The Object Data Model

What is different:

-- object identity

-- complex structure

-- classes: attributes, methods, encapsulation

-- inheritance

--more complex query languages

Object identity

-- built-in notion, similar to entities in ER model
-- independent of value (unlike in relational dbs)

Example

parent	age	child	age
Peter	40	John	15
Sue	41	John	15
……………………………...			

Same child ?

With object identity:

Peter → John
Sue → John

Object identity

-- built-in notion, similar to entities in ER model
-- independent of value (unlike in relational dbs)

Example

parent	age	child	age
Peter	40	John	15
Sue	41	John	15
……………………………...			

Same child ?

With object identity:

Peter → John
Sue → John

Class: type + extent

--extent: set of objects in class (like database instance)

--type: structure + methods

encapsulation: separate interface from implementation

Pseudocode example:

```
class employee {
        string name;            int annual salary();
        string address;         string get-name();
        int salary;             int employment-length() };
```

attributes methods

Inheritance: class is a special case of given class

--what is inherited: structure + methods

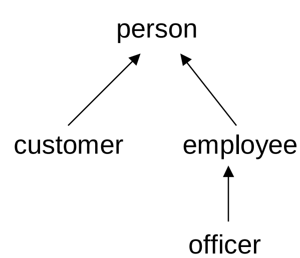
Example

```
class person {
        string name;
        string address; }

class cutomer isa person {
        int credit-rating; }

class employee isa person {
        date start-date;
        int salary;  }

class officer isa employee {
        int office-number;
        int expense-account-number; }
```

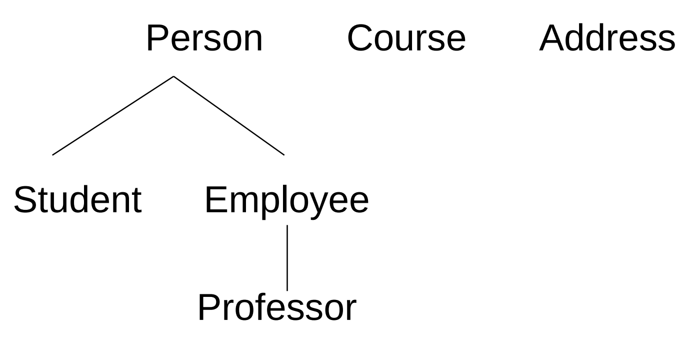


Multiple inheritance: when one class is subclass of several classes -- potential ambiguity

Relationships

- **Beside attributes, an object may have some relationships with other objects.**
- **These can be 1-1, 1-n or m-n relationships.**
- **Let us take an example:**

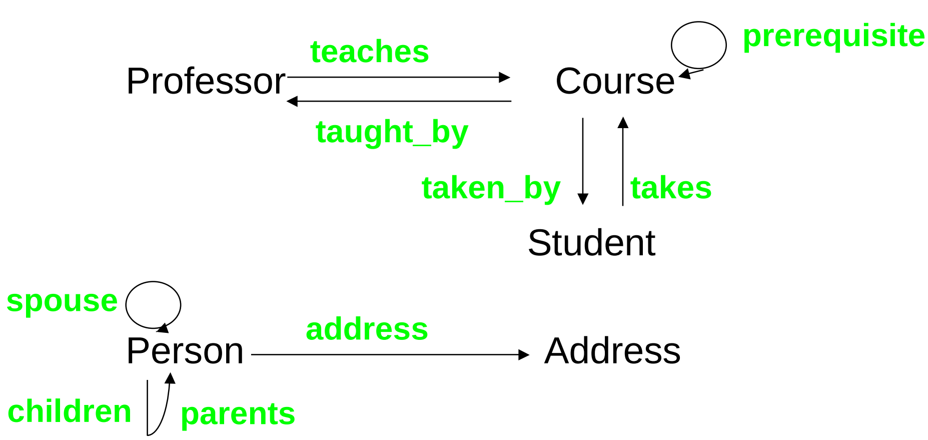
Consider these classes:



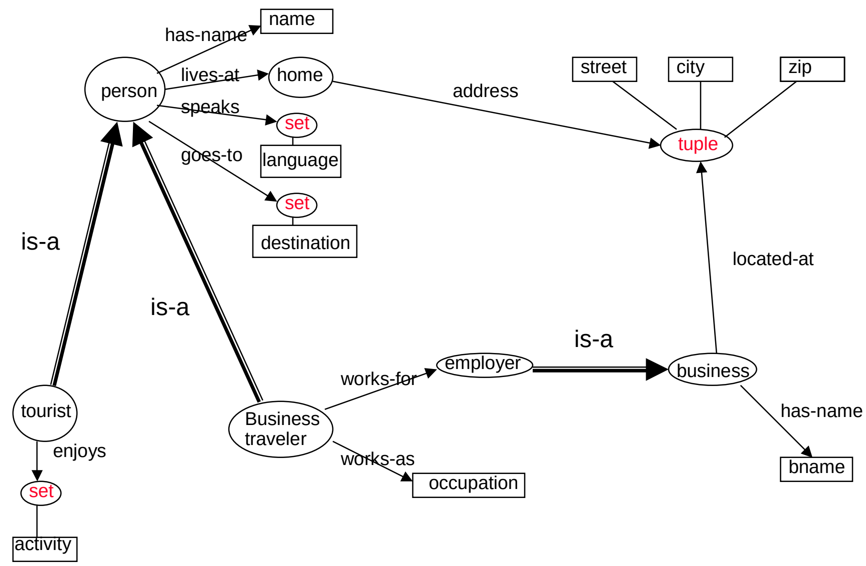
Attributes and Operations

Person	*name, birth_date, age()*
Student	*takes*, *activities()*
Employee	*number, activities()*
Professor	*grade*
Course	*subject, notes()*
Address	*street, city*

Relationships



Graphic representation of more complex structure: travel agency example



Another complex structure (nested relation)

Director	Movies: Title	Movies: Actors
Hitchcock	The Birds	Tippi Hedren Rod Taylor Alfred Hitchcock
	Psycho	Anthony Perkins Janet Leigh Alfred Hitchcock
Coppola	Apocalypse	Robert Duvall Marlon Brando
	Godfather	Marlon Brando Al Pacino Diane Keaton

Cinema:
tuple (Director: char,
Movies: set(tuple
(Title: char, Actors: set (char))))

Other structures: list, bag

Example query:
find all movies whose director
is not an actor in the movie

select m.Title
from f **in** Cinema, m **in** f.Movies
where f.Director **not in**
(**select** a **from** a **in** m.Actors)

Complex structure

Director	Movies	
	Title	Actors
Hitchcock	The Birds	Tippi Hedren Rod Taylor Alfred Hitchcock
	Psycho	Anthony Perkins Janet Leigh Alfred Hitchcock
	Title	Actors
Coppola	Apocalypse	Robert Duvall Marlon Brando
	Godfather	Marlon Brando Al Pacino Diane Keaton

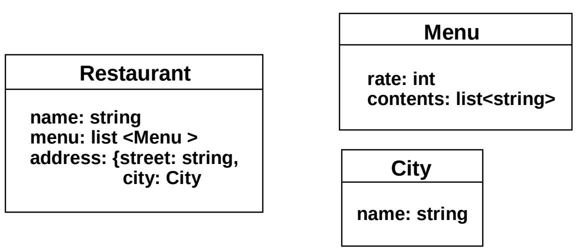
Another example query:
find all directors that have not acted
in any Hitchcock movie

select f.Director
from f **in** Films
where f.Director **not in flatten**
  (**select** m.Actors
  from g **in** Films, m **in** g.Movies
  where g.Director = “Hitchcock”)

flatten operator:
constructs the union of sets of actors

there are more operators to deal with
complex structures

Example: building complex answers



select struct(restaurant: r.name, choices: select menu.contents
from r.menu as menu
where menu.rate < 40)

from Restaurant as r

where r.address.city.name = "Paris" and
(exists menu in r.menu: menu.rate < 40)