Objects in GIS
AND
set theory
AND
Boolean operations

This is lecture 4

Data models versus structures

- Brief review:
- What is a data model?
- What is a data structure, and how is it related to a data model?

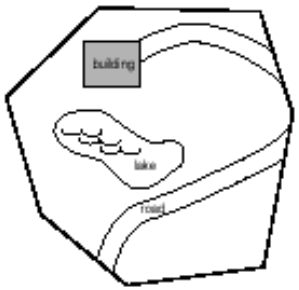
Definition of a data model

- a means of conceptualizing geographic space for computational implementation.
- It is the way we imagine space inside the computer.
- Data structures are a means of implementing data models. They are simply tools for organizing data.
- Data models a higher level of abstraction than data structures.

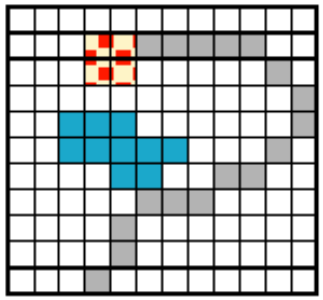
Object oriented Database structures

- OODBS are the most recent and innovative model.
- Problem is that OO can refer to different levels of implementation from conceptualization to formalization to DBS.

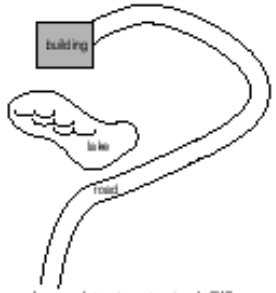
Objects: geographic entities without containers



polygon with geographic features (road, lake and building)



raster grid with same geographic features



In an object-oriented GIS, the container or perimeter is removed. Objects exist in space independent of jurisdictional boundaries.

Object-oriented hierarchical database model

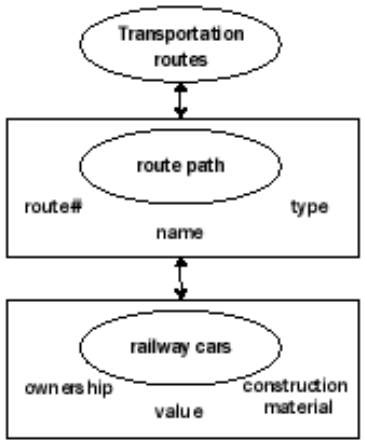


diagram derived from Burroughs, 1998, p. 49

Transportation route data organized in an object-oriented hierarchy

(note the lack of redundancy)

Object-oriented data models consist of geographic objects which belong to object classes. Each class inherits characteristics from its parent class. Objects and classes are organized hierarchically. Once data is encapsulated in objects, they can only be accessed by queries or messages.

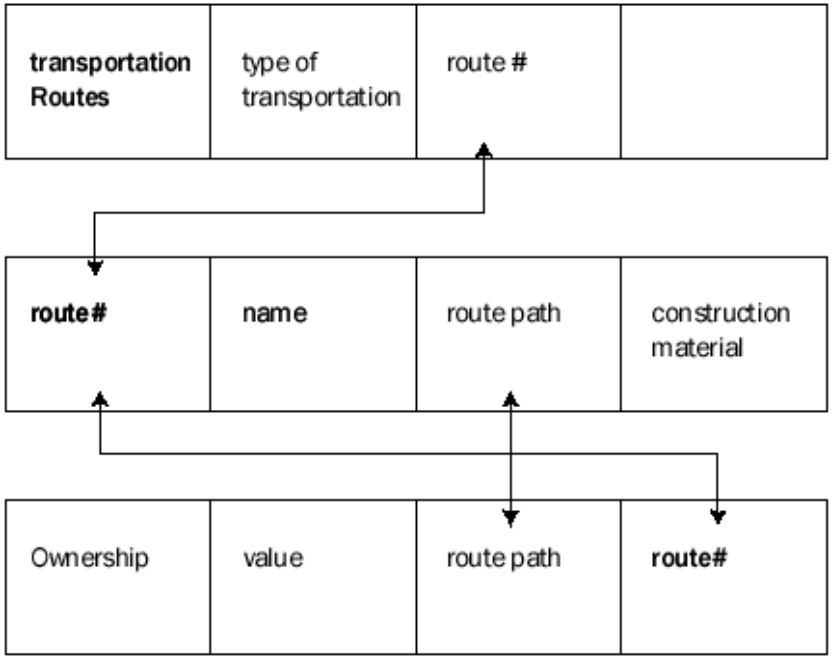
Early OO

- The basic ideas underlying OO were introduced 25 years ago through the Simula programming language. Today the best known OOPLA is C++ and now JAVA.
- OO migrated into geography in the early 1990s.
- OO changed data model possibilities from raster/vector to field/object.

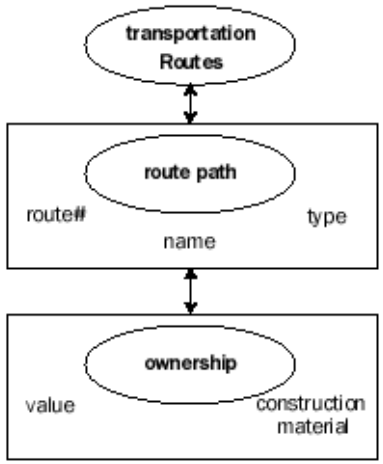
What is different about OO?

- OO is an attempts to get around problems of *impedance mismatch* (where certain problems defy formalization).
- Impedance mismatch is partly produced when the way that ideas (or objects) are constructed at one level is completely different at another.

Difference between relational and obect-oriented databases



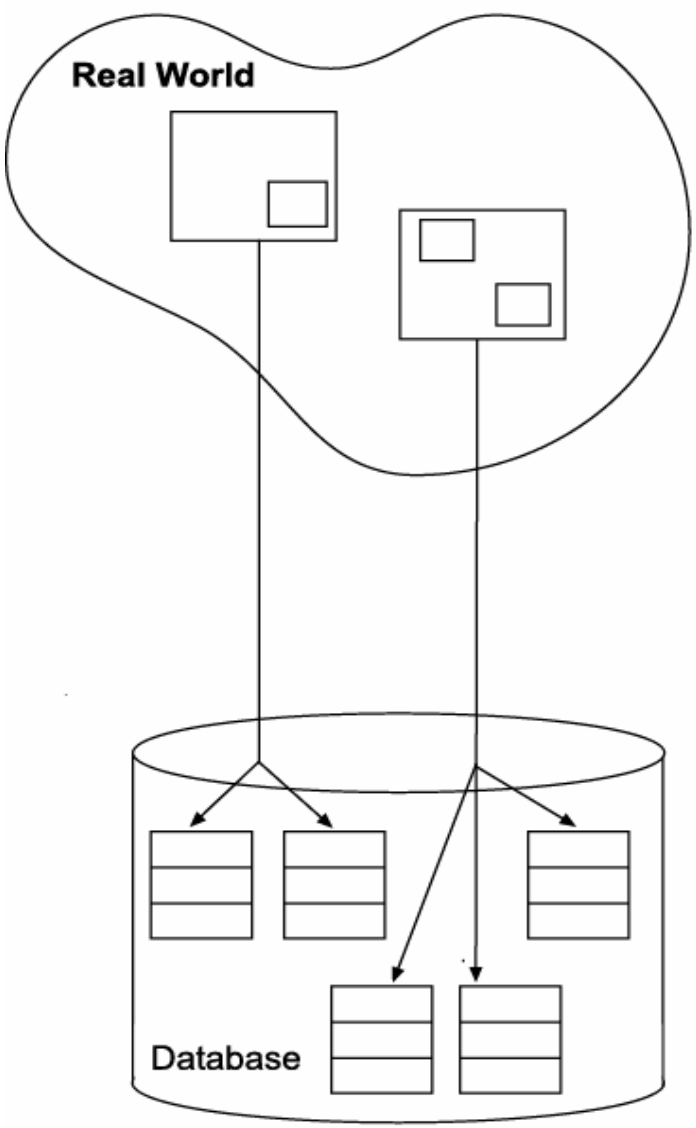
Relational database model for transportation routes



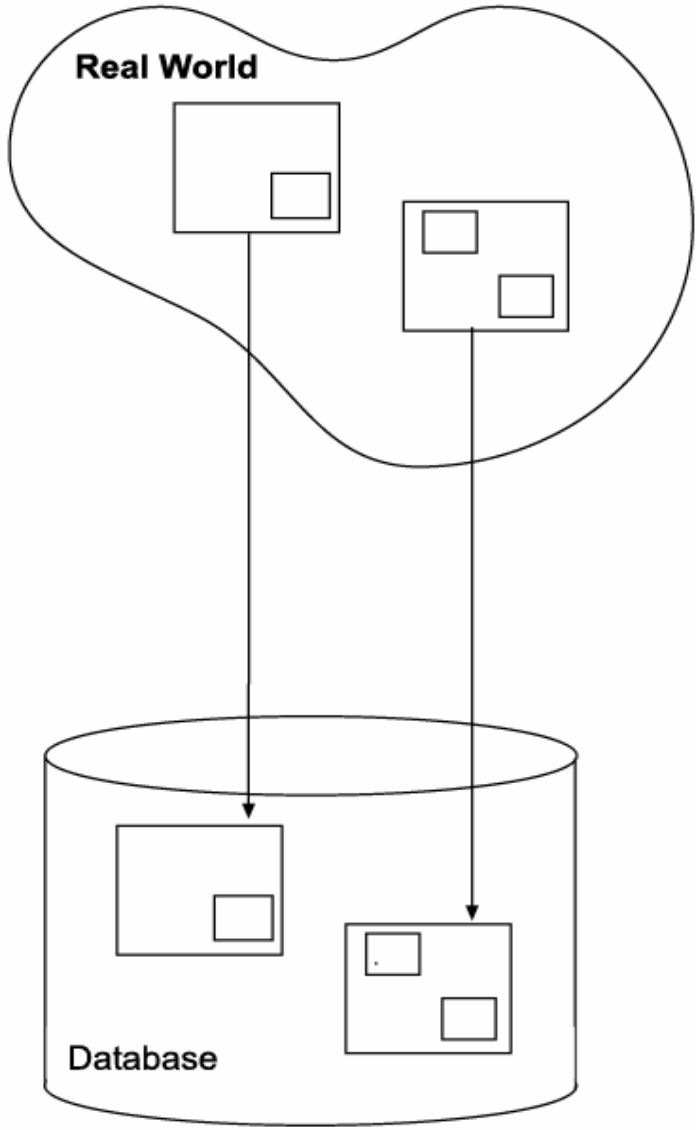
The same data organized in an object-oriented hierarchy

(note the lack of redundancy)

Differences between RDBMS and OODBMS



Relational Model
One Object → Many Records



Object Oriented Model
One (real) entity → one (database) object

What does OO do differently?

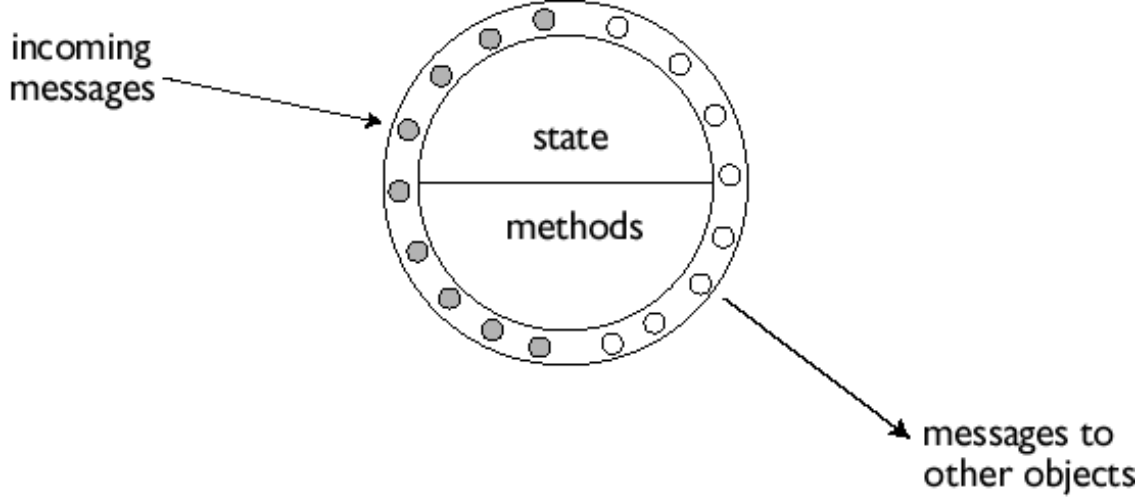
- Objects include dynamic behaviour which is not possible in a RDBS.
- The static qualities of an object are listed as *attributes*, just like in a RDBS. The totality of attributes values for an object at any one time is its *state*.
- The dynamic behaviour of objects is also included. This is expressed as a set of operations that the object can perform. So a car wouldn't just be defined by a list of its parts (attributes) but also the operations it can carry out (like *drive* and *park*).

Attributes, Methods and State

- The key notion is that *object* + *state* = *functionality*.

Object Methods and Messaging

- The code that describes operations objects can execute refer to *methods*.
- This is a much more active type of relation than we see in the ER model, underlying the RBDS.



Object messaging

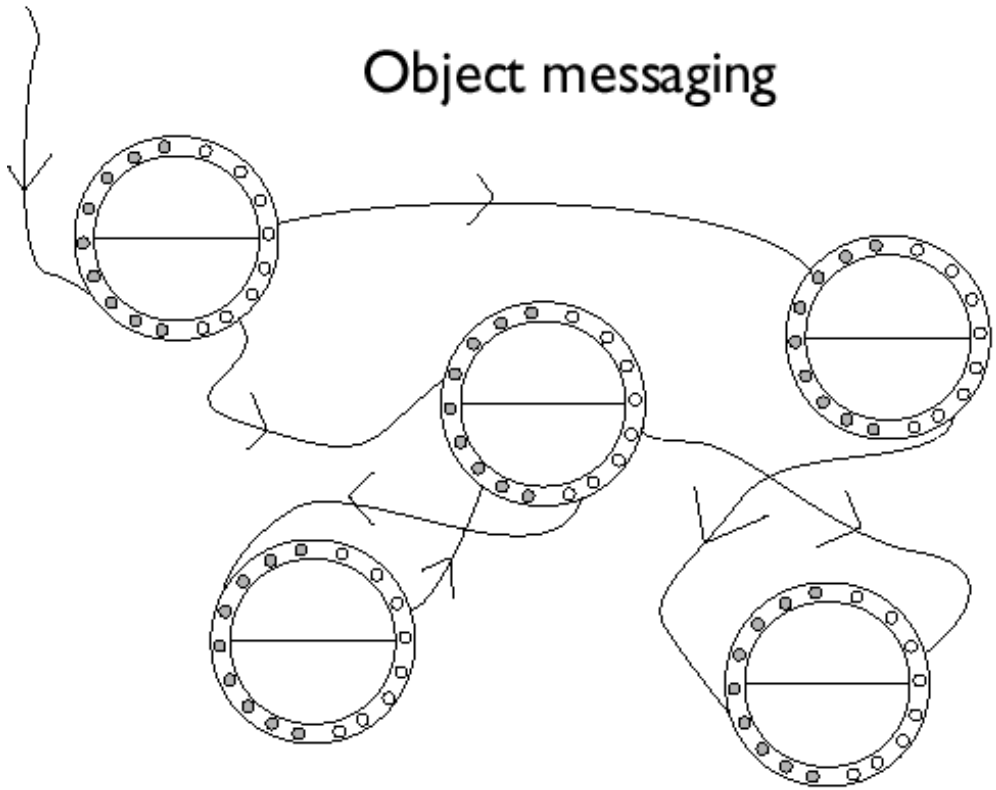


diagram adopted from Wotroys, M. 1995. GIS: A Computing Perspective. London: Taylor & Francis: 379.

Static and dynamic objects

- Objects send messages to other objects and receive them.
- Objects are both static and dynamic. There attributes are static while their potential operations are dynamic.

Object Classes

- Objects that have the same types of instance variables are in the same *class*. Classes share similar data structures and methods
- Objects with similar behaviour have the same *type*.
- An object *class* is an implementation construct while an object *type* is a semantic construct. Classes are based on organizational structure while types are based on behaviour.

Object Identity

- Objects are given an identify even though their attributes might change.
- Most OO systems support object identity.
- NOTE that RDBMS do not allow identity.

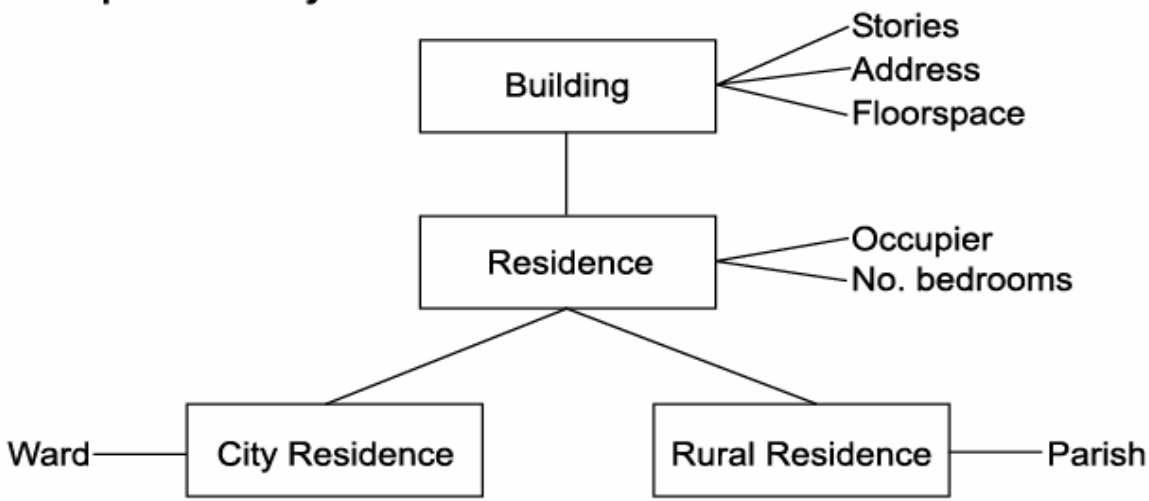
Object Encapsulation

- Objects clearly don't use the E/R model that links entities and their relationships to different attributes.
- Instead they use *enscapsulation*.
- **At any one time, the state of an object is determined by the values of the items that are encapsulated in the wrapper.**
- All of these data items together are referred to as *instance variables.*
- This is distinct from the ER model in which there is a two-part structure which separates entities from the relations.

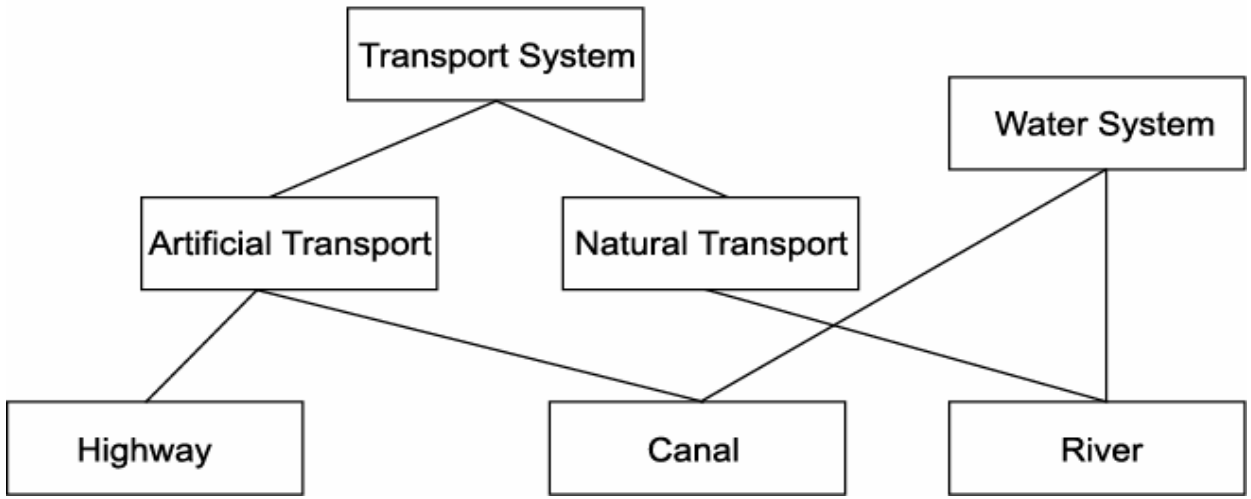
Object Inheritance

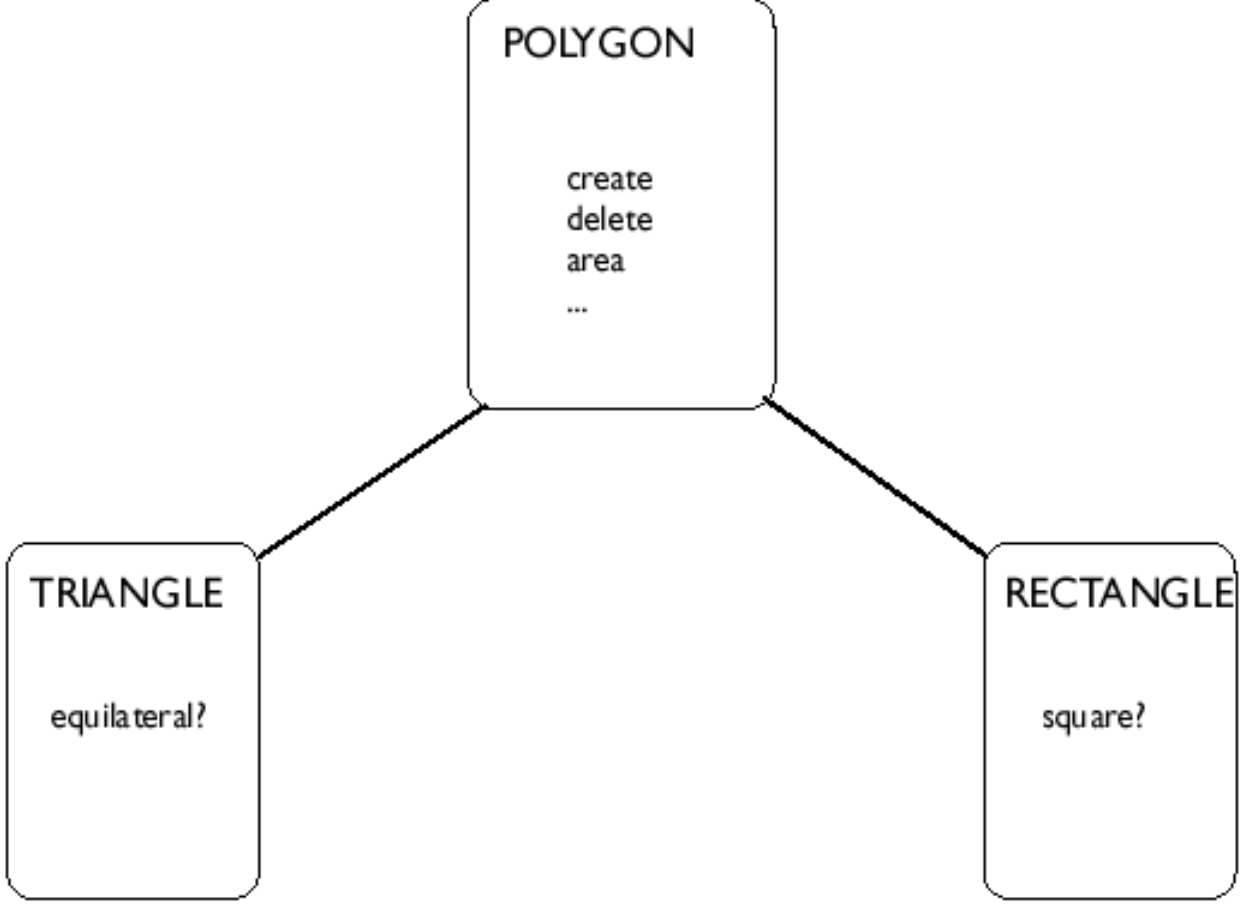
- *Inheritance* is another key concept associated with OO systems. Inheritance is why encapsulation works.
- Using inheritance, objects inherit characteristics from other objects.

Simple Hierarchy



Complex Hierarchy





Object inheritance of operations between types and sub-types

Generalization and specialization

- Inheritance works in two ways that are the inverse of each other. The first is *generalization* and the second is *specialization.*
- **Generalization**: abstracting the properties of several object types. For instance, car, motorcycle, motorboat, train, and truck generalize to **vehicle**
- **Specialization**: partitioning object types, according to use. For instance **settlement** specializes to city, town, village and hamlet.
- Subtypes inherit all the attributes and methods from the supertype but also add their own. So, for example, the Type Polygon gives rise to the sub-types rectangle and square and triangle.

Aggregate objects

- Objects can be made up of *aggregate* objects. DEFINITION: “An aggregate object is semantically an extended object that is treated as a unit in many operations, although physically made of several lesser objects” (Worboys, 1995, 89)
- An aggregate object is distinct from *association* of objects, in which a collection of objects is formed, but all are of the same type (like an association of lakes).

Polymorphism

- Methods have different results when applied to different objects. This characteristic is called *polymorphism.*
- Different responses will be evoked using the same method, on different objects.
- Polymorphism enables tremendous flexibility in OO systems because operations are consolidated only at implementation with a specific object.

Building Objects

- There are two ways to approach building an OO system
  1. extend an existing RDBS to handle OO programming or
  2. Build a database system around an OO programming language.

Final note on OO

- OO approach is much more complex than the relational model but hasn’t yet established universally accepted critieria for operations.