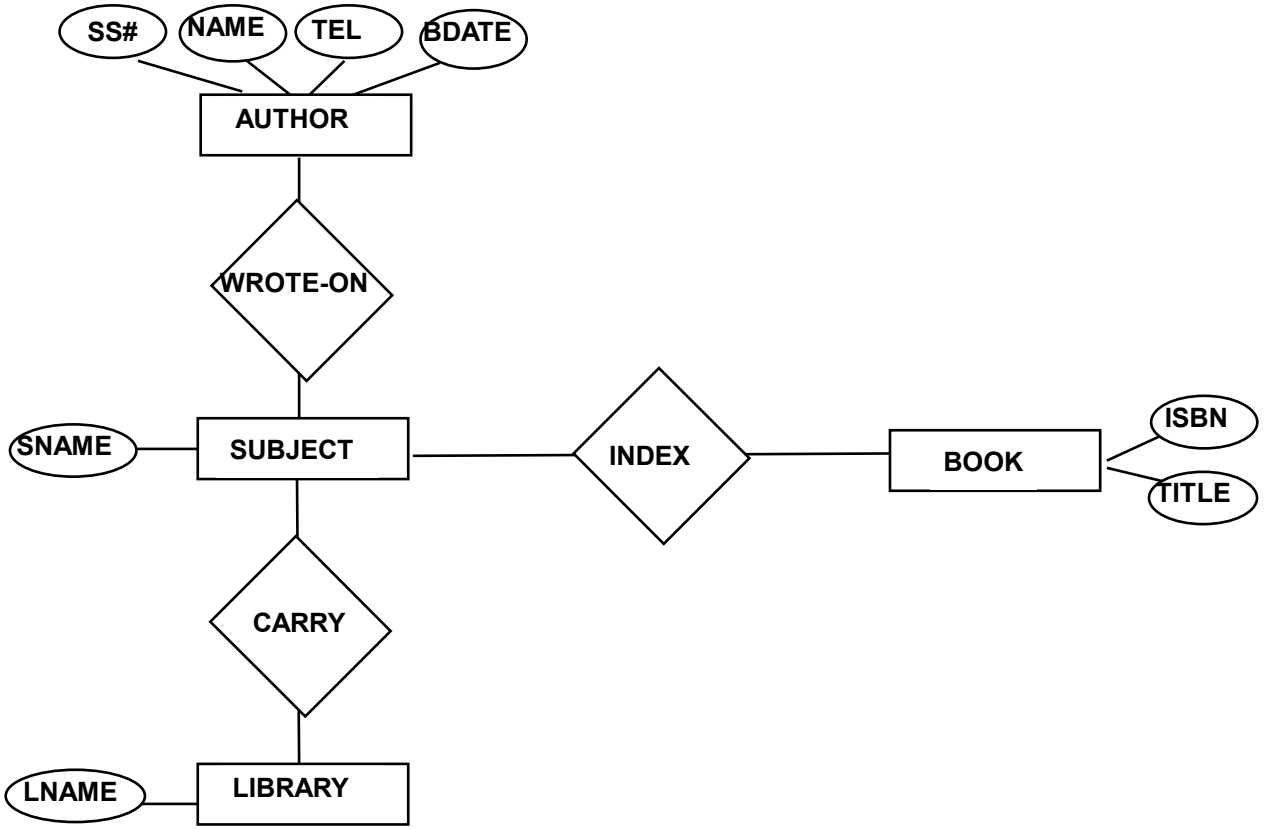


Relational Model

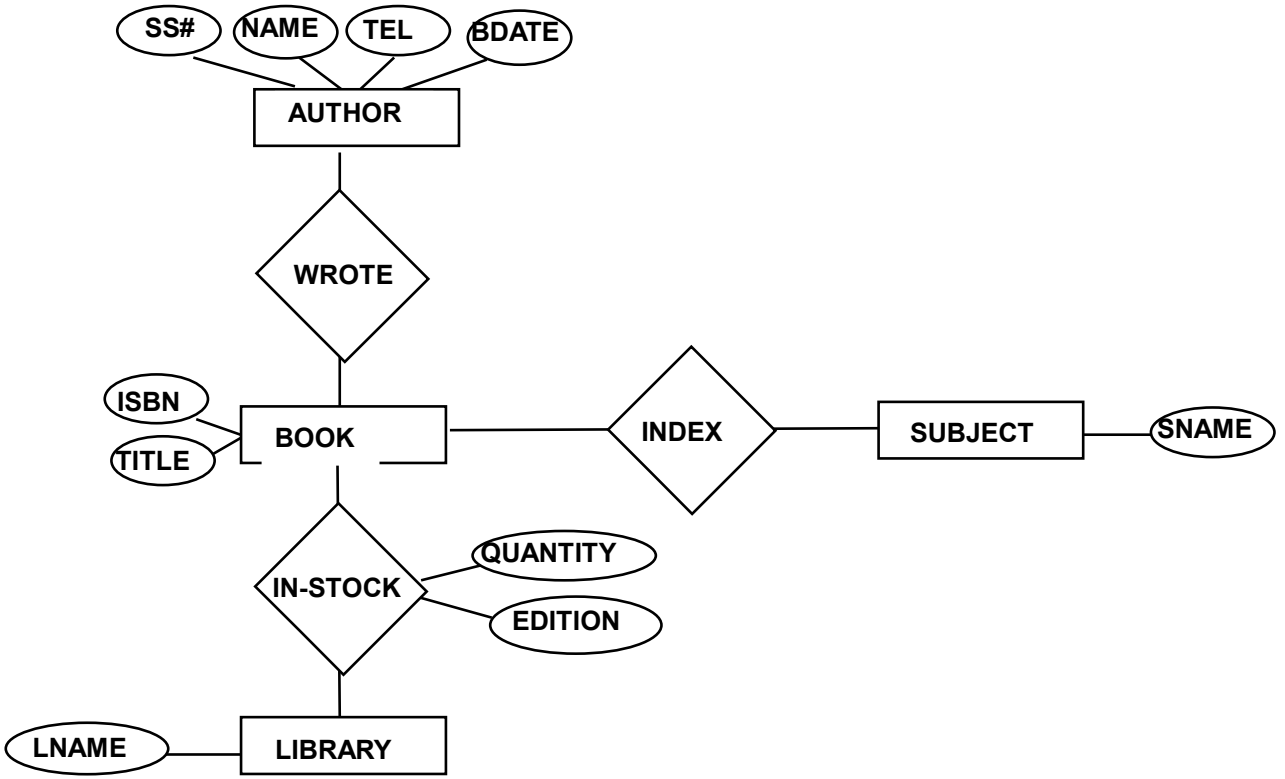
Typically used for logical database design

Problems in our First Design

- does not capture the fact that a library carries books of a specific author
- does not capture the fact that a library carries a specific book
- does not capture the fact that an author has written a specific book
- does not store which edition of the book the library has, how many copies, etc.



2nd Attempt to the Library Design

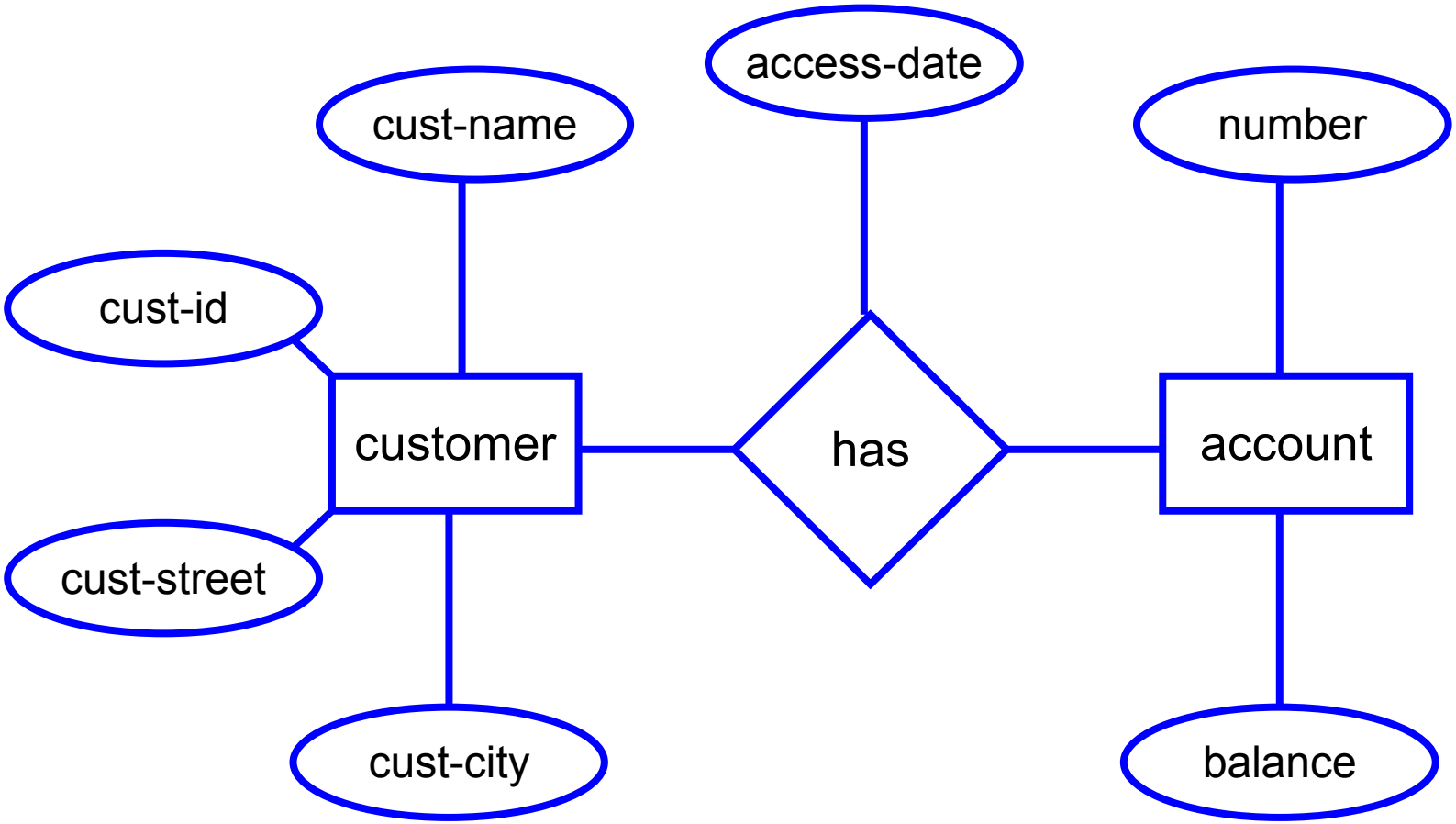


- Much better

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



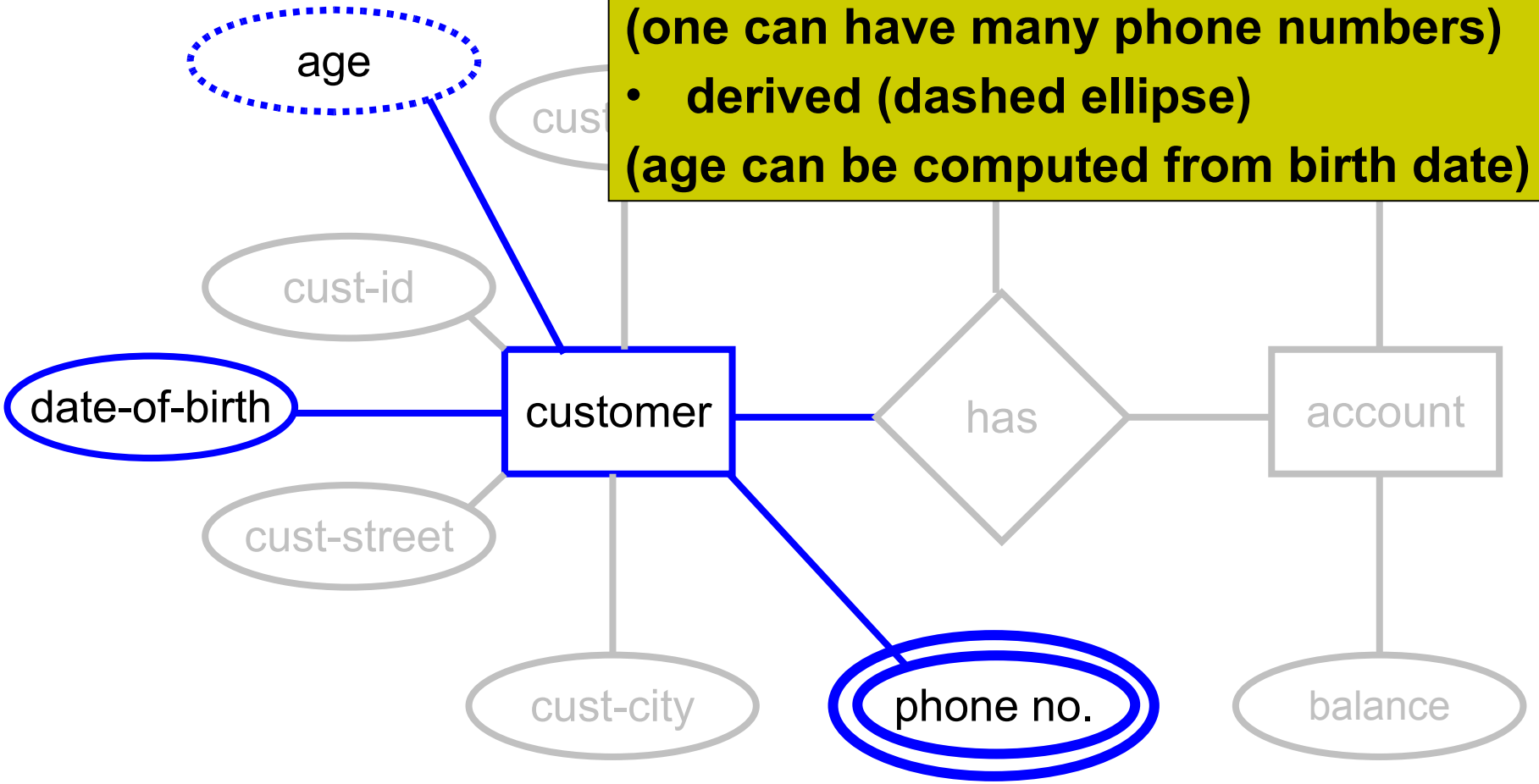
Types of Attributes

- **multi-valued (double ellipse)**

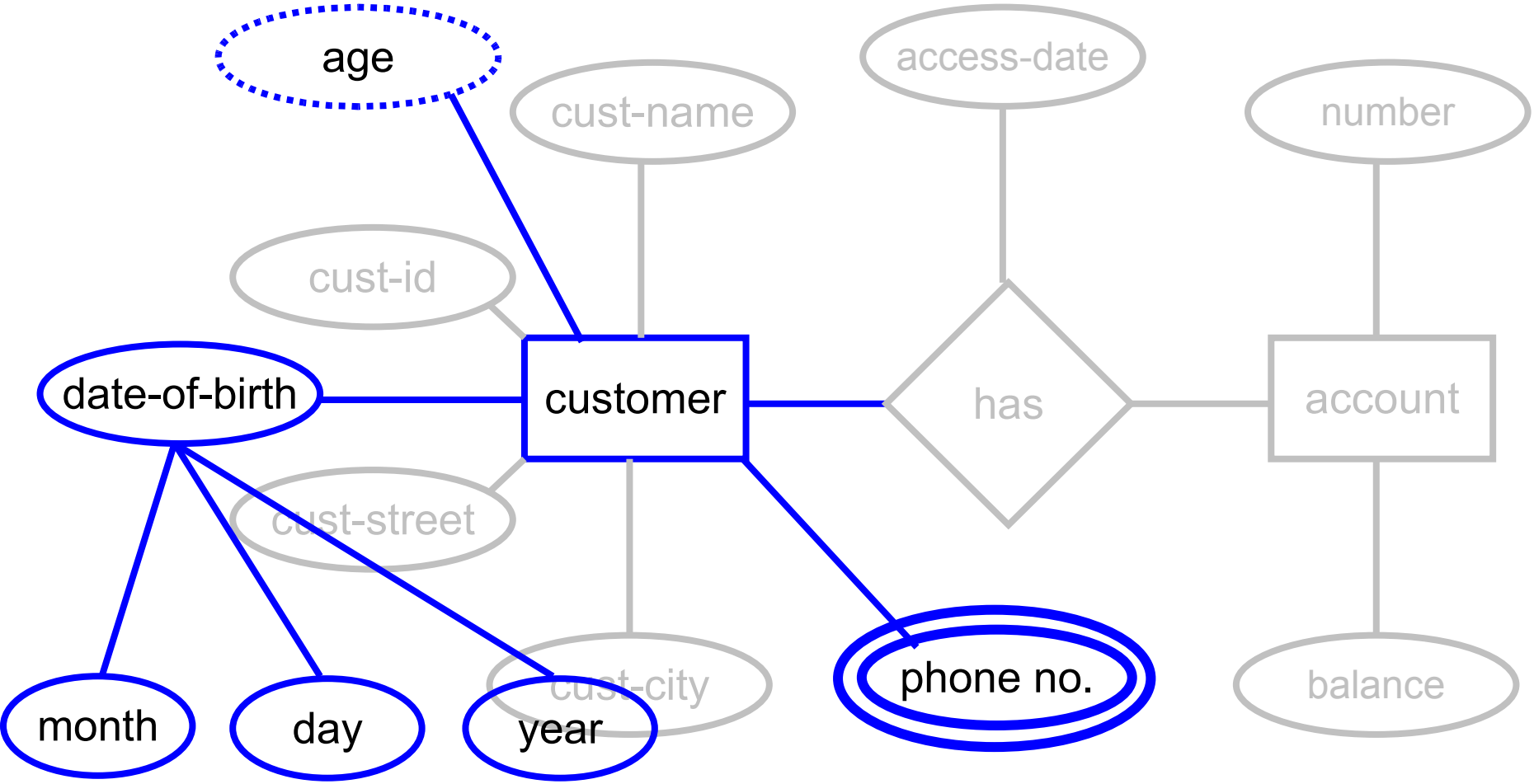
(one can have many phone numbers)

- **derived (dashed ellipse)**

(age can be computed from birth date)



Types of Attributes

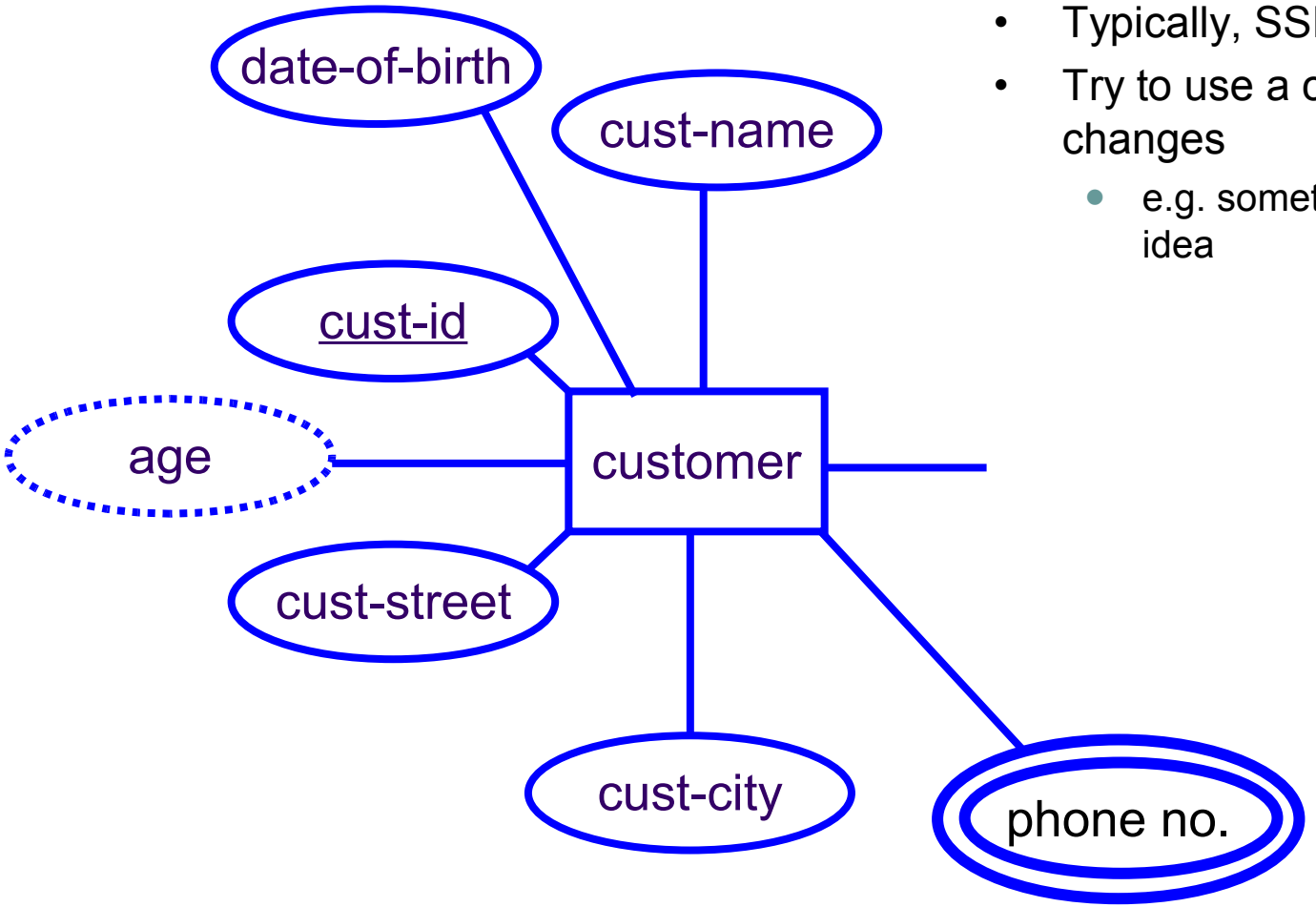


Composite Attribute

Next: Keys

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

Entity Keys

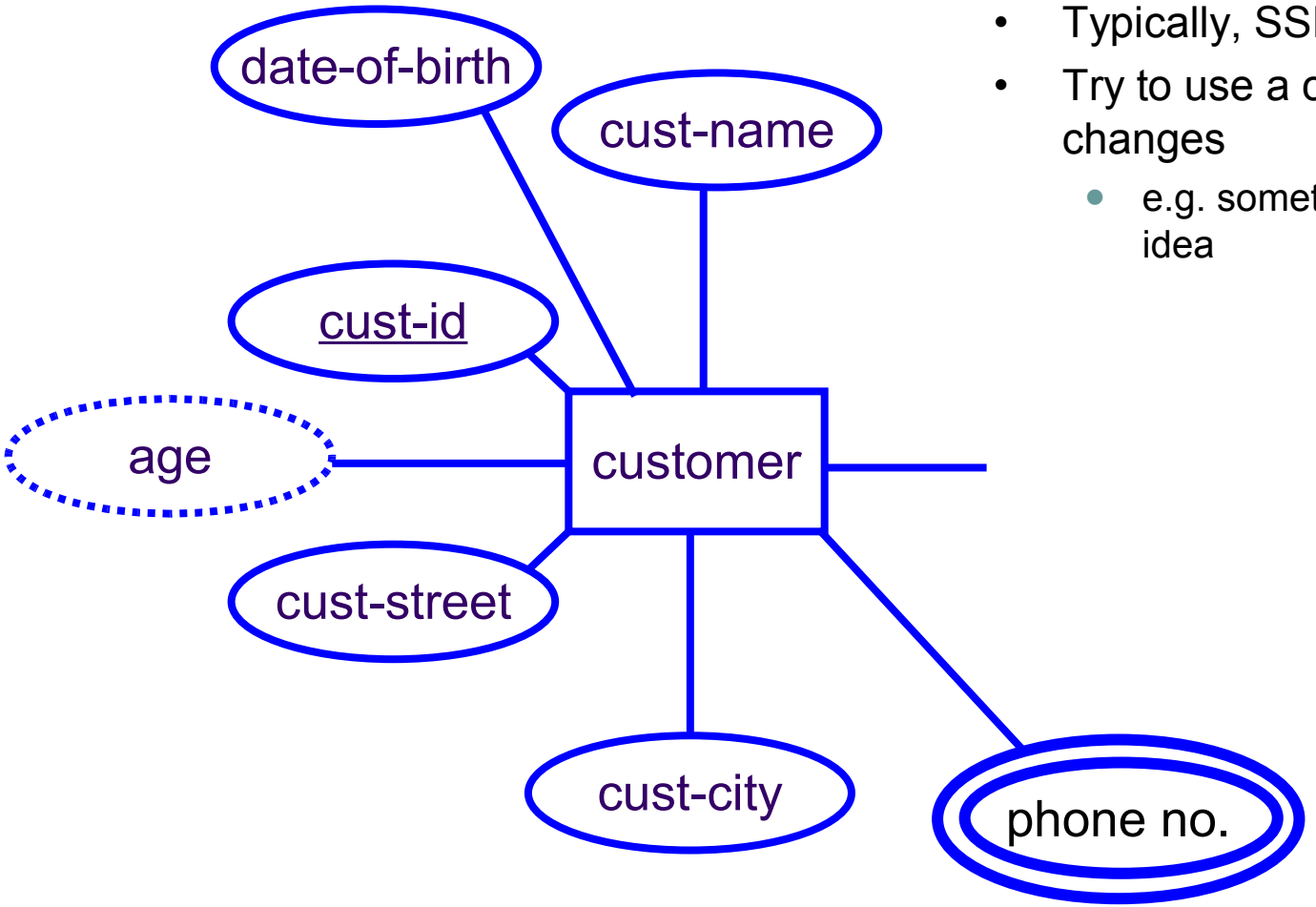


- {*cust-id*} is a natural primary key
- Typically, SSN forms a good primary key
- Try to use a candidate key that rarely changes
  - e.g. something involving address not a great idea

Entity Keys

- *Superkey*
  - any set of attributes 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
      - {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

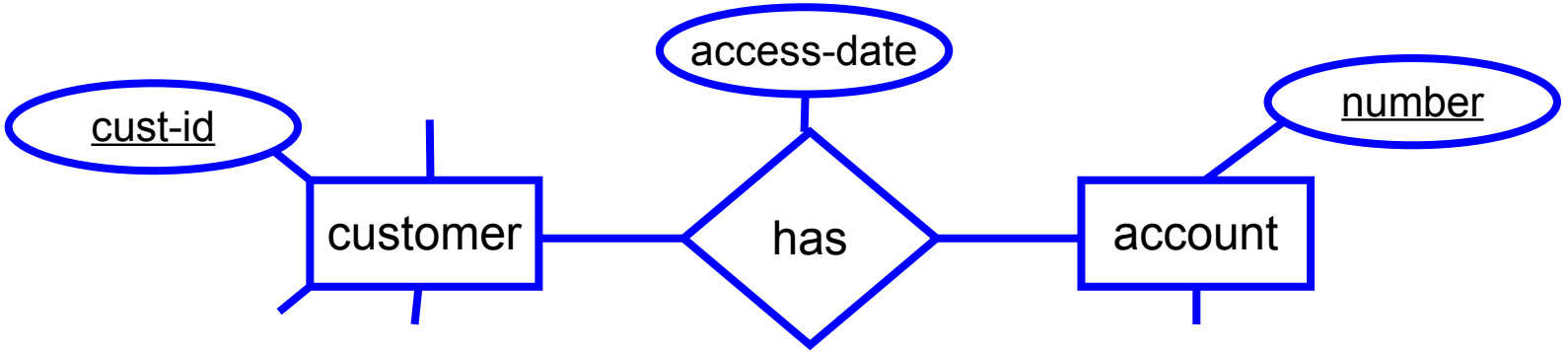
Entity Keys



- {*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

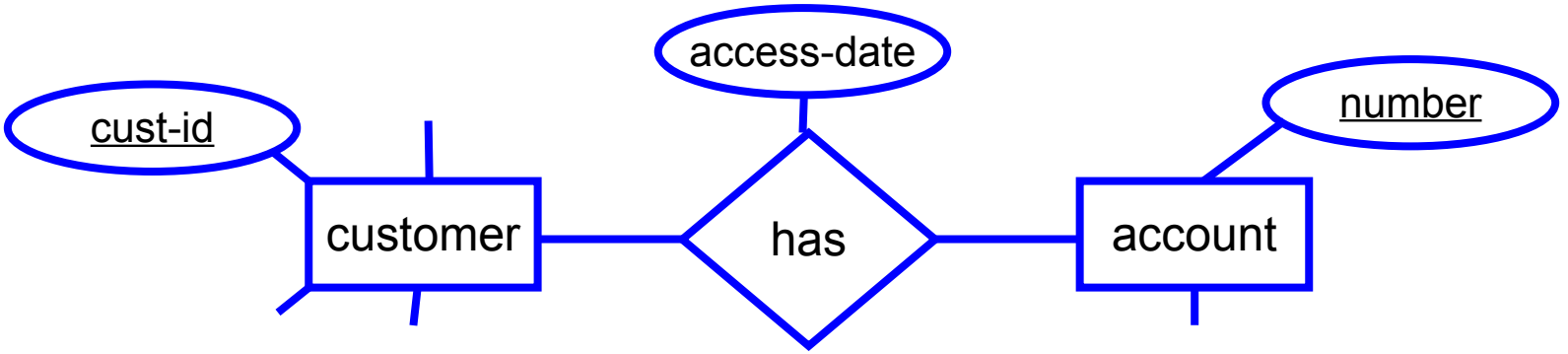
- What attributes are needed to represent a relationship completely and uniquely ?
  - Union of primary keys of the entities involved, and relationship attributes



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

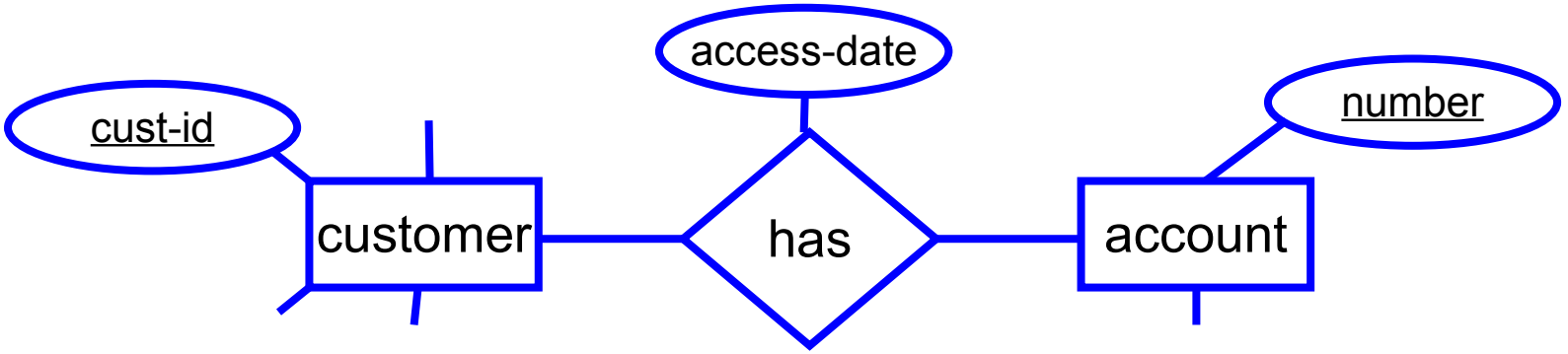
Relationship Set Keys

- Is *{cust-id, access-date, account number}* a candidate key *?*
  - No. Attribute *access-date* can be removed from this set without losing key-ness
  - In fact, union of primary keys of associated entities is always a superkey



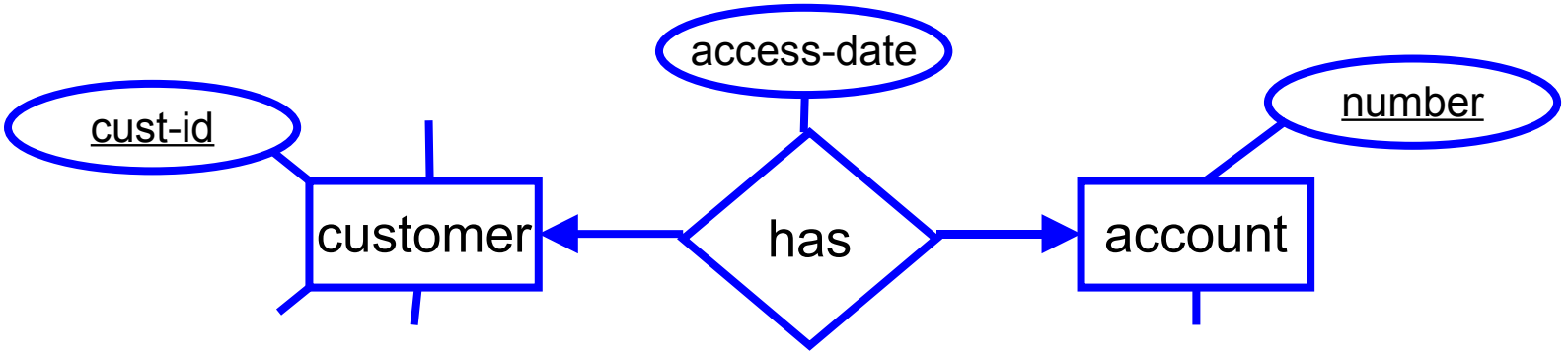
Relationship Set Keys

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



Relationship Set Keys

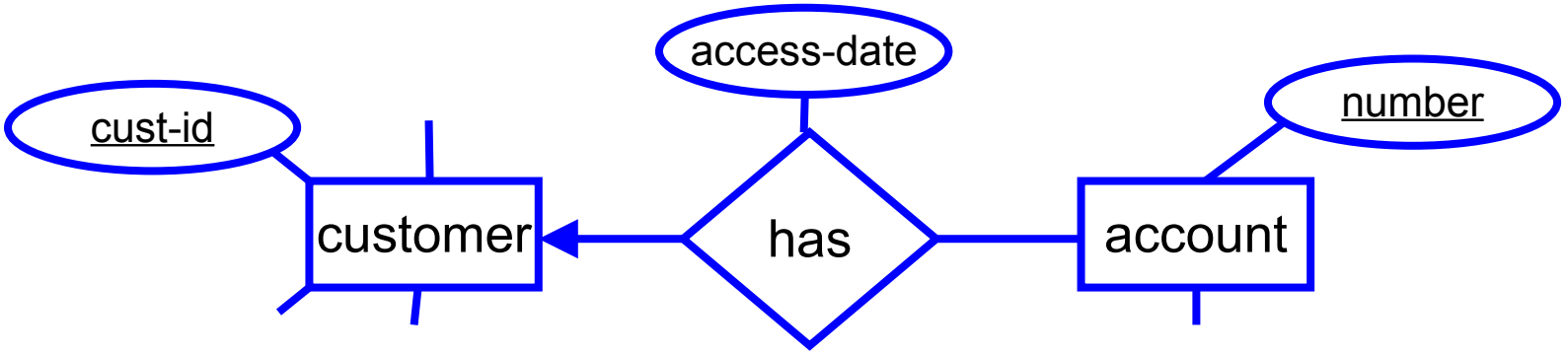
- Is {cust-id, account-number} a candidate key ?
  - Depends



- 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
  - Ditto *account*

Relationship Set Keys

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



- If one-to-many relationship (as shown), *{account-number}* is a candidate key
  - A given customer can have many accounts, but at most one account holder per account allowed

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

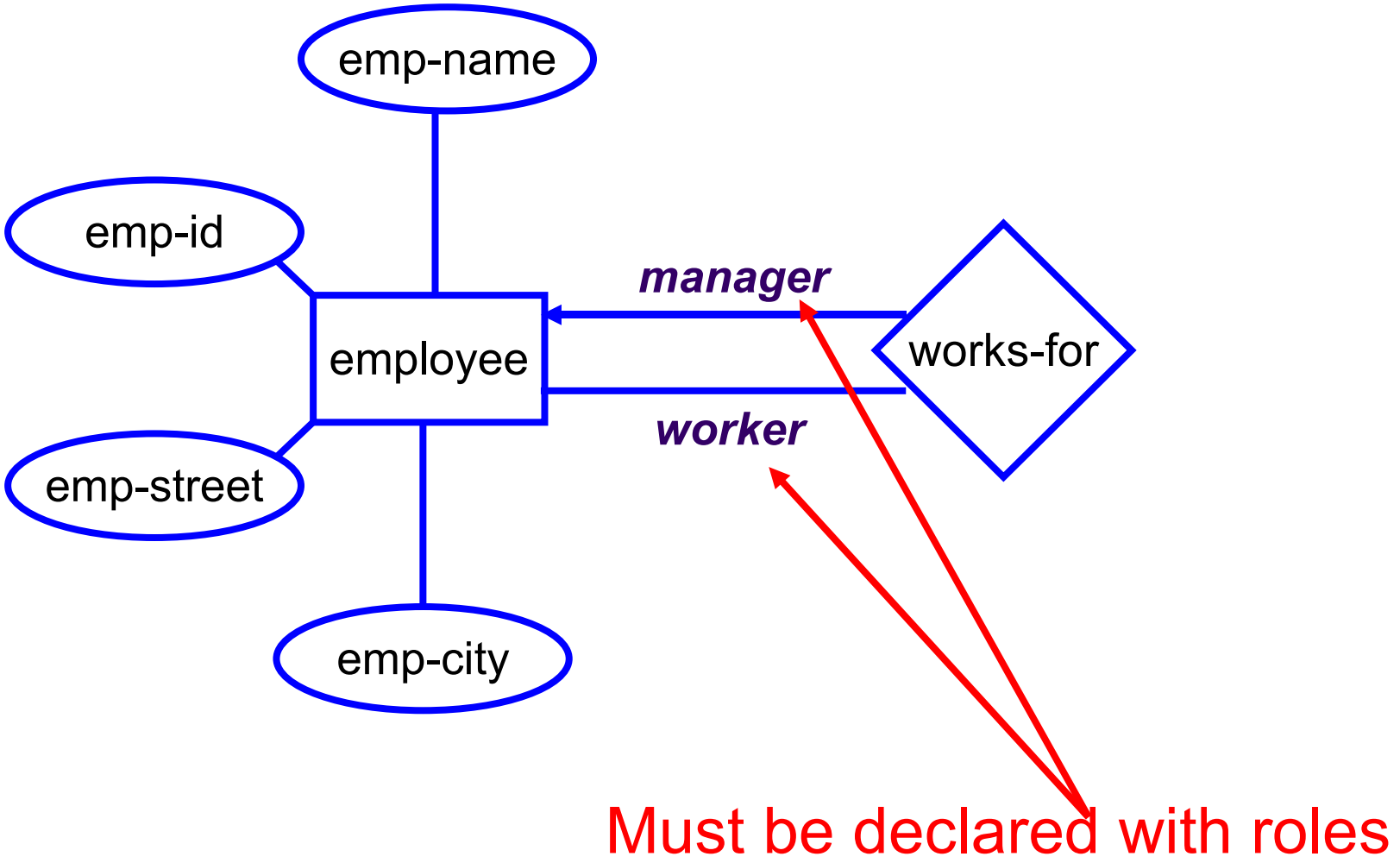
…

- What have we been doing
- Why ?
- Understanding this is important
  - Rest are details !!
  - That’s what books/manuals are for.

Next: Recursive Relationships

- Sometimes a relationship associates an entity set to itself

Recursive Relationships



Next: Weak Entity Sets

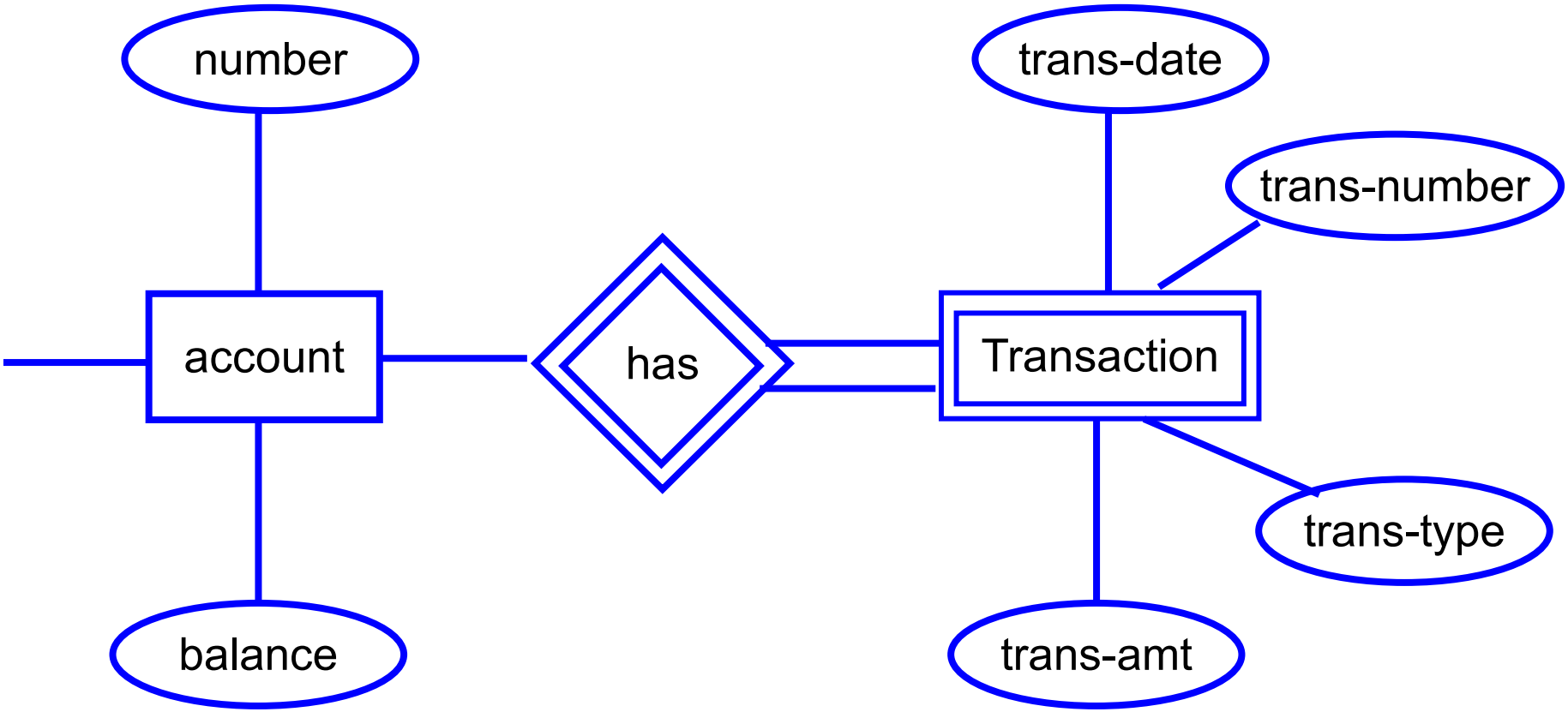
- An entity set without enough attributes to have a primary key
- E.g. Transaction Entity
  - Attributes:
    - transaction-number, transaction-date, transaction-amount, transaction-type
    - transaction-number: may not be unique across accounts

Weak Entity Sets

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

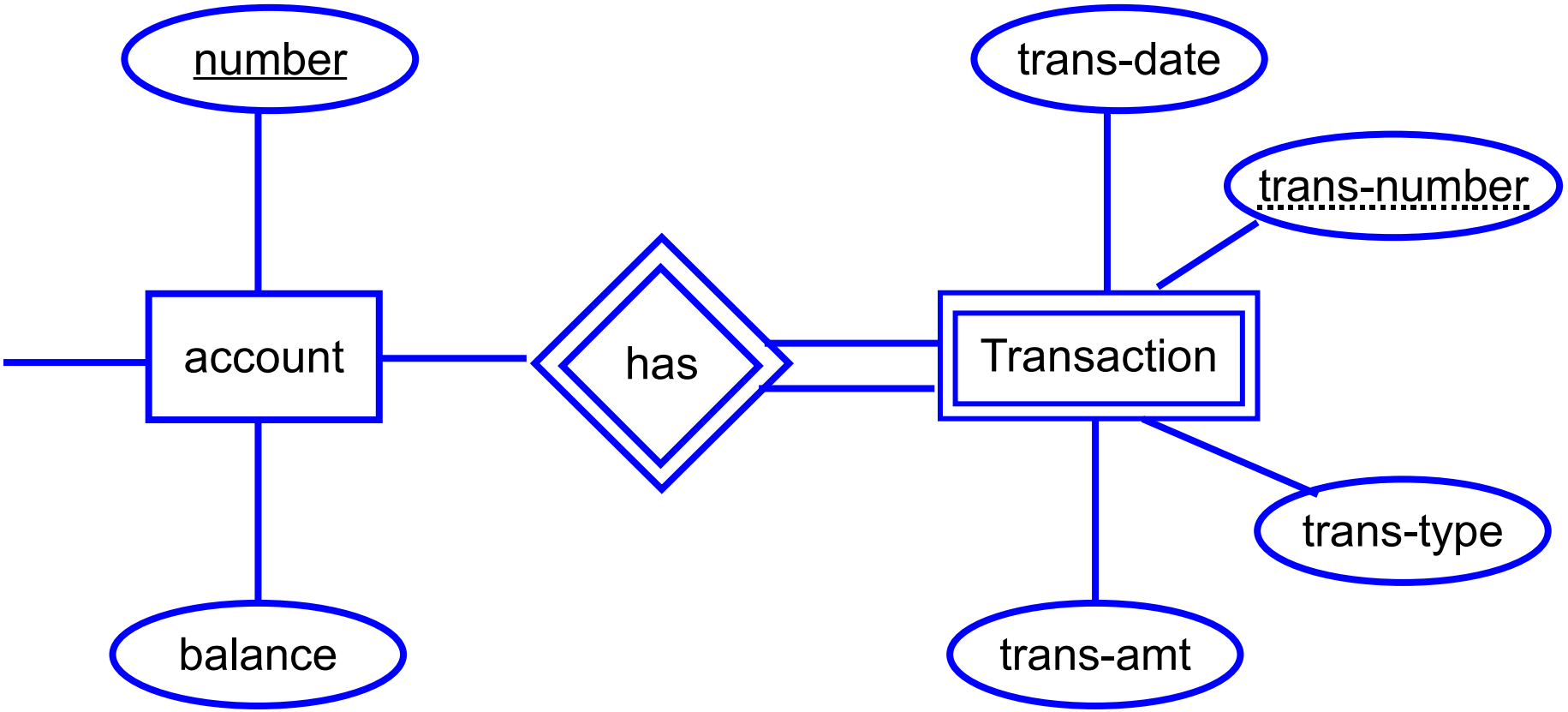
Weak Entity Sets

Still need to be able to distinguish between different weak entities associated with the same strong entity



Weak Entity Sets

Discriminator: A set of attributes that can be used for that

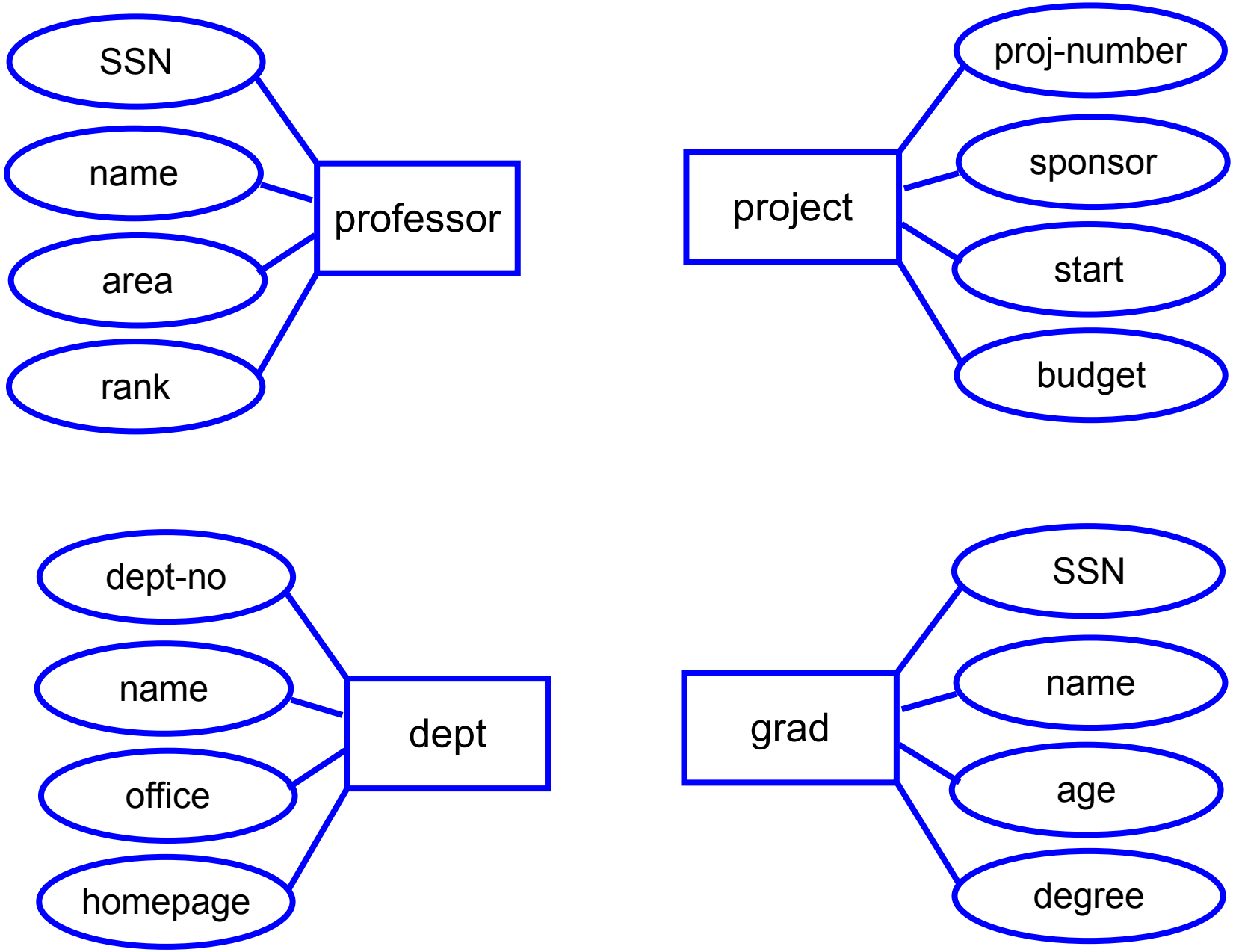


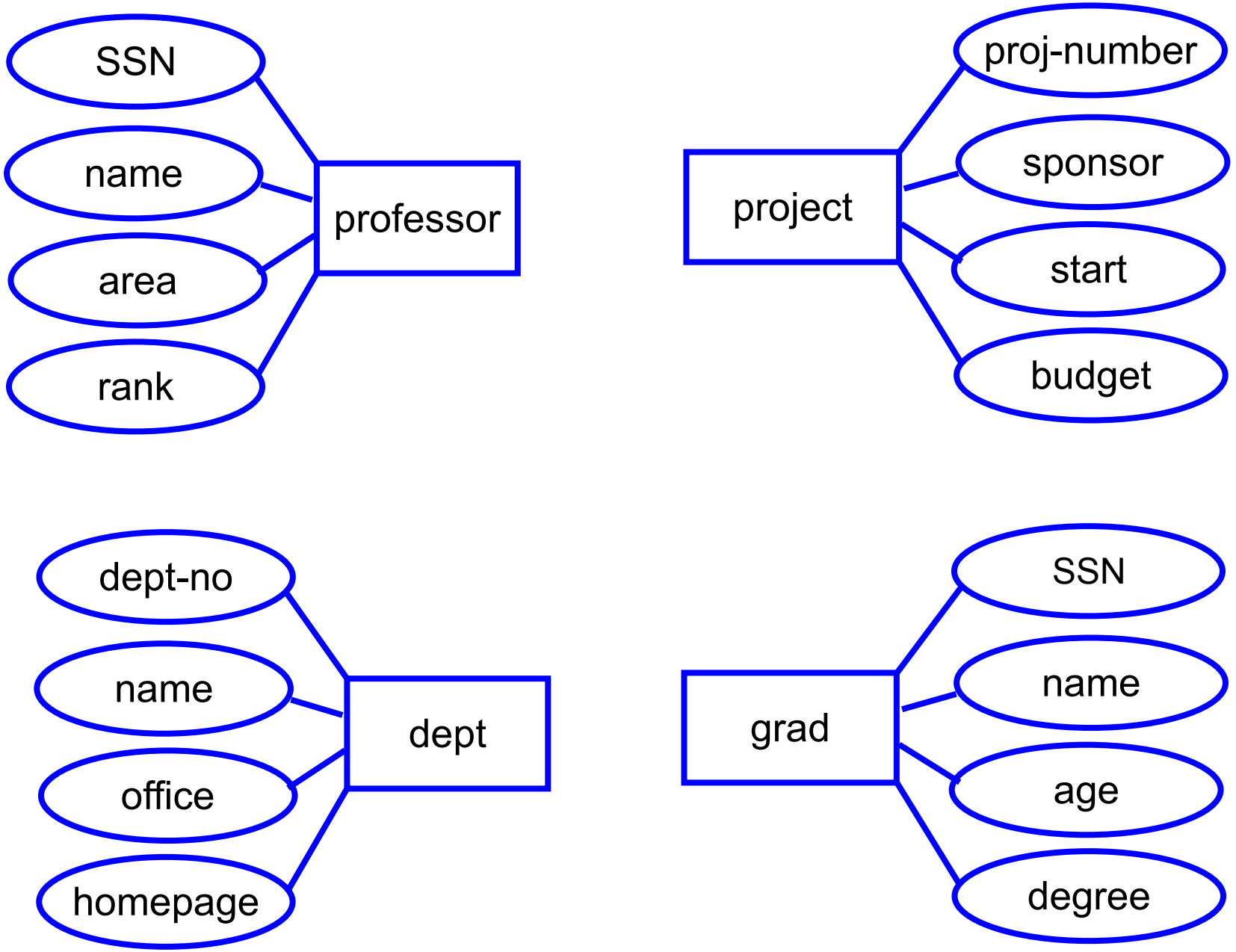
Weak Entity Sets

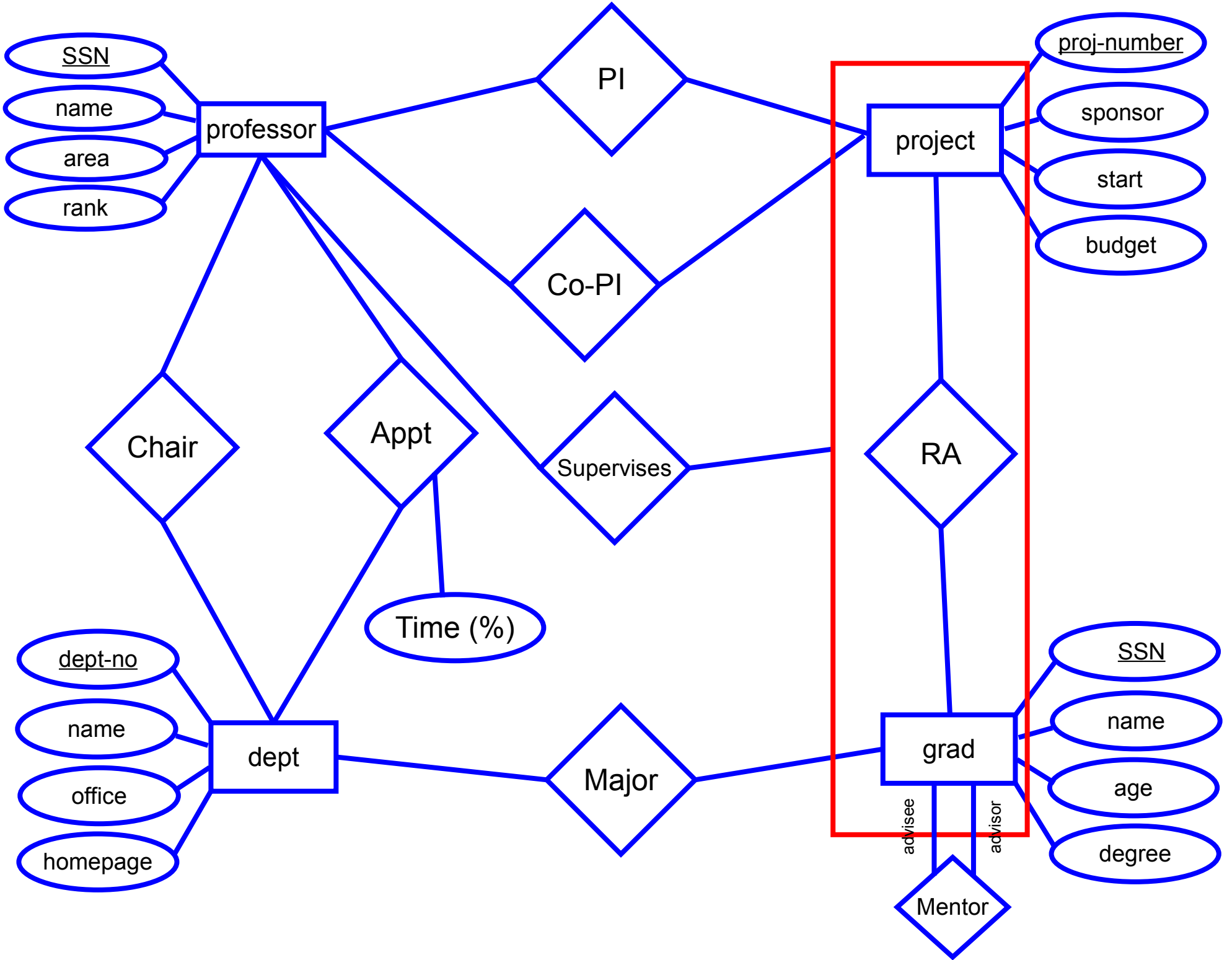
- Primary key:
  - Primary key of the associated strong entity + discriminator attribute set
  - For Transaction:
    - *{account-number, transaction-number}*

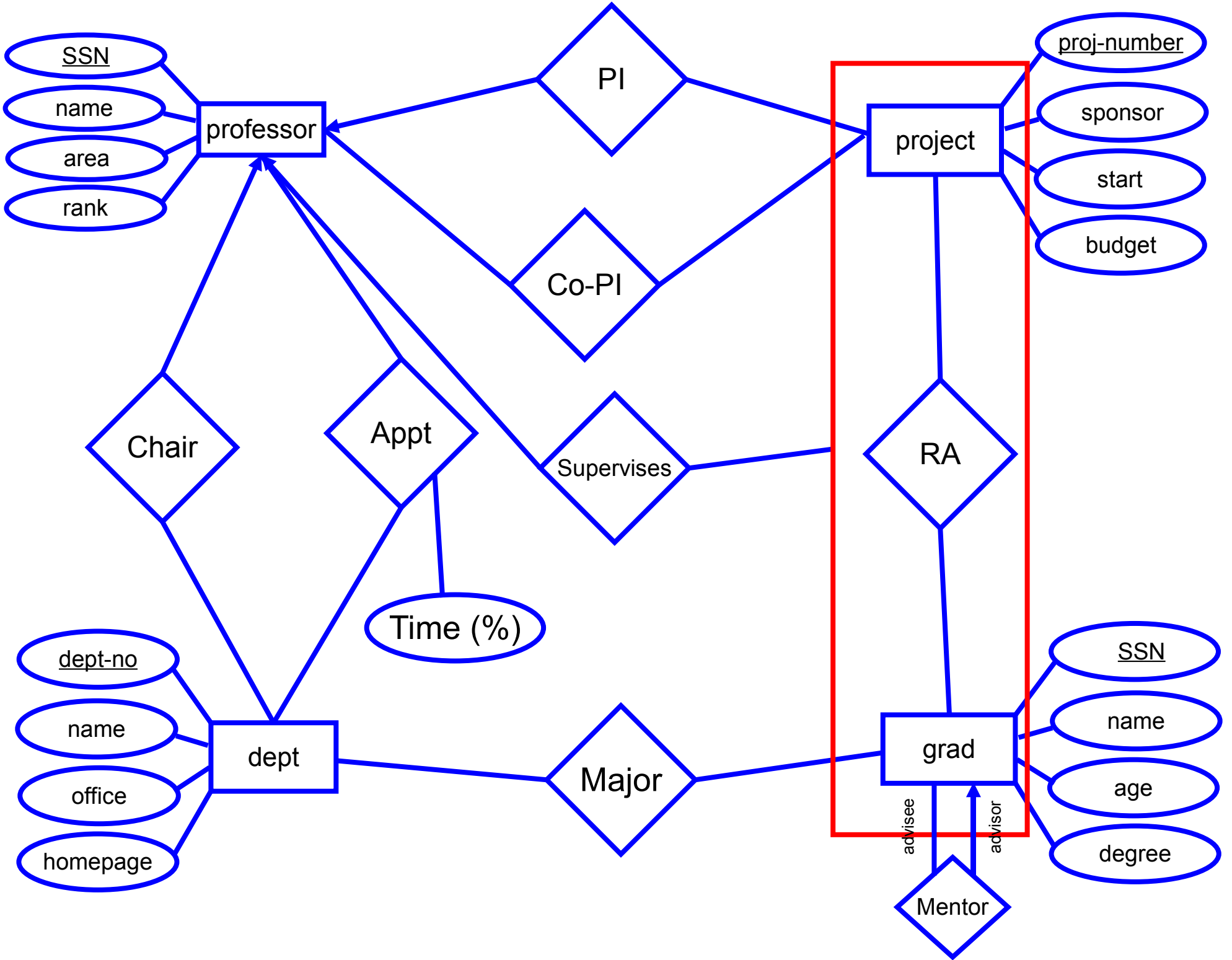
Example Design

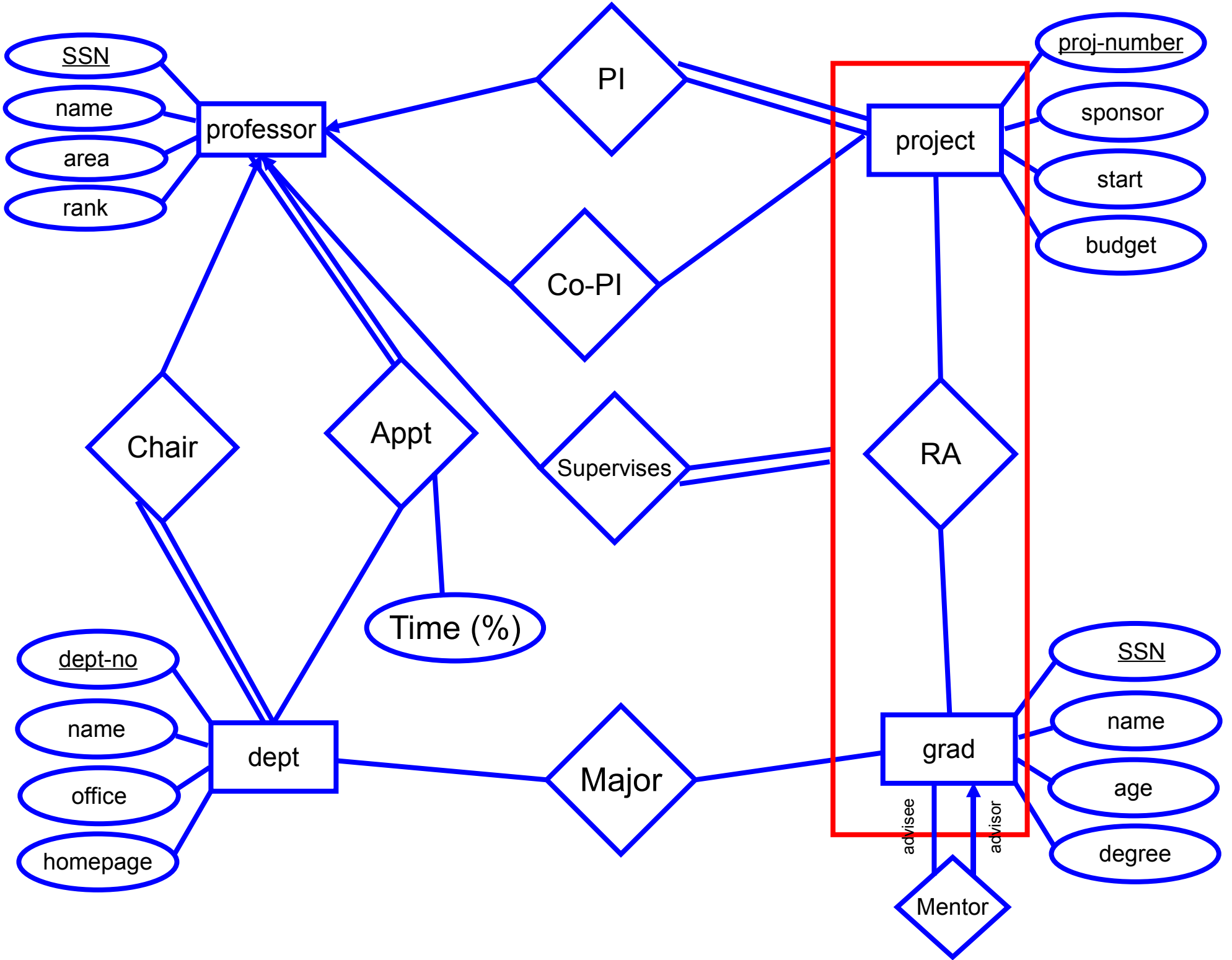
- We will model a university database
  - Main entities:
    - Professor
    - Projects
    - Departments
    - Graduate students
    - etc…


SSN
name
area
rank
professor
proj-number
sponsor
start
budget
project
dept-no
name
office
homepage
dept
SSN
name
age
degree
grad


SSN
name
area
rank
professor
proj-number
sponsor
start
budget
project
dept-no
name
office
homepage
dept
SSN
name
age
degree
grad


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
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
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

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
  - Consider *telephone* (entity: customer, attribute: telephone)
    or (entity: telephone, entity: customer, relationship: customer_phone)
- Entity sets vs Relationship sets
  - Consider *loan* (entity or relationship between customer and bank?)
- N-ary vs binary relationships
  - Possible to avoid n-ary relationships, but there are some cases where it is advantageous to use them
- It is not an exact science !!