

**Object = vector GIS**

# Object, element, feature class dimension/property/dimension entities

**Category of dimensions (dimensions=properties) of entity values** (objects):

1. spatial dimension
  2. topological dimension
  3. attribute dimension
  4. graphic dimension – presentation of spatial objects
  5. time dimension
  6. textual numeric dimension
- 
- A white arrow originates from the right side of the first list item, 'spatial dimension', and points downwards to the right side of the sixth list item, 'textual numeric dimension'. The arrow is composed of a horizontal line segment, a vertical line segment, and a downward-pointing arrowhead.



# Object, element, feature class

## 1. spatial dimension

3

### Spatial objects

= located in the space = **contained** = (embedding spaces)

**A space** is **usually** thought ne of the following spaces:

- ▶ **2D** (= 2-dimensional) does not contain height information
- ▶ **2.5D** height is given as descriptive (attribute) data
- ▶ **3D** 3 dimensions
- ▶ **4D** time is the 4th dimension
  
- ▶ For more, see geodesy – coordinate systems

# Object , element, feature class

## 1 . spatial dimension

### Geometric dimension of spatial objects in GIS

- **dimensionless** - 0D – points
- **one dimensional** - 1D – line objects
- **two dimensional** - 2D – planar objects of finite size
- **three dimensional** - 3D – solids of finite volume with finite surface area (polyhedrons)

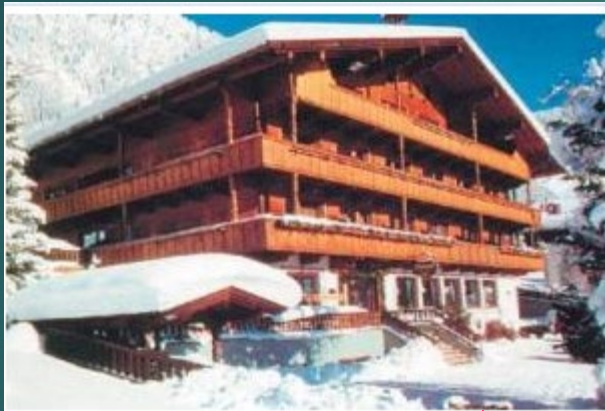


# Object , element, feature class

## 1. spatial dimension

Your choice

real object



You have 2 options for choosing which geometric type to choose in GIS:

model



- **1) point object** = the **symbol** for the building is used
- **2) surface object** = is displayed as a **polygon**

# Object , element, feature class

## 1. spatial dimension

Your choice

Real object



A possible **model**  
3 options:



**area** object



**line** object

**point** object = symbol



# Object, element, feature class

## 2. Topological dimension

### Topological dimension of a spatial object:

- ▶ **0. Vertex** – the point forming the shape of the edge = the shape of the object
- ▶ **1. Node** – the end/beginning of the object
- ▶ **2. Edge** – connector of 2 nodes
- ▶ **3. Line string** – more edges
- ▶ **4. Polygon** – area formed by a closed line string

# Object, element, feature class

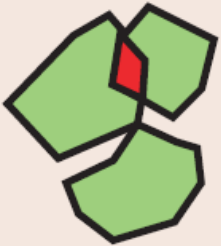

## 2. Topological dimension – topology check

**Polygons** of the same class must not overlap (parcel boundaries)


**Polygon**

### Must not overlap

Polygons must not overlap within a feature class or subtype. Polygons can be disconnected or touch at a point or touch along an edge.



Polygon errors are created from areas where polygons overlap.



A voting district map cannot have any overlaps in its coverage.

*Use this rule to make sure that no polygon overlaps another polygon in the same feature class or subtype.*



# Object, element, feature class

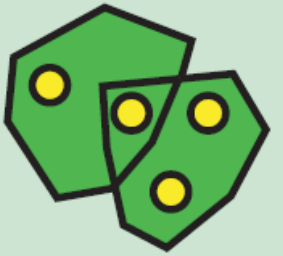
## 2. Topological dimension – topology check

**Polygons** – each must contain one point (parcel and its number)


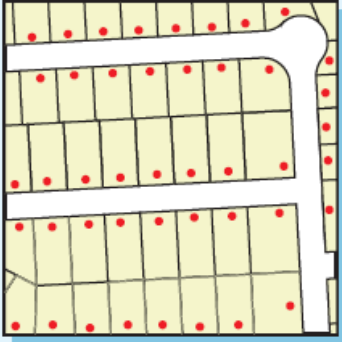

**Polygon**

### Contains point

Each polygon of the first feature class or subtype must contain within its boundaries at least one point of the second feature class or subtype.



Polygon errors are created from the polygons that do not contain at least one point. A point on the boundary of a polygon is not contained in that polygon.



Use this rule to make sure that all polygons have at least one point within their boundaries. Overlapping polygons can share a point in that overlapping area.

Parcels must contain at least one address point.

# Object, element, feature class

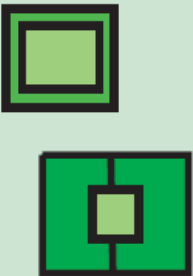
## 2. Topological dimension – topology check

### Polygons

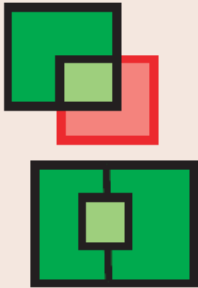
Each polygon of one class must be covered by polygons of the other class (districts are parts of regions)

**Polygon**

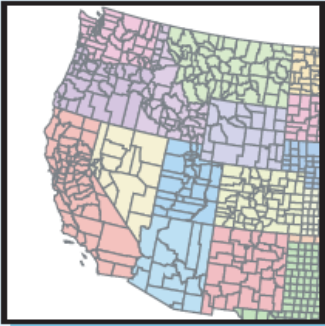
**Must be covered by feature class of**



The polygons in the first feature class or subtype must be covered by the polygons of the second feature class or subtype.



Polygon errors are created from the uncovered areas of the polygons in the first feature class or subtype.



States are covered by counties.

*Use this rule when each polygon in one feature class or subtype should be covered by all the polygons of another feature class or subtype.*



# Object, element, feature class

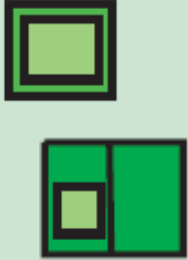
## 2. Topological dimension – topology check

**Polygons** – polygons of one class must be covered by a part of one polygon of another class

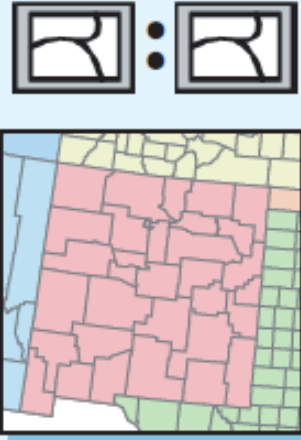

**Polygon**

**Must be covered by**

Polygons in one feature class or subtype must be covered by a single polygon from another feature class or subtype.



Polygon errors are created from polygons from the first feature class or subtype that are not covered by a single polygon from the second feature class or subtype.



Counties must be covered by states.

*Use this rule when you want one set of polygons to be covered by some part of another single polygon in another feature class or subtype.*

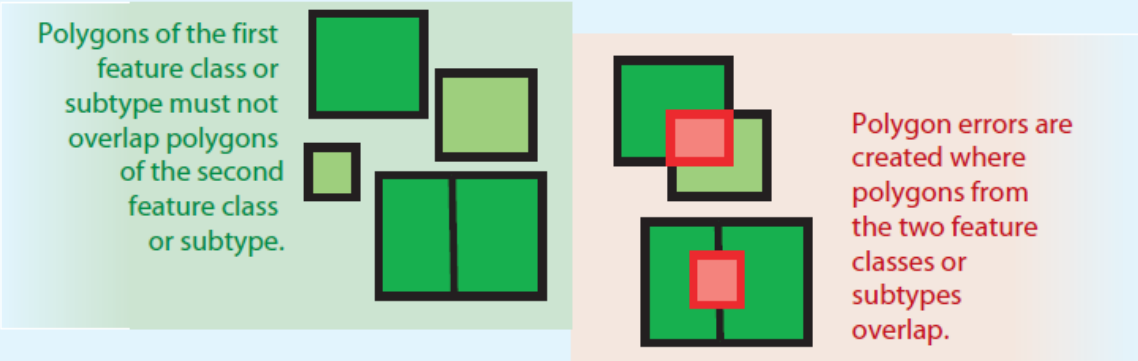
# Object, element, feature class

## 2. Topological dimension – topology check

**Polygons** – of one class must not be covered by polygons of another class


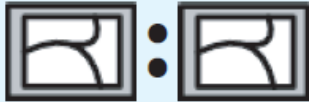
**Polygon**

### Must not overlap with



Polygons of the first feature class or subtype must not overlap polygons of the second feature class or subtype.

Polygon errors are created where polygons from the two feature classes or subtypes overlap.



Lakes and land parcels from two different feature classes must not overlap.

*Use this rule when polygons from one feature class or subtype should not overlap polygons of another feature class or subtype.*



# Object, element, feature class

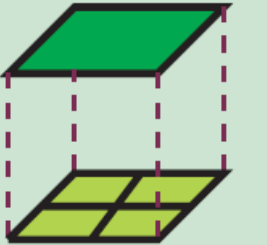
## 2. Topological dimension – topology check

**Polygons** – the boundaries of polygons of one class must be covered by the boundary of the other class  
(regions are divided into districts)



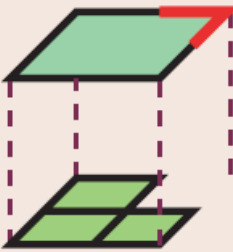
**Polygon**

### Area boundary must be covered by boundary of

The boundaries of polygons in one feature class or subtype must be covered by the boundaries of polygons in another feature class or subtype.



Line errors are created where polygon boundaries in the first feature class or subtype are not covered by the boundaries of polygons in another feature class or subtype.



Subdivision boundaries are coincident with parcel boundaries, but do not cover all parcels.

*Use this rule when the boundaries of polygons in one feature class or subtype should align with the boundaries of polygons in another feature class or subtype.*

# Object, element, feature class

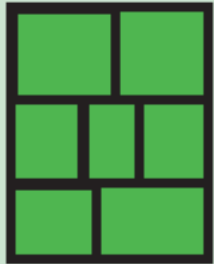
## 2. Topological dimension – topology check

14

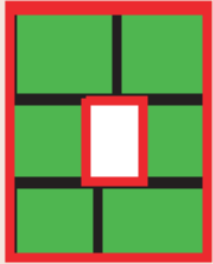
**Polygons** – the area of the territory must not contain holes

**Polygon**


### Must not have gaps



Polygons must not have a void between them within a feature class or subtype.



Line errors are created from the outlines of void areas in a single polygon or between polygons. Polygon boundaries that are not coincident with other polygon boundaries are errors.



Soil polygons cannot include gaps or form voids—they must form a continuous fabric.

*Use this rule when all of your polygons should form a continuous surface with no voids or gaps.*



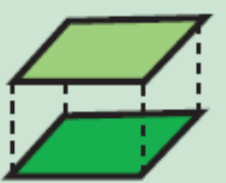
# Object, element, feature class

## 2. Topological dimension – topology check

**Polygons** of one class must be identical to polygons of another class

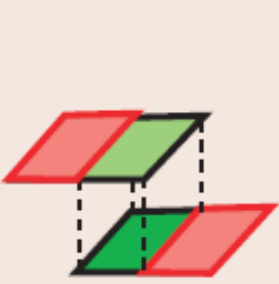
**Polygon**

### Must cover each other

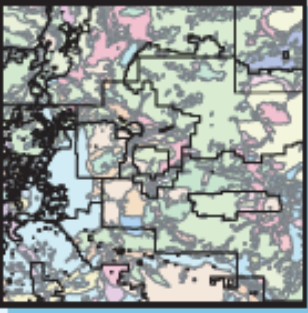


All polygons in the first feature class and all polygons in the second feature class must cover each other.


- FC1 Must be covered by feature class of FC2.
- FC2 Must be covered by feature class of FC1.



Polygon errors are created where any part of a polygon is not covered by one or more polygons in the other feature class or subtype.



Vegetation and soils must cover each other.



*Use this rule when you want the polygons from two feature classes or subtypes to cover the same area.*

# Object, element, feature class

## 2. Topological dimension – topology check

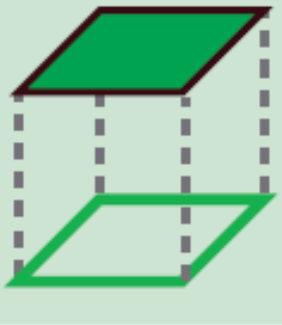
16

**Polygons × polygon boundaries** – polygons of one class must be covered by another line class (parcels and their boundaries)



**Polygon**

### Boundary must be covered by

Polygon boundaries in one feature class or subtype must be covered by the lines of another feature class or subtype.




Line errors are created where polygon boundaries are not covered by a line of another feature class or subtype.



Use this rule when polygon boundaries should be coincident with another line feature class or subtype.

Major road lines form part of outlines for census blocks.





# Object, element, feature class

## 2. Topological dimension - topology check

### Polygons × boundaries between them



- if neighboring boundaries are within tolerance of each other, they will coincide during topology cleaning – distance threshold must be set

**Line or Polygon**


### Must be larger than cluster tolerance

Cluster tolerance is the minimum distance between vertices of features.

Vertices that fall within the cluster tolerance are defined as coincident and are snapped together.



Any polygon or line feature that would collapse when validating the topology is an error.



Soil polygons must be larger than the cluster tolerance.

*This rule is applied to all line and polygon feature classes that participate in the topology.*


# Object, element, feature class

## 2. Topological dimension – topology check


**Polygons × points** – all points must lie completely inside polygons (capitals lie inside states)

**Point**

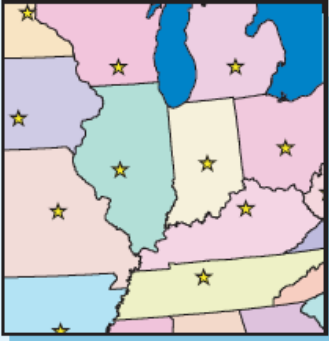
### Must be properly inside polygons




Points in one feature class or subtype must be inside polygons of another feature class or subtype.



Point errors are created where the points are outside or touch the boundary of the polygons.



State capitals must be inside each state.






# Object, element, feature class

## 2. Topological dimension – topology check


**Polygons × points** – all points of one class lie on the border of polygons (all border crossings lie on the border of states)

**Point**


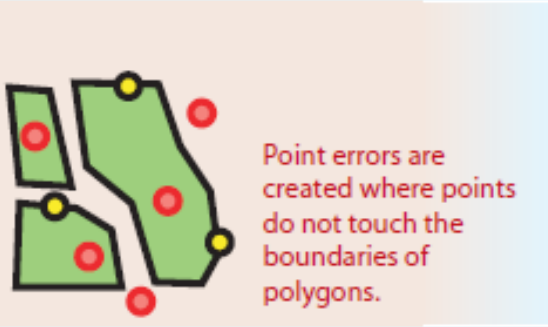
**Must be covered by boundary of**



Points in one feature class or subtype must touch boundaries of polygons from another feature class or subtype.



Point errors are created where points do not touch the boundaries of polygons.



Use this rule when you want points to align with the boundaries of polygons.

Utility service points might be required to be on the boundary of a parcel.

# Object, element, feature class

## 2. Topological dimension – topology check


20

**Lines** must not have free ends (parcel boundaries must be closed to form area boundaries)



**Line**

### Must not have dangles

The end of a line must touch any part of one other line or any part of itself within a feature class or subtype.



Point errors are created at the end of a line that does not touch at least one other line or itself.



*Use this rule when you want lines in a feature class or subtype to connect to one another.*

A street network has line segments that connect. If segments end for dead-end roads or cul-de-sacs, you could choose to set as exceptions during an edit session.



# Object, element, feature class

## 2. Topological dimension – topology check


21

**Lines** of one class must not have the same position as another line of the same class, but they can cross and touch



**Line**

### Must not overlap

Lines must not overlap any part of another line within a feature class or subtype. Lines can touch, intersect, and overlap themselves.



Line errors are created where lines overlap.



Lot lines cannot overlap one another.

*Use this rule with lines that should never occupy the same space with other lines.*

# Object, element, feature class

## 2. Topological dimension – topology check

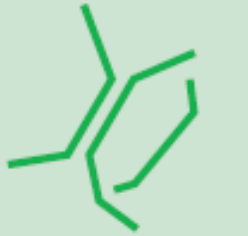
22

**Line** of the same class may not have the same position as another line of the same class, nor cross it, but may touch it



**Line**

### Must not intersect

Lines must not cross or overlap any part of another line within the same feature class or subtype.



Line errors are created where lines overlap, and point errors are created where lines cross.



Use this rule with lines whose segments should never cross or occupy the same space with other lines.

Lot lines cannot intersect or overlap, but the endpoint of one feature can touch the interior of another feature.



# Object, element, feature class

## 2. Topological dimension – topology check


23

**Lines** can touch at the end nodes of the line, but not cross or overlap

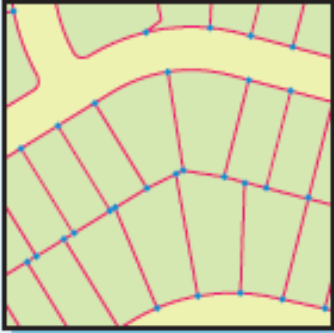

**Line**

### Must not intersect or touch interior

Lines can only touch at their ends and must not overlap each other within a feature class or subtype. Lines can touch, intersect, and overlap themselves.

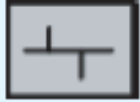


Line errors are created where lines overlap, and point errors are created where lines cross or touch.



Use this rule when you only want lines to touch at their ends and not intersect or overlap.

Lot lines cannot intersect or overlap and must connect to one another only at the endpoint of each line feature.



# Object, element, feature class

## 2. Topological dimension – topology check

24

**Line** – point objects must always lie on nodes – water meter (one class) is at the end of each connection (second class)

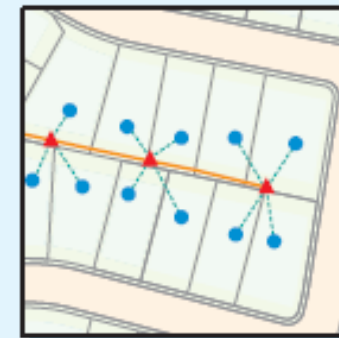
**Line**

### Endpoint must be covered by

The ends of lines in one feature class or subtype must be covered by points in another feature class or subtype.



Point errors are created at the ends of lines that are not covered by a point.



*Use this rule when you want to model the ends of lines in one feature class or subtype that are coincident with point features in another feature class.*

Endpoints of secondary electric lines must be capped by either a transformer or meter.



# Object, element, feature class

## 2. Topological dimension – topology check

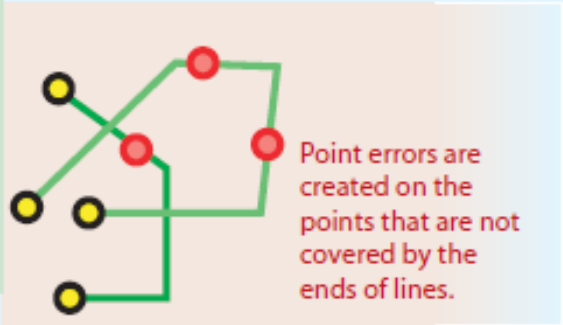

25

**Line × points** – each point must lie only on the end point of the line


**Point**

**Must be covered by endpoint of**


Points in one feature class or subtype must be covered by the ends of lines in another feature class or subtype.



Point errors are created on the points that are not covered by the ends of lines.



Street intersections must be covered by the endpoints of street centerlines.



# Object, element, feature class

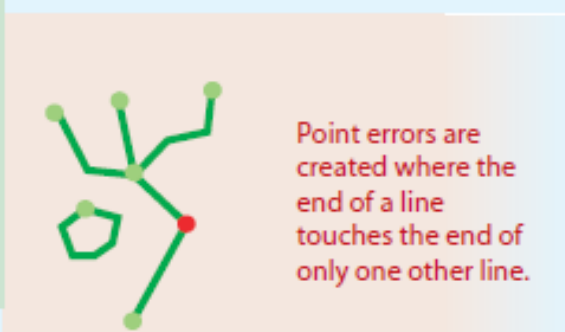
## 2. Topological dimension – topology check

26

**Line × Nodes** – it must not have nodes where only one line of the same class exits and enters – to clean the topology

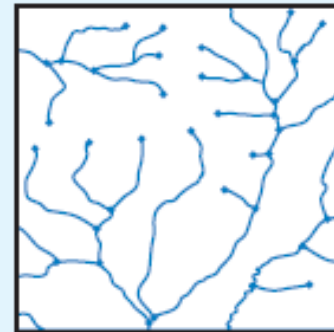
**Line**

### Must not have pseudonodes




The end of a line cannot touch the end of only one other line within a feature class or subtype. The end of a line can touch any part of itself.

Point errors are created where the end of a line touches the end of only one other line.



Use this rule to clean up data with inappropriately subdivided lines.

For hydrologic analysis, segments of a river system might be constrained to only have nodes at endpoints or junctions.





# Object, element, feature class

## 2. Topological dimension – topology check

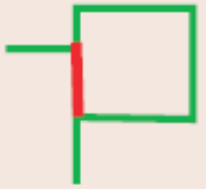

27

**Line** – must not overlap itself

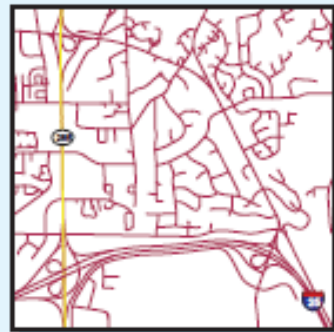
**Line**

### Must not self overlap

Lines must not overlap themselves within a feature class or subtype. Lines can touch, intersect, and overlap lines in another feature class or subtype.



Line errors are created where lines overlap themselves.



Use this rule with lines whose segments should never occupy the same space as another segment on the same line.

For transportation analysis, street and highway segments of the same feature should not overlap themselves.

# Object, element, feature class

## 2. Topological dimension – topology check

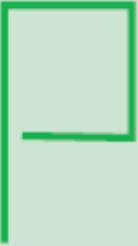
28

**Line** – must not cross itself – a node must be inserted


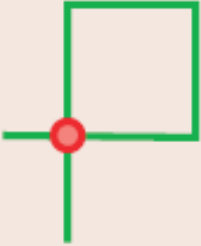
**Line**

### Must not self intersect

Lines must not cross or overlap themselves within a feature class or subtype. Lines can touch themselves and touch, intersect, and overlap other lines.

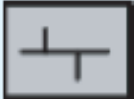


Line errors are created where lines overlap themselves, and point errors are created where lines cross themselves.



Use this rule when you only want lines to touch at their ends without intersecting or overlapping themselves.

Contour lines cannot intersect themselves.





# Object, element, feature class

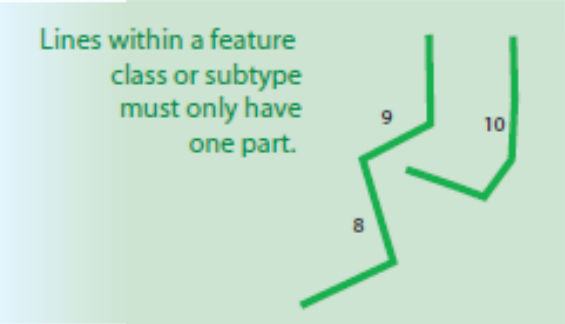
## 2. Topological dimension – topology check

29

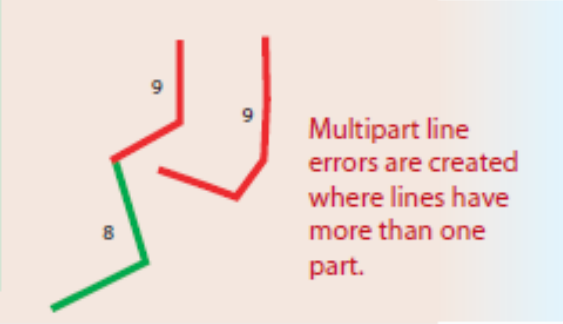
**Lines** – cannot have 2 separate lines marked as one continuous line  
(line 8, 9, 10 on the left × 8, 9 and 9 on the right)

**Line**


### Must be single part



Lines within a feature class or subtype must only have one part.



Multipart line errors are created where lines have more than one part.



A highway system is made up of individual features where any one feature is not made up of more than one part.

*Use this rule when you want lines to be composed of a single series of connected segments.*

# Object, element, feature class


## 2. Topological dimension – topology check

30


**Line × line** – for 2 different classes must have the same geometry  
(a bus route is identical to the road network)

**Line**


**Must be covered** by feature class of



Lines in one feature class or subtype must be covered by lines in another feature class or subtype.




Line errors are created on the lines in the first feature class that are not covered by lines in the second feature class.



Lines that make up bus routes must be on top of lines in a road network.

Use this rule when you have multiple groups of lines describing the same geography.





# Object, element, feature class

## 2. Topological dimension – topology check


31

**Line** – the line of one class must not overlap the line of another class  
(the watercourse must not have the same position as the road adjacent to it)



**Line**

### Must not overlap with

Lines in one feature class or subtype must not overlap any part of another line in another feature class or subtype.



Line errors are created where lines from two feature classes or subtypes overlap.



Use this rule for lines that should never occupy the same space with lines in another feature class or subtype.

Highways can cross and come close to rivers, but road segments cannot overlap river segments.

# Object, element, feature class


## 2. Topological dimension – topology check

32

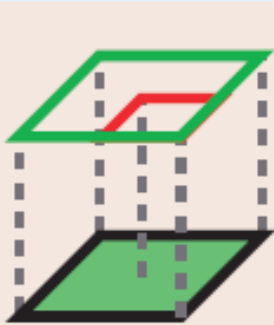
**Lines × Polygons** – the line must be covered by the boundary of the polygon

**Line**


**Must be covered by boundary of**



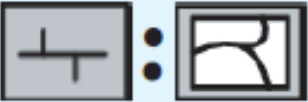
Lines in one feature class or subtype must be covered by the boundaries of polygons in another feature class or subtype.



Line errors are created on lines that are not covered by the boundaries of polygons.



Polylines used for displaying block and lot boundaries must be covered by parcel boundaries.



*Use this rule when you want to model lines that are coincident with the boundaries of polygons.*



# Object, element, feature class

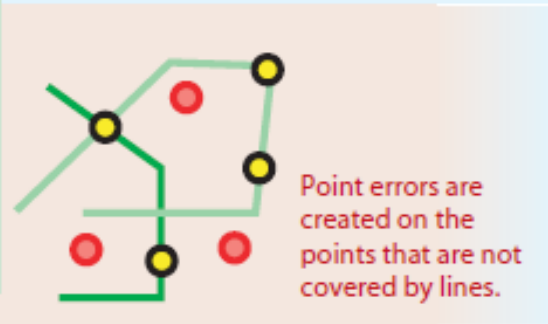
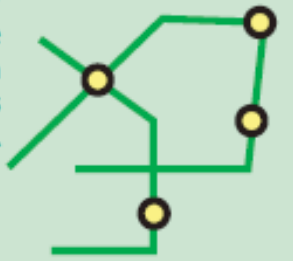
## 2. Topological dimension – topology check

### Points × Line – points must lie on a line

**Point**

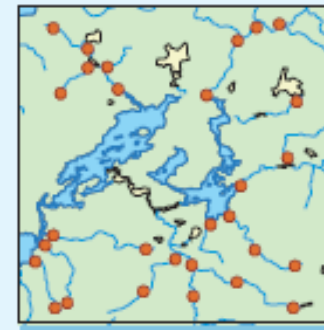
#### Point must be covered by line

Points in one feature class or subtype must be covered by lines in another feature class or subtype.



Point errors are created on the points that are not covered by lines.

*Use this rule when you want to model points that are coincident with lines.*



Monitoring stations must fall along streams.

# Object, element, feature class

## 3. Attribute dimension = descriptive data in GIS

34

Descriptive properties = **attributes** = descriptive data

- they specify individual elements – they define dimensionally or qualitatively properties = they represent the quality of the object,
- one method of measurement cannot be established for all

### **examples of attributes:**

- ▶ *number of floors*
- ▶ *owner*
- ▶ *connection to the public water supply*
- ▶ *last fix*



# Object, element, feature class

## 3. Attribute dimension

35

**Attribute dimension** is given by the number of attributes:

- ▶ at least one appears = the **key attribute** is marked
- ▶ **maximum number of** attributes is not limited
- ▶ each object has its **own individual attribute values**
- ▶ each object **may not** have all attribute values listed
- ▶ each class has its own list of attributes

# Object , element, feature class

## 3. Attribute Dimension: from class, entity and attribute records

1 **class** is displayed in 1 table

1 **object** is displayed in 1 line

1 **attribute** (descriptive data) – listed in 1 column

**key attribute** = unique (often referred to as **ID** – see below)  
must always be present for all entities, other attributes do not have to be and all values from 1 attribute do not

Attribute 1= <b>ID</b>	Attribute 2	Attribute 3	Attribute 4	Attribute 5
1	Vltava	280	...	...
2	Berounka	152	...	...
3	Labe	....	...	...



# Object , element, feature class

## 3. Attribute dimension: data type

37

### Attribute data type (value domain specification)

= Specifies applicable values for individual attributes (green)

- ▶ **Attribute** data type
- ▶ **number of floors** = number (integer type)
- ▶ **owner** = text string (specify number of characters)
- ▶ **connection to the public water supply** = boolean type (yes/no)
- ▶ **last fix** = number (date type)

# Object, element, feature class

## 3. Attribute dimension: data types

38

### data types

#### Automatic number

**key attribute / primary key** = base attribute that must be u  
of each object for each class entered !!

Specifies uniqueness within a class that is unique in the database

**An automatic number** ensures the uniqueness of the key attribute –  
from set as data type (=auto number)



# Object, element, feature class

## 3. Attribute dimension – descriptive data in GIS – data types

39

### Numeric data types

**Number** – can be used to checkmate. operation

*subtypes* :

#### 1 . integer

**integer (Integer = Smallint)**

= **2 bytes** = 16 bits signed :  $-\frac{1}{2} \cdot 2^{16}$  ,  $+\frac{1}{2} \cdot 2^{16} - 1$

= 16 bits unsigned : 0,  $+ 2^{16} - 1$

**long integer = 4 bytes = 32 bits** :

-2,147,483,648 to + 2,147,483,647

For reference:

**1 byte** (byte) = 8 bits (0 – 255,  $256 = 2^8$ )

# Object, element, feature class

## 3. Attribute dimension – descriptive data in GIS – data types

40

### Numeric data types

#### 2. Decimal number

**single precision** (single precision , FLOAT) = decimal number - positive and negative values

size 4 bytes = 32 bits, total number of values is  $2^{32}$

**double precision** = decimal number - positive and negative values

size 8 bytes = 64 bits, total number of values  $2^{64}$

**more detailed decimal number -**

size 12 bytes = 96 bits, total number of values  $2^{96}$

***b-bit data takes up size  $b/8$  flats***



# Object, element, feature class

## 3. Attribute dimension – descriptive data in GIS – data types

41

### Numeric data types

#### 3. other numerical data types

**Date/time** - date of birth, entry in the cadastre – DB machines can handle it (selects the dates of one month, adds a week, ...)

**Currency** – data with precision from 1 to 4 decimal places,  
size 8 flats

# Object , element, feature class

## 3. Attribute dimension = descriptive data in GIS : domain

42

**Domain** = definition of the values of individual attributes (including system coordinates):

eg domain: for year: can be 1900 – 2005,

for text: number of characters,

Reason: **prevention** of errors

**prevention** redundant data volume sizes



# Object, element, feature class

43

## 3. Attribute dimension = descriptive data in GIS

### Data type **Text string**

alphanumeric characters + other characters –  
digits cannot be used for calculations

**domain** for text string -

- ▶ character limit,
- ▶ selection from a pre-prepared list of attribute values

# Object, element, feature class

44

## 3. Attribute dimension = descriptive data in GIS

**Data type BLOB** – binary large objects

it is a long sequence of binary numbers – image or multimedia data



# Object, element, feature class

45

## 3. Attribute dimension = descriptive data in GIS

### Data type **Boolean**

only two possible values (=1-bit value, possible values: 0, 1):

Yes × No

# Object, element, feature class

## 3. Attribute dimension = descriptive data in GIS

46

### Descriptive data in GIS

- ◉ **elements within one class** – are different from each **other** values of *individual attributes*
- ◉ Based on attribute values, classes can either be **split** or **merged**
- ◉ The division into classes does **not have to be immutable**!
  - ◉ Pay attention to the division of area classes



# Object, element, class of elements

## 4. Graphic dimension = descriptive data in GIS - graphic attributes

47

class **view** method:

**point** – symbol, size

**line** – line type, thickness, colour

**polygon** – line type for the border,  
area filling (yes, no)  
method of filling

# Object, element, class of elements

## 5. Time dimension

48

Time changes of geobjects = **dynamics**

- **time resolution** = determined by the GIS processor – different for different objects – geology × transport network × state of the atmosphere (time resolution)
- **chrono** = smallest time unit
- **time interval** = (time span) = set of time-ordered chronomes (time interval)



# Object, element, feature class

## time dimension /dimension

49

### Kinds of time:

- ◉ **valid time** = moment of occurrence of the event (valid time)
- ◉ **transaction time** = time when the change is entered into the database (transaction time)
- ◉ **user time** = time if the event is used as data in the database system (user time)

# Object, element, feature class

## 5. Time dimension -- spatio-temporal processes

50

### Possible changes

- **change of position** = spatial limitation (spatial displacement) - changes geometry and maybe topology
- **spatial reduction (enlargement)** = (spatial expansion / reduction) - the topology does not need to be changed
- **changing attributes**

**time in GIS** – complex, difficult manipulation



# Object, element, feature class

## Summary – class definition

51

### So the feature class is

the sum of elements (objects or phenomena) that are

- ▶ defined by a single set of attributes
- ▶ are of the same geometric type
- ▶ they may not always be displayed with the same graphic attribute

*Ex. cities,*

*water courses,*

*local authorities,*

*construction completion date...*

# Object, element, feature class

summary

entity = element = phenomenon or object (specific element)  
or phenomenon

**Entity (= element)** in GIS must be:

- ▶ **identifiable** = distinguishable from others (**database, class** and **ID**)
- ▶ **relevant** = required for the given application (see section on **modelling**)
- ▶ **describable** = by its properties (choice of **attributes**)