Spatial Database Systems

Tutorial Notes

Ralf Hartmut Güting

Fernuniversität Hagen

Praktische Informatik IV

D-58084 Hagen

Germany

gueting@fernuni-hagen.de

1 What is a Spatial Database System?

Requirement: Manage data related to some *space*.

Spaces:

2D (or"2.5D") • geographic space (surface of the earth, at large or small scales)

→ GIS, LIS, urban planning, ...

3D • the universe

→ astronomy

2D • a VLSI design

3D • a model of the brain (or someone's brain)

→ medicine

3D • a molecule structure

→ biological research

Characteristic for the supporting technology: capability of managing *large collections of relatively simple geometric objects*

Terms:

pictorial database system

image

geometric

geographic

spatial

A database may contain

collections of *objects* in some space	raster images of some space

clear identity, location, extent

spatial database system	image database system

analysis, feature extraction

My personal definition of a spatial DBMS:

(1) A spatial database system is a database system

(2) It offers *spatial data types* in its data model and query language

(3) It supports spatial data types in its implementation, providing at least *spatial indexing* and efficient algorithms for *spatial join*.

Focus of this tutorial: describe fundamental problems and known solutions in a coherent manner.

2 Modeling

3 Querying

4 Tools for Implementation: Data Structures and Algorithms

5 System Architecture

Tutorial based on article:

R.H. Güting, An Introduction to Spatial Database Systems. *VLDB Journal 3 (4), 1994*, pp. 357-399.

but revised and extended recently

Additional references there.

2 Modeling

2.1 What needs to be represented?

2.2 Discrete Geometric Bases

2.3 Spatial Data Types / Algebras

2.4 Spatial Relationships

2.5 Integrating Geometry into the DBMS Data Model

2.1 What needs to be represented?

Two views:

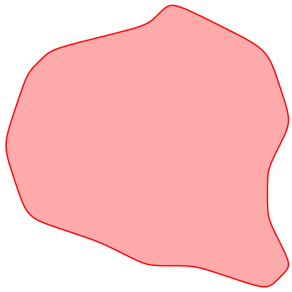
(i) objects in space

(ii) space itself

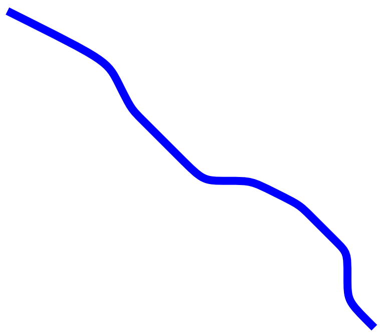
(i) Objects in space

city Berlin, …, population: 3 000 000,

city area:



river Rhine, …, route:



(ii) Space

Statement about every point in space (↔ raster images)

- land use maps (“thematic maps”)
- partitions into states, counties, municipalities, …

We consider:

1. modeling single objects
2. modeling spatially related collections of objects

1. Basic abstractions for modeling single objects:

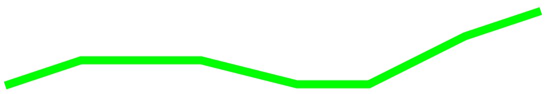
- *point*

city

geometric aspect of an object, for which only its *location* in space, but not the *extent*, is relevant

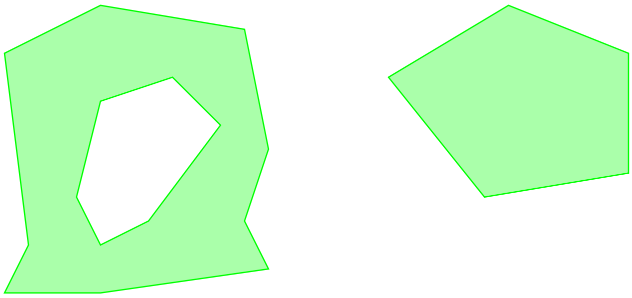
- *line* (polyline)

river
cable
highway



moving through space, connections in space

- *region*

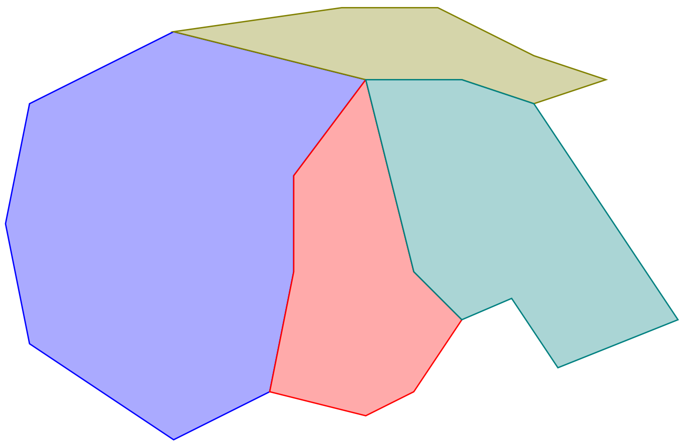


forest
lake
city

abstraction of an object with extent

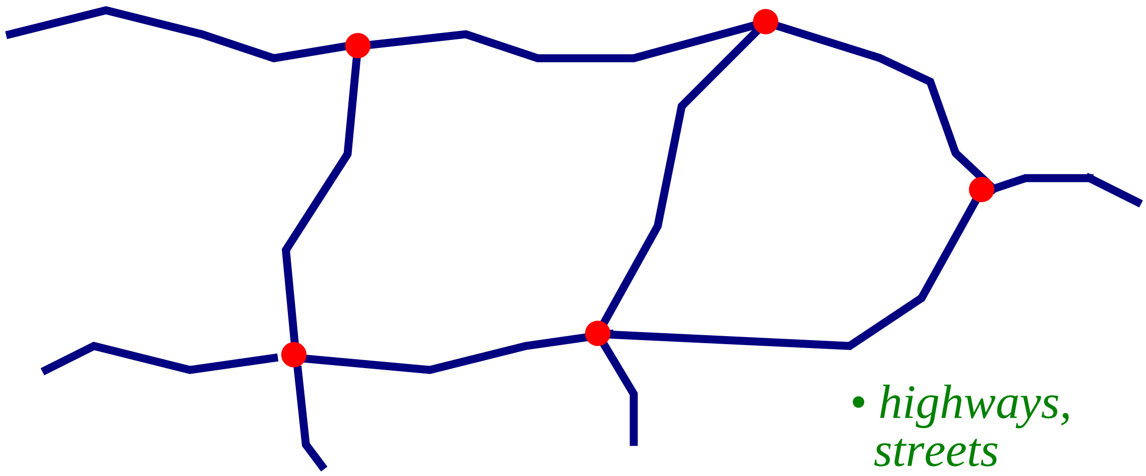
2. Basic abstractions for spatially related collections of objects

- *Partition*



- *land use*
- *districts*
- *land ownership*
- *“environments” of points* → *Voronoi diagram*

- Spatially embedded *network* (graph)



- *highways, streets*
- *railways, public transport*
- *rivers*
- *electricity, phone*

Others:

- nested partitions
- digital terrain models

2.2 Organizing the Underlying Space: Discrete Geometric Bases

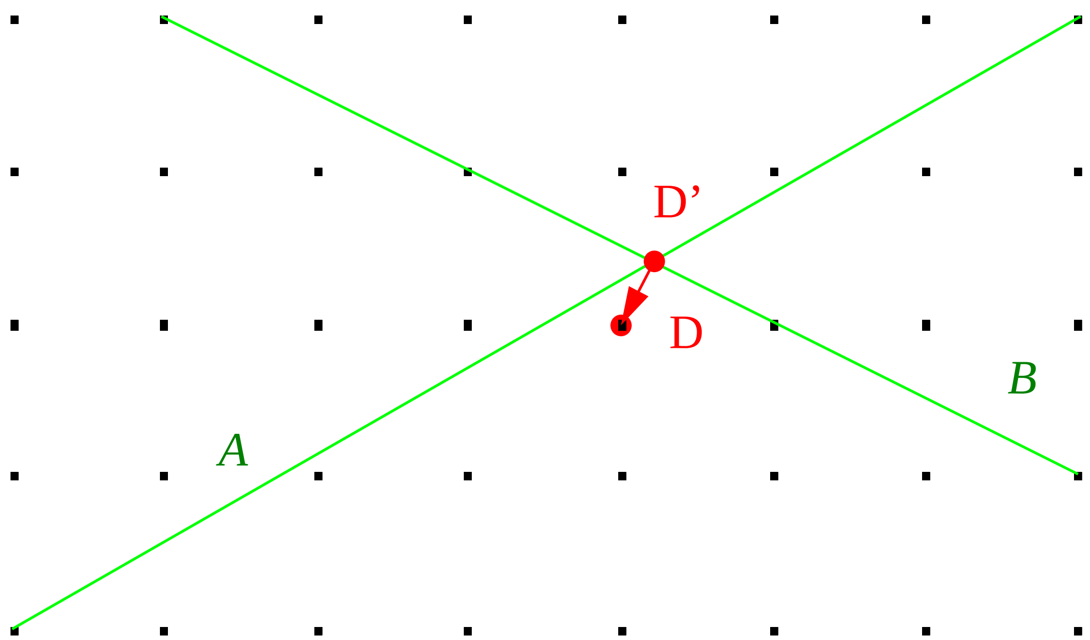
Is Euclidean geometry a suitable base for modeling?

Problem: space is continuous

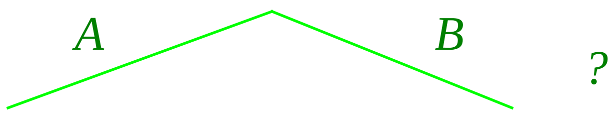
computer numbers are discrete

$p = (x, y) \in \mathbb{R}^2$

$p = (x, y) \in$ real $\times$ real



- Is D *on* A?
- Is D *properly contained* in the area below



Goal: Avoid computation of any new intersection points within geometric operations

Definition of geometric types and operations

geometric basis

Treatment of numeric problems upon updates of the geometric basis

Two approaches:

- *Simplicial complexes*

  Frank & Kuhn 86

  Egenhofer, Frank & Jackson 89

- *Realms*

  Güting & Schneider 93

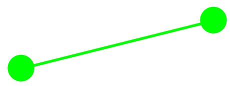
  Schneider 97

Simplicial Complexes

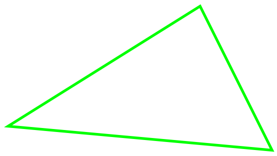
(from combinatorial topology)

d-simplex: minimal object of dimension *d*

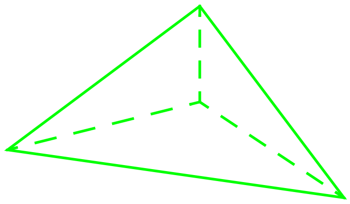
0-simplex



1-simplex



2-simplex

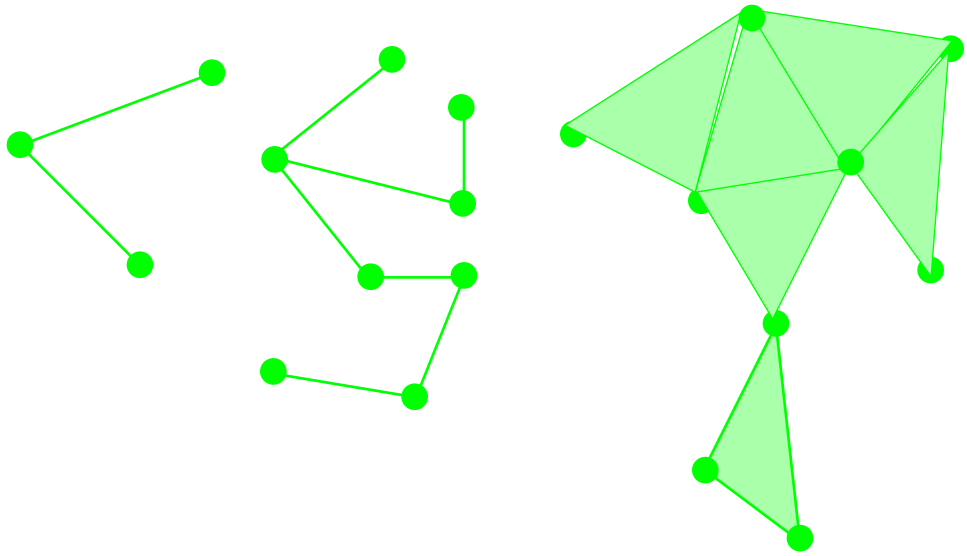


3-simplex

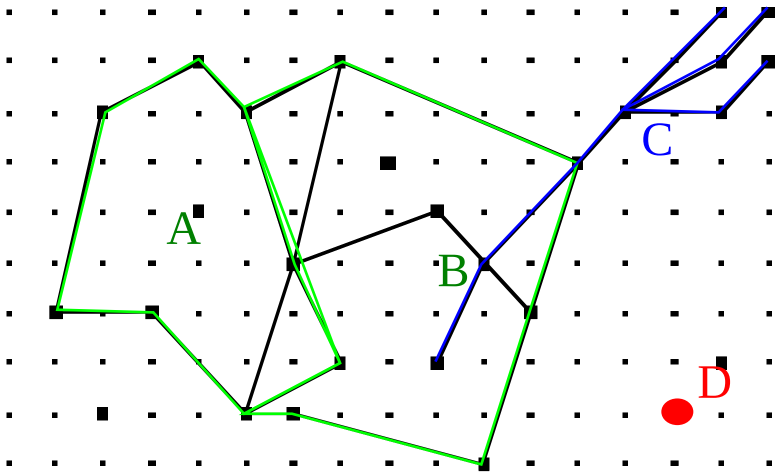
etc.

d-simplex consists of *d*+1 simplices of dimension *d*-1.

Components of a simplex are called *faces*. *Simplicial complex*: finite set of simplices such that the intersection of any two simplices is a face.

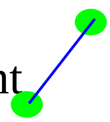


Realms

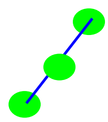


Realm (intuitive notion): Complete description of the geometry (all points and lines) of an application.

Realm (formally): A finite set of points and line segments defined over a grid such that:

(i) each point or end point of a segment is a grid point 

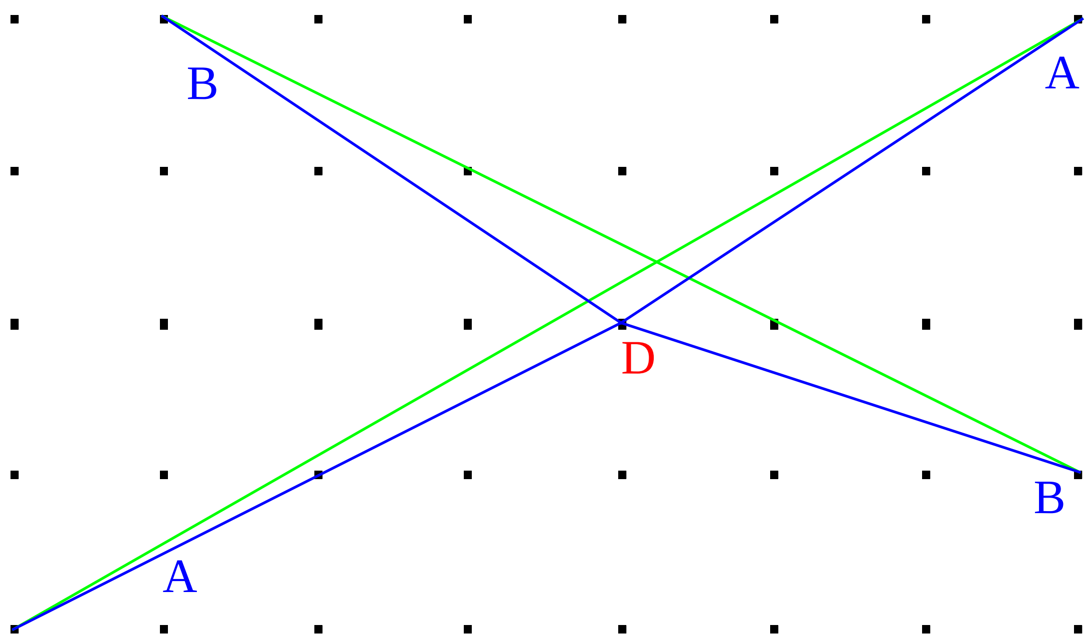
(ii) each end point of a segment is also a point of the realm

(iii) no realm point lies *within* a segment 

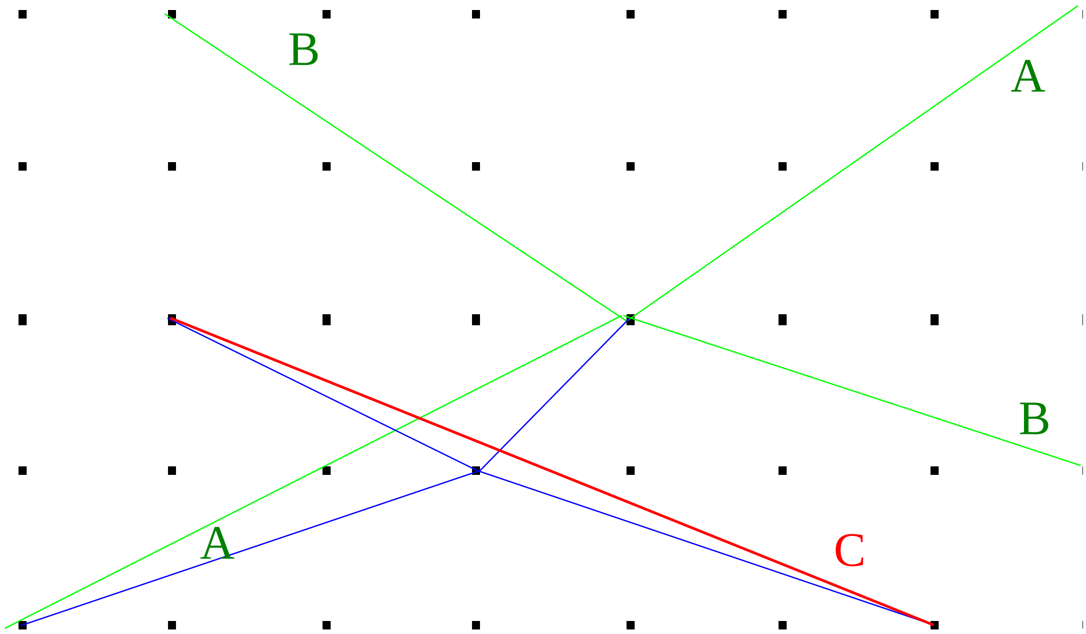
(iv) any two distinct segments do neither intersect nor overlap

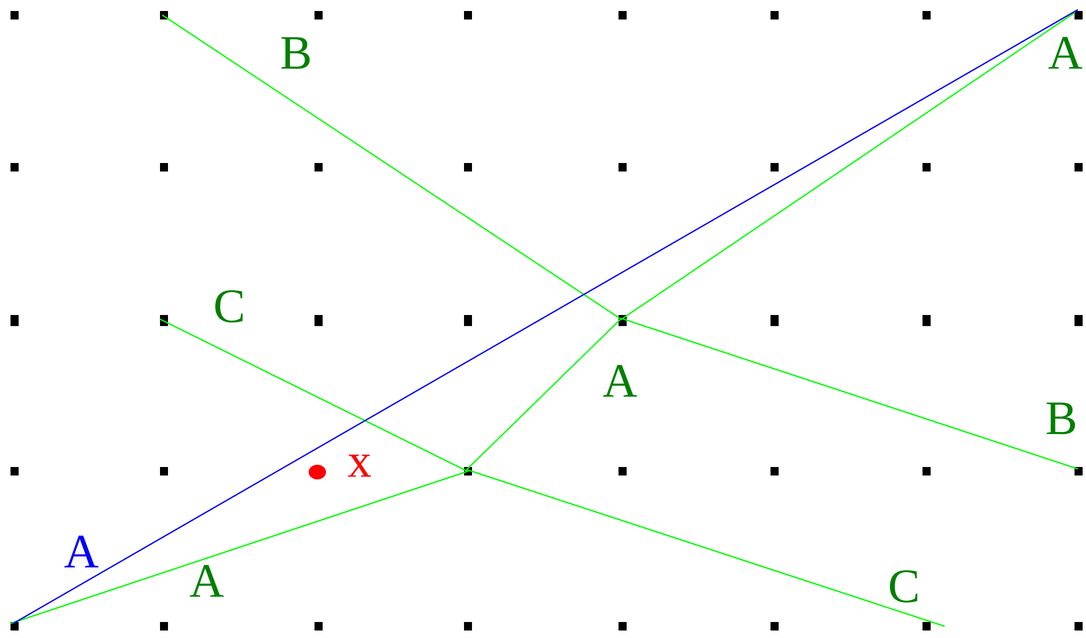
Numeric problems are treated *below* the realm layer:

Application data are sets of points and *intersecting* line segments. Need to insert a segment intersecting other segments. Basic idea: slightly distort both segments.



Solution?

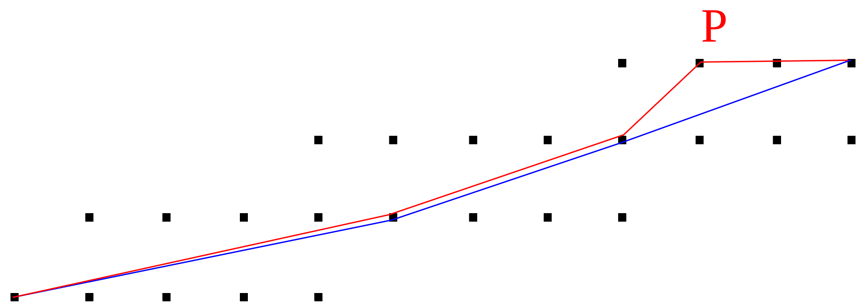




Segments can move!

Point *x* is now on the wrong side of *A*!

Concept of Greene & Yao (1986): *Redraw* segments within their *envelope*.



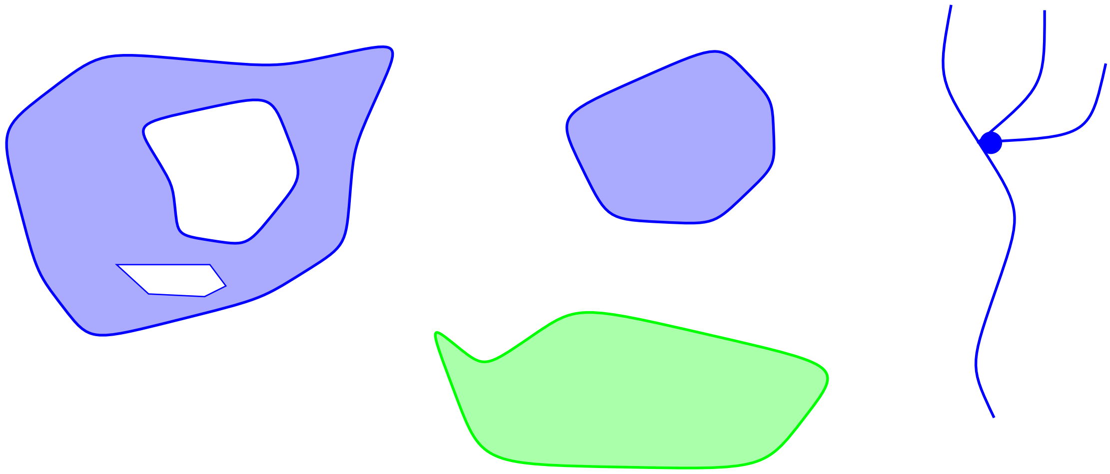
Segments are “captured” within their envelope; can never cross a grid point.

2.3 Spatial Data Types / Algebras

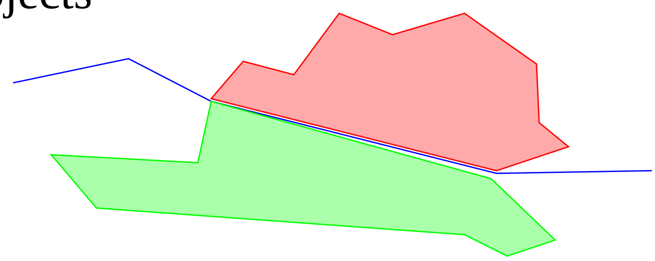
(spatial data type = SDT)

Requirements:

- “general structure” of values ↔ closed under set operations on the underlying point sets



- precise formal definition of SDT values and functions
- definition in terms of finite precision arithmetics
- support for geometric consistency of spatially related objects



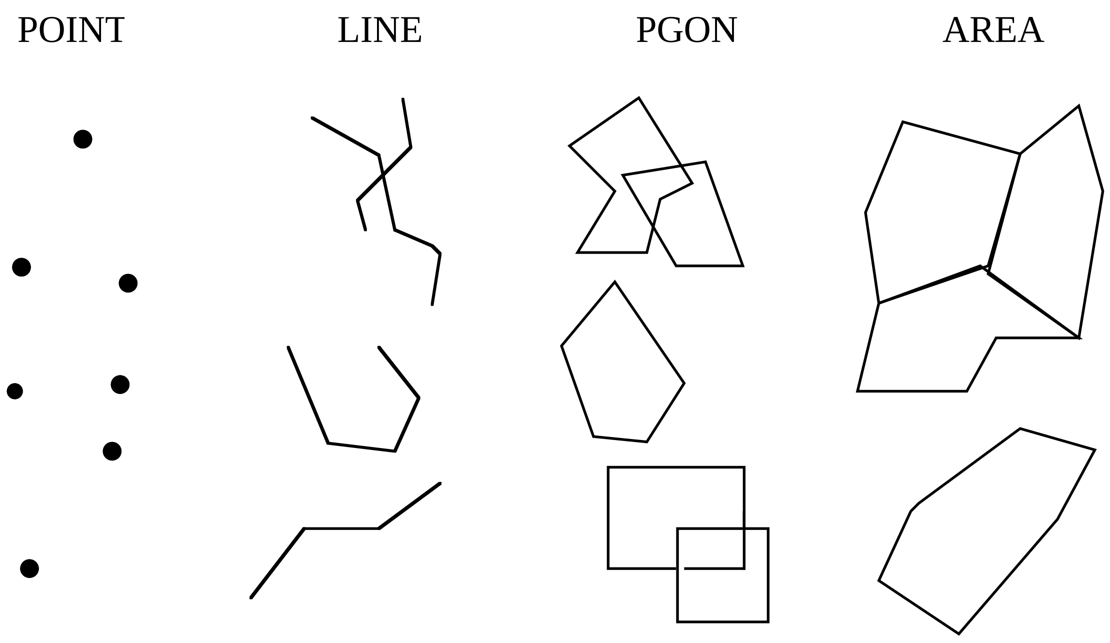
- independent of particular DBMS data model, but cooperating with any

Geo-Relational Algebra (Güting 1988)

Relational algebra viewed as a *many-sorted algebra* (relations + atomic data types)

Sorts:

REL	NUM	POINT	
	STR	LINE	
	BOOL	REG	PGON
			AREA

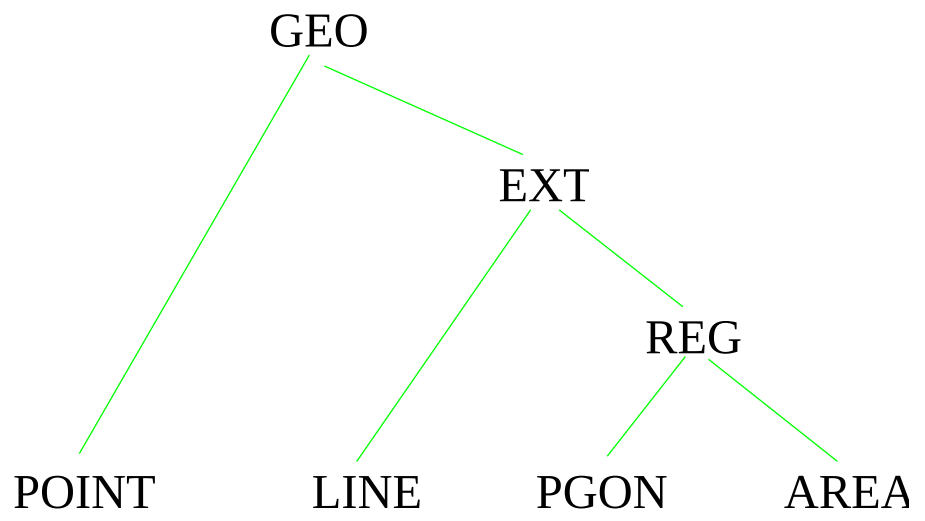


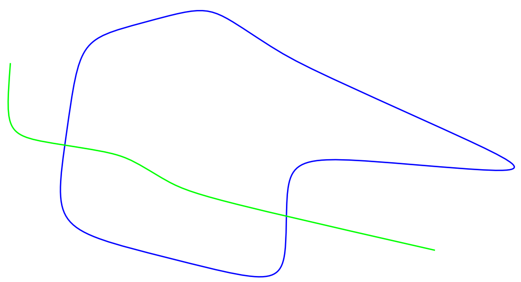
cities	cname	center	cpop
	STR	POINT	NUM

Operations:

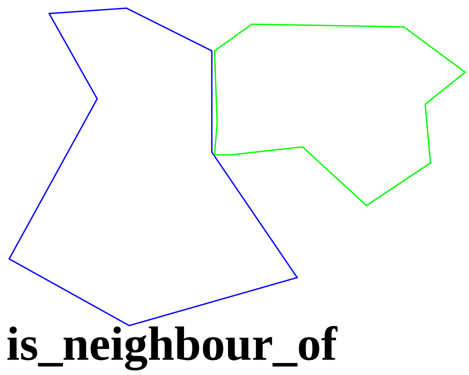
Geometric predicates

POINT × POINT	→ BOOL	=, ≠
LINE × LINE	→ BOOL	
REG × REG	→ BOOL	
GEO × REG	→ BOOL	**inside**
EXT × EXT	→ BOOL	**intersects**
AREA × AREA	→ BOOL	**is_neighbour_of**





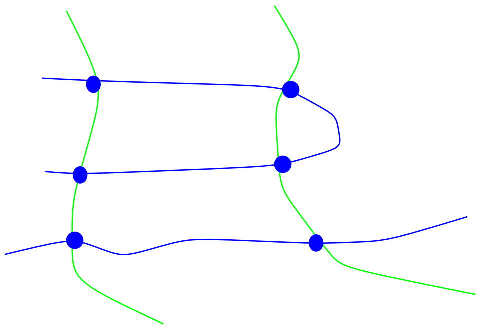
intersects



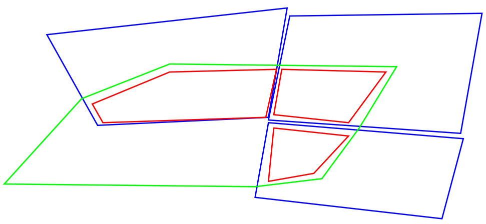
is_neighbour_of

Geometric relation operations

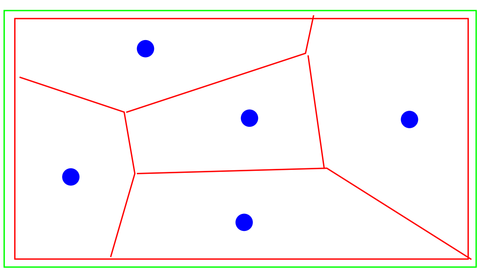
LINE* $\times$ LINE*	$\rightarrow$ POINT*	**intersection**
LINE* $\times$ REG*	$\rightarrow$ LINE*	
PGON* $\times$ REG*	$\rightarrow$ PGON*	
AREA* $\times$ AREA*	$\rightarrow$ AREA*	**overlay**
EXT*	$\rightarrow$ POINT*	**vertices**
POINT* $\times$ REG	$\rightarrow$ AREA*	**voronoi**
POINT* $\times$ POINT	$\rightarrow$ REL	**closest**



intersection



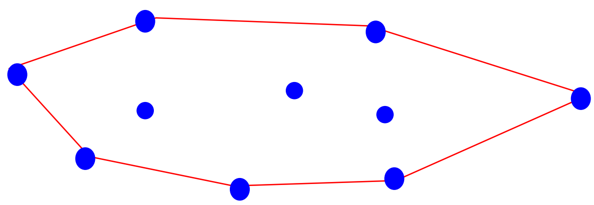
overlay



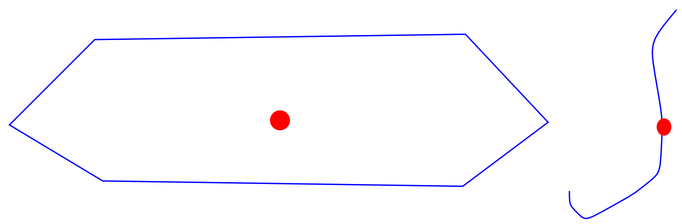
voronoi

Operations returning atomic geometric objects

POINT*	$\rightarrow$ PGON	**convex_hull**
POINT*	$\rightarrow$ POINT	**center**
EXT	$\rightarrow$ POINT	



convex_hull



center

Operations returning numbers

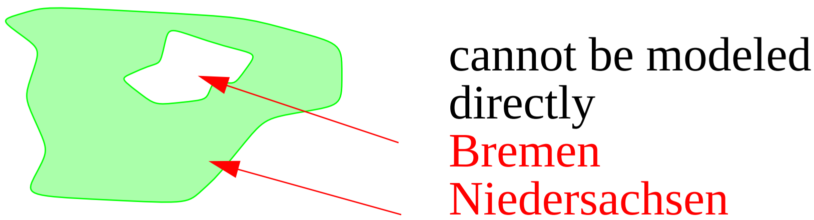
POINT $\times$ POINT	$\rightarrow$ NUM	**dist**
GEO $\times$ GEO	$\rightarrow$ NUM	**mindist, maxdist**
POINT*	$\rightarrow$ NUM	**diameter**
LINE	$\rightarrow$ NUM	**length**
REG	$\rightarrow$ NUM	**perimeter, area**

Compare to requirements:

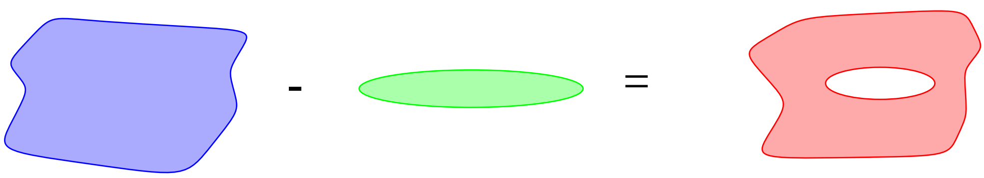
- general structure, closed under point set ops –
- formal definition +
- finite precision arithmetics –
- support for geometric consistency (–)
- data model independent –

Problems:

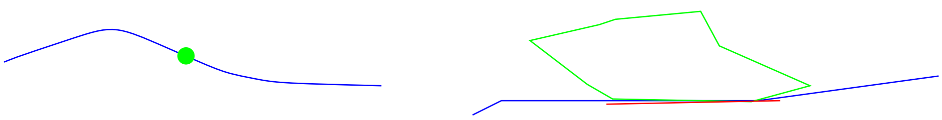
- only simple polygons



- forming the intersection of two geometric objects must be embedded in a relation operation
- no difference of regions



- no numerically critical operations included



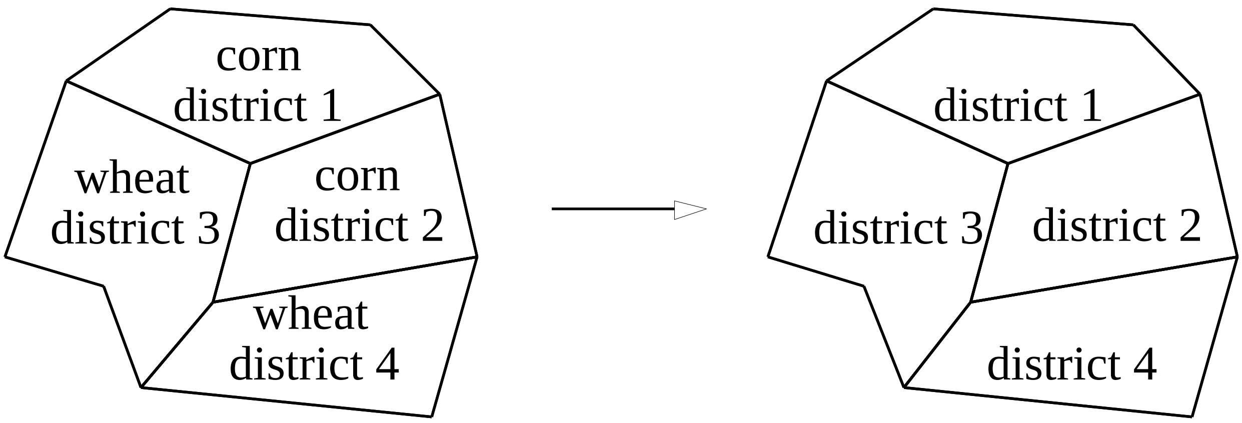
But also advantages:

- conceptually simple geometric objects
- simple formal definitions
- implementation: simple data structures + algorithms

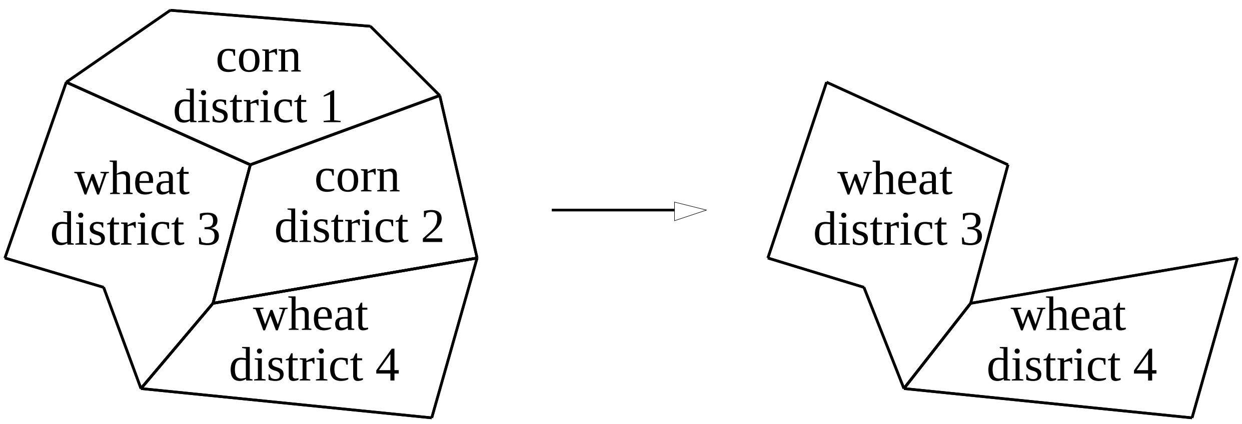
Scholl & Voisard 89

Algebra for manipulating “maps” = partitions

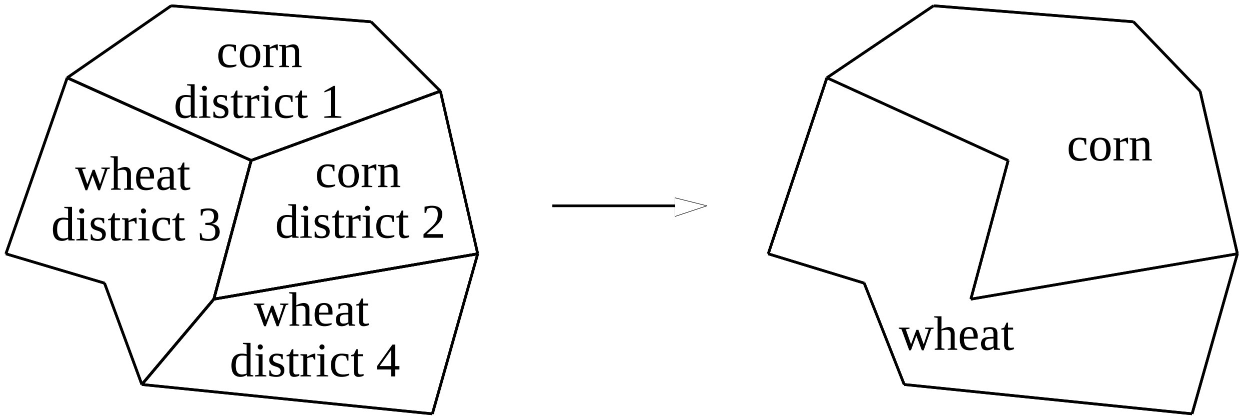
Goal: formal modeling of map operations, including the following:



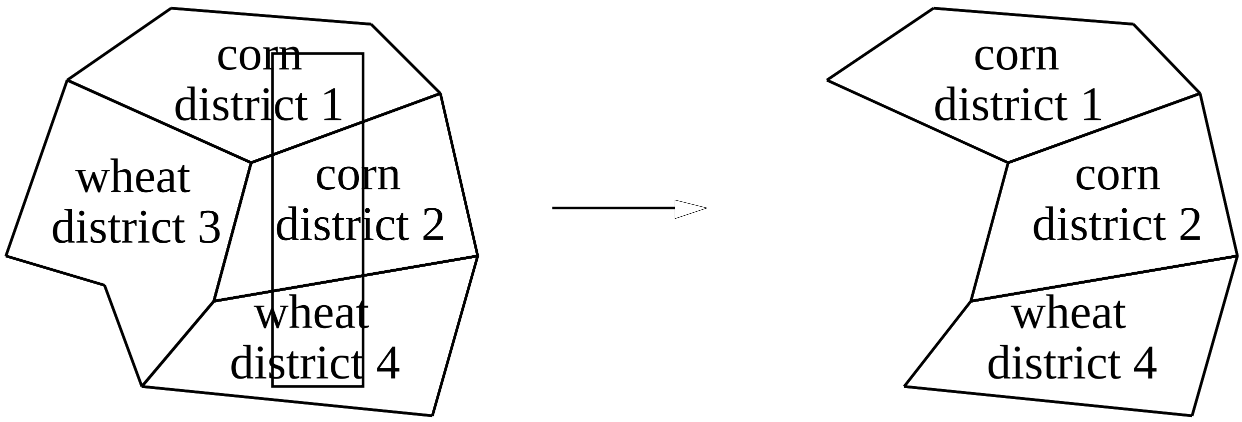
Projection



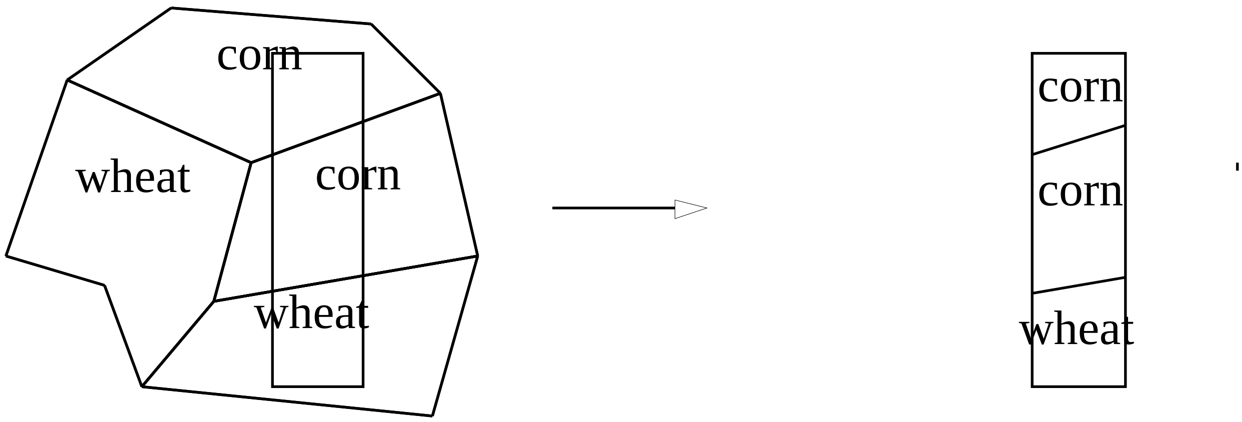
Selection



Projection with merging ("*fusion*")



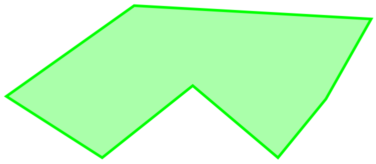
"*Windowing*"



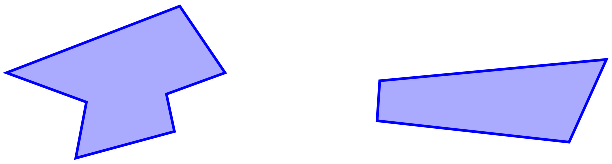
"*Clipping*"

Data type for regions:

Elementary region γ :



or



$\rightarrow$ region of type γ

region of type $\{\gamma\}$

Embedded into a model for complex objects. A map is a set of tuples with a region attribute.

{[height: integer, land_use: string, area: γ]

or area: $\{\gamma\}$]

Algebra for complex objects used (in particular, operator for *nesting*)

{[A: string, R: γ]} $\xrightarrow{\text{Nest}_A}$ {[A: string, S: {R: γ }]}

+ some primitives on regions

+ some geometric set operations

Several other designs:

Gargano, Nardelli & Talamo 91

Larue, Pastre & Viemont 93

Svensson & Huang 91

Tomlin 90

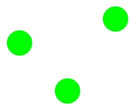
Chan & Zhu 96

ROSE Algebra (Güting & Schneider 95)

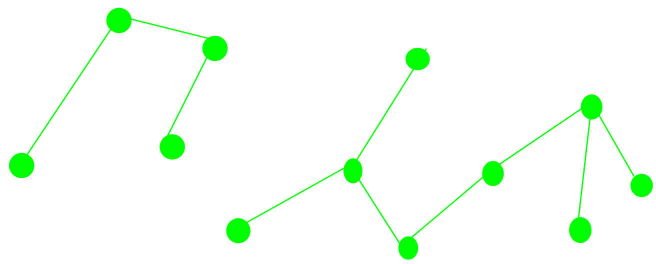
ROSE = RObust Spatial Extension

A system of realm-based spatial data types → objects composed from realm elements

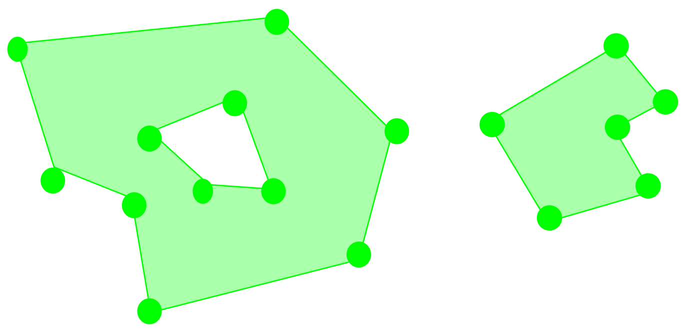
Types *points*, *lines*, *regions*



a *points* value

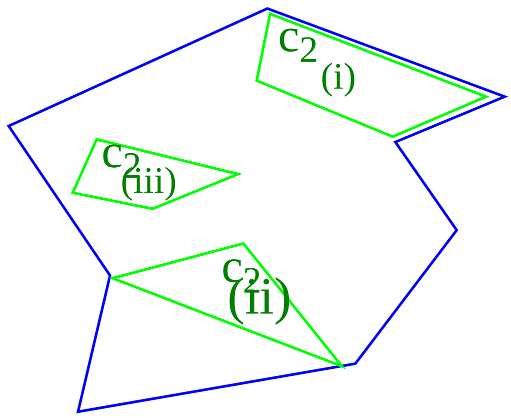


a *lines* value



a *regions* value

More complex structure of objects must be treated in definitions (and implementation).



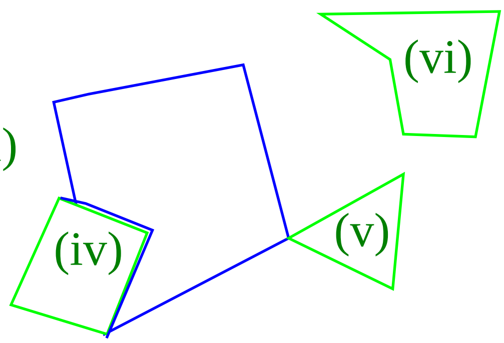
c_2 is

- (area-) inside (i, ii, iii)
- edge-inside (ii, iii)
- vertex-inside (iii)

c_1.

c_1 and c_2 are

- area-disjoint (iv, v, vi)
- edge-disjoint (v, vi)
- (vertex-) disjoint (vi)



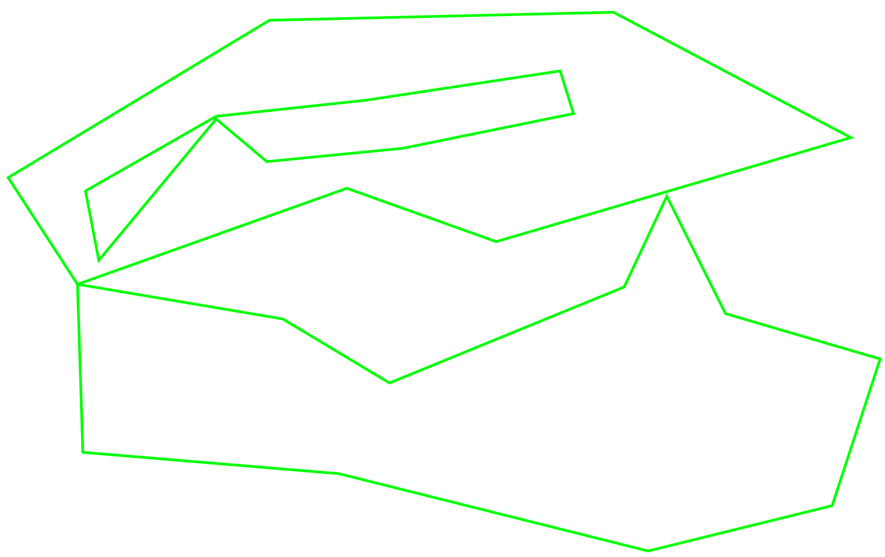
An *R-face* f is a pair (c, H), with c an R-cycle and $H = \{h_1, \ldots, h_m\}$ a set of R-cycles, such that:

(i) $\forall\, i \in \{1, \ldots, m\}$: h_i *edge-inside* c

(ii) $\forall\, i, j \in \{1, \ldots, m\}, i \neq j$: h_i and h_j are *edge-disjoint*

(iii) "no other cycle" can be formed from the segments of f

Last condition enforces unique representations.



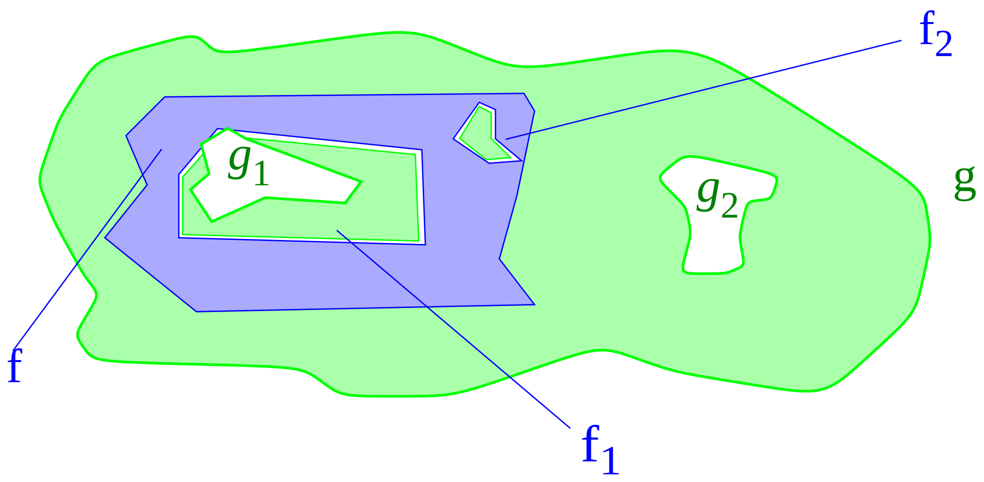
Let $f = (f_0, F)$ and $g = (g_0, G)$ be two R-faces.

f *area-inside* g

$:\Leftrightarrow$ f_0 area-inside g_0

$\wedge \quad \forall\, g \in G{:}\, g$ area-disjoint f_0

$\vee\, \exists\, f \in F{:}\ g$ area-inside f



A *regions* value F is a set of edge-disjoint R-faces.

Let F, G be two *regions* values.

F *area-inside* G

$:\Leftrightarrow\ \forall\, f \in F\ \exists\, g \in G{:}\ f$ area-inside g

ROSE algebra offers precisely defined operations for manipulating such values:

(GEO = {<u>*points*</u>, <u>*lines*</u>, <u>*regions*</u>}, EXT = {<u>*lines*</u>, <u>*regions*</u>})

$\forall$ *geo* in GEO. $\forall$ *ext* in EXT.

geo × <u>*regions*</u>	→ <u>*bool*</u>	**inside**
<u>*regions*</u> × <u>*regions*</u>	→ <u>*bool*</u>	**area-disjoint**, **edge-disjoint**, **edge-inside**, **vertex-inside**

Contains also (otherwise) numerically critical operations:

<u>*points*</u> × *ext*	→ <u>*bool*</u>	**on_border_of**
ext × <u>*regions*</u>	→ <u>*bool*</u>	**border_in_common**

Any intersection, union, difference can be formed due to general structure of values:

<u>*points*</u> × <u>*points*</u>	→ <u>*points*</u>	**intersection**
<u>*lines*</u> × <u>*lines*</u>	→ <u>*points*</u>	**intersection**
<u>*regions*</u> × <u>*regions*</u>	→ <u>*regions*</u>	**intersection**
<u>*regions*</u> × <u>*lines*</u>	→ <u>*lines*</u>	**intersection**
geo × *geo*	→ *geo*	**plus**, **minus**

(no embedding into e.g. relation operations needed.)

Spatial operations for manipulating collections of database objects defined via general "*object model interface*":

$\forall$ *obj* in OBJ. $\forall$ *geo*, geo_1, geo_2 in GEO.

$\underline{set}(obj) \times (obj \rightarrow geo) \rightarrow geo$ **sum**

$\underline{set}(obj) \times (obj \rightarrow geo_1) \times geo_2 \rightarrow \underline{set}(obj)$ **closest**

(also **overlay**, **fusion**)

Compare to requirements:

- general structure, closed under point set ops +
- formal definition +
- finite precision arithmetics +
- support for geometric consistency +
- data model independent +

But also disadvantages:

- No operations possible that create new geometries (leave the realm closure), e.g. **voronoi**, **center**, **convex_hull**
- Integrating realms into database systems somewhat difficult. Updates of the realm must be propagated to realm-based attribute values in objects.

General issues concerning spatial data types / algebras:

- extensibility
- completeness
- one or more types ($\rightarrow$ type “geometry”)?
- operations on sets of DB objects

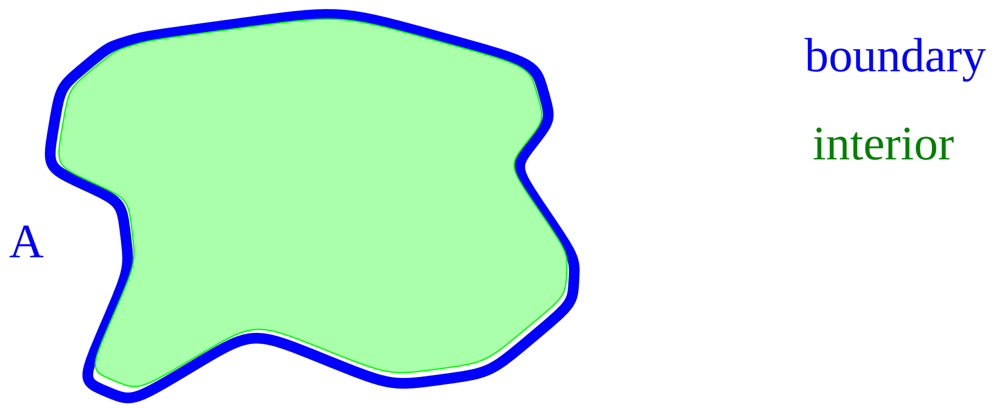
2.4 Spatial Relationships

Most important operations of spatial algebras (predicates). E.g. find all objects in a given relationship to a query object.

- topological: inside, intersects, adjacent ...
  (invariant under translation, rotation, scaling)
- direction: above, below, north_of, ...
- metric: distance < 100

Topological relationships studied in some depth. Any completeness criteria ?

Yes! Egenhofer 89 and subsequent work. Originally for simple regions only (no holes, connected)



$\partial A_1 \cap \partial A_2$	$\partial A_1 \cap A_2°$	$A_1° \cap \partial A_2$	$A_1° \cap A_2°$	relationship name
$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	A_1 disjoint A_2
$\varnothing$	$\varnothing$	$\varnothing$	$\neq\varnothing$	
$\varnothing$	$\varnothing$	$\neq\varnothing$	$\varnothing$	
$\varnothing$	$\varnothing$	$\neq\varnothing$	$\neq\varnothing$	A_2 in A_1
$\varnothing$	$\neq\varnothing$	$\varnothing$	$\varnothing$	
$\varnothing$	$\neq\varnothing$	$\varnothing$	$\neq\varnothing$	A_1 in A_2
$\varnothing$	$\neq\varnothing$	$\neq\varnothing$	$\varnothing$	
$\varnothing$	$\neq\varnothing$	$\neq\varnothing$	$\neq\varnothing$	
$\neq\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	A_1 touch A_2
$\neq\varnothing$	$\varnothing$	$\varnothing$	$\neq\varnothing$	A_1 equal A_2
$\neq\varnothing$	$\varnothing$	$\neq\varnothing$	$\varnothing$	
$\neq\varnothing$	$\varnothing$	$\neq\varnothing$	$\neq\varnothing$	A_1 cover A_2
$\neq\varnothing$	$\neq\varnothing$	$\varnothing$	$\varnothing$	
$\neq\varnothing$	$\neq\varnothing$	$\varnothing$	$\neq\varnothing$	A_2 cover A_1
$\neq\varnothing$	$\neq\varnothing$	$\neq\varnothing$	$\varnothing$	
$\neq\varnothing$	$\neq\varnothing$	$\neq\varnothing$	$\neq\varnothing$	A_1 overlap A_2

4-intersection method (4 intersection sets for two objects)

$4^2 = 16$ combinations

Extensions:

- include point and line features (Egenhofer & Herring 92, de Hoop & van Oosterom 92)
- consider also intersection of complements A^{-1} (Egenhofer 91b).

Clementini et. al. 93: consider dimension of the intersection (empty, 0D, 1D, or 2D in 2-space) $\rightarrow 4^4 = 256$ combinations, 52 are valid (*dimension extended method*).

Too many to be remembered !

Alternative: 5 basic relationships

- touch
- in
- cross
- overlap
- disjoint

defined in terms of dimension ext. method, e.g.

$<\lambda_1 \text{ touch } \lambda_2> \; :\Leftrightarrow$

$$(\lambda_1^\circ \cap \lambda_2^\circ = \emptyset) \wedge (\lambda_1 \cap \lambda_2 \neq \emptyset)$$

+ 3 operators b, f, t to get boundaries.

One can prove:

- 5 relationships mutually exclusive
- 5 relationships + 3 boundary operators can distinguish all 52 configurations of the dimension extended method.

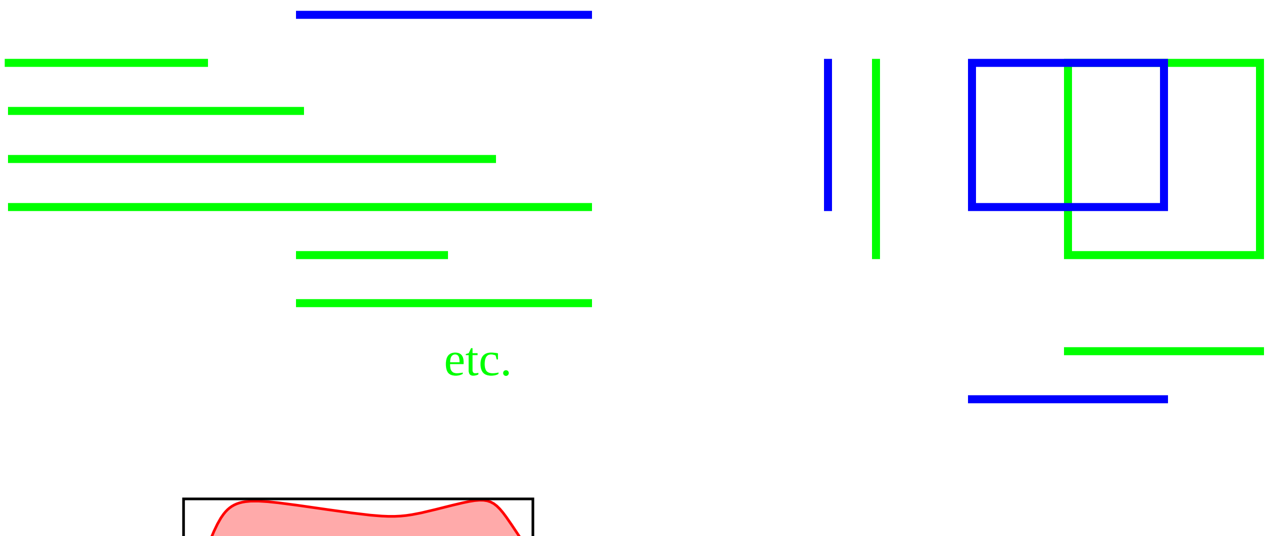
Further extensions:

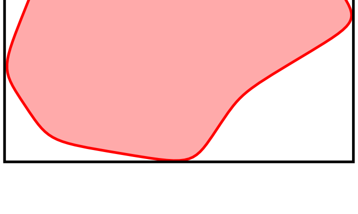
- consider regions with holes (Egenhofer, Clementini & Di Felice 94)
- consider composite regions (Clementini, Di Felice & Califano 95) *several disjoint components*

Papadias et al. 95: What do relations between bounding boxes tell us about the topological relations between their enclosed regions?

13 possible relations on intervals in 1D space (Allen 1983)

$\rightarrow$ 169 relations on rectangles in 2D space





2.5 Integrating Geometry into the DBMS Data Model

Basic idea: represent "spatial objects" by *objects* (of the DBMS data model) *with at least one SDT attribute*.

DBMS model must be open for user-defined types (→ abstract data type support, → extensibility).

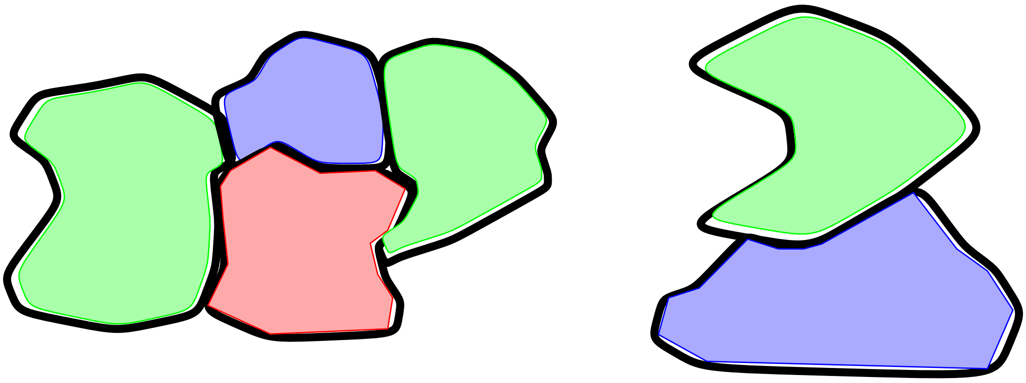
E.g. for relational model:

```
relation states (sname: STRING; area: REGION;
    spop: INTEGER)
relation cities (cname: STRING; center: POINT;
    ext: REGION; cpop: INTEGER)
relation rivers (rname: STRING; route: LINE)
```

(representation of (sets of) single objects)

Representation of spatially related collections of objects:

Partitions



Set of objects with REGION attribute ?

Loses some information:

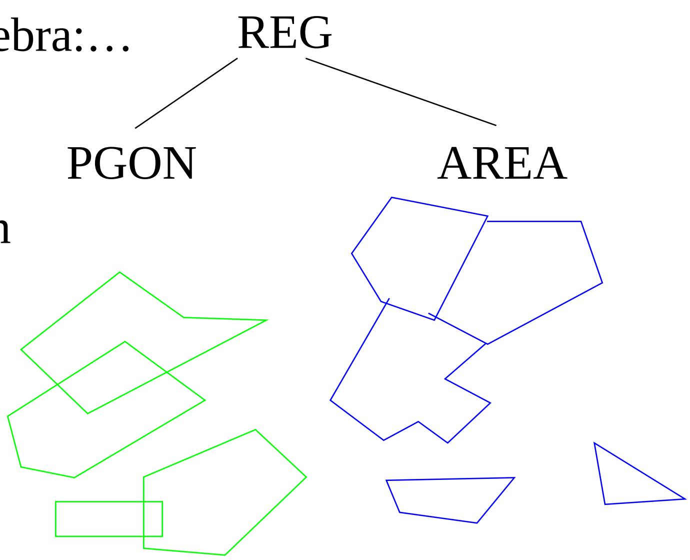
- regions are disjoint
- adjacency relationship important (e.g. for building an "adjacency join index")

In the geo-relational algebra:...

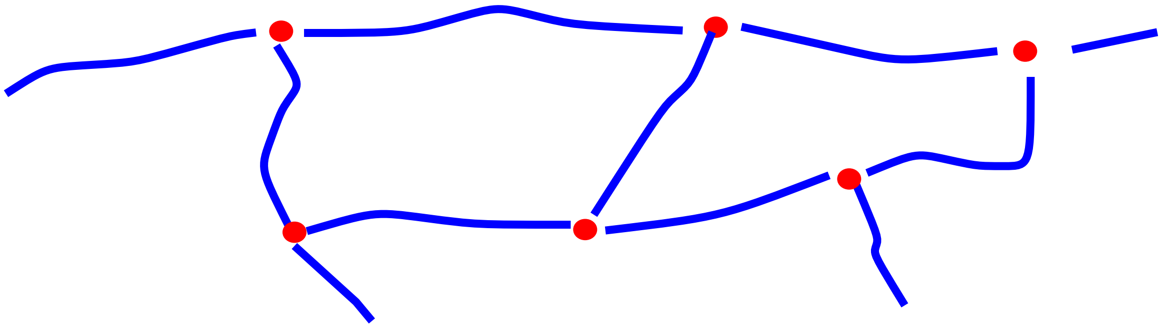
but should rather be

an integrity constraint on

a relation



Networks



Not much research on *spatially embedded networks* (but a lot on graphs in databases in general). Usually graphs represented by given facilities of a data model. Disadvantage: Graph not visible to user; not very well supported by system. GraphDB (Güting 94) has explicit graphs integrated into an OO model.

```
class vertex = pos: POINT;
vertex class junction = name: STRING;
vertex class exit = nr: INTEGER;
link class section = route: LINE, no_lanes: INTEGER,
    top_speed: INTEGER from vertex to vertex;
path class highway = name: STRING as section+;
```

3 Querying

Connect operations of a spatial algebra to the facilities of a DBMS query language.

Issues:

- fundamental operations (algebra) for manipulating sets of database objects
- graphical input and output
- extending query languages

3.1 Fundamental Operations (Algebra)

- spatial selection
- spatial join
- spatial function application
- other set operations

"Spatial selection" ≡ selection based on a spatial predicate.

"Find all cities in Bavaria."

```
cities select[center inside Bavaria]
```

a REGION value

"Find all rivers intersecting a query window."

```
rivers select[route intersects Window]
```

a REGION value

"Find all big cities no more than 100 km from Hagen."

```
cities select[dist(center,Hagen) ≤ 100 and
    pop > 500 000]
```

a POINT value

"Spatial join" ≡ join based on a predicate comparing spatial attribute values.

"Combine cities with their states."

```
cities states join[center inside area]
```

"For each river, find all cities within less than 50 kms."

```
cities rivers join[dist(center, route) < 50]
```

Spatial Function Application

How can we use operations of a spatial algebra computing new SDT values? E.g.

$\underline{\textit{regions}} \times \underline{\textit{lines}} \rightarrow \underline{\textit{lines}}$ **intersection**

- In selection conditions.
- Object algebra operators allow one to apply functions to each member of a set:
  - filter operator (FAD)
  - replace
  - map
  - extend / λ

“For each river going through Bavaria, return the name, the part inside Bavaria and the length of that part.”

```
rivers select[route intersects Bavaria]
  extend[intersection(route,Bavaria) {part}]
  extend[length(part) {plength}]
  project[rname, part, length]
```

Other Set Operations

Manipulate whole sets of spatial objects in a special way:

- operation is a conceptual unit
- separation between DBMS object set manipulation and spatial algebra SDT manipulation (often) not possible.

For example:

- overlay
- fusion
- voronoi

Section 2.3

Interfacing the spatial algebra with the DBMS more difficult.

3.2 Graphical Input and Output

Needed:

- Graphical presentation of SDT values in query results
- Entering SDT values (“constants” for queries)
- Overlay query results (build a taylored picture of the space)

Requirements for spatial querying (Egenhofer 94)

1. Spatial data types
2. Graphical display of query results
3. Graphical combination (overlay) of several query results
4. Display of context
5. A facility for checking the content of a display
6. Extended dialog
7. Varying graphical representations
8. Legend
9. Label placement
10. Scale selection
11. Subarea for queries

Graphical user interface may have three (sub-)windows:

- *text window* for textual representation of a collection of objects
- *graphics window* for graphical representation of a collection of objects
- text window for entering queries and display of system messages

  $\rightarrow$ text-graphics interaction

Query may consist of three parts (Egenhofer 94):

- describe set of objects to be retrieved,
- partition this set by *display queries* into subsets,
- describe for each subset how to render its spatial attributes.

$\rightarrow$ GPL (*graphical presentation language*) Egenhofer 91a

Tioga-2: Integrate the visual representation of a query with a visualization of the result. Query represented as a dataflow graph (boxes and arrows). Query and visualization constructed incrementally. (Aiken et al. 96)

3.3 Integrating Geometry into a Query Language

Three aspects:

- denoting SDT values / graphical input
- expressing the four classes of fundamental operations
- describing the presentation of results

Denoting SDT values. It is useful if the DBMS allows one to define named atomic data type values. For example:

```
DEFINE Bavaria
ELEMENT SELECT s.area
    FROM s in states
    WHERE s.sname = "Bavaria"
```

Allows one to decouple input via graphical input device and use in query. Other proposal: keyword in query leads to interaction for input:

```
SELECT sname FROM cities WHERE center inside PICK
```

Expressing the four classes of fundamental operations:

- selection
- join
- function application
- other set operations

no problem

also ok

do not fit with SFW-construct

```
SELECT *
FROM rivers
WHERE route intersects Window

SELECT cname, sname
FROM cities, states
WHERE center inside area

SELECT rname, intersection(route, Bavaria),
  length(intersection(route,Bavaria))
FROM rivers
WHERE route intersects Bavaria
```

Describing presentation of results:

Could be

- part of query language
- separate language
- defined by GUI manipulation

Presentation language needs some querying capabilities.

Other directions:

- deductive database approach (e.g. Abdelmoty, Williams & Paton 93)
- visual querying – draw a sketch of spatial configurations of interest in a query (e.g. Maingenaud & Portier 90, Meyer 92)
- virtual reality exploration – fast navigation through large topographic scenes (Pajarola et al. 98)
- query by spatial structure: find all *n*-tuples of objects fulfilling a set of specified relationships. Can be viewed as a generalization of spatial join. (Papadias, Mamoulis & Delis 98)

4 Tools for Spatial DBMS Implementation: Data Structures and Algorithms

Problem: Implementation of a spatial algebra in such a way that it can be integrated into a database system's query processing :

- representations for the types
- algorithms/procedures for the operations

Use of predicates in set-oriented query processing:

- spatial selection
- spatial join
- algorithms for set operations of a spatial algebra

4.1 **Representing SDT Values and Implementing Atomic SDT Operations**

4.2 **Spatial Indexing - Supporting Spatial Selection**

4.3 **Supporting Spatial Join**

4.1 Representing SDT Values and Implementing Atomic SDT Operations

Representation of an SDT value must be simultaneously compatible with two views:

DBMS view:

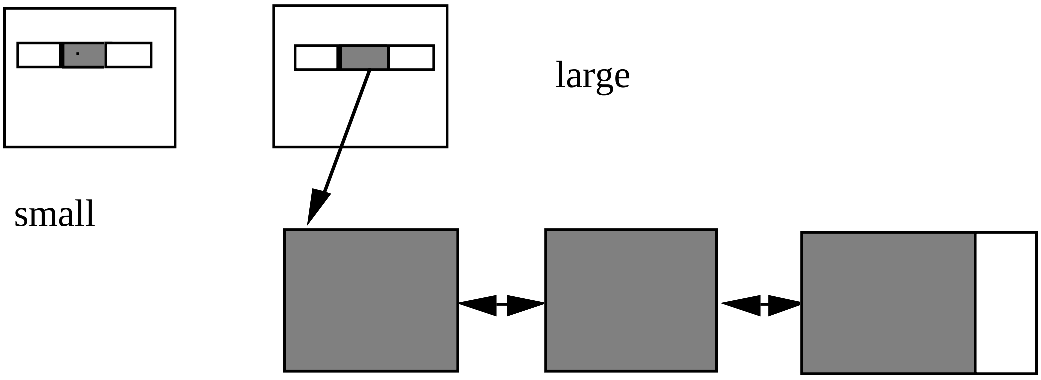
- like values of other types w.r.t. generic operations
- varying and possibly very large size
- resides permanently on disk in one page or a set of pages
- can be loaded efficiently into memory (value of a pointer variable there)
- offers type-specific implementations of generic operations

Spatial algebra implementation view:

- value of a programming language data type
- is some arbitrary, possibly complex data structure
- supports computational geometry algorithms
- not designed to support just one particular algorithm, but balanced to support many well enough

Support DBMS view:

- paged data structure compatible with DBMS support for long fields or large attribute values



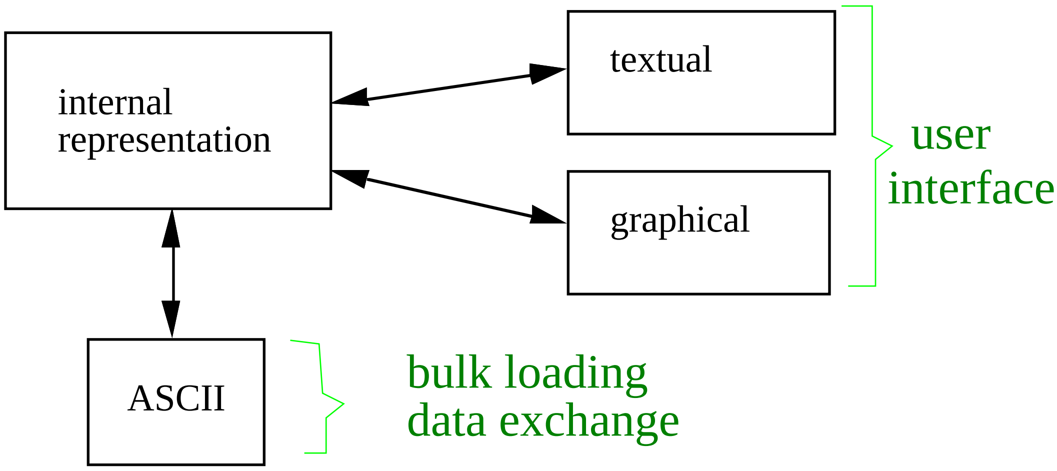
- info part and exact geometry part separately loadable

small, constant size — possibly large, varying size

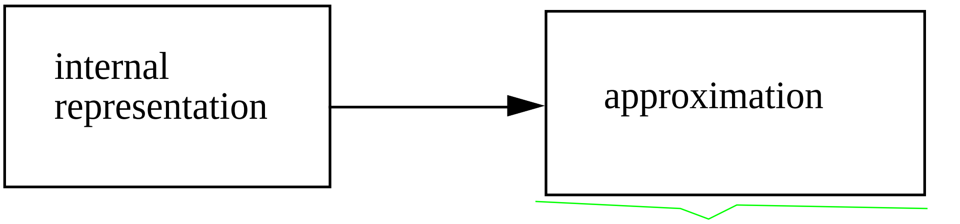
type polygon = **pointer to**

info part
```
record area : real;
       perimeter : real;
       bbox : rectangle;
       no_vertices : integer;
```
exact geometry part
```
       vertices : array[1..1000000]
       of point
end
```

Generic operations, for example:

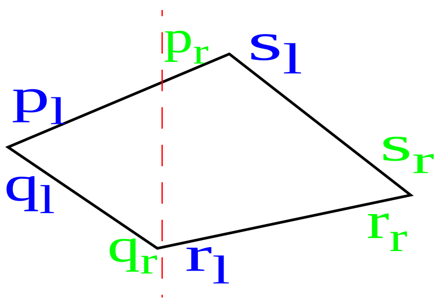


Specifically for *spatial* data types



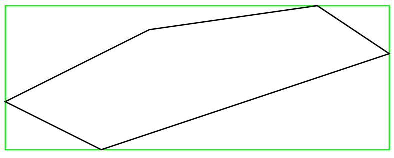
Support spatial algebra view:

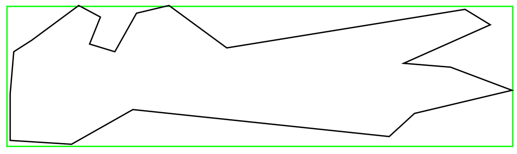
- Representation contains *plane sweep sequence*



q_l	p_l	q_r	r_l	p_r	s_l	s_r	r_r

- Representation contains *approximations*





- Representation contains *stored unary function values*

  e.g. area, perimeter of region computed at construction of SDT value or delayed to first use.

Implementation of SDT operations:

- prechecking on approximations
- looking up stored function values
- use plane-sweep

Generally, use efficient algorithms from Computational Geometry.

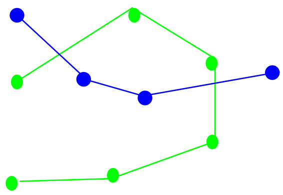
Güting, de Ridder & Schneider 95

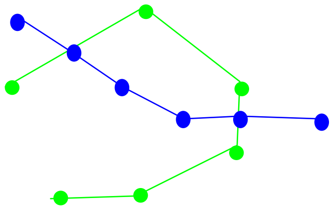
Chan & Ng 97

Special case of *realm-based* SDTs: Never any new intersection points computed; all known in advance, occur in both objects.

- often a parallel scan of two SDT values suffices where otherwise a plane sweep is needed

<u>*lines*</u> $\times$ <u>*lines*</u> $\rightarrow$ <u>*points*</u> **intersection**





plane-sweep needed	parallel scan of halfsegment sequences suffices
(O(n log n + k), complex)	(O(n+k), simple)

- plane sweep also simplified (only static sweep-event structure needed)

(Güting, de Ridder & Schneider 95)

4.2 Spatial Indexing - Supporting Spatial Selection

Spatial indexing supports

- spatial selection
- spatial join
- other operations

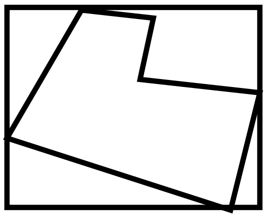
Organizes space and the objects in it. Two strategies:

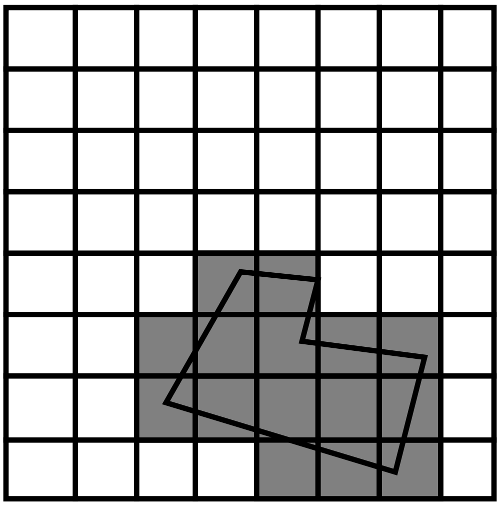
(i) dedicated external data structures

(ii) map spatial objects into 1D space and use a standard index structure (B-tree)

Fundamental idea for indexing and query processing in general: *use of approximations* (Frank 81, Orenstein 86, Orenstein & Manola 88)

- continuous approximation
- grid approximation





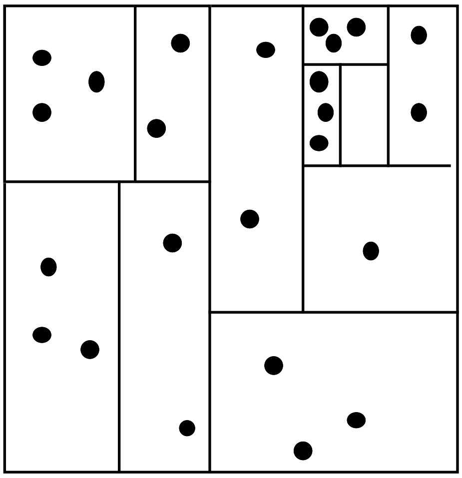
→ *filter and refine strategy*

Dedicated external data structures designed to store a set of points or a set of rectangles.

Operations: insert, delete, member + query operations:

- range query
- nearest neighbour
- distance scan
- intersection query
- containment query

Space decomposed into buckets with associated bucket regions:



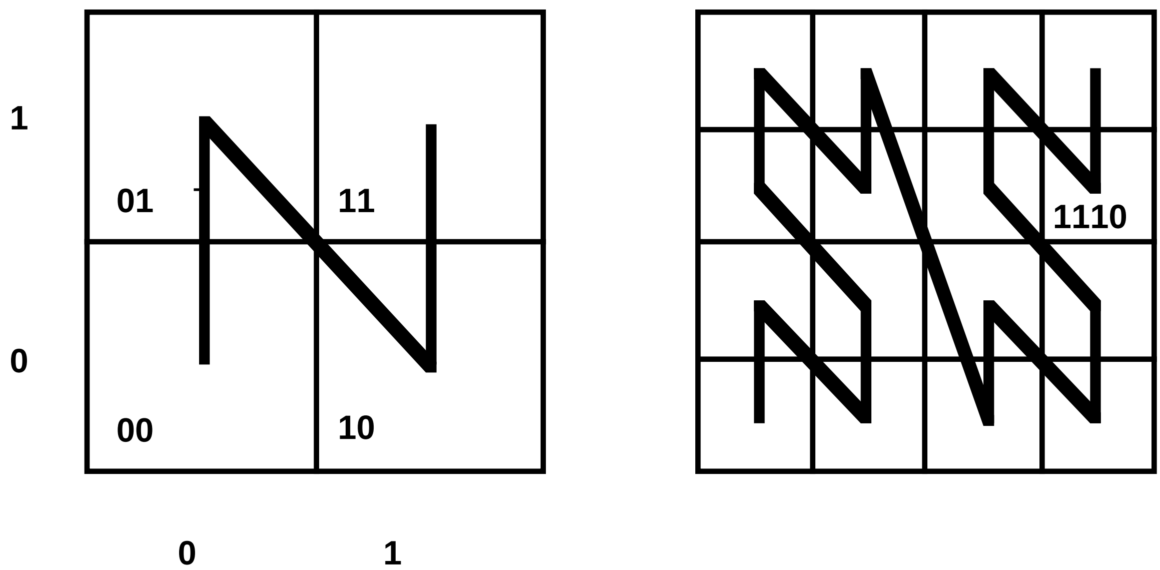
Clustering or secondary index.

4.2.1 One-Dimensional Embedding of Grid Approximations

Basic idea:

(i) find a linear order for the cells of the grid preserving proximity

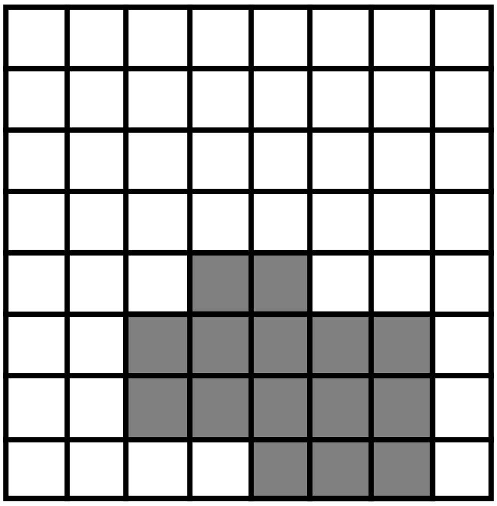
(ii) define this order recursively for a grid corresponding to a hierachical subdivision of space

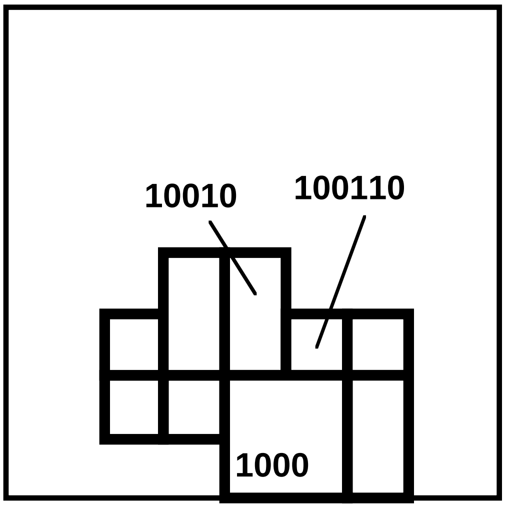


z-order, Morton-order (Morton 66), bit interleaving. General basis for query processing in PROBE (Orenstein 86).

Order imposed on all cells of a hierarchical subdivision is the *lexicographical order of the bit strings*.

Represent any shape by a set of bit strings, called *z-elements*.





Put z-elements as spatial keys into a B-tree *B*.

Containment query with rectangle *r*:

- determine z-elements for *r*
- for each z-element *z* scan a part of the leaf sequence of *B* having *z* as a prefix
- check theses candidates for actual containment, avoid duplicate reports

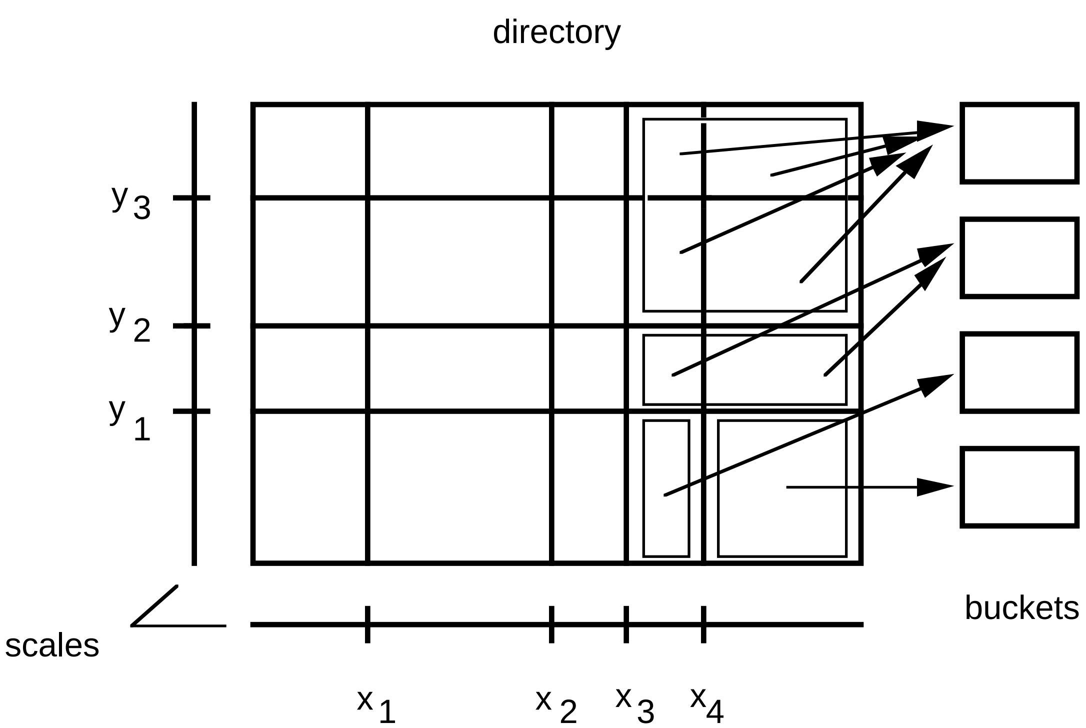
4.2.2 Spatial Index Structures for Points

Long tradition as structures for *multi-attribute retrieval*

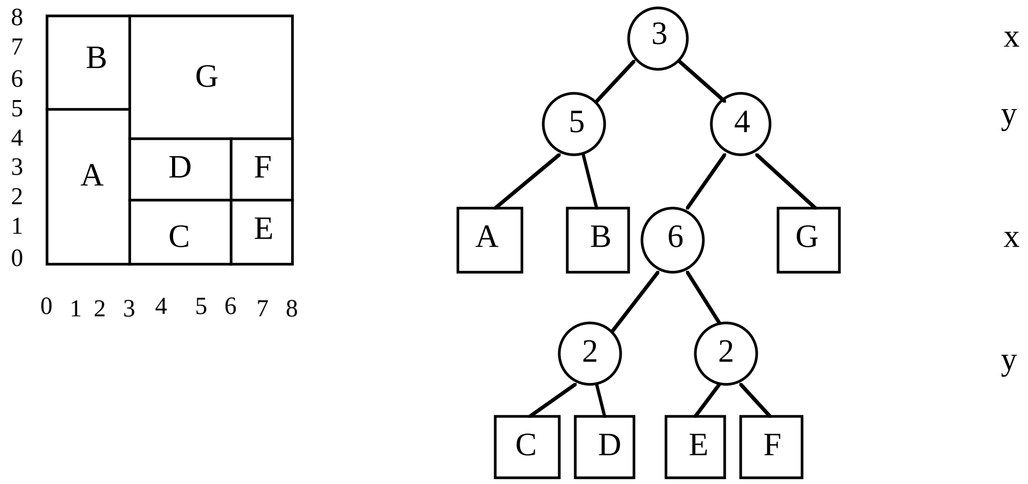
Tuple $t = (x_1, \ldots, x_k)$ point in k dimensions

- grid file
- kd tree
- KDB tree
- LSD tree
- EXCELL
- buddy hash tree
- BANG file
- hB - tree

Grid file Nievergelt, Hinterberger & Sevcik 84



kd-tree (Bentley 75) originally an internal structure:



KDB-tree (Robinson 81): introduce buckets, paginate the binary tree, all leaves at the same level (like B-tree)

LSD-tree (Henrich, Six & Widmayer 89): abandon strict cycling through dimensions

- clever paging algorithm keeps external path length balanced even for very unbalanced binary trees

→ important for *transformation approach*

4.2.3 Spatial Index Structures for Rectangles

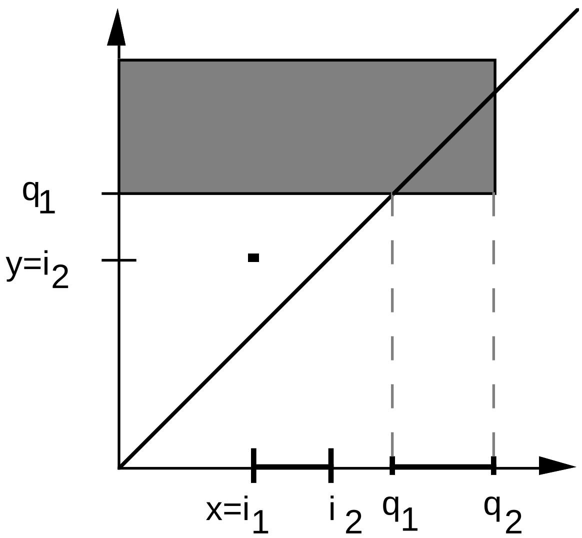
Rectangles more difficult than points, do not fall into a single cell of a bucket partition. Three strategies:

- transformation approach
- overlapping bucket regions
- clipping

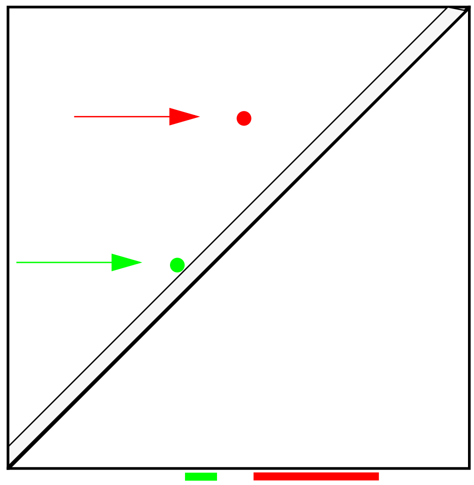
Transformation approach: Hinrichs 85

Seeger & Kriegel 88

Rectangle (x_l, x_r, y_b, y_t) viewed as 4D point. Queries map to regions of 4D space.



Leads to skewed distributions of points.

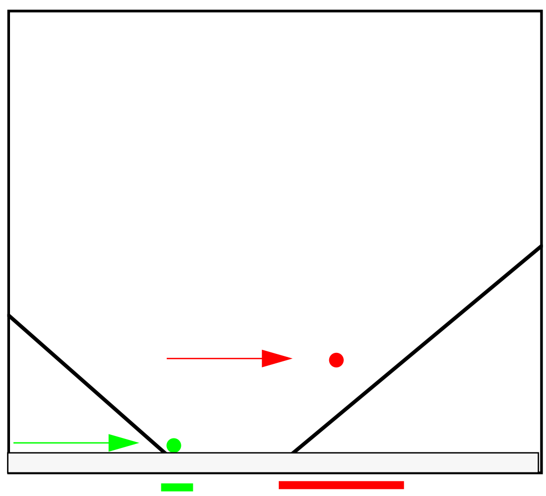


corner
representation

LSD-tree designed to adapt to such skewed distributions.

Center representation: rectangle represented by

(x, y, x-ext, y-ext)



For intervals:

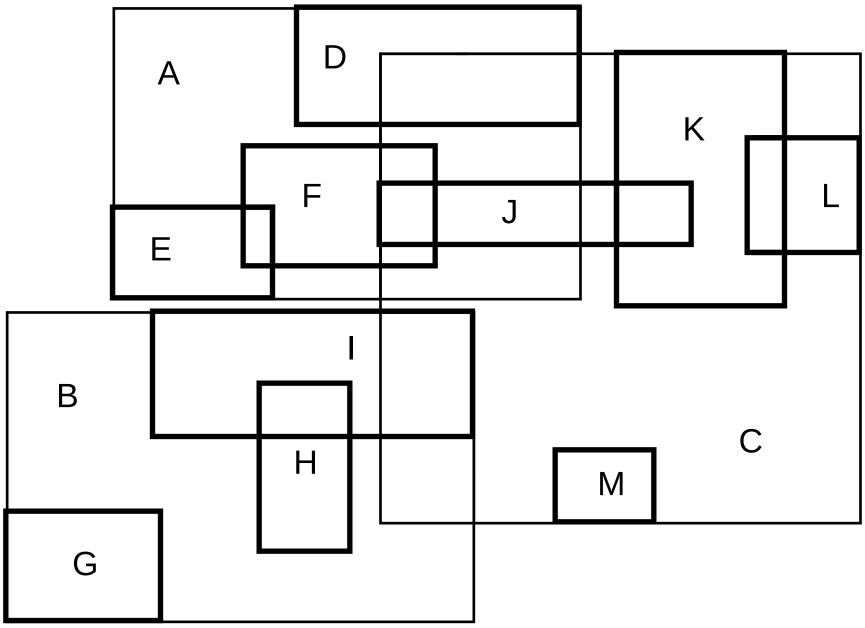
(x, x-ext)

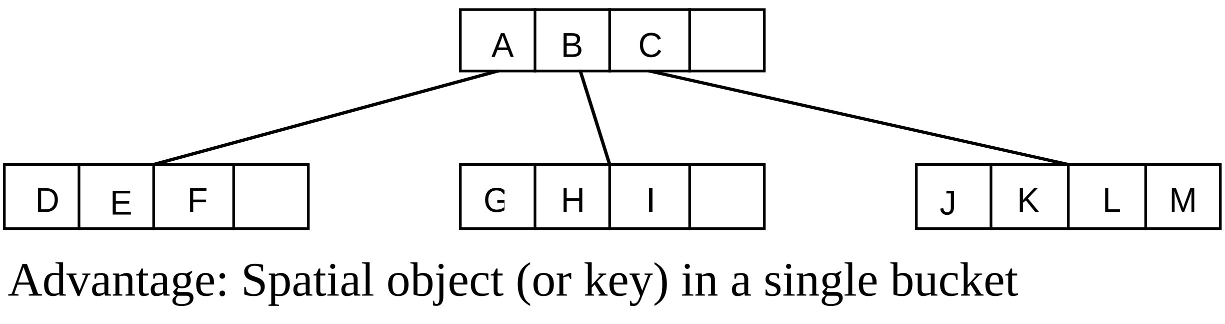
Leads to cone-shaped query regions.

Overlapping bucket regions:

Prime example: the *R-tree* Guttmann 84

Beckmann et al. 90 (R*-tree)





Advantage: Spatial object (or key) in a single bucket

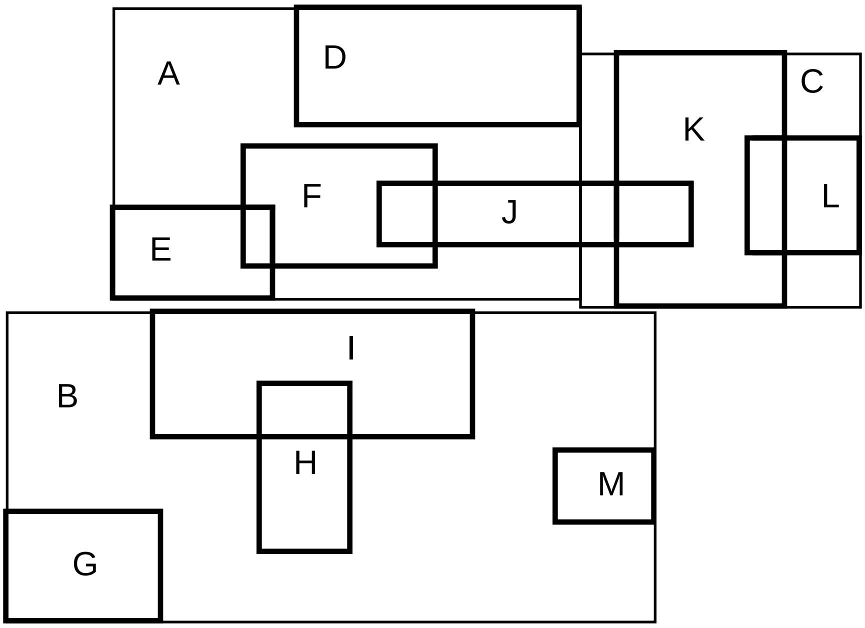
Disadvantage: multiple search paths due to overlapping bucket regions

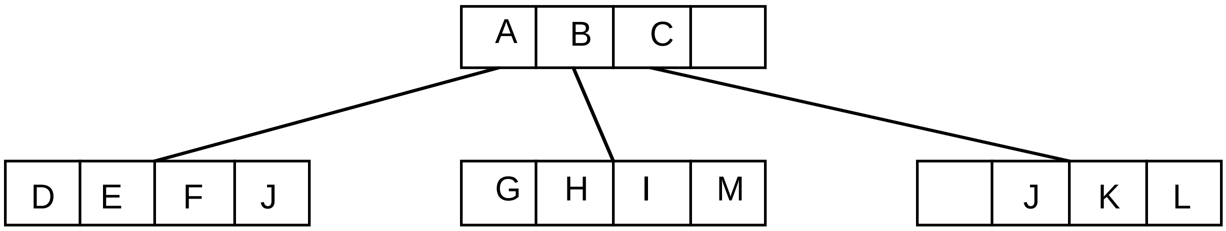
Clipping:

R^+-tree: bucket regions disjoint; data rectangles cut into several pieces, if necessary.

Sellis, Rossopoulos & Faloutsos 87

Faloutsos, Sellis & Rossopoulos 87





Advantage: less branching in search

Disadvantage: multiple entries for a single spatial object (not good as a clustering index)

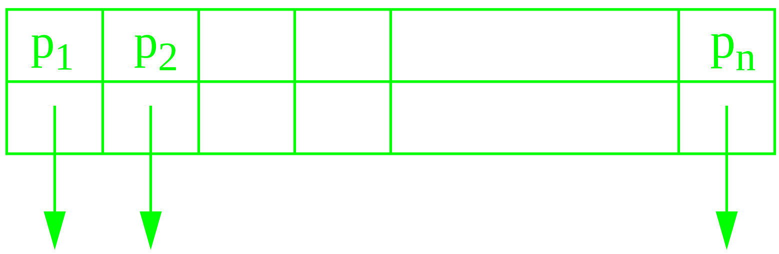
Other directions in spatial indexing, for example

- quad tree variants (Samet 90)
- cell trees (Günther 88; ...)

Generalized Search Trees (GiST; Hellerstein, Naughton & Pfeffer 95)

Generic, customizable tree structure. Key type (class) supplied by user with six required methods:

- consistent (p, q) *p a key (predicate), q a predicate*
- union ($\{p_1, \ldots, p_n\}$) *support for e.g. "bounding box"*
- compress (p)
- decompress (π)
- penalty (p_1, p_2) *support for insertion*
- pickSplit (P) *support for splitting*



pi key predicates, hold for all tuples in subtree

Can implement B-tree, R-tree, and many others.

Further issues:

- provide concurrency control for the new structures (necessary to make them really usable in a production DBMS)

  Kornacker & Banks 95 *R-tree*

  Chakrabarti & Mehrotra 98 *R-tree*

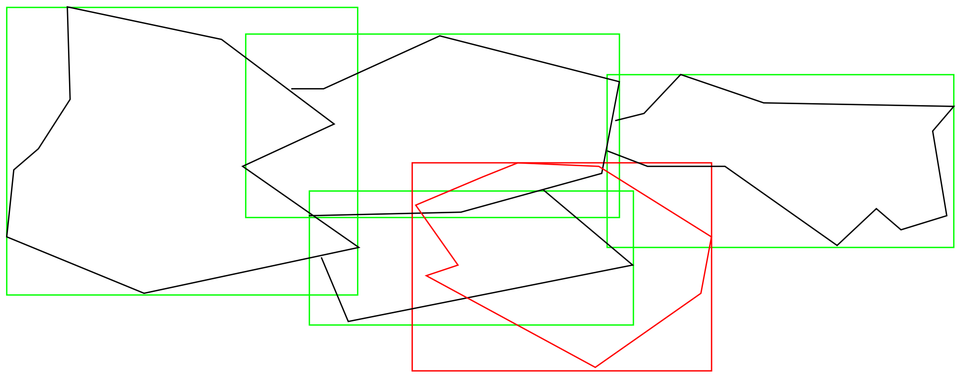
  Kornacker, Mohan & Hellerstein 97 *GiST*

- bulk loading techniques → better clustering, less time for building the index

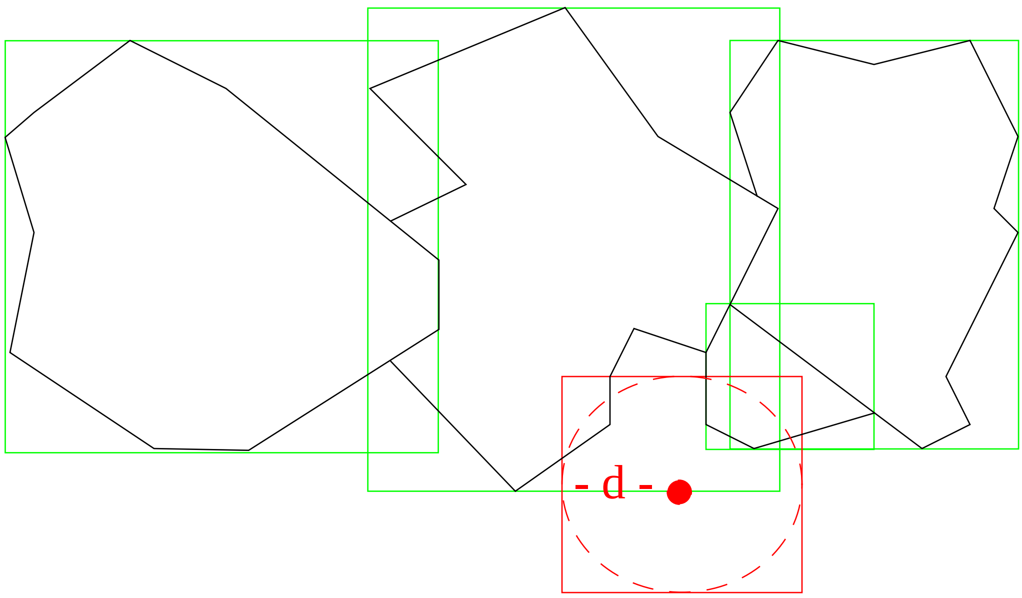
  Van den Bercken, Seeger & Widmayer 97

Surveys on spatial indexing: Widmayer 91, Günther 98 (Section 3.3), Gaede & Günther 98.

Index structures offering a few fundamental query operations support selection with many different predicates (by filter + refine) E.g. intersection query



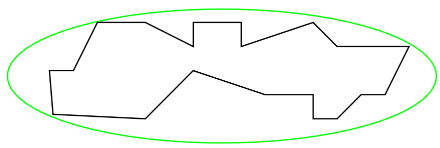
Find regions adjacent to *A*.



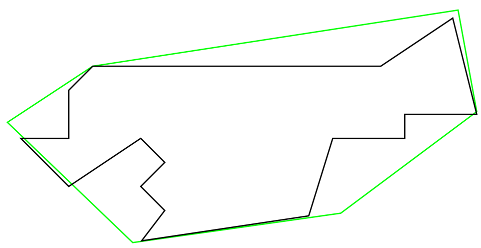
Find regions within distance *d* from *B*.

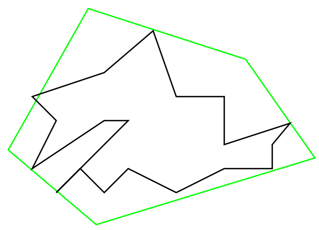
Extensions of the filter + refine strategy: Add a second filter step with better approximation than bounding box (Brinkhoff, Kriegel and Schneider 93), for example

- minimum bounding ellipse



- convex hull   min. bounding 5-corner





Conservative approximation: encloses object

$\rightarrow$ exclude *false hits*

Progressive approximation: contained in the object

$\rightarrow$ identify *hits*

- e.g. maximum enclosed circle, rectangle

Goal: avoid loading the exact geometry for many objects (SDT values).

Using orthogonal polygons as bounding structures has also been investigated (Esperanca & Samet 97).

4.3 Supporting Spatial Join

Very active area in the last few years.

- sort / merge join
- hash join
- filtering the cartesian product

not applicable

too expensive

Central ideas:

- filter + refine
- use of spatial index structures

Classification of strategies:

- grid approximation / bounding box
- none / one / both operand sets represented in a spatial index

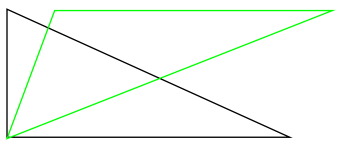
The Filter Step

Grid approximations

E.g., overlap $\rightarrow$ parallel scan through the sets of z-elements

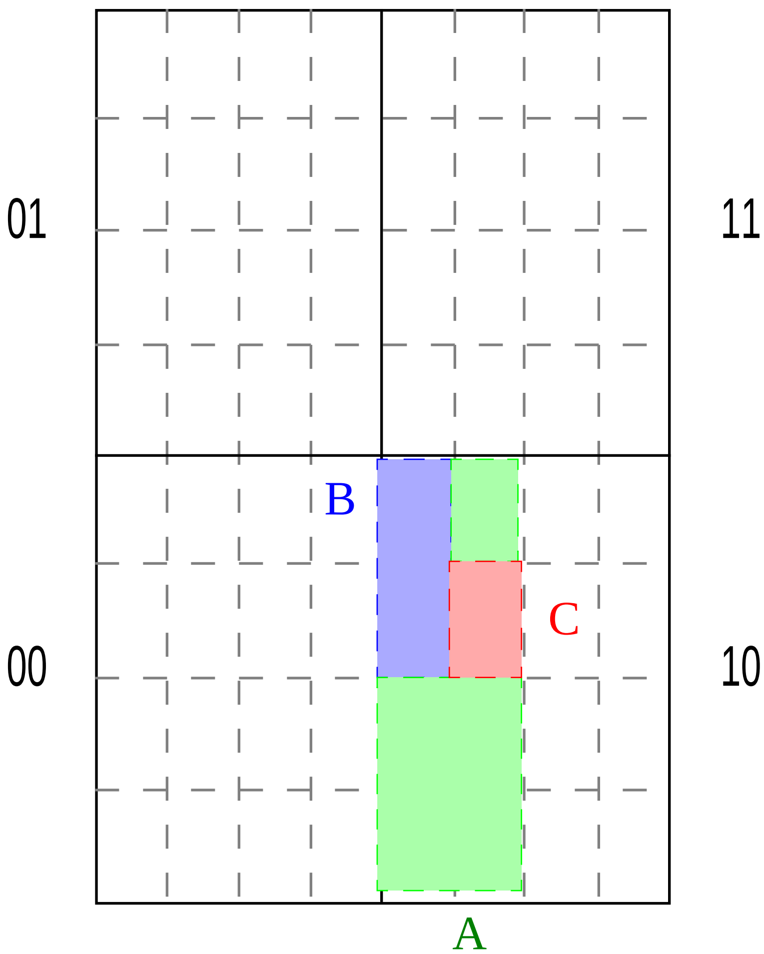
B C

0110, 10, 10010, 100110, 10110, ..



0111, 100, 1010, 1011, 1101 $\rightarrow$ (A,B), (A,C) , ...

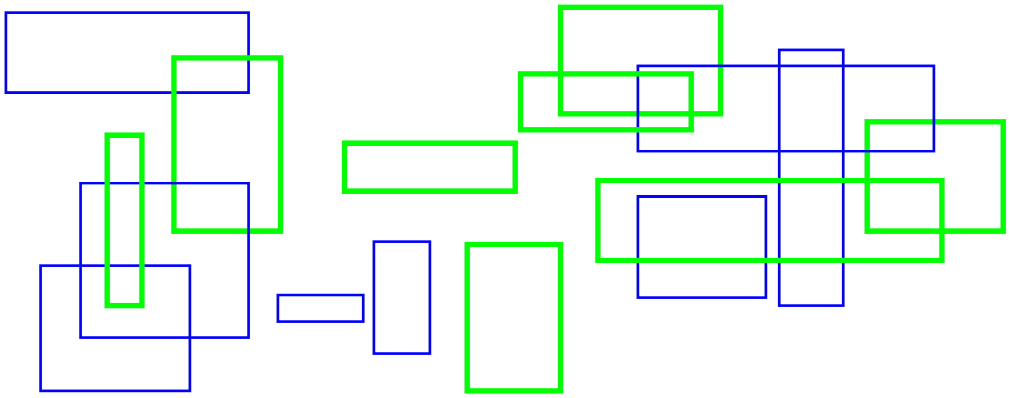
A



Bounding box approximations

(1) None of the operands represented in a spatial index

→ rectangle intersection problem



determine all pairs (p, q), p intersects q

→ bb-join operation, general basis for spatial join; plays the role of sort/merge join for standard data types.

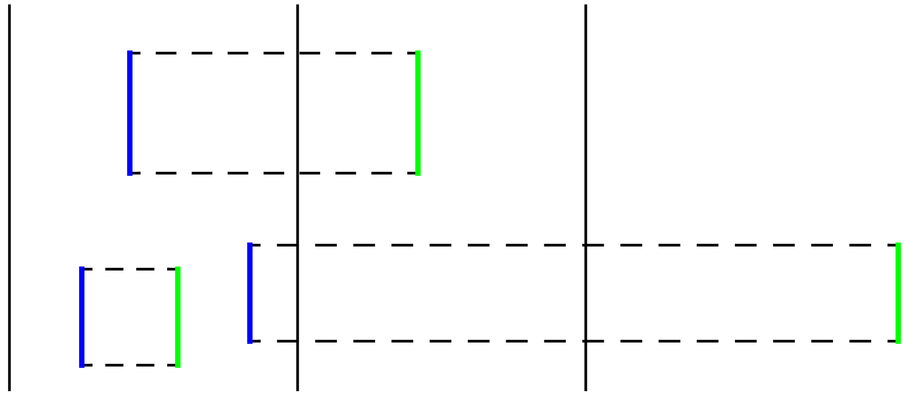
Needed as a query processing method in any case:

```
cities select[pop > 500 000]
states select[language = "french"]
join[center inside area]
```

no index any more

Proposed solutions:

- External, or "sweeping", *divide-and-conquer algorithm* (Güting & Schilling 87), adapted from internal computational geometry algorithm. Finds all intersecting pairs in *one* set of rectangles. Simple modification to treat two sets implemented as *bbjoin* method in the Gral system, Becker & Güting 92).



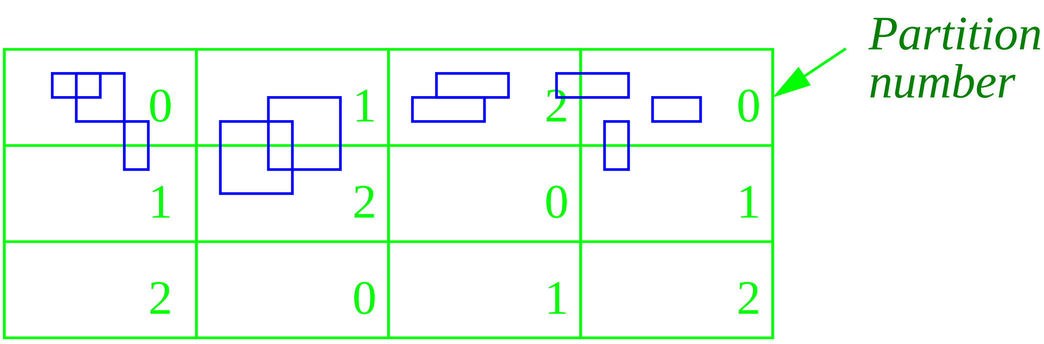
Divide plane into vertical stripes such that each stripe contains about *c* vertical edges of rectangles. Compute intersections between rectangles represented in the stripe by internal DAC algorithm. *External*: Merge adjacent stripes bottom-up, as in external merge sort, writing intermediate structures into files again.

Sweeping: Keep the result of merging stripes 1 through *j* in memory. In each step, read stripe *j*+1 and merge with it.

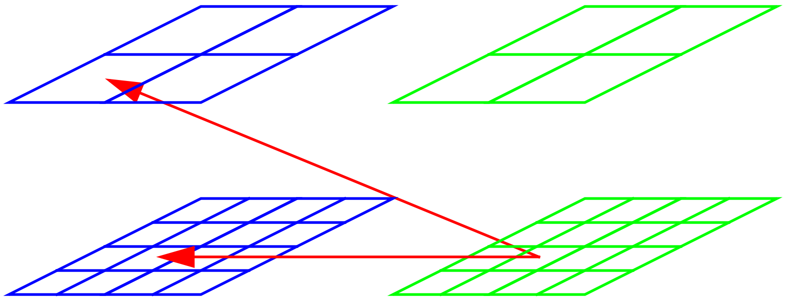
- "*Spatial hash join*" (Lo & Ravishankar 96, Patel & De Witt 96). Assign the two sets of rectangles to two sets of buckets; process pairs of buckets internally. Many design choices; nice analysis in Lo & Ravishankar 96. Concrete proposals and experiments:

  Lo & Ravishankar 96: Overlapping bucket areas for set B; each rectangle assigned to *one* bucket which grows under insertion (initial bucket areas by sampling). Bucket areas for A chosen equal to those of B; each rectangle in A goes into *all* buckets it overlaps.

  Patel & De Witt 96: Decompose plane into regular grid; map grid cells into buckets by round-robin to deal with skew in rectangle distribution.

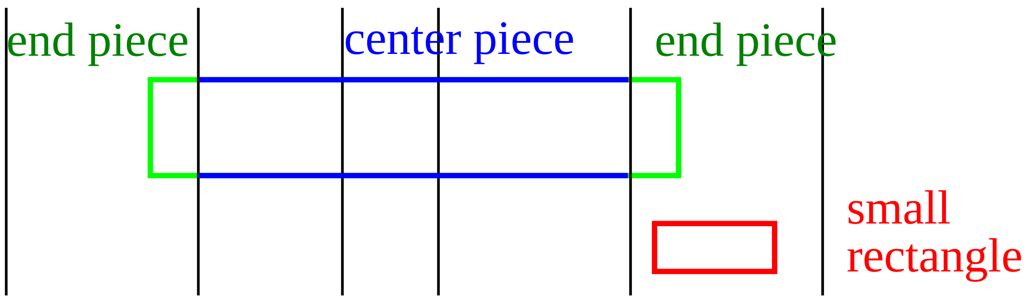


- “*Size separation spatial join*” (Koudas & Sevcik 97). Distribute rectangles into two hierarchical grid partitions. Each rectangle falls into lowest level grid cell that wholly contains it.



Each bucket from set B is merged with all buckets from set A of the same or higher level. No replication of rectangles necessary, each page read exactly once.

- “*Scalable sweeping-based spatial join*” (Arge et al. 98). Somewhat similar to Güting & Schilling 87. Divide plane into vertical stripes. Process each stripe internally by plane sweep.



Worst-case I/O analysis, theoretically optimal.

(2) One of the operands represented in a spatial index

$\rightarrow$ index join, repeated search join. Scan “outer” operand set; for each object perform a search with the bounding box on the index for the “inner” operand.

Or build an index on the fly, starting from the known bucket boundaries of the existing index. (“*Seeded trees*”, Lo & Ravishankar 94)

(3) Both operands are represented in a spatial index

Basic idea: Synchronized, parallel traversal of the two data structures

- grid files

  (Rotem 91, Becker, Hinrichs & Finke 93)

- R-trees

  (Brinkhoff, Kriegel & Seeger 93; refined version with breadth-first traversal: Huang, Jing & Rundensteiner 97)

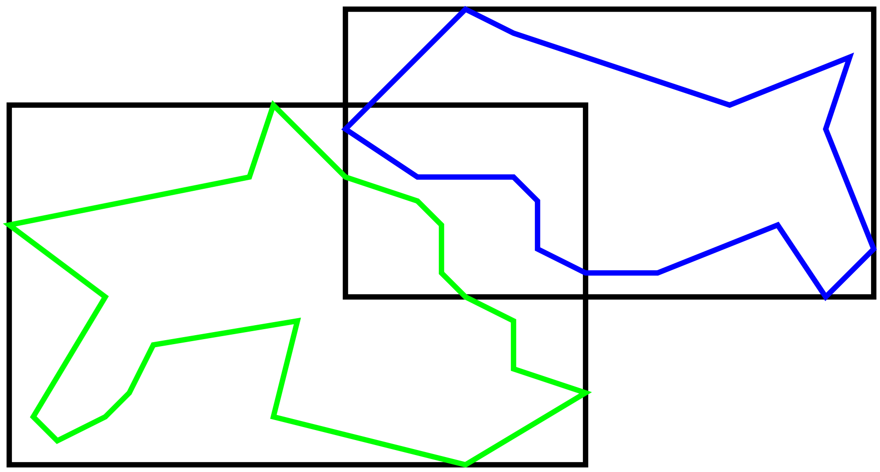
- generalization trees(Günther 93)

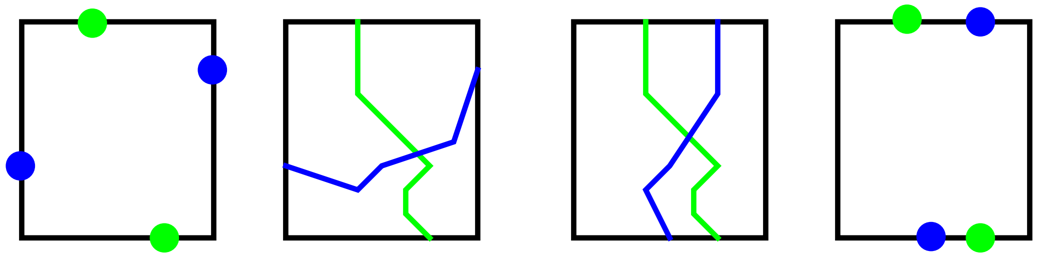
Further idea: Use of join indices (Rotem 91, Lu & Han 92)

Second Filter Step

Introduce a second filter step with better approximations (Brinkhoff et al. 94), as before for spatial selection.

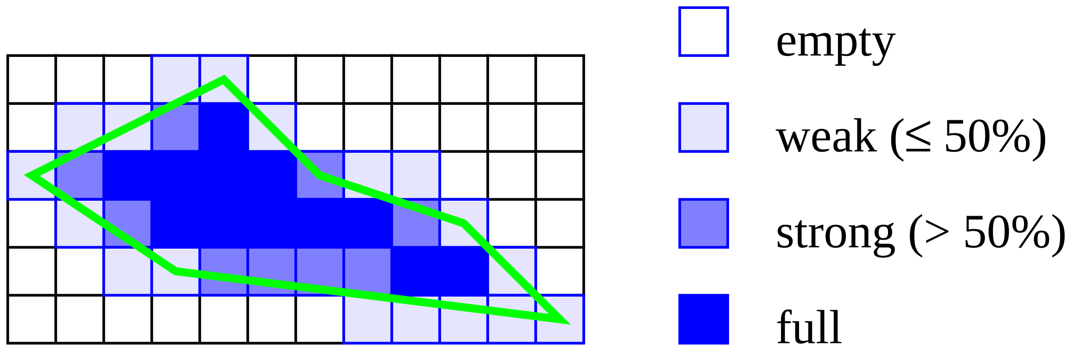
Huang, Jones & Rundensteiner 97: "*Symbolic intersect detection*." Consider intersection area of two bounding boxes, clip contained polygons by this; examine configuration of boundaries in this area.





Some configurations of boundaries imply that polygons must intersect; no further plane-sweep needed.

Zimbrao & Souza 98: For second filter step, represent polygon as a four-colour bitmap.



Scale polygons for comparison. Compare cells: only *weak-weak* and *weak-strong* are inconclusive.

Further work on:

- spatial join in distributed DBMSs (Abel et al. 95) "*spatial semi-join*"
- parallel processing of spatial join (Brinkhoff, Kriegel & Seeger 96, Zhou, Abel & Truffet 97)
- "*incremental distance join*": enumerate pairs by increasing distance; possibly restrict by distance bounds or number of results. (Hjaltason & Samet 98)

5 System Architecture

5.1 Requirements

Integrate the tools from Section 4 into the system architecture. Accomodate the following extensions:

- representations for data types of a spatial algebra
- procedures for atomic operations
- spatial index structures
- access operations for spatial indices
- spatial join algorithms
- cost functions for all these operations
- statistics for estimating selectivity of spatial selection and spatial join
- extensions of the optimizer to map queries into the specialized query processing methods
- spatial data types and operations within data definition and query language
- user interface extensions for graphical I/O

Clean solution: Integrated architecture using an extensible DBMS.

5.2 GIS - Architectures — Using a Closed DBMS

First generation: built on top of file system

→ no high level data definition,

no flexible querying,

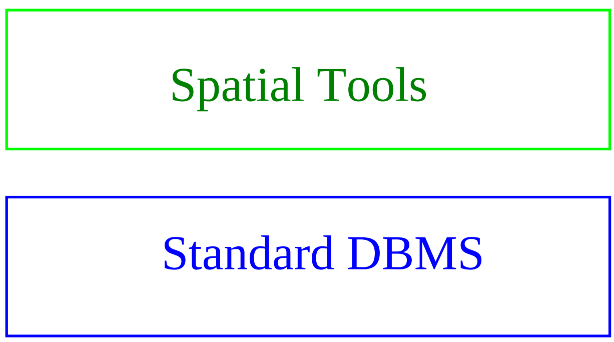
no transaction management,

...

Using a standard (mostly relational) DBMS:

- layered architecture
- dual architecture

Layered architecture



Representation of SDT values:

(1) Decompose SDT value into a set of tuples, one tuple per point or line segment

states	sid	sname	pop
	s01	"Bavaria"	7 000 000

edges	sid	x1	y1	x2	y2	edge
	s01	134.78	92.514	137.13	93.84	1
	s01	137.13	93.84	139.11	96.37	2
	s02					
	...					

Obviously terrible.

(2) Represent SDT values in “long fields” of DBMS

GEOVIEW	Waugh & Healey 87
SIRO-DBMS	Abel 89

DBMS handles geometries only as uninterpreted byte strings; any predicate or other operation on the exact geometry can only be evaluated in the top layer.

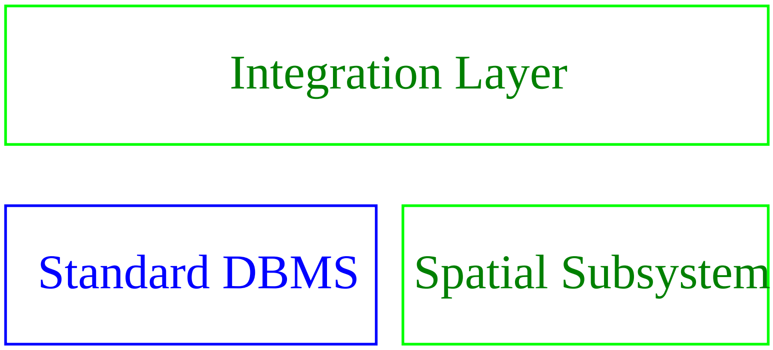
Indexing: maintain sets of z-elements in special relations; index these with a B-tree.

Some variant of these techniques currently used in *Oracle8 Spatial Cartridge* (Oracle 97). Spatial tools layer offers types point, line, polygon, and polygon with holes. A line or polygon value is represented in a set of tuples. Each tuple has a sequence number and fields to take 125 coordinate pairs. Unused fields are padded with zeroes. Hence small lines or polygons can be represented in a single tuple.

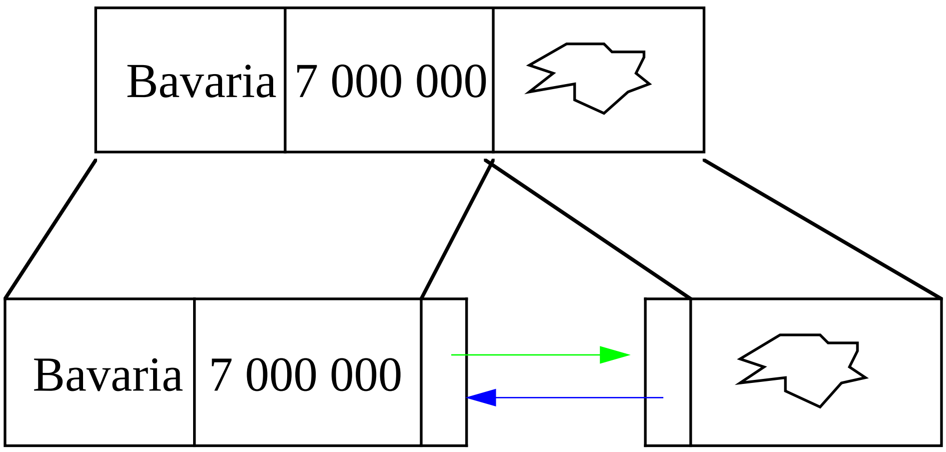
Spatial indexing by sets of z-elements, as mentioned above.

Oracle intends to move to spatial data types as ADTs in columns (Oracle 97).

Dual architecture



Spatial object representation broken into two pieces:



Advantage: freedom to use efficient data structures and algorithms in the spatial subsystem.

Problems: Query must also be decomposed into two parts → complex query processing.

No global query optimization possible.

For example, either spatial or standard index can be used for a given query, but standard DBMS does not reveal its cost estimate.

Used in most commercial GIS; some research prototypes, e.g.

ARC / INFO(Morehouse 89)

SICAD(Schilcher 85)

Ooi, Sacks-Davis & McDonell 89

Dual architecture also in

Aref & Samet 91a, 91b

but build a new system (not using a standard DBMS).

5.3 Integrated Spatial DBMS Architecture — Using an Extensible DBMS

Research into *extensible database systems* tried to make extensions required above possible.

POSTGRES	Starburst
Probe	Gral
EXODUS	Sabrina
GENESIS	DASDBS
Predator	

Leads to *integrated architecture*:

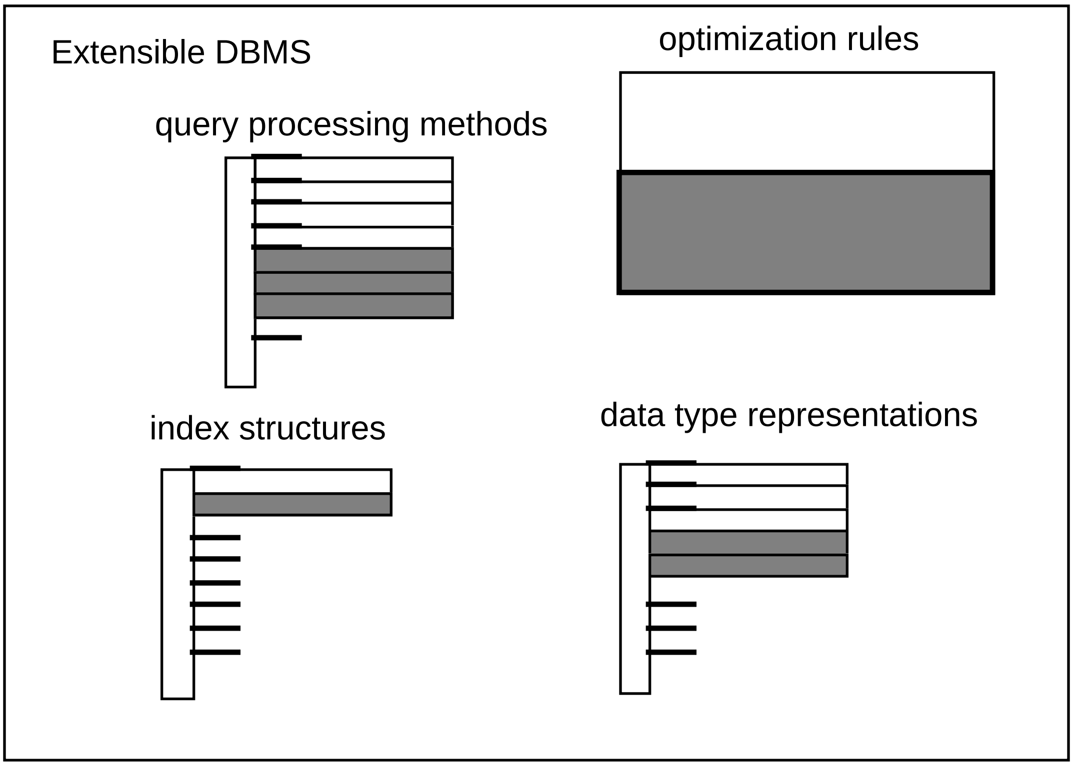
(1) No difference in principle between a “standard” data type (STRING) and a spatial data type (REGIONS); same for operations.

(2) No difference in principle between index for standard attributes (B-tree) and for spatial attributes (R-tree).

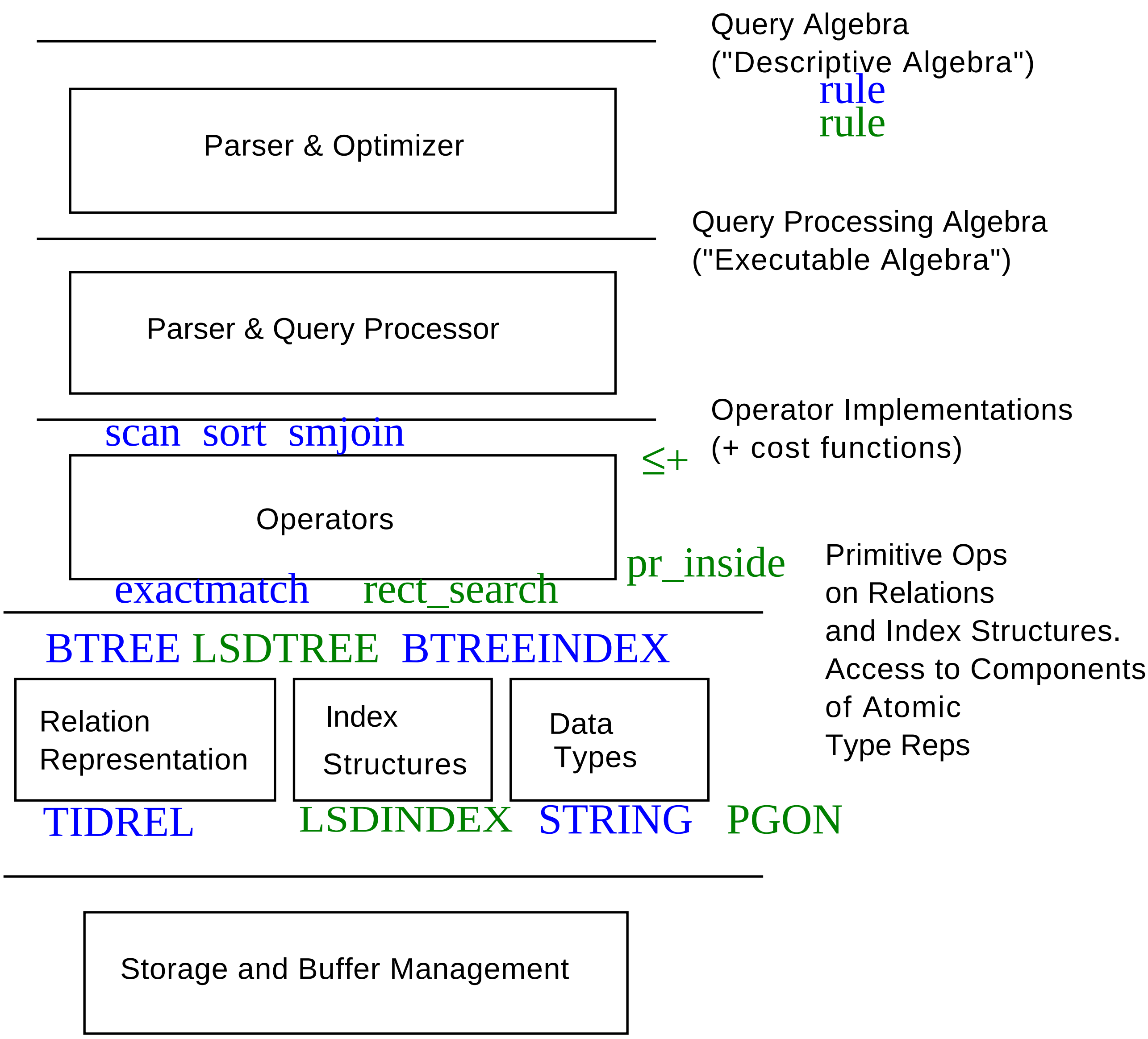
(3) Sort/merge join and bounding-box join are basically the same.

(4) Mechanisms for query optimization do not distiguish spatial or other operations.

Extensible architecture (in general)



Extensible architecture (specifically for Gral)



Some spatial DBMS prototypes built based on extensible systems:

PROBE Orenstein 86

Orenstein & Manola 88

- functional data model
- general POINT-SET data type, SDTs as refinements
- "approximate geometry" processing: SDT values as sets of z-elements; spatial indexing and spatial join in the system kernel (filter + refine strategy).

DASDBS Schek et al. 90

Wolf 89

- nested relational model
- EDT ("external data type") concept

  $\rightarrow$ external "geometric computation service"
- spatial access methods partitioning data space into cells are assumed (e.g. grid file, R-tree); each EDT must offer *clip* and *compose* functions.

GRAL Güting 89

Becker & Güting 92

- relational model
- many-sorted algebra as a formal basis; used to describe query language and query processing system
- rule-based optimizer
- bounding box as interface to access methods (LSD tree)

PARADISE DeWitt et al. 94

Patel et al. 97

- parallel spatial DBMS; emphasis on parallel processing techniques
- treatment of satellite images / large *n*-dimensional arrays
- implemented on SHORE storage manager
- treatment of very large data sets (120 GB)

MONET Boncz, Quak & Kersten 96

- parallel, main memory, spatial DBMS
- many-sorted algebra extensibility (as in Gral)
- "decomposed storage" model (binary relations)

Further prototypes:

Geo^{++} Oosterom & Vijlbrief 91

on POSTGRES Vijlbrief & Oosterom 92

GéoSabrina Larue, Pastre & Viémont 93

on Sabrina

Commercial extensible DBMS with spatial extensions, for example:

- Informix Universal Server, with *Geodetic DataBlade* (Informix 97)
- IBM DB2 *Spatial Extender* (Davis 98)

Not covered in this tutorial:

- image database systems, raster data management

Lots of other interesting issues (only selected references):

- *spatio-temporal databases, moving objects databases*

  Worboys 94, Sistla et al. 97, Güting et al. 98

- *spatial objects with imprecise boundaries*

  Erwig & Schneider 97

- *multi-scale modeling / cartographic generalization*

  Puppo & Dettori 95, Rigaux & Scholl 95

- *data lineage*

  Woodruff & Stonebraker 97

- *query processing for spatial networks*

  Shekhar & Liu 97, Huang, Jing & Rundensteiner 97

- *nearest neighbor / similarity search*

  Rossopoulos, Kelley & Vincent 95, Berchtold et al. 98, Hjaltason & Samet 95

- *spatial constraint databases*

  Paredaens 95, Belussi, Bertino & Catania 97, Grumbach, Rigaux & Segoufin 98

References

Abdelmoty, A.I., M.H. Williams, and N.W. Paton, Deduction and Deductive Databases for Geographic Data Handling. Proc. 3rd Intl. Symposium on Large Spatial Databases, Singapore, 1993, 443-464.

Abel, D.J., B.C. Ooi, K.L Tan, R. Power, and J.X. Yu, Spatial Join Strategies in Distributed Spatial DBMS. Proc. 4th Intl. Symposium on Large Spatial Databases, 1995, 348-367.

Abel, D.J., SIRO-DBMS: A Database Tool Kit for Geographical Information Systems. *Intl. J. of Geographical Information Systems 3 (1989)*, 103-116.

Aiken, A., J. Chen, M. Stonebraker, and A. Woodruff, Tioga-2: A Direct Manipulation Database Visualization Environment. Proc. 12th Intl. Conf. on Data Engineering, 1996, 208-217.

Allen, J.F., Maintaining Knowledge about Temporal Intervals. *Communications of the ACM 16 (1983)*, 832-843.

Aref, W., and H. Samet, Extending a DBMS with Spatial Operations. Proc. 2nd Intl. Symposium on Large Spatial Databases, Zürich, 1991, 299-318.

Aref, W., and H. Samet, Optimization Strategies for Spatial Query Processing. Proc. 17th Intl. Conf. on Very Large Data Bases, Barcelona, 1991, 81-90.

Arge, L., O. Procopiuc, S. Ramaswamy, T. Suel, and J.S. Vitter, Scalable Sweeping-Based Spatial Join. Proc. 24th Intl. Conf. on Very Large Data Bases, 1998, 570-581.

Becker, L., and R.H. Güting, Rule-Based Optimization and Query Processing in an Extensible Geometric Database System. *ACM Transactions on Database Systems 17 (1992)*, 247-303.

Becker, L., K. Hinrichs, and U. Finke, A New Algorithm for Computing Joins with Grid Files. Proc. 9th Intl. Conf. on Data Engineering, Vienna, 1993, 190-198.

Beckmann, N., H.P. Kriegel, R. Schneider, and B. Seeger, The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. Proc. ACM SIGMOD Conf. 1990, 322-331.

Belussi, A., E. Bertino, and B. Catania, Manipulating Spatial Data in Constraint Databases. Proc. 5th Intl. Symposium on Large Spatial Databases, Berlin, 1997, 115-141.

Bentley, J.L., Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM 18 (1975)*, 509-517.

Berchtold, S., B. Ertl, D.A. Keim, H.P. Kriegel, and T. Seidl, Fast Nearest Neighbor Search in High-Dimensional Space. Proc. 14th Intl. Conf. on Data Engineering, 1998, 209-218.

Boncz, P.A., W. Quak, and M.L. Kersten, Monet and its Geographic Extensions: A Novel Approach to High Performance GIS Processing. Proc. 5th Intl. Conf. on Extending Database Technology, 1996, 147-166.

Brinkhoff, T., H.P. Kriegel, and B. Seeger, Efficient Processing of Spatial Joins Using R-Trees. Proc. ACM SIGMOD Conf., Washington, 1993, 237-246.

Brinkhoff, T., H.P. Kriegel, and B. Seeger, Parallel Processing of Spatial Joins Using R-trees. Proc. 12th Intl. Conf. on Data Engineering, 1996, 258-265.

Brinkhoff, T., H.P. Kriegel, and R. Schneider, Comparison of Approximations of Complex Objects Used for Approximation-Based Query Processing in Spatial Database Systems. Proc. 9th Intl. Conf. on Data Engineering, Vienna, 1993, 40-49.

Brinkhoff, T., H.P. Kriegel, R. Schneider, and B. Seeger, Multi-Step Processing of Spatial Joins. Proc. ACM SIGMOD Conf., Minneapolis, 1994, 197-208.

Chakrabarti, K., and S. Mehrotra, Dynamic Granular Locking Approach to Phantom Protection in R-Trees. Proc. 14th Intl. Conf. on Data Engineering, 1998, 446-454.

Chan, E.P.F., and J. N.H. Ng, A General and Efficient Implementation of Geometric Operators and Predicates. Proc. 5th Intl. Symposium on Large Spatial Databases, Berlin, 1997, 69-93.

Chan, E.P.F., and R. Zhu, QL/G – A Query Language for Geometric Data Bases. Proc. First Intl. Conf. on GIS in Urban and Environmental Planning, Samos, Greece, 1996, 271-286.

Clementini, E., P. Di Felice, and G. Califano, Composite Regions in Topological Queries. *Information Systems 20 (1995)*, 579-594.

Clementini, E., P. Di Felice, and P. van Oosterom, A Small Set of Formal Topological Relationships Suitable for End-User Interaction. Proc. 3rd Intl. Symposium on Large Spatial Databases, Singapore, 1993, 277-295.

Davis, J.R., IBM's DB2 Spatial Extender: Managing Geo-Spatial Information Within the DBMS. Technical report, IBM Corp., May 1998.

DeWitt, D.J., N. Kabra, J. Luo, J.M. Patel, and J. Yu, Client-Server Paradise. Proc. 20th Intl. Conf. on Very Large Data Bases, Santiago, 1994, 558-569.

Egenhofer, M., A Formal Definition of Binary Topological Relationships. Proc. 3rd Intl. Conf. on Foundations of Data Organization and Algorithms, Paris, 1989, 457-472.

Egenhofer, M., A. Frank, and J.P. Jackson, A Topological Data Model for Spatial Databases. Proc. First Intl. Symposium on Large Spatial Databases, Santa Barbara, 1989, 271-286.

Egenhofer, M., and J. Herring, Categorizing Binary Topological Relationships between Regions, Lines, and Points in Geographic Databases. University of Maine, Orono, Maine, Dept. of Surveying Engineering, Technical Report, 1992.

Egenhofer, M., Extending SQL for Cartographic Display. *Cartography and Geographic Information Systems 18 (1991)*, 230-245.

Egenhofer, M., Reasoning about Binary Topological Relations. Proc. 2nd Intl. Symposium on Large Spatial Databases, Zürich, 1991, 143-160.

Egenhofer, M., Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering 6 (1994)*, 86-95.

Egenhofer, M.J., E. Clementini, and P. Di Felice, Topological Relations between Regions with Holes. *Intl. Journal of Geographical Information Systems 8 (1994)*, 129-142.

Erwig, M., and M. Schneider, Vague Regions. Proc. 5th Intl. Symposium on Large Spatial Databases, Berlin, 1997, 298-320.

Esperança, C., and H. Samet, Orthogonal Polygons as Bounding Structures in Filter-Refine Query Processing Strategies. Proc. 5th Intl. Symposium on Large Spatial Databases, Berlin, 1997, 197-220.

Faloutsos, C., T. Sellis, and N. Rossopoulos, Analysis of Object-Oriented Spatial Access Methods. Proc. ACM SIGMOD Conf., San Francisco, 1987, 426-439.

Frank, A., and W. Kuhn, Cell Graphs: A Provable Correct Method for the Storage of Geometry. Proc. 2nd Intl. Symposium on Spatial Data Handling, Seattle, 1986, 411-436.

Frank, A., Application of DBMS to Land Information Systems. Proc. 7th Intl. Conf. on Very Large Data Bases, Cannes, 1981, 448-453.

Gaede, V., and O. Günther, Multidimensional Access Methods. *ACM Computing Surveys 30 (1998)*, 170-231.

Gargano, M., E. Nardelli, and M. Talamo, Abstract Data Types for the Logical Modeling of Complex Data. *Information Systems 16, 5 (1991).*

Greene, D., and F. Yao, Finite-Resolution Computational Geometry. Proc. 27th IEEE Symp. on Foundations of Computer Science, 1986, 143-152.

Grumbach, S., P. Rigaux, and L. Segoufin, The DEDALE System for Complex Spatial Queries. Proc. ACM SIGMOD Conf., 1998, 213-224.

Günther, O., Efficient Computation of Spatial Joins. Proc. 9th Intl. Conf. on Data Engineering, Vienna, 1993, 50-59.

Günther, O., Efficient Structures for Geometric Data Management. LNCS 337, Springer, 1988.

Günther, O., Environmental Information Systems. Springer-Verlag, Berlin-Heidelberg-New York, 1998.

Güting, R.H., An Introduction to Spatial Database Systems. *VLDB Journal 3 (1994),* 357-399.

Güting, R.H., and M. Schneider, Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal 4 (1995)*, 100-143.

Güting, R.H., and M. Schneider, Realms: A Foundation for Spatial Data Types in Database Systems. Proc. 3rd Intl. Symposium on Large Spatial Databases, Singapore, 1993, 14-35.

Güting, R.H., and W. Schilling, A Practical Divide-and-Conquer Algorithm for the Rectangle Intersection Problem. *Information Sciences 42 (1987)*, 95-112.

Güting, R.H., Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. In: J.W. Schmidt, S. Ceri, M. Missikoff (eds.), Proc. EDBT 1988, 506-527.

Güting, R.H., Gral: An Extensible Relational Database System for Geometric Applications. Proc. 15th Intl. Conf. on Very Large Data Bases, Amsterdam, 1989, 33-44.

Güting, R.H., GraphDB: A Data Model and Query Language for Graphs in Databases. Proc. 20th Intl. Conf. on Very Large Data Bases, Santiago, 1994, 297-308.

Güting, R.H., M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis, A Foundation for Representing and Querying Moving Objects. FernUniversität Hagen, Informatik-Report 238, September 1998.

Güting, R.H., T. de Ridder, and M. Schneider, Implementation of the ROSE Algebra: Efficient Algorithms for Realm-Based Spatial Data Types. Proc. 4th Intl. Symposium on Large Spatial Databases, Portland, 1995.

Guttmann, R., R-Trees: A Dynamic Index Structure for Spatial Searching. Proc. ACM SIGMOD Conf., 1984, 47-57.

Hellerstein, J.M., J.F. Naughton, and A. Pfeffer, Generalized Search Trees for Database Systems. Proc. 21st Intl. Conf. on Very Large Data Bases, 1995, 562-573.

Henrich, A., H.-W. Six, and P. Widmayer, The LSD-Tree: Spatial Access to Multidimensional Point- and Non-Point-Objects. Proc. 15th Intl. Conf. on Very Large Data Bases, Amsterdam, 1989, 45-53.

Hinrichs, K., The Grid File System: Implementation and Case Studies of Applications. Doctoral Thesis, ETH Zürich, 1985.

Hjaltason, G., and H. Samet, Ranking in Spatial Databases. Proc. 4th Intl. Symposium on Large Spatial Databases, 1995, 83-95.

Hjaltason, G.R., and H. Samet, Incremental Distance Join Algorithms for Spatial Databases. Proc. ACM SIGMOD Conf., 1998, 237-248.

Hoop, S. de, and P. van Oosterom, Storage and Manipulation of Topology in Postgres. Proc. 3rd European Conf. on Geographical Information Systems, Munich, 1992, 1324-1336.

Huang, Y.-W., N. Jing, and E.A. Rundensteiner, Spatial Joins Using R-Trees: Breadth-First Traversal with Global Optimizations. Proc. 23rd Intl. Conf. on Very Large Data Bases, 1997, 396-405.

Huang, Y.W., M. C. Jones, and E. A. Rundensteiner, Improving Spatial Intersect Joins Using Symbolic Intersect Detection. Proc. 5th Intl. Symposium on Large Spatial Databases, Berlin, 1997, 165-177.

Huang, Y.W., N. Jing, and E.A. Rundensteiner, Integrated Query Processing Strategies for Spatial Path Queries. Proc. 13th Intl. Conf. on Data Engineering, 1997, 477-486.

Informix Geodetic DataBlade Module: User's Guide. Informix Press, June 1997.

Kornacker, M., and D. Banks, High-Concurrency Locking in R-Trees. Proc. 21st Intl. Conf. on Very Large Data Bases, 1995, 134-145.

Kornacker, M., C. Mohan, and J.M. Hellerstein, Concurrency and Recovery in Generalized Search Trees. Proc. ACM SIGMOD Conf., 1997, 62-72.

Koudas, N., and K.C. Sevcik, Size Separation Spatial Join. Proc. ACM SIGMOD Conf., 1997, 324-335.

Larue, T., D. Pastre, and Y. Viémont, Strong Integration of Spatial Domains and Operators in a Relational Database System. Proc. 3rd Intl. Symposium on Large Spatial Databases, Singapore, 1993, 53-72.

Lo, M.L., and C.V. Ravishankar, Spatial Hash-Joins. Proc. ACM SIGMOD Conf., 1996, 247-258.

Lo, M.L., and C.V. Ravishankar, Spatial Joins Using Seeded Trees. Proc. ACM SIGMOD Conf., Minneapolis, 1994, 209-220.

Lu, W., and J. Han, Distance-Associated Join Indices for Spatial Range Search. Proc. 9th Intl. Conf. on Data Engineering, Vienna, 1992, 284-292.

Maingenaud, M., and M. Portier, Cigales: A Graphical Query Language for Geographical Information Systems. Proc. 4th Intl. Symposium on Spatial Data Handling, Zürich, 1990, 393-404.

Meyer, B., Beyond Icons: Towards New Metaphors for Visual Query Languages for Spatial Information Systems. In: R. Cooper (ed.), Interfaces to Database Systems, Springer, 1992, 113-135.

Morehouse, S., The Architecture of ARC/INFO. Proc. Auto-Carto 9, Baltimore, 1989.

Morton, G.M., A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing. IBM, Ottawa, Canada, 1966.

Nievergelt, J., H. Hinterberger, and K.C. Sevcik, The Grid File: An Adaptable, Symmetric Multikey File Structure. *ACM Transactions on Database Systems 9 (1984),* 38-71.

Ooi, B.C., R. Sacks-Davis, and K.J. McDonell, Extending a DBMS for Geographic Applications. Proc. 5th Intl. Conf. on Data Engineering, Los Angeles, 1989, 590-597.

Oosterom, P. van, and T. Vijlbrief, Building a GIS on Top of the Open DBMS POSTGRES. Proc. 2nd European Conf. on Geographical Informations Systems (EGIS 91), Brussels, 1991, 775-787.

Oracle8: Spatial Cartridge. An Oracle Technical White Paper. Oracle Corporation, June 1997.

Orenstein, J., and F. Manola, PROBE Spatial Data Modeling and Query Processing in an Image Database Application. *IEEE Trans. on Software Engineering 14 (1988)*, 611-629.

Orenstein, J.A., Spatial Query Processing in an Object-Oriented Database System. Proc. ACM SIGMOD Conf. 1986, 326-336.

Pajarola, R., T. Ohler, P. Stucki, K. Szabo, and P. Widmayer, The Alps at Your Fingertips: Virtual Reality and Geoinformation Systems. Proc. 14th Intl. Conf. on Data Engineering, 1998, 550-557.

Papadias, D., N. Mamoulis, and V. Delis, Algorithms for Querying by Spatial Structure. Proc. 24th Intl. Conf. on Very Large Data Bases, 1998, 546-557.

Papadias, D., Y. Theodoridis, T. Sellis, and M. Egenhofer, Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-Trees. Proc. ACM SIGMOD Conf., 1995, 92-103.

Paredaens, J., Spatial Databases, The Final Frontier. Proc. 5th Intl. Conf. on Database Theory, 1995, 14-32.

Patel, J.M., and D.J. DeWitt, Partition Based Spatial-Merge Join. Proc. ACM SIGMOD Conf., 1996, 259-270.

Patel, J.M., J.B. Yu, N. Kabra, K. Tufte, B. Nag, J. Burger, N.E. Hall, K. Ramasamy, R. Lueder, C. Ellman, J. Kupsch, S. Guo, D.J. DeWitt, and J. Naughton, Building a Scaleable Geo-Spatial DBMS: Technology, Implementation, and Evaluation. Proc. ACM SIGMOD Conf., 1997, 336-347.

Puppo, E., and G. Dettori, Towards a Formal Model for Multi-Resolution Spatial Maps. Proc. 4th Intl. Symposium on Large Spatial Databases, 1995, 152-169.

Rigaux, P., and M. Scholl, Multi-Scale Partitions: Application to Spatial and Statistical Databases. Proc. 4th Intl. Symposium on Large Spatial Databases, 1995, 170-183.

Robinson, J.T., The KDB-Tree: A Search Structure for Large Multidimensional Dynamic Indexes. Proc. ACM SIGMOD Conf., 1981, 10-18.

Rossopoulos, N., S. Kelley, and F. Vincent, Nearest Neighbor Queries. Proc. ACM SIGMOD Conf., 1995, 71-79.

Rotem, D., Spatial Join Indices. Proc. 7th Intl. Conf. on Data Engineering, Kobe, Japan, 1991, 500-509.

Samet, H., The Design and Analysis of Spatial Data Structures. Addison-Wesley, 1990.

Schek, H.J., H.B. Paul, M.H. Scholl, and G. Weikum, The DASDBS Project: Objectives, Experiences, and Future Prospects. *IEEE Transactions on Knowledge and Data Engineering 2 (1990)*, 25-43.

Schilcher, M., Interactive Graphic Data Processing in Cartography. *Computers & Graphics 9 (1985)*, 57-66.

Schneider, M., Spatial Data Types for Database Systems. LNCS 1288. Springer-Verlag, Berlin-Heidelberg-New York, 1997.

Scholl, M., and A. Voisard, Thematic Map Modeling. Proc. First Intl. Symp. on Large Spatial Databases, Santa Barbara, 1989, 167-190.

Seeger, B., and H.P. Kriegel, Techniques for Design and Implementation of Efficient Spatial Access Methods. Proc. 14th Intl. Conf on Very Larga Data Bases, Los Angeles, 1988, 360-371.

Sellis, T., N. Rossopoulos, and C. Faloutsos, The R^+-Tree: A Dynamic Index for Multi-Dimensional Objects. Proc. 13th Intl. Conf. on Very Large Data Bases, Brighton, 1987, 507-518.

Shekhar, S., and D.R. Liu, CCAM: A Connectivity-Clustered Access Method for Networks and Network Computations. *IEEE Transactions on Knowledge and Data Engineering 9 (1997)*, 102-119.

Sistla, A.P., O. Wolfson, S. Chamberlain, and S.D. Dao, Modeling and Querying Moving Objects. Proc. 13th Intl. Conf. on Data Engineering, 1997, 422-432.

Svensson, P., and Z. Huang, Geo-SAL: A Query Language for Spatial Data Analysis. Proc. 2nd Intl. Symposium on Large Spatial Databases, Zürich, 1991, 119-140.

Tomlin, C.D., Geographic Information Systems and Cartographic Modeling. Prentice-Hall, 1990.

Van den Bercken, J., B. Seeger, and P. Widmayer, A Generic Approach to Bulk Loading Multidimensional Index Structures. Proc. 23rd Intl. Conf. on Very Large Data Bases, 1997, 406-415.

Vijlbrief, T., and P. van Oosterom, The GEO++ System: An Extensible GIS. Proc. 5th Intl. Symposium on Spatial Data Handling, Charleston, South Carolina, 1992, 40-50.

Waugh, T.C., and R.G. Healey. The GEOVIEW Design: A Relational Data Base Approach to Geographical Data Handling. *Intl. Journal of Geographical Information Systems 1 (1987),* 101-118.

Widmayer, P., Datenstrukturen für Geodatenbanken (Data Structures for Spatial Databases). In: G. Vossen (ed.), Entwicklungstendenzen bei Datenbanksystemen. Oldenbourg, München, 1991, 317-361.

Wolf, A., The DASDBS GEO-Kernel: Concepts, Experiences, and the Second Step. Proc. First Intl. Symposium on Large Spatial Databases, Santa Barbara, 1989, 67-88.

Woodruff, A., and M. Stonebraker, Supporting Fine-grained Data Lineage in a Database Visualization Environment. Proc. 13th Intl. Conf. on Data Engineering, 1997, 91-102.

Worboys, F., A Unified Model for Spatial and Temporal Information. *The Computer Journal 37 (1994)*, 25-34.

Zhou, X., D. J. Abel, and D. Truffet Data Partitioning for Parallel Spatial Join Processing. Proc. 5th Intl. Symposium on Large Spatial Databases, Berlin, 1997, 178-196.

Zimbrao, G., and J. Moreira de Souza, A Raster Approximation for Processing of Spatial Joins. Proc. 24th Intl. Conf. on Very Large Data Bases, 1998, 558-569.