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

<table>
<thead>
<tr>
<th>parent</th>
<th>age</th>
<th>child</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>40</td>
<td>John</td>
<td>15</td>
</tr>
<tr>
<td>Sue</td>
<td>41</td>
<td>John</td>
<td>15</td>
</tr>
</tbody>
</table>

With object identity:

Peter
Sue
John
Object identity

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

Example

<table>
<thead>
<tr>
<th>parent</th>
<th>age</th>
<th>child</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>40</td>
<td>John</td>
<td>15</td>
</tr>
<tr>
<td>Sue</td>
<td>41</td>
<td>John</td>
<td>15</td>
</tr>
</tbody>
</table>

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

Pseudo-code example:

class employee {
  string name;
  string address;
  int salary;
  int annual salary();
  string get-name();
  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 customer 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:

Person       Course       Address

Student      Employee      

Professor
<table>
<thead>
<tr>
<th>Class</th>
<th>Attributes and Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>name, birth_date, age()</td>
</tr>
<tr>
<td>Student</td>
<td>takes, activities()</td>
</tr>
<tr>
<td>Employee</td>
<td>number, activities()</td>
</tr>
<tr>
<td>Professor</td>
<td>grade</td>
</tr>
<tr>
<td>Course</td>
<td>subject, notes()</td>
</tr>
<tr>
<td>Address</td>
<td>street, city</td>
</tr>
</tbody>
</table>
Relationships

Professor \(\xrightarrow{\text{teaches}}\) Course

\(\text{taught}_\text{by}\)

Student \(\xleftarrow{\text{takes}}\) Course

\(\text{taken}_\text{by}\)

Person \(\xrightarrow{\text{address}}\) Address

\(\text{spouse}\)

\(\text{children}\) \(\xleftarrow{\text{parents}}\)

\(\text{prerequisite}\)
Graphic representation of more complex structure: travel agency example
Another complex structure (nested relation)

<table>
<thead>
<tr>
<th>Director</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitchcock</td>
<td>The Birds</td>
</tr>
<tr>
<td></td>
<td>Tippi Hedren, Rod Taylor, Alfred Hitchcock</td>
</tr>
<tr>
<td></td>
<td>Psycho</td>
</tr>
<tr>
<td></td>
<td>Anthony Perkins, Janet Leigh, Alfred Hitchcock</td>
</tr>
<tr>
<td>Coppola</td>
<td>Apocalypse</td>
</tr>
<tr>
<td></td>
<td>Robert Duvall, Marlon Brando</td>
</tr>
<tr>
<td></td>
<td>Godfather</td>
</tr>
<tr>
<td></td>
<td>Marlon Brando, Al Pacino, Diane Keaton</td>
</tr>
</tbody>
</table>

Cinema:
set ( 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 m.Actors
### Complex structure

<table>
<thead>
<tr>
<th>Director</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitchcock</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Birds</td>
</tr>
<tr>
<td></td>
<td>Psycho</td>
</tr>
<tr>
<td>Coppola</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Apocalypse</td>
</tr>
<tr>
<td></td>
<td>Godfather</td>
</tr>
</tbody>
</table>

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

```sql
select f.Director
from f in Cinema
where f.Director not in flatten
(select m.Actors
from g in Cinema, 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
Another 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 = « San Francisco » and
(exists menu in r.menu: menu.rate < 40)
```