

# Object-Oriented Databases

## The OM Data Model

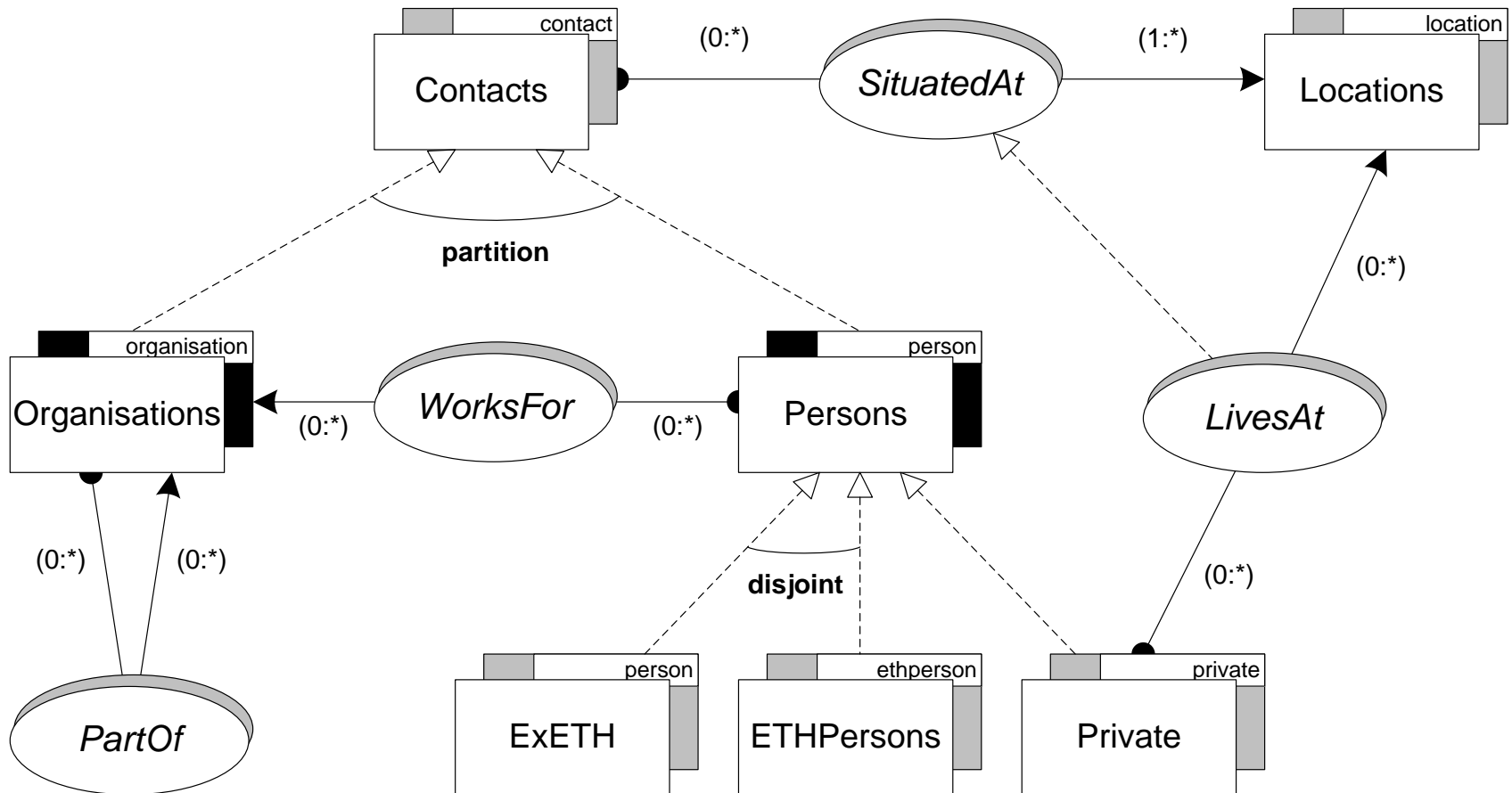
- Multiple Inheritance, Instantiation and Classification
- Collections and Associations
- Cardinality, Classification and Evolution Constraints



# OM Data Model

- Extended Entity-Relationship model for object-oriented data management
- Distinguishing features
  - distinguishes typing and classification
  - data represented as objects, i.e. attributes and methods
  - multiple inheritance, multiple instantiation, multiple classification
  - collections and binary associations as first-order concept
  - constraints for integrity, classification and evolution
  - data definition, manipulation and query language OML
- Implementations
  - OMS Pro, OMS/Java, eOMS, OMS Avon

# OM Data Model



# Typing and Classification

## Typing

- representation of entities
- defines format of data values
- defines operations
- defines inheritance properties

```
public class Person {  
    String name;  
    Date birthdate;  
  
    public Person(String name) {  
        this.name = name;  
    }  
}
```

## Classification

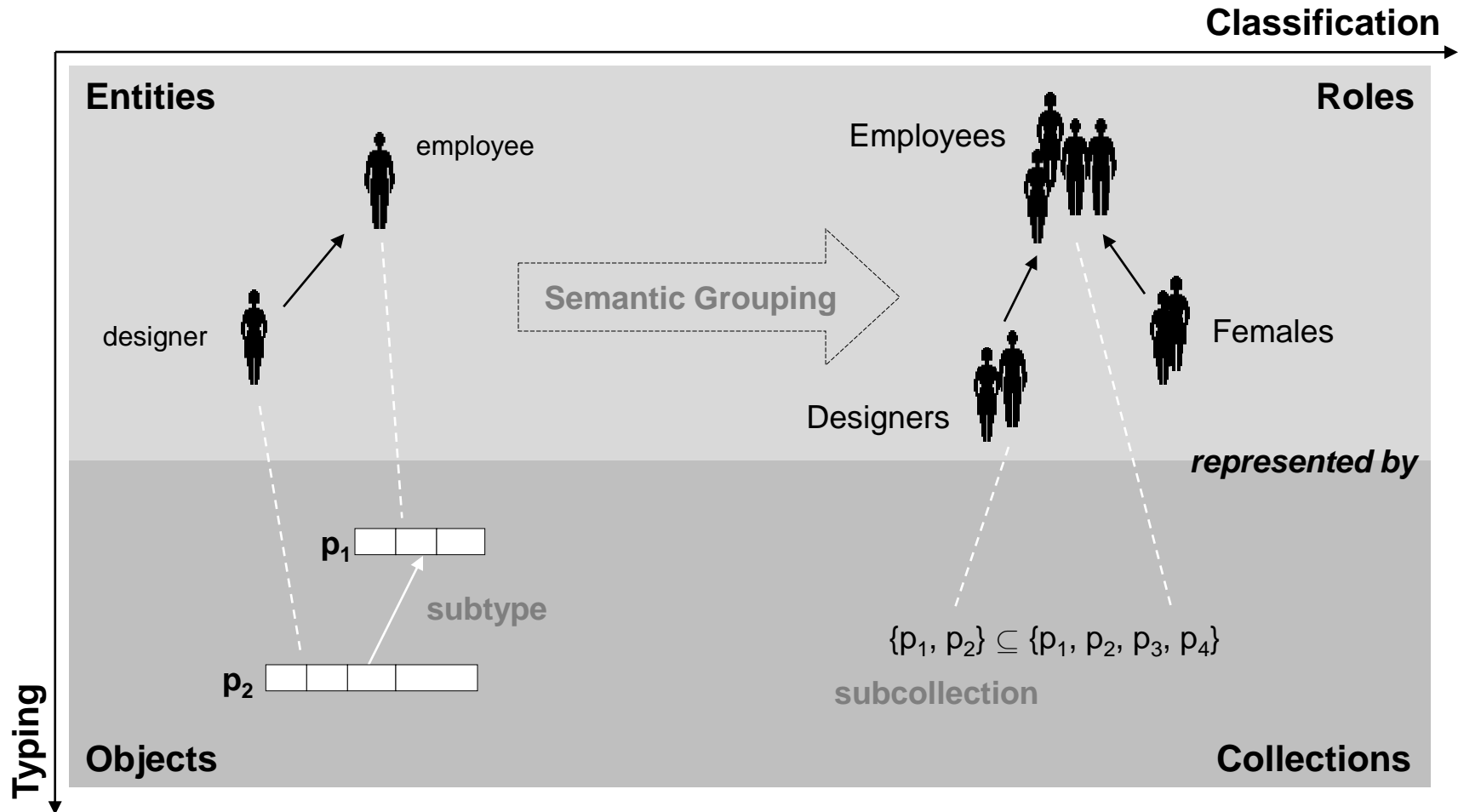
- roles of entities
- defines semantic groupings as collections of values
- defines constraints among collections

```
Person alex =  
    new Person("Alex de Spindler");  
Collection<Person> friends =  
    new HashSet<Person>();  
Collection<Person> coWorkers =  
    new HashSet<Person>();  
friends.add(alex);  
coWorkers.add(alex);
```

# Typing and Classification

- Better understanding of issues
  - important to recognise the two concepts even if they are somehow merged together in a particular model or system
- Reduces complexity of type graphs
  - no need to introduce subtypes to represent each classification
- Rich classification structures
- Support for relationships between objects
  - associations over collections rather than embedded in objects
- Integration of database and programming languages

# Typing and Classification



# Typing and Classification

- Collections
  - semantic groupings of objects
- Member types of collections
  - constrain membership in a collection
  - can define a view of objects accessed in context of collection
- Object evolution
  - objects can gain and lose roles by being added to and deleted from collections
  - type change only required if an object is not an instance of member type of collection

# OM Data Model Layers

- OM distinguishes typing from classification based on a two-layered Entity-Relationship model
- Type layer
  - defines object representation
  - multiple inheritance
  - multiple instantiation
- Classification layer
  - defines semantic groupings
  - multiple classification
  - collections and associations



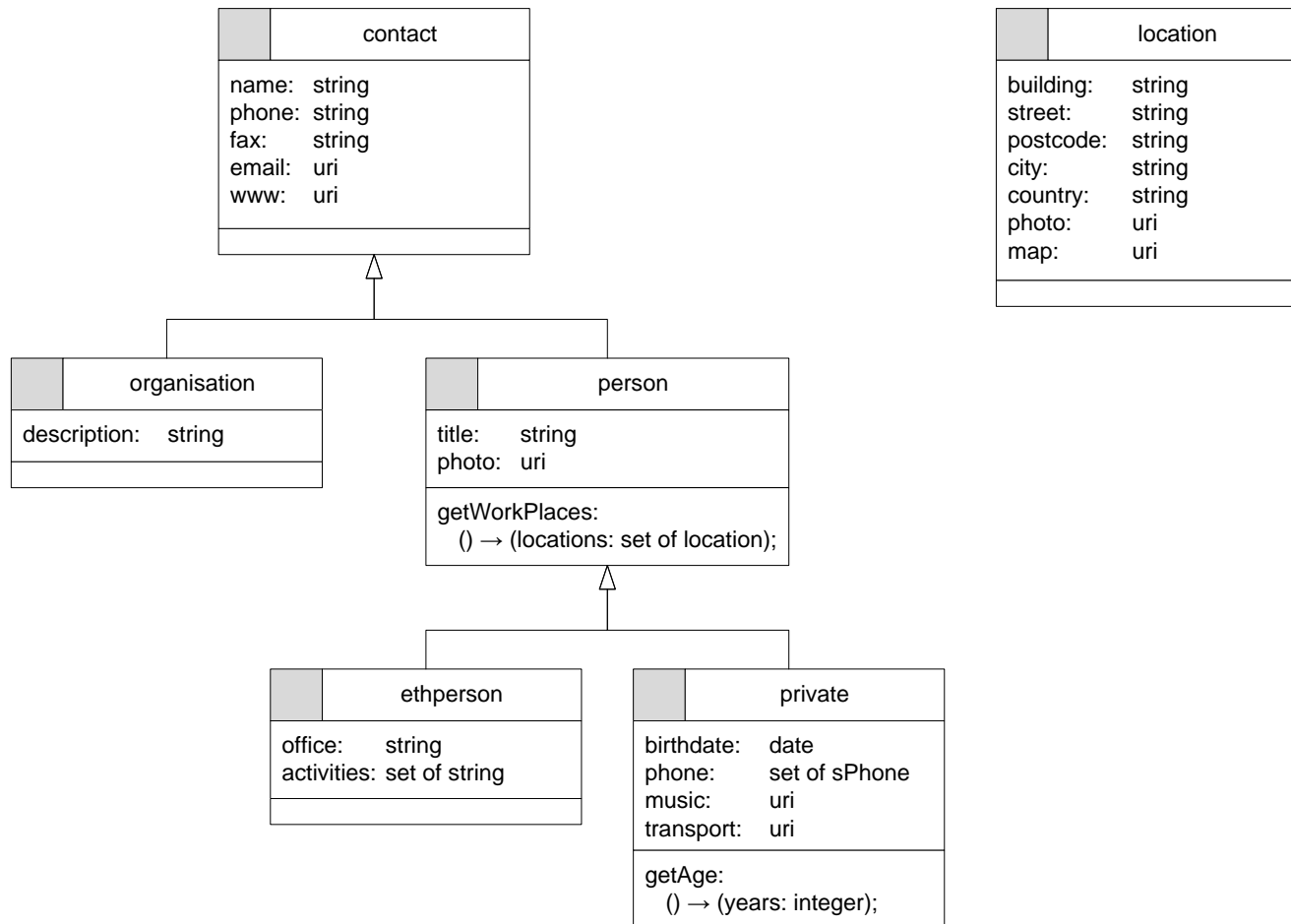
# OM Type Layer

- Objects are represented by object types
  - object types define type units
  - instances of object types have corresponding information units
- Objects can gain and lose types dynamically
  - dress operation adds a type to an object
  - strip operation removes a type from an object
- Multiple inheritance
  - an object type can have multiple supertypes
  - conflicts and name clashes must be handled by developer
- Multiple instantiation
  - objects can have types from parallel inheritance hierarchies
  - objects can have types that are completely unrelated

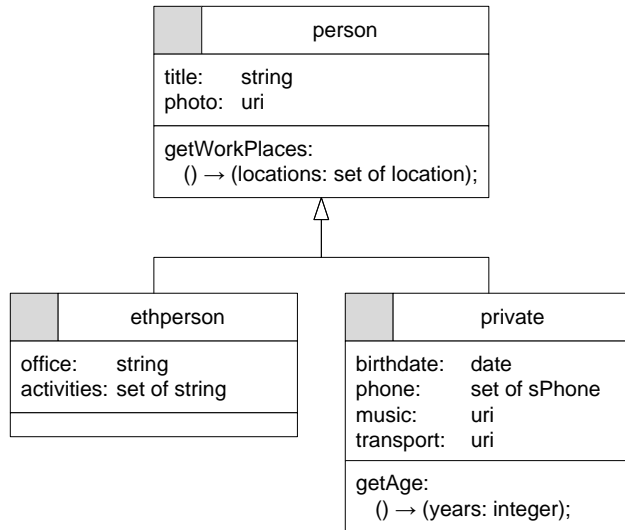
# OM Type Layer

- Values of different types of types are supported in the OM data model
- Base types
  - define basic built-in values without identity
  - string, integer, real, boolean, date, uri, ...
- Object types
  - define representations of objects with identity
- Structured types
  - define structure or record values without identity
- Bulk types
  - define collection values of given member type without identity

# Object Types



# Type Units



- Each type defines a type unit
  - direct correspondence to type
  - no inherited fields
- Attributes have a name, a type and a bulk
  - uni
  - set, bag, sequence, ranking
- Methods are managed separately and linked to types

person	(title, string, uni), (photo, uri, uni)
--------	---

ethperson	(office, string, uni), (activities, uri, set)
-----------	---

private	(birthdate, date, uni), (phone, sPhone, set), (music, uri, uni), (transport, uri, uni)
---------	--

# Information Units

## Object

o327	
contact	name: "Moir C. Norrie" phone: "+41 44 632 72 42" ...
person	title: "Prof." ...
ethperson	office: "IFW D 45.1" ...
private	birthdate: null phone: { ("mobile", "+4179...") } ...

## Information Units

o327	contact	"Moir C. Norrie", "+41 44 632 72 42", ...
------	---------	---

o327	person	"Prof.", ...
------	--------	--------------

o327	ethperson	"IFW D 45.1", ...
------	-----------	-------------------

o327	private	null, {("mobile", "+4179...")}, ...
------	---------	-------------------------------------

# Browsing Objects

## Information Units

o327	contact	"Maira C. Norrie", "+41 44 632 72 42", ...
o327	person	"Prof.", ...
o327	ethperson	"IFW D 45.1", ...
o327	private	null, {"mobile", "+4179..."}, ...

### Instance person

person(o327)
name: "Maira C. Norrie"
phone: "+41 44 632 72 42"
...
title: "Prof"
...

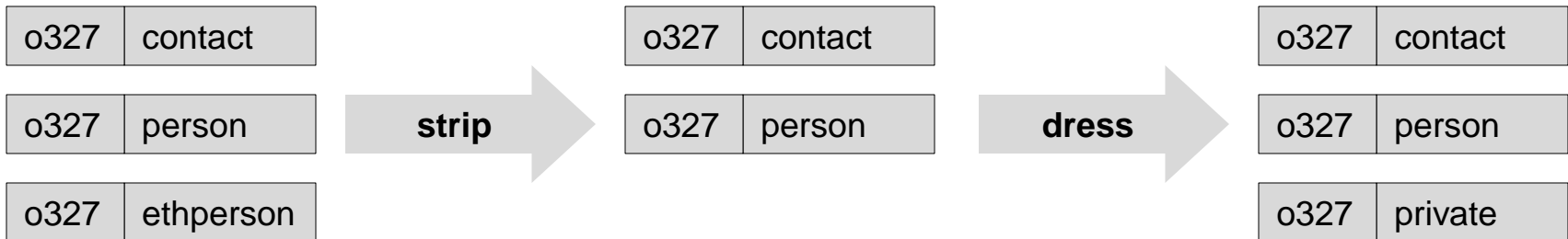
### Instance ethperson

ethperson(o327)
name: "Maira C. Norrie"
phone: "+41 44 632 72 42"
...
title: "Prof"
...
office: "IFW D 45.1"
...

### Instance private

private(o327)
name: "Maira C. Norrie"
...
title: "Prof"
...
birthdate: null
phone: {"mobile", "+4176..."}

# Dressing and Stripping Objects



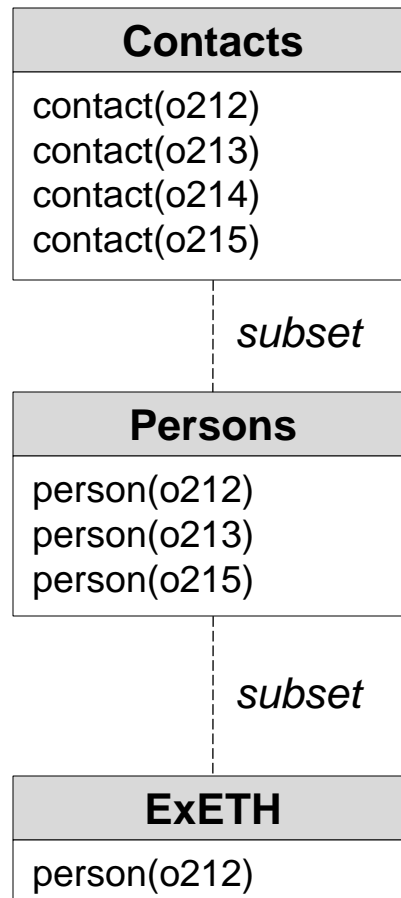
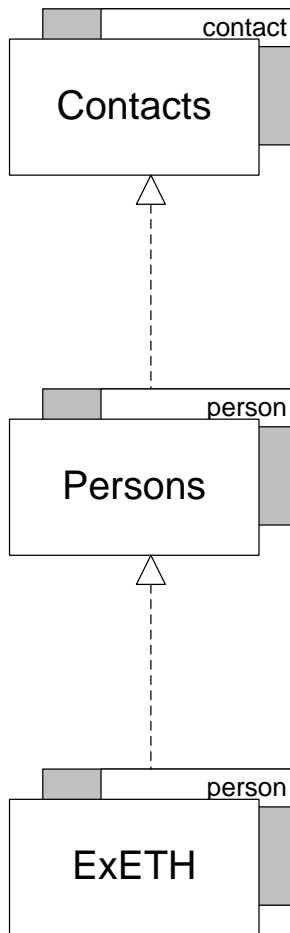
- Objects can gain and lose type instances dynamically
- Dress operation creates an information unit of given type, initialised it with null values, and adds it to object
- Strip operation removes an information unit of given type and discards values stored in it

# OM Classification Layer

- Classification is defined based on types from type layer
- Collections as a concept for semantic groupings
  - collection membership constrained by type
  - associations to link objects together
  - multiple collections can share the same member type
  - different types of collection behaviour
  - kinds and roles
- Constraints to raise semantic expressiveness
  - subcollection and supercollection constraints
  - classification structures and classification constraints
  - cardinality constraints to describe associations
  - evolution constraints that govern object lifecycle



# Collections and Subcollections



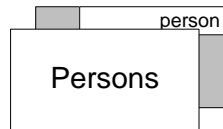
$\text{ext}(\text{Contacts}) =$   
 $\{ \text{contact(o212)}, \text{contact(o213)},$   
 $\text{contact(o214)}, \text{contact(o215)} \}$

$\text{ext}(\text{Persons}) \subseteq \text{ext}(\text{Contacts})$

$\text{ext}(\text{ExETH}) \subseteq \text{ext}(\text{Persons})$

# Collection Behaviour

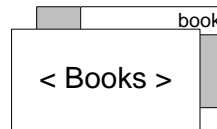
## Set



Persons
person(o111)
person(o112)
person(o113)
person(o114)

no duplicates  
no order

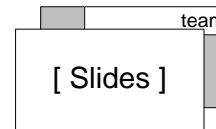
## Bag



Books
book(211)
book(212)
book(212)
book(212)
book(213)
book(213)

duplicates  
no order

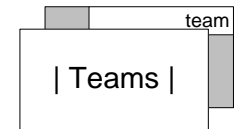
## Sequence



Slides
slide(311)
slide(312)
slide(313)
slide(314)
slide(315)
slide(311)

duplicates  
order

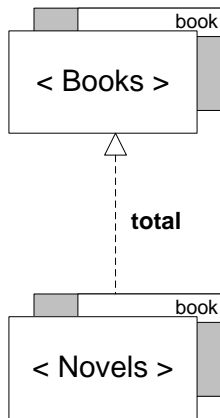
## Ranking



Teams
team(411)
team(415)
team(413)
team(412)
team(416)
team(414)

no duplicates  
order

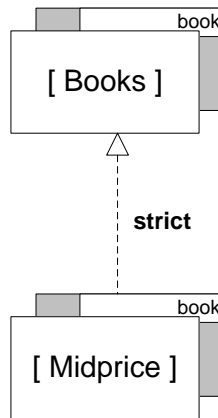
# Subcollection Behaviour



control duplicates

all copies of a novel  
that are in Books are  
also in Novels

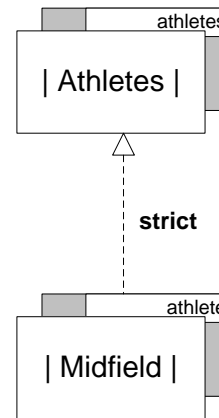
$$\{(b_1, 2), (b_2, 3), (b_3, 1)\}$$

$$\{(b_1, 2), (b_3, 1)\}$$


control order

subcollection is a  
strict subsequence  
of supercollection

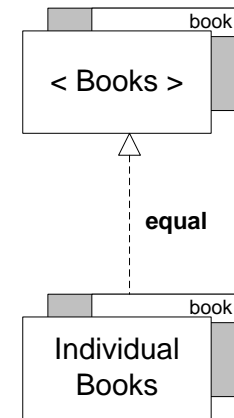
$$[b_1, b_1, b_2, b_2, b_2, b_3]$$

$$[b_1, b_2, b_2, b_2]$$


control order

subcollection is a  
strict subranking of  
supercollection

$$|a_1, a_2, a_3, a_4, a_5, a_6|$$

$$|a_3, a_4|$$


control membership

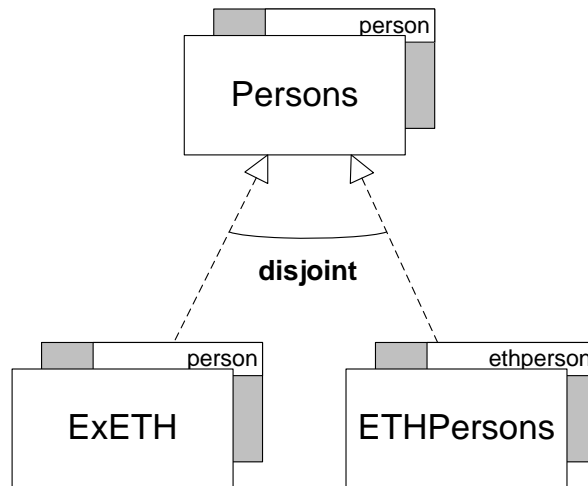
collection behaviour  
is different, elements  
are the same

$$\{(b_1, 2), (b_2, 3), (b_3, 1)\}$$

$$\{b_1, b_2, b_3\}$$

# Classification Structures

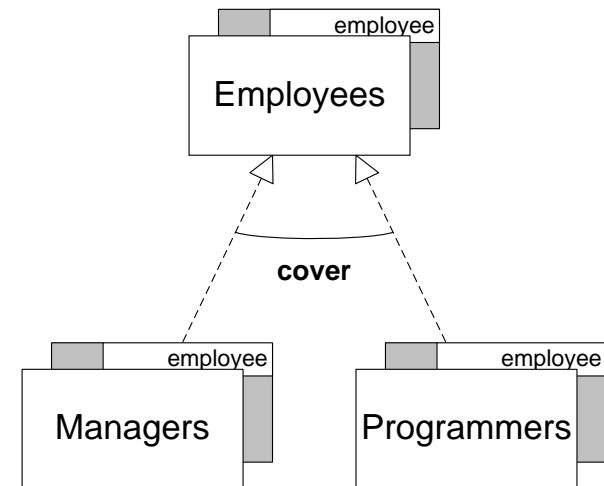
## Disjoint



No person can be employed by ETH and, at the same time, have left ETH

$$\text{ext}(\text{ExETH}) \cap \text{ext}(\text{ETHPersons}) = \emptyset$$

## Cover

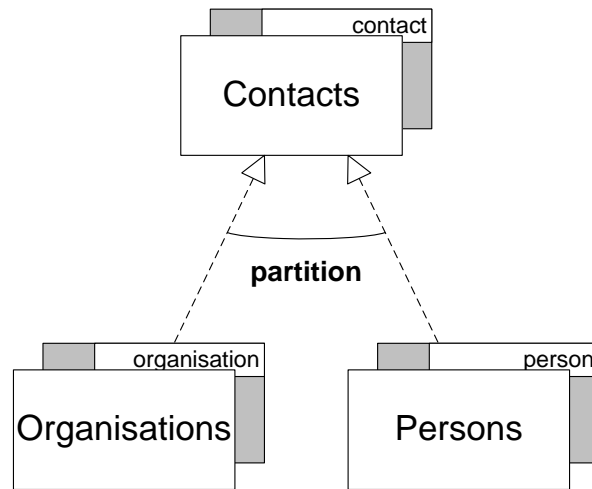


Every employee is either a manager, a programmer or both

$$\text{ext}(\text{Managers}) \cup \text{ext}(\text{Programmers}) = \text{ext}(\text{Employees})$$

# Classification Structures

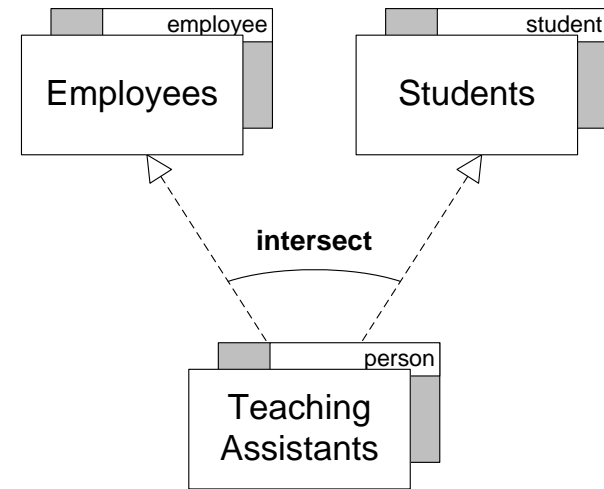
## Partition



A contact is either an organisation or a person, but not both

$$\begin{aligned} \text{ext}(\text{Organisations}) \cap \text{ext}(\text{Persons}) &= \emptyset \\ \wedge \text{ext}(\text{Organisations}) \cup \text{ext}(\text{Persons}) &= \text{ext}(\text{Contacts}) \end{aligned}$$

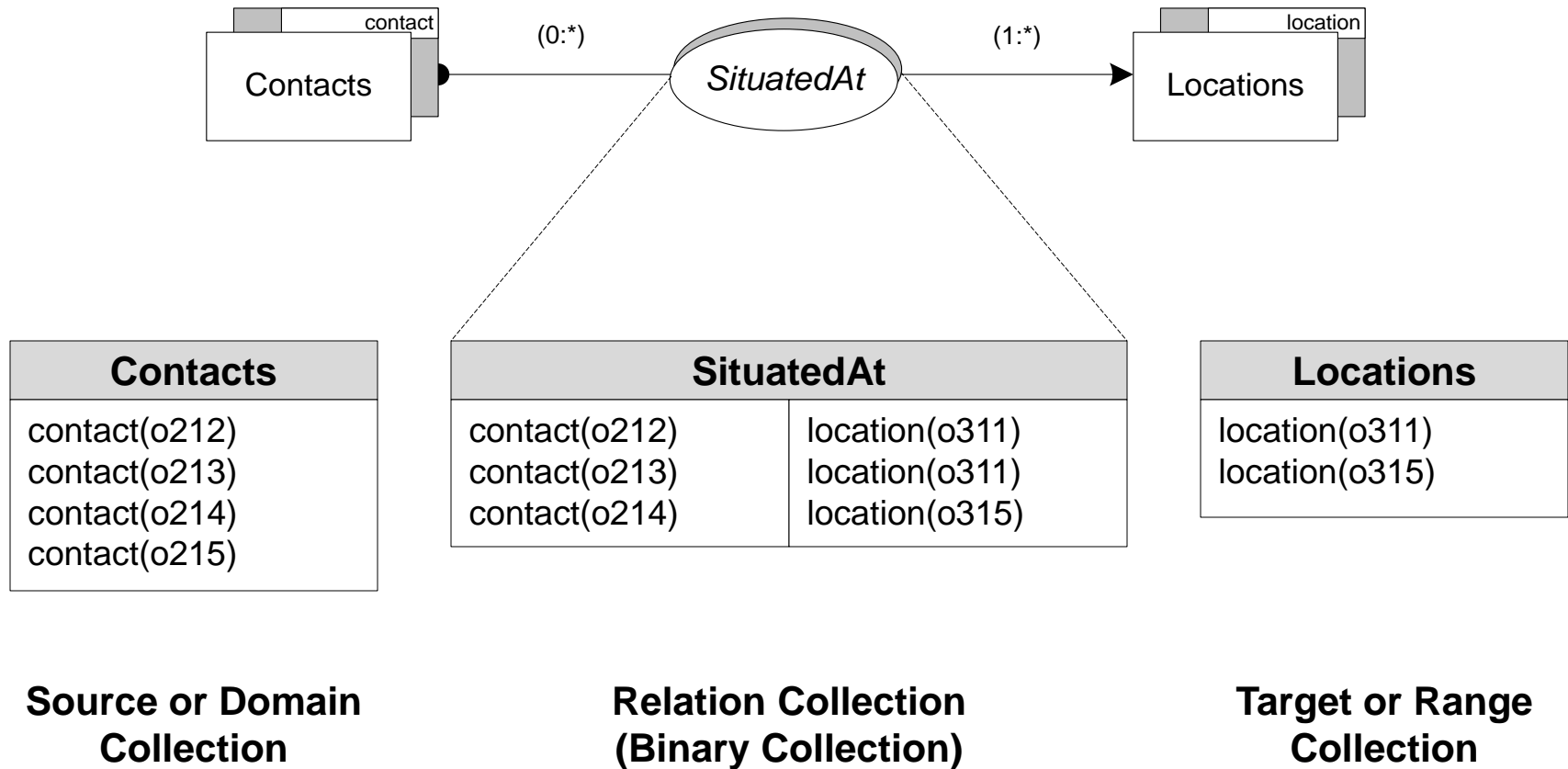
## Intersection



Every person that is an employee and a student is a teaching assistant

$$\begin{aligned} \text{ext}(\text{Employees}) \cap \text{ext}(\text{Students}) &= \\ &= \text{ext}(\text{Teaching Assistants}) \end{aligned}$$

# Associations

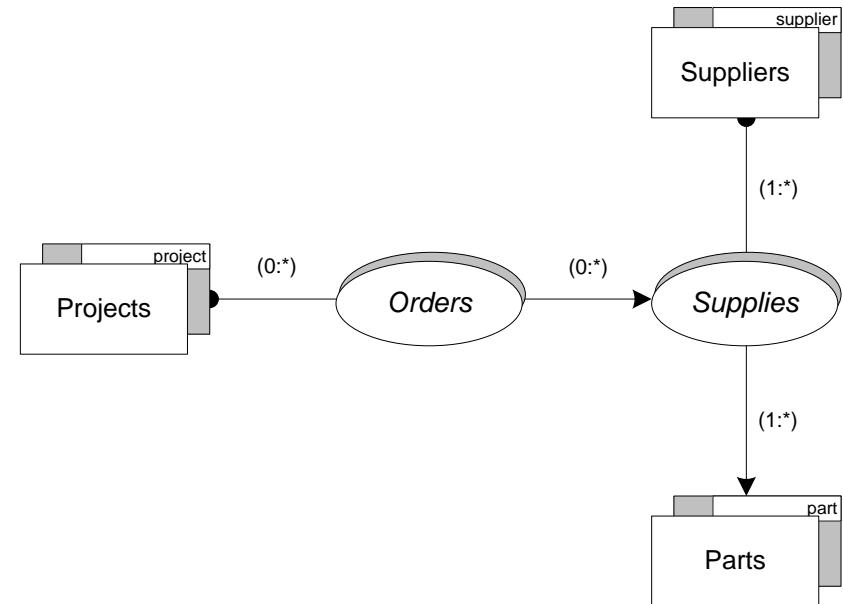


# Associations

- Cardinality constraints
  - (0:\*) constraint on domain expresses that an organisation can be situated at any number of locations
  - (1:\*) constraint on range expresses that each location has to be associated with at least one organisation
  - notation differs from the one commonly used in E/R models
- Behaviour
  - relation collection of an association is a normal collection
  - can be set, bag, sequence or ranking
- No ternary association and no relationship attributes
  - OM does not support ternary or n-ary associations
  - OM does not support attributes for associations

# Nested Associations

- Domain and range of an association can be associations
- Can be used to model
  - n-ary relationships
  - relationship attributes
- Advantages
  - decomposition of ternary relationship into primary and secondary association
  - clearer semantics of cardinality constraints
  - allows uniform query language constructs to be used

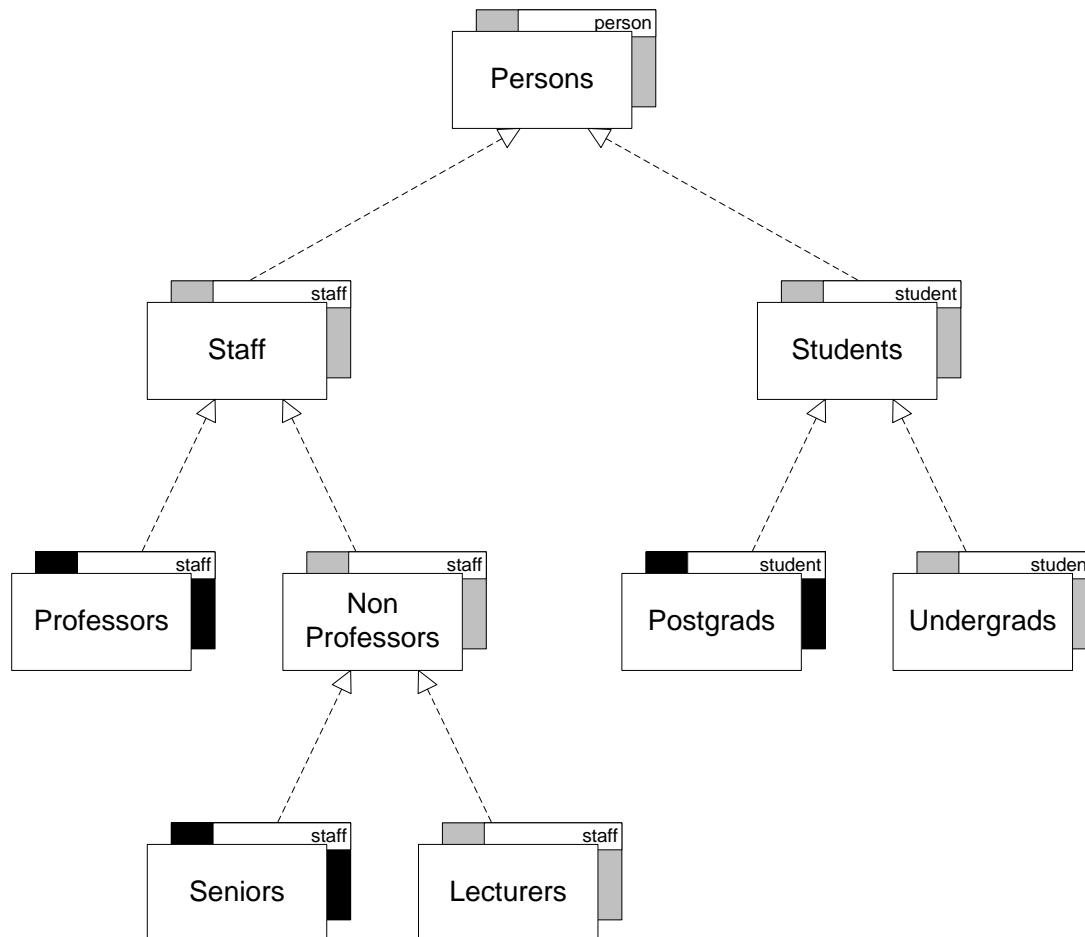




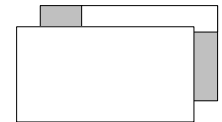
# Kinds and Roles

- Kinds
  - fundamental fixed classification
  - similar to Stan Zdonik's notion of essential types but he does not distinguish typing and classification and only has types to deal with both representation and classification
  - kinds can change if certain conditions fulfilled
- Roles
  - roles change during entity lifecycles
- Kinds and roles also control the evolution of a database

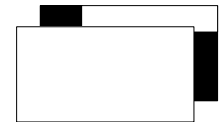
# Example



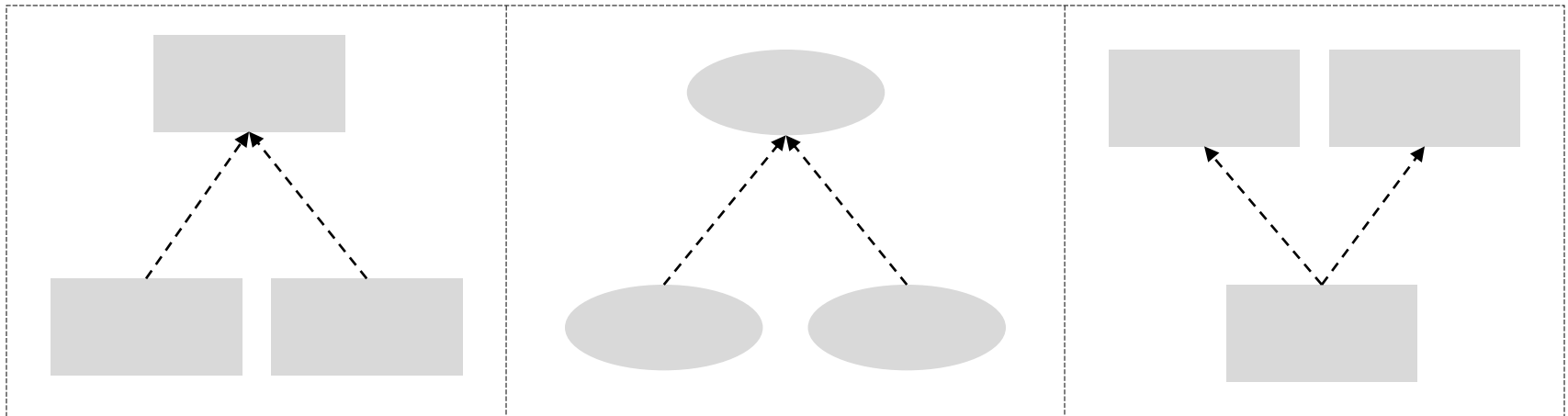
Roles



Kinds



# Classification Graphs



- $C_2 \preccurlyeq C_1$  denotes that  $C_2$  is a subcollection of  $C_1$
- For a collection  $C$ 
  - $\text{kinds } C = \{ K \mid C \preccurlyeq K \wedge K \text{ is a kind} \}$
  - $\text{roles } C = \{ R \mid C \preccurlyeq R \wedge R \text{ is a role} \}$   
 $= \{ R \mid C \preccurlyeq R \wedge R \text{ is not a kind} \}$

# Controlling Evolution

- Assumptions
  - each classification structure has a single root
  - a root is a kind
  - if  $C \preceq C_1$  and there is no  $C_2$  with  $C_1 \preceq C_2$  then  $C_1$  is the maximal collection of  $C$
  - each collection has a single maximal collection
- Migration  $x :: C_1 \rightarrow C_2$  valid if
  - a)  $x$  does not belong to a subcollection of  $C_1$
  - b)  $x$  can lose a kind only if it loses the contextual role of that kind  
 $\forall K \in (\text{kinds } C_1 - \text{kinds } C_2): \exists R \in \text{roles } K \Rightarrow R \notin \text{roles } C_2$

# Migration Example

- $x :: \text{Postgrads} \rightarrow \text{Lecturers}$ 
  - no  $x$  can belong to a subcollection of Postgrads
  - $\text{kinds Postgrads} - \text{kinds Lecturers}$   
 $= \{\text{Postgrads}, \text{Persons}\} - \{\text{Persons}\} = \{\text{Postgrads}\}$
  - $\forall K \in \{\text{Postgrads}\}: \exists R \in \text{roles } K \Rightarrow R \notin \text{roles Lecturers}$
  - $K = \text{Postgrads}$   
 $\exists R \in \text{roles Postgrads} \Rightarrow R \notin \text{roles Lecturers}$   
 $\exists R \in \{\text{Students}\} \Rightarrow R \notin \{\text{Lecturers}, \text{NonProfessors}, \text{Staff}\}$
- Migration is valid

# Migration Example

- $x :: \text{Postgrads} \rightarrow \text{Undergrads}$ 
  - no  $x$  can belong to a subcollection of Postgrads
  - $\text{kinds Postgrads} - \text{kinds Undergrads}$   
 $= \{\text{Postgrads}, \text{Persons}\} - \{\text{Persons}\} = \{\text{Postgrads}\}$
  - $\forall K \in \{\text{Postgrads}\}: \exists R \in \text{roles } K \Rightarrow R \notin \text{roles Lecturers}$
  - $K = \text{Postgrads}$   
 $\exists R \in \text{roles Postgrads} \Rightarrow R \notin \text{roles Undergrads}$   
 $\exists R \in \{\text{Students}\} \Rightarrow R \notin \{\text{Undergrads}, \text{Students}\}$
- Migration is invalid

# Literature

- Moira C. Norrie: **An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems**, In: *Proceedings of ER*, 390-401, 1993
- Moira C. Norrie: **Distinguishing Typing and Classification in Object Data Models**, In: *Information Modelling and Knowledge Bases VI*, H. Kangassalo, H. Jaakkola, K. Hori and T. Kitahashi, eds, IOS Press, 1995
- Alain P. Würigler: **OMS Development Framework: Rapid Prototyping for Object-Oriented Databases**, *PhD Thesis*, ETH No. 13512, 2000

# Next Week

## Object Model Language: OML

- Collection Algebra
- Language Design
- Data Definition, Manipulation and Query Language

