Last week

- We have discussed the different means to store spatial data in databases with focus on PostGIS
- Spatial data handling requires:
  - Specification of geometries
  - Specification of geometry type and subtype
  - Vertex (coordinate values) as numbers / strings / arrays
  - Spatial reference systems
- Different methods enable the construction of geometries to save them to the respective spatial data type in the database.

Geo875 | FS25
University of Zürich

3. Lecture Spatial Databases

Topology in Spatial Databases

Rolf Meile

Eidg. Forschungsanstalt für Wald, Schnee und Landschaft (WSL)

Swiss Federal Institute for Forest, Snow and Landscape Research

Esra Suel

Dept. of Geography, University of Zürich

Learning Objectives

- Understand how spatial databases can efficiently handle spatial (topological) relationships
- Understand how topological relationships can be enforced and stored in a persistent manner in order to ensure data integrity
- Get an idea how software driven topology implementations can guarantee topologically correct base data
- Understand how topological relationships between objects are computed on the fly and how you can query them

Overview

1. **Objects in space**
2. Storing topological relationships
3. Evaluating topological relationships
4. Querying topological relationships using SQL

Exercise 1

a) What are your spontaneous associations when you hear of "topological relationships?

b) You have already learnt some things about topological relationships. Discuss with your neighbour three applications where you analysed / enforced / saved topological relationships;

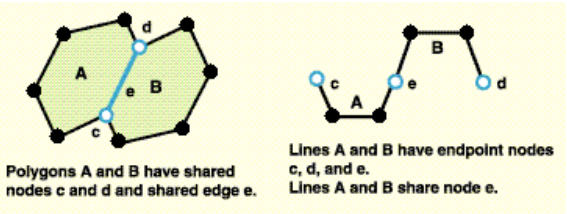
Objects in space

- Topological relationships – recall Geo243
  - Relationships that are invariant under continuous distortion;
  - A circle is topologically equivalent with an ellipse or a square.
- Topological relationships are **queried** when answering questions as:
  - Is this tree inside this parcel?
  - What are the neighbouring parcels?
  - What streets connect to this street?
- These are *qualitative* relationships (as we know from ER relationship types = the ‘verbs’)
- We need to ensure that some qualitative relationships are **enforced** in a DB (e.g., parcels not overlapping)
- We will only focus on topology for vector objects (no raster/TINs)

Application of topology in spatial databases

- Increasing speed and efficiency
  - The more vertices a geometry does have, the longer it takes to evaluate binary relationships of geometries; but we have spatial indices;
  - **Precomputing** and **storing** topology enables the use of simple algebraic or combinatorial approaches rather than complex evaluation over all vertices;
  - Standard procedure with GIS-Client and file-based data: always storing intermediate results; not necessarily the case with DBs;
- Reducing storage (and transfer) size:
  a. Adjacent geometries may share edges (if 2D) or nodes (1D) – explicit storage of topology enables the reduction of storage size and assures integrity;
  b. But it requires an additional physical data model to store this;
- Assuring data integrity:
  - assuring that modeled **relationships are enforced** (very important for, e.g, legal, money reasons – land ownership)

Source: Esri docs

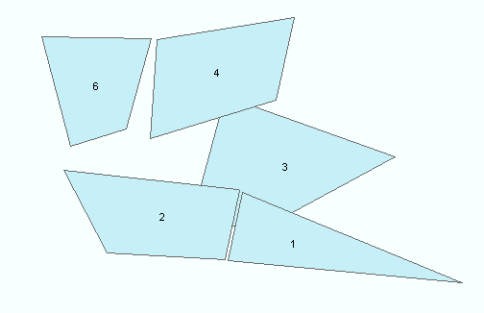


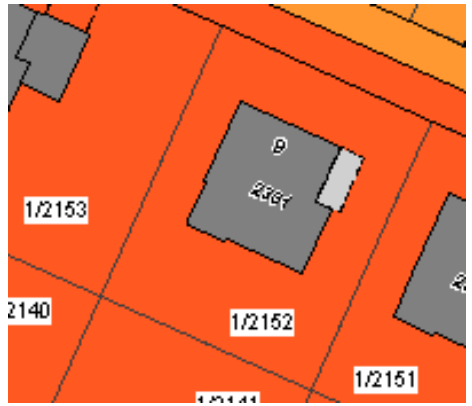
Overview

1. Objects in space
2. **Storing topological relationships**
3. Evaluating topological relationships
4. Querying topological relationships using SQL

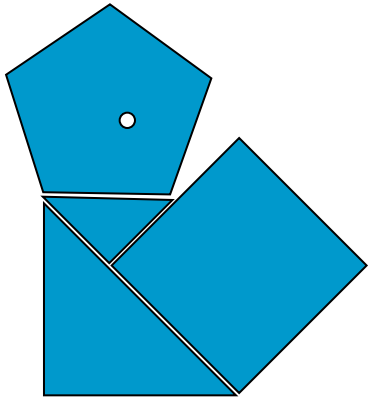
Storing topological relationships

Geometric view of things

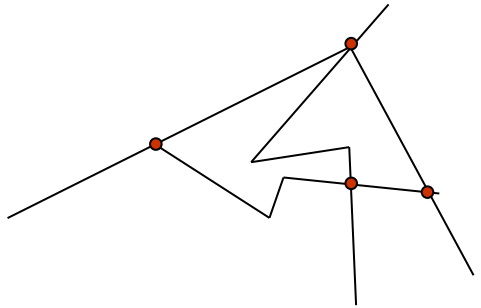




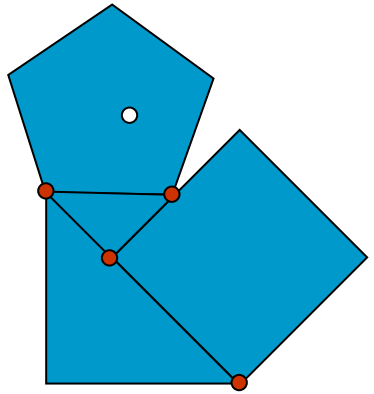
Typology of saved datasets



Spaghetti data model



Network data model

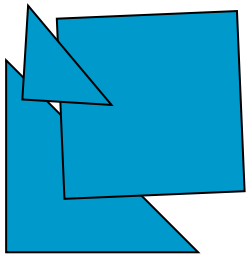


Topological data model

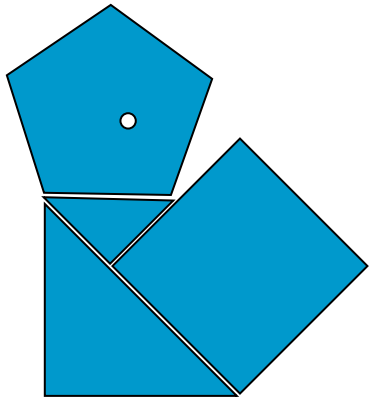
Spaghetti data model

Does not store any topological relationship

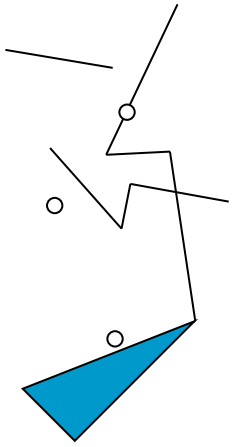
1. Does not enforce any topological relationships (data integrity) of spatial records neither in the **same** table nor **between multiple spatial** tables
2. Topological relationships between objects need to be computed ‘live’ by spatial database means or other external tools
3. On the fly checks: assuring data integrity could happen during data creation/update with additional logic (= software in DB or external)
4. Spaghetti model is the "standard" spatial data storage



or



or



polygons only (1 layer)

points & polygons (2 layers)

points, polygons & lines (3 layers)

Spaghetti data model (cont.)

1. **Pros**
   a. Simplicity in data modeling and loading
   b. Independently stored geometries – when there is no previously known constraint on their spatial relationship
2. **Cons**
   a. If certain relationships have to be maintained, this must be done in the application logic layer; usually complex; no ‘shortcut’ possible;
   b. Redundancy of storage of shared boundary edges – no way around this!
   c. Evaluation of relationships with classical spatial index
   d. Pre-storing would speed up some things;

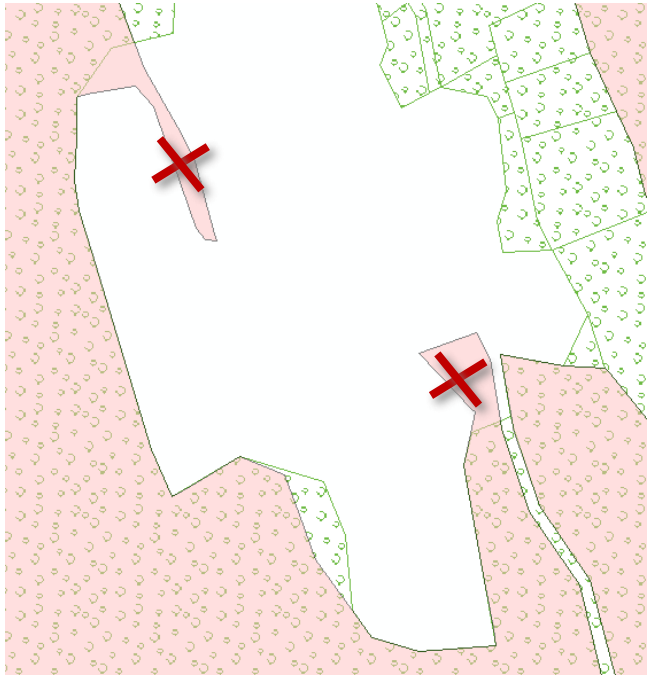
Inconsistency - a recent example from WSL

Expected and 'communicated' data

a. For a Swiss Canton: Its forested area (green)
b. In addition: Areas where cantonal contracts with forest owners exist (red); subsidies for not using forest resources;
c. We thought: Implicitly, first dataset spatially contains second dataset

Data recieved

a. Detailed communication on **both** sides is important
b. Know more about the data
c. Do **never** expect correct topological data
d. Labels of datasets and columns are not necessarily self-explaining; more metadata reading or talking required; forest <> forest;
e. Who does the 'topology job'?



should be topologically **contained** in forest?!

Exercise 2

How would you have checked for this issue with the tools you know so far? Have you ever done that with data you recieved? Discuss with your neighbours.

Spaghetti data model - with software topology

Data stored the 'spaghetti way' but software on top checks topological consistency

Topology Application

- Capable of dealing with all kind of (topological) rules
- Managing different users working at the same time with the same spatial data
- Reconciling data versions of user and versions over time
- On the fly checks of rule chains -> spatial queries!
- Mechanisms for solving contradictory results - a partially manual process

Geo-DB containing...

- Spatial datasets (of spaghetti type)
- Rulesets for topology software - created by users
- Versions of edited, unedited or partially edited data

Tidy up rules for spaghettis

- ArcGIS applicable rules for spatial datasets in geodatabases

  - Must Be Larger Than Cluster Tolerance
  - Must Not Overlap
  - Must Not Have Gaps
  - Must Not Overlap With
  - Must Be Covered By Feature Class Of
  - Must Cover Each Other
  - Must Be Covered By
  - Boundary Must Be Covered By
  - Area Boundary Must Be Covered By Boundary Of
  - Contains Point
  - Contains One Point
  - Must Be Larger Than Cluster Tolerance
  - Must Not Overlap
  - Must Not Intersect
  - Must Not Intersect With
  - Must Not Have Dangles
  - Must Not Have Pseudo Nodes
  - Must Not Intersect Or Touch Interior
  - Must Not Intersect Or Touch Interior With
  - Must Not Overlap With
  - Must Be Covered By Feature Class Of
  - Must Be Covered By Boundary Of
  - Must Be Inside
  - Endpoint Must Be Covered By
  - Must Not Self-Overlap
  - Must Not Self-Intersect
  - Must Be Single Part
  - Must Coincide With
  - Must Be Disjoint
  - Must Be Covered By Boundary Of
  - Must Be Properly Inside
  - Must Be Covered By Endpoint Of
  - Point Must Be Covered By Line

Spaghetti data model - with topology (cont.)

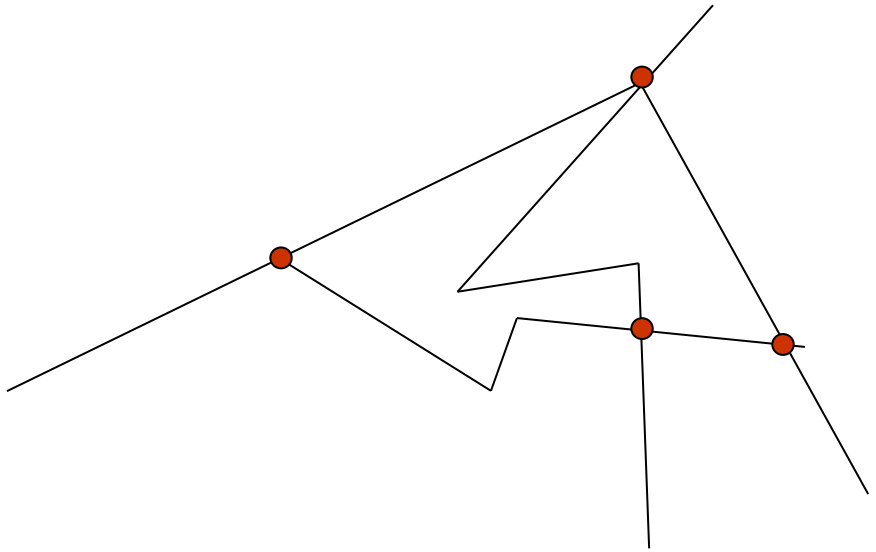
1. Pros
   a. A lot of advanced scenarios possible and implemented
   b. Available rules can be combined to chains and processes
   c. Expandable models

2. Cons
   a. Additional software / storage mechanism needed
   b. Different versions of data makes querying temporarily difficult
   c. Access through other software e.g. SQL more complicated

We will see an additional example for that under 'network data model'. The example was established with software and underlying 'spaghetti data model';

Network data model

1. Explicitly stores topological relationships in a (mathematical) graph structure
   a. Edges, or arcs (if oriented)
   b. Nodes
2. Suitable for storage of relationships between e.g., linear structures (utilities, ducts, streets [incl. lanes]), social networks, transport networks
3. Data structures:
   1. Adjacency matrix
   2. Edge-node list

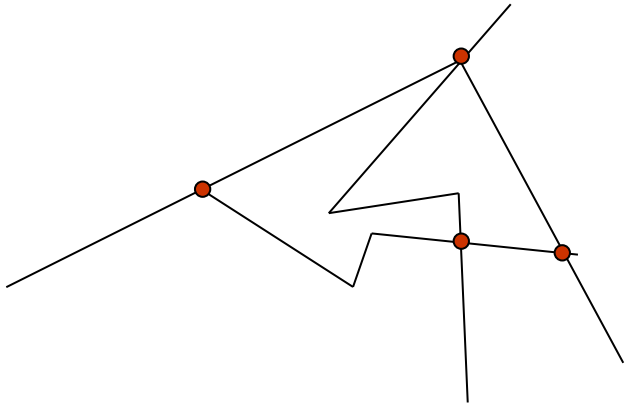


Network data model (cont.)

1. Pros:
   a. Suitable for routing and structural analysis (shortest paths, centrality analysis);
   b. Explicit description of the topology
   c. Removes the concern for geometry although we still need it
   d. Because **references** to surrounding geometries are stored (with explicit attributes) -> this leads to graphs -> fast calculations
2. Cons:
   a. Not simple to store information about relationship of 2D geometries
   b. Additional data are stored beyond the geometry itself (but this applies to all explicit topology storage)

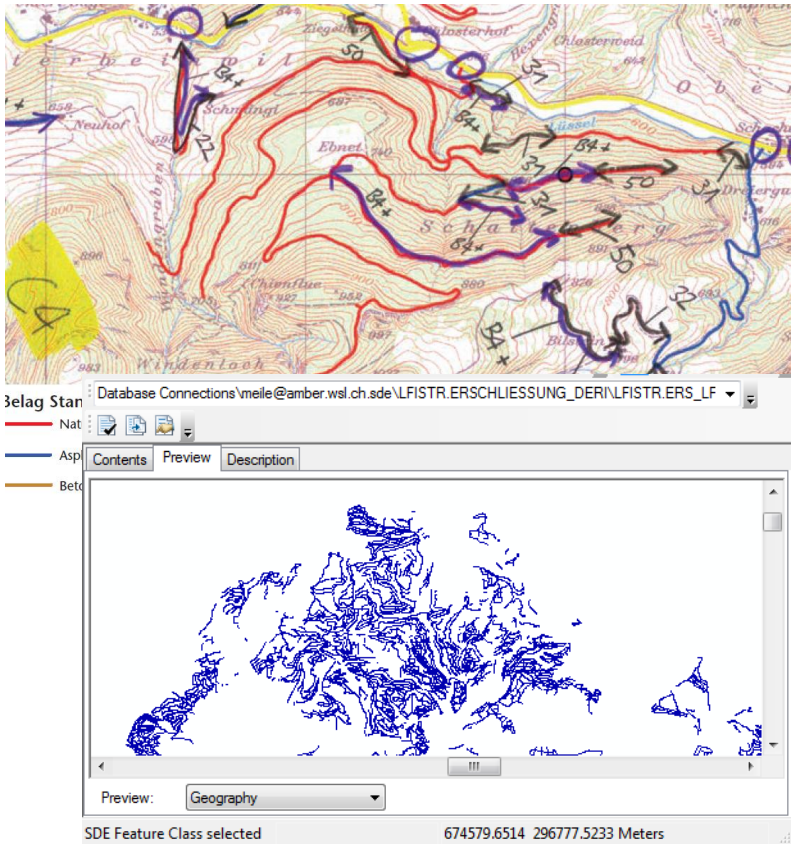
Graph databases (e.g. Neo4j) as a storage for topology?

- Yes, possible; with spatial extension; not really used for topological analyses (yet);



Network data - but a 'spaghetti' example

1. **What we did**
   a. Collection of forest road information from local forestry service
   b. Why? Potential wood supply and accessibility -> e.g. energy wood
   c. Classifying new roads plus roads on the base of swissTLM (only in forests)
   d. Digitize from analog drawings
2. **What we got**
   a. New attributes for swissTLM roads
   b. New 'unknown' roads
   c. Accessibility for trucks - wood transport
   d. Dataset with no topology,
      no connectivity




```
SELECT COUNT(*) anz_lines
FROM LFISTR.ERS_LFI4;
-- result: 911923
```

http://szf-jfs.org/doi/pdf/10.3188/szf.2016.0136

Network data - an example (cont.)

3. What we did to polish up the dataset

a. Creating a fully-fledged road network; needs to be a topologically correct line network; forest roads have to be connected to all other Swiss roads;

b. Why? Routing (and therefore resource planning) for wood trucks

c. Initial dataset was digitized with snapping features enabled (**client side "topology"**)

d. Analysis of the current data by means of topological queries and topological tools;

e. Build a "network" (ArcGISPro); further analysis and error seeking is done;

f. Network analyst functions used on the 'network':
   - dangling roads
   - small 'sliver' roads
   - service area function (trafficability)
   - origin distance cost matrix; closest facility
   - checking graphically (!)

g. Pitfalls: missing connectors; errors in base dataset (no snapping during creation); complex geometries; tolerance for snaping too small;

Network data - an example (cont.)

4. Conclusions

a. Final dataset is of 'spaghetti' type; one spatial table with all the forest roads

b. This dataset reflects - but is not - a network because it was checked with topological functions (querying) and tools

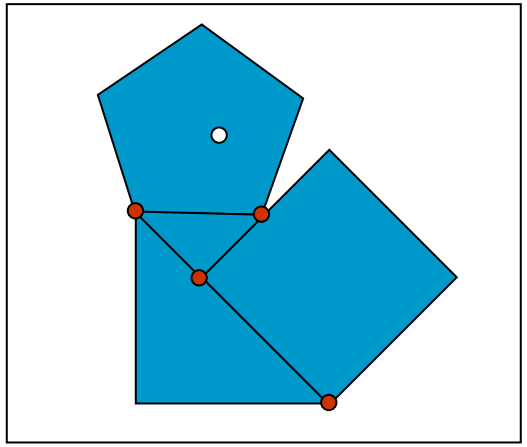
c. This example intensively uses the iterative method of
   - on-the-fly search for topological errors
   - fix the error in the dataset
   - recheck til no further corrections are needed

d. No 'structural topology' was used; But client-side 'software topology'
   such as *Build a network in ArcGISPro* in the previous slide - it created a local (not DB!) dataset that implemented a temporary local network model; some topo-functions only work with this type of data model;

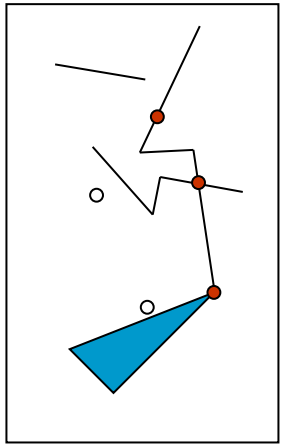
For routing calculations -> we can now transfer the dataset to a structural network model in the DB

Topological data model

1. Explicitly stores topological relationships of 0-*n*D geometries
2. Topological elements:
   a. Nodes [point, edge]
   b. Edges (~arc, if orientation is important) [node-start, node-end,right-face, left-face]
   c. Faces (= polygon) [{edges}]
3. Distinguishes:
   a. Start and End nodes (for **connectivity**)
   b. Left and Right faces (for **adjacency**)



Or



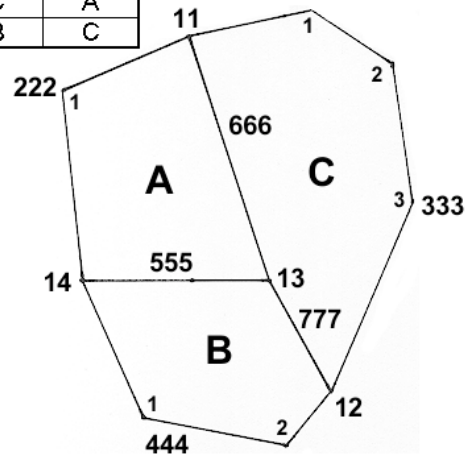
Topological data model (cont.)

1. The arc/node model:
   a. Simple example (coverage files)
   b. Complete with thematic attributes
2. Other models:
   1. Winged edge topology
   2. Half-edge topology
   3. Doubly connected edge list…

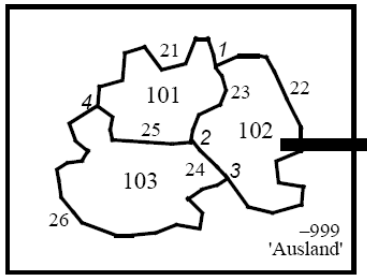
Arc / Node

	Fnode	Tnode	Lpoly	Rpoly
222	11	14	A	D
333	11	12	D	C
444	12	14	D	B
555	13	14	B	A
666	11	13	C	A
777	12	13	B	C

Node ID	x	y
11	x_{11}	y_{11}
12	x_{12}	y_{12}
13	x_{13}	y_{13}
14	x_{14}	y_{14}



Gemeinden: **Geometrie**



Topologie

Polygon-Ketten-Liste

Poly-ID	Ketten-ID
101	21, 23, 25
102	22, 23, 24
103	24, 25, 26

Ketten-Liste

Ketten-ID	AK	EK	PL	PR
21	4	1	–999	101
22	1	3	–999	102
23	2	1	101	102
24	2	3	102	103
25	4	2	101	103
26	4	3	103	–999

Metrik

Ketten-Koordinaten-Liste

Ketten-ID	Anz. Pte	Koords
21	19	x1,y1 x2,y2 x3,y3 . .
22	45	x1,y1 x2,y2 x3,y3 . .
23	11	x1,y1 x2,y2 x3,y3 . .
24	9	x1,y1 x2,y2 x3,y3 . .
25	19	x1,y1 x2,y2 x3,y3 . .
26	48	x1,y1 x2,y2 x3,y3 . .

Knoten-Koord-Liste

Knoten-ID	XK	YK
1	x	y
2	x	y
3	x	y
4	x	y

Gemeinden: **Thematik**

Attribut-Tabelle

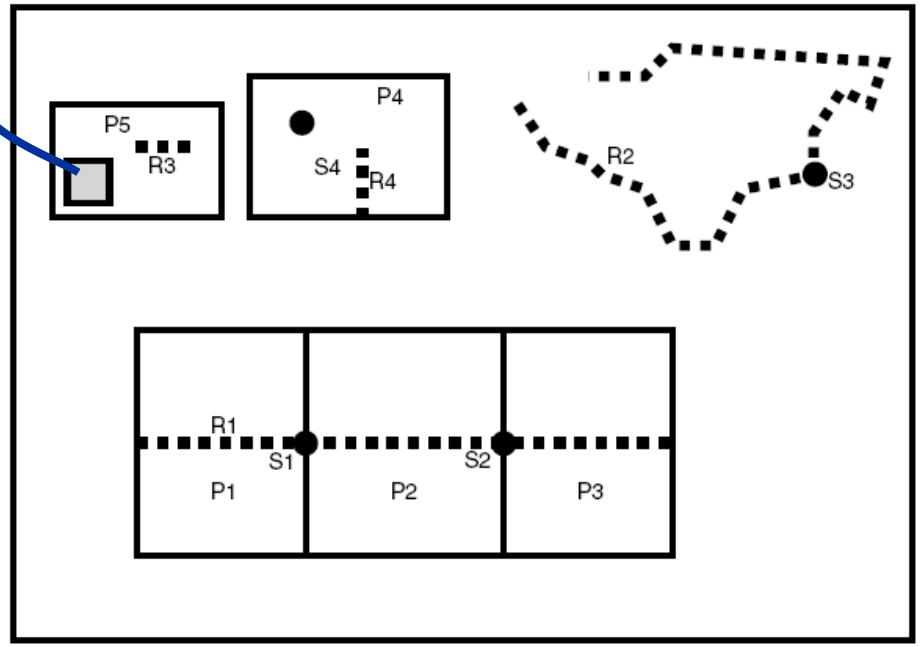
Poly-ID	Name	Fläche	Bev. 80	Bev. 90
101	GISwil	8.3	2021	2286
102	Vectoraz	10.7	4233	4112
103	Rasteringen	11.3	1225	1432

Structural Topology - Oracle

1. Oracle stores the topological relationships in a second set of tables
   a. Geometry tables
   b. Topology tables

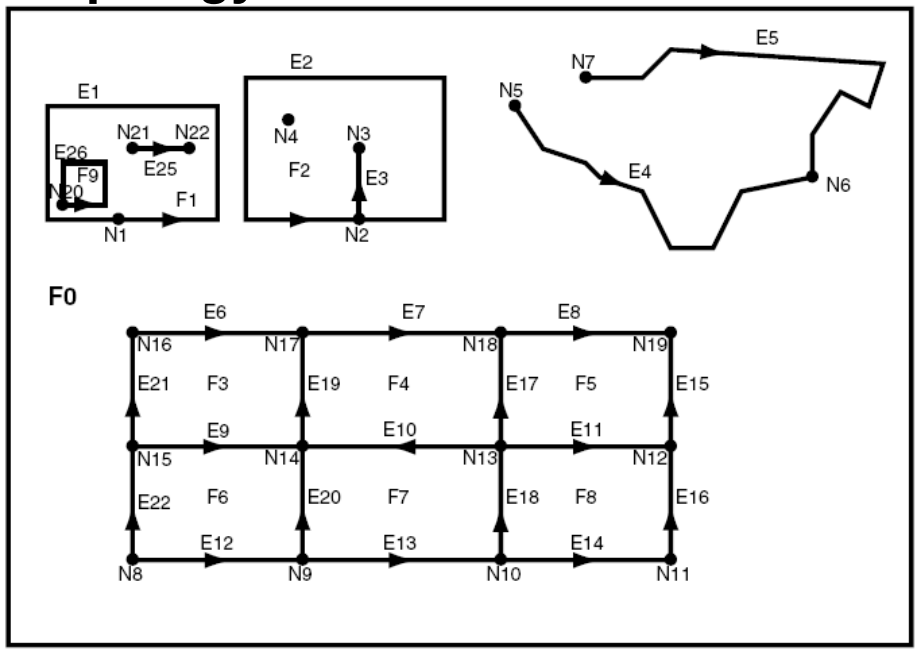
Hole in parcel

Geography view



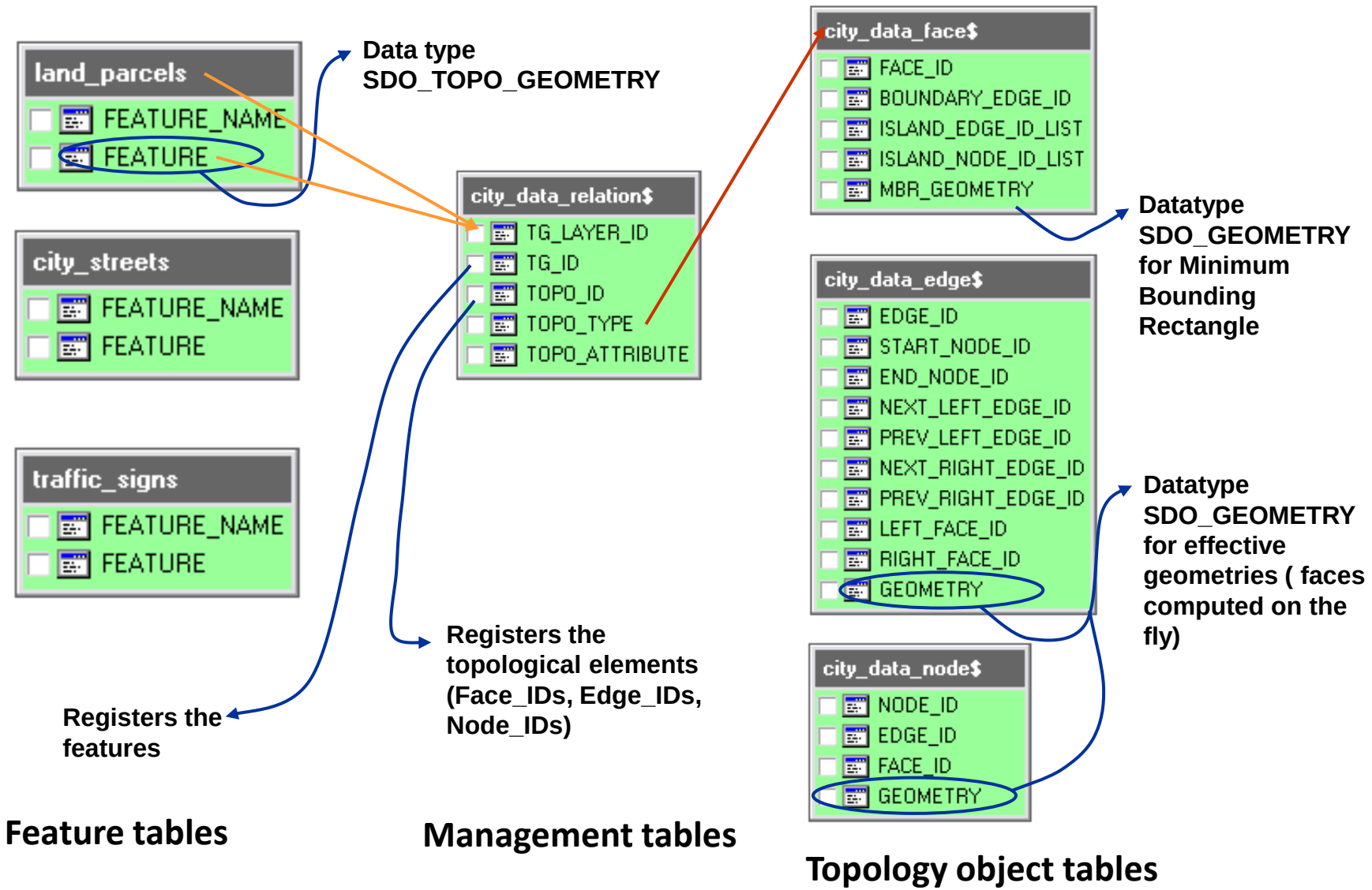
Parcels, Roads, Signals

Topology view



Faces, Edges, Nodes

Structural Topology – Oracle (cont.)



Structural Topology - Postgis with PostgreSQL

- Structurally similar with Oracle's implementation - means edge, node, face tables;
- Registering categories of things taking part in the topology using a topogeometry type (e.g. highways, state borders, signals); they are called 'layers’ in this context;
- What to insert through SQL and some PostGIS-functions now?
  - coming from the topological view: 1) insert edges, nodes, faces 2) 'register' the topogeometry thing to 'layers'. E.g. a certain highway with two previously inserted edges
  - coming from the geometry view: 1) insert your geometry e.g. a state and register it to 'layers' with one function 2) Having done that, edges, nodes and faces are created automatically
- Querying quite easy - visualizing possible through QGIS
- Editing with SQL but mostly with scripting; Versioning?

Topological data model - summary

1. Pros:
    1. Assures consistency (of stored geometries and their relationships)
    2. Reduces redundancy (?)
    3. Efficient neighbourhood or routing querying

2. Cons:
    a. Stores some objects that do not have geographical base (nodes);
    b. Stores additional information about organisation of nodes/edges/faces;
    c. Complex data structures for storage;
    d. Slower insert – topology has to be constructed and updated;
    e. If faces are not stored (see oracle example), area has to be calculated on the fly

Overview

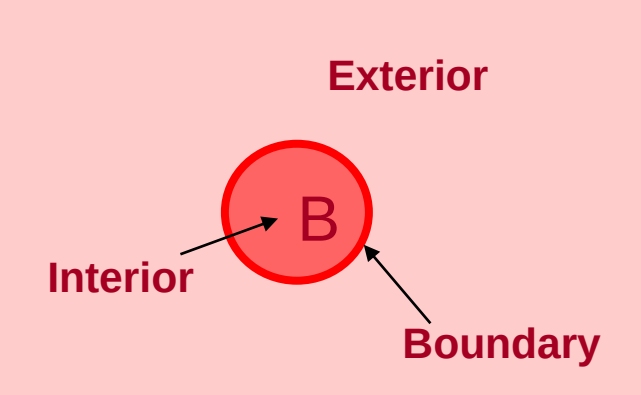
1. Objects in space
2. Storing topological relationships
3. **Evaluating topological relationships**
4. Querying topological relationships using SQL

Evaluating topological relationships

- The foundation behind the previously discussed models – effective evaluation of topological relationships
- Data models (with own types and extra structure) can explicitly store the topological relationships (Oracle / PostGIS) aka **structural topology**
- Comparison methods of geometries' boundaries, interiors and exteriors enable computation of their relationships
- This allows 'on the fly' querying - **adhoc spatial SQL –** between any desired geometries in any desired (even the same) table
- All **software-driven topology** is built on spatial queries (e.g. Esri's software stack)
- **Structural topology** is much less dependent on spatial queries but used less often in real life systems

9-intersection model (from Geo243)

- Egenhofer and others worked on a formal model of **topological spatial relations** in the late 1980s and early 1990s
- They came up with the 9-intersection model which describes **binary topological relationships** in terms of **interiors**, **boundaries** and **exteriors** for two spatial entities in **two dimensions**



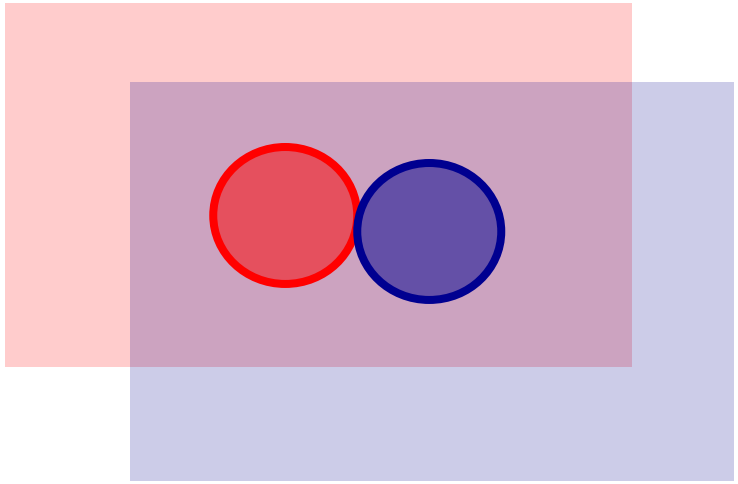
The **boundary** of an object separates its **interior** from its **exterior** (infinite)

➲ Egenhofer and Franzosa (1991)

9-intersection model – e.g. “A touches B”

Specific matrix for two polygons!

		Entity B		
		Interior	Boundary	Exterior
Entity A	Interior	False	F	T
	Boundary	F	True	T
	Exterior	T	T	T



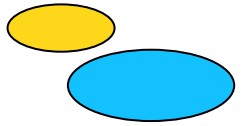
“A touches B” implies that:

Interior of A intersects Interior of B?	F
Interior of A intersects Boundary of B?	F
Interior of A intersects Exterior of B?	T
Boundary of A intersects Interior of B?	F
Boundary of A intersects Boundary of B?	T
Boundary of A intersects Exterior of B?	T
Exterior of A intersects Interior of B?	T
Exterior of A intersects Exterior of B?	T
Exterior of A intersects boundary of B?	T

9IM (cont.)

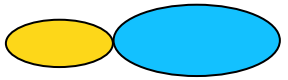
8 matrices are meaningful in 2D and have natural names

A disjoint B



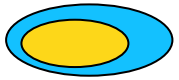
	B I	B B	B E
A I	F	F	*
A B	F	F	*
A E	*	*	*

A touches (meets) B



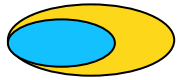
	B I	B B	B E
A I	F	*	*
A B	*	T	*
A E	*	*	*

A inside (within) B



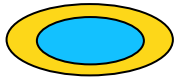
	B I	B B	B E
A I	T	*	F
A B	*	*	F
A E	*	*	*

A covers B



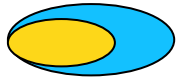
	B I	B B	B E
A I	*	*	*
A B	*	*	*
A E	F	F	*

A contains B



	B I	B B	B E
A I	T	*	*
A B	*	*	*
A E	F	F	*

A covered by B

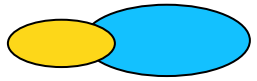


	B I	B B	B E
A I	*	*	F
A B	*	*	F
A E	*	*	*

A equals B

	B I	B B	B E
A I	T	*	F
A B	*	*	F
A E	F	F	*

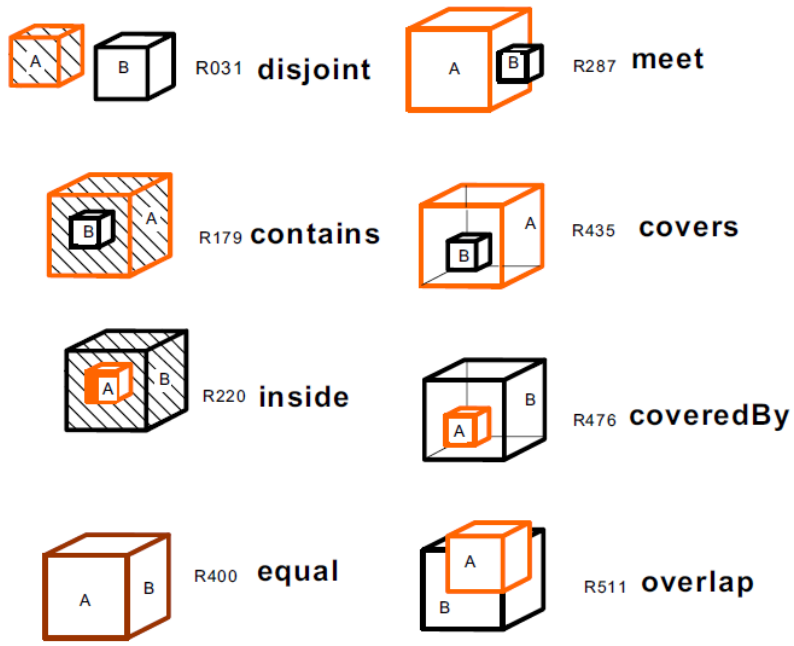
A overlaps B



	B I	B B	B E
A I	T	*	T
A B	*	*	*
A E	T	*	*

9IM+ (in 3D)

1. Natural language descriptions become complicated;
2. Combinatorially more complex



➲ Zlatanova, S., Abdul Rahman, A., Shi W., 2004. In Computers & Geosciences, Vol. 30, pp. 419-428

Overview

1. Objects in space
2. Storing topological relationships
3. Evaluating topological relationships
4. **Querying topological relationships using SQL**

Evaluating topological relationships in SQL

- Topological relationships between two (sets of) objects can be **tested** using SQL predicates
- Any **number** of spatial objects based on a given relationship can be done; methods apply in SQLs in the **WHERE / ON** clause;
- Tests using natural language convenience methods (return True/False):

```
ST_Contains(geom1, geom2)
ST_Crosses(geom1, geom2)
ST_Disjoint(geom1, geom2)
ST_Equals(geom1, geom2)
ST_Intersects(geom1, geom2)
ST_Overlaps(geom1, geom2)
ST_Touches(geom1, geom2)
ST_Within(geom1, geom2)
```

- Tests using the intersection matrix (linearized, row first, * for any):

```
ST_Relate (geom1, geom2, patternmatrix)
```

For `patternmatrix` use e.g., `'T*F**FFF*'`

	B I	B B	B E
A I	T	*	F
A B	*	*	F
A E	F	F	*

Exercise 3

What do we check here? Which convenience method could/should we have used?

```
SELECT a.comnr, a.comname, b.canr, b.caname
FROM   community a JOIN canton b
ON     ST_Relate(a.geom, b.geom, 'T*F**FFF*');

-- variants: checking for TRUE
ST_Relate(s.shape, n.shape, 'T*F**FFF*') = TRUE;
ST_Relate(s.shape, n.shape, 'T*F**FFF*') = 't';

-- variants: checking for FALSE
ST_Relate(s.shape, n.shape, 'T*F**FFF*') = FALSE;
ST_Relate(s.shape, n.shape, 'T*F**FFF*') = 'f';
```

Evaluating topological relationships in SQL (cont.)

- Integration in the SQL SELECT ... **JOIN ... ON / WHERE** clause
- Use the natural language named functions; they use the spatial index;
- Beware, always check the definition of the topological methods for different DBs; naming may also differ from Egenhofer;

```
SELECT a.comnr, a.comname, b.canr, b.caname
FROM   community a JOIN canton b
ON     ST_Intersects(a.geom, b.geom) -- spatial join!
WHERE  b.caname='Basel-Stadt';


SELECT a.comnr, a.comname,
       ST_AsText(a.shape) AS wktxt
FROM   community a
WHERE  ST_Touches(ST_GeomFromText(
       'POINT(2701100 1261100)', 2056),a.geom);
```

Other spatial relationships

- We can evaluate also other, spatial and thematic relationships using SQL
  - Direction
  - Length
  - Area
  - Perimeter
  - Attribute…
- These spatial relationship functions can be combined in a query; they are applicable mostly in the **SELECT** part of the SFW block;
- Why preferably **not** in the **WHERE** block?
- Refer to the PostGIS documentation on "Measurements"
  https://postgis.net/docs/reference.html#Measurement_Functions

Spatial functions for transforming geometries

- Derivating one ore more geometries and 'creating' new ones; the spatial DBs gives us the most important tools
  - Buffering
  - ConvexHull
  - Difference of geometries
  - Intersection of geometries (this is not the ST_Intersects! but ST_Intersection)
  - Symmetric difference
  - Union of geometries
  - Aggregates
- They apply in the **SELECT** part of the SFW block
- Refer to the PostGIS documentation on "Geometry Processing"
  https://postgis.net/docs/reference.html#Geometry_Processing

Summary

- We have discussed spatial data models;
  - Spaghetti (without topological storage);
  - Network (for linear structures);
  - Topological (storing the topology explicitly);
- We have discussed what explicitly stored topology is for (efficiency of computation and storage, consistency enforcement);
- We have discussed how topological relationships are evaluated in principle;
- We have shown how this is implemented in SQL; more details in next lecture;