Spatial Database Management Systems

Guillaume Raschia

université de Nantes

Last update : November 28, 2012

[Source: Tutorial Notes from Ralf H. Güting, Fernuniversität Hagen]

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 VLSI design
  - 3D Model of the brain (or someone's brain): medicine
  - 3D Molecule structure: biological research
- Characteristic for the supporting technology: capability of managing **large collections of relatively simple geometric objects**

Terms

- pictorial database system
- image db sys
- geometric db sys
- geographic db sys
- spatial db sys

A database may contain

- A collections of objects in some space with **clear identity, location, extent**

  Spatial database management system

- Raster images of some space

  Image database management system

- From I-DBMS to S-DBMS: Analysis, feature extraction

S-DBMS

A personal view from R.H. Güting:

1. A spatial database system is a database system
2. It offers **spatial data types** (SDT) in its data model and query language
3. It supports SDT in its implementation, providing at least **spatial indexing** and efficient algorithms for **spatial join**

Contents

Models

SDT & Algebras

Querying & Storage

Access Methods

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

Modeling

1. What needs to be represented?
2. Discrete Geometric Bases
3. Spatial Data Types / Algebras
4. Spatial Relations

What needs to be represented?

Two views:

(i) Objects in space: **Entity-based model**

(ii) Space itself: **Field-based model**

Entity-based model

City Nantes

- mayor: J.-M. Ayrault
- population: 283 025
- ...
- city area: 

River Loire

- length: 1 013km
- ...
- route: 

Field-based model

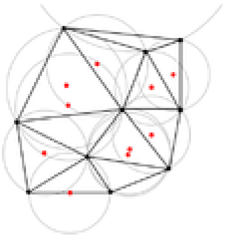
Statement about **every point in space**

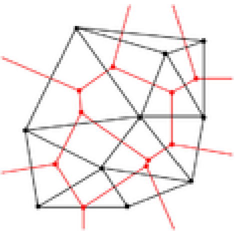
Examples

- Partitions: Administrative boundaries, etc.
- Land use maps: Zoning, Utilization maps
- Continuous functions over space (climate, pollution, etc.)
- Digital Elevation Model (DEM)
    - Triangulated Irregular Network (TIN)

Two Special Cases

Duality of **Delaunay Triangulation** and **Voronoï Diagram**





We consider

1. Modeling **single** objects
2. Modeling **spatially related collections** of objects

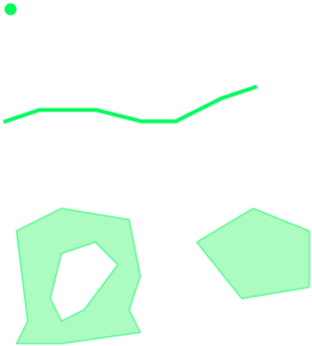
Single Objects

Basic abstractions for modeling spatial objects

0D Point: individual, building, city
Geometric aspect of an object, for which only its **location** in space, but not the extent, is relevant

1D Line (polyline): river, cable, highway
Moving through space, **connections** in space

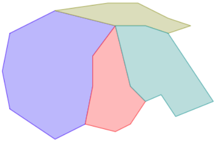
2D Region: country, forest, lake, city, building
Abstraction of an object **with extent**



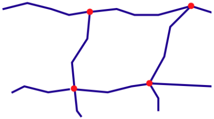
Collections

Basic abstractions for spatially related collections of objects

- Partition
  - digital terrain models
  - land use
  - districts
  - land ownership
  - "environment" of points for Voronoï diagram



- Spatially embedded network (graph)
  - highways, streets
  - railways, public transport
  - rivers
  - electricity, phone



- Others: nested partitions. . .

Discrete Geometric Bases

Organizing the underlying space

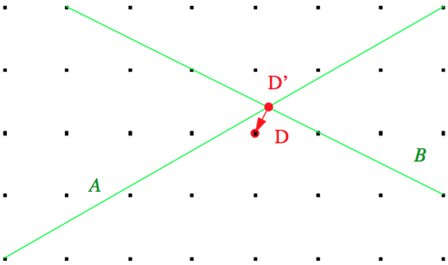
- Is Euclidean geometry a suitable base for modeling?
- Problem:
  - Space is **continuous**

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

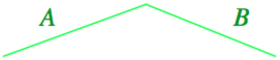
  - Computer numbers are discrete

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

Discrete Geometric Bases (cont'd)



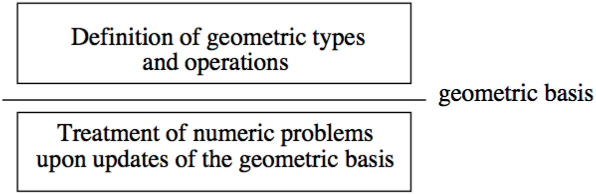
- Is D **on** A ?
- Is D **properly contained** in the area below



Discrete Geometric Bases (cont'd)

Goal

Avoid computation of any new intersection points within geometric operations



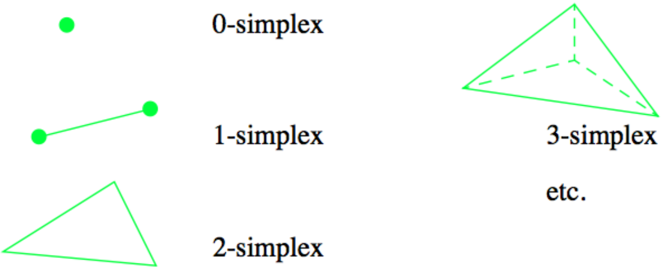
Two approaches:

- **Simplicial complexes**: Frank et al. (1986, 1989)
- **Realms**: Schneider et al. (1993, 1997)

Simplicial Complexes

d-simplex

(from combinatorial topology) Minimal object of dimension d

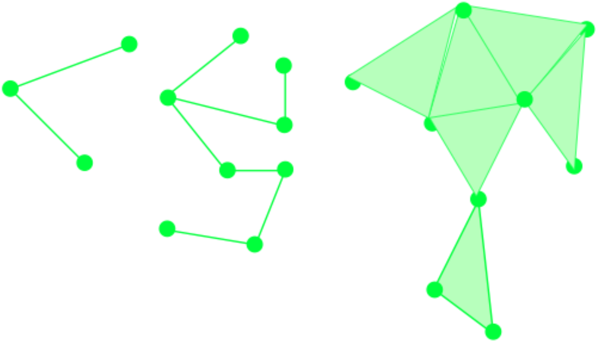


- d-simplex consists of $d+1$ simplices of dimension $d-1$
- Components of a simplex are called **faces**

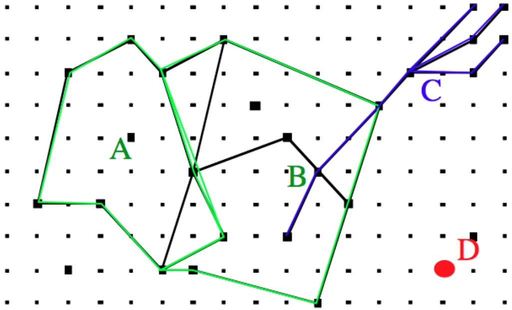
Simplicial Complexes (cont'd)

Definition (Simplicial complex)

Finite set of simplices such that the intersection of any two simplices is a **face**



Realm



About **Realm**: comprehensive description of the geometry (all points and lines) of an application

Realm (cont'd)

Definition (Realm)

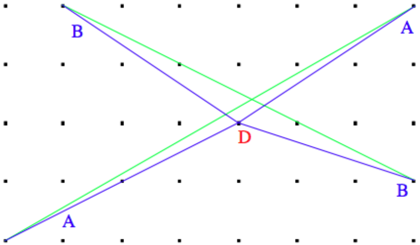
A finite set of points and line segments defined over a grid such that:

(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

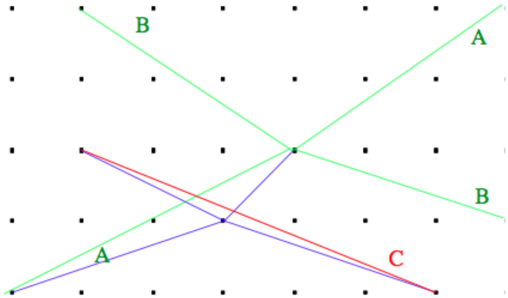
Realm update

Numerical problems are treated **below** the realm layer

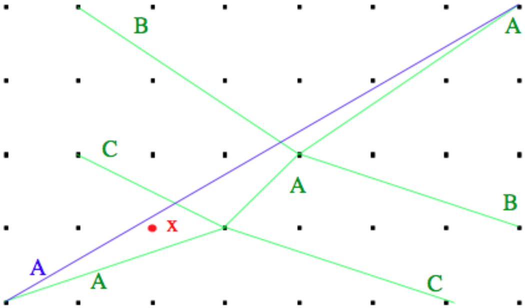
- Data are sets of points and **intersecting** segments
- Problem: Need to insert a segment intersecting other segments
- Basic idea: slightly **distort** both segments!



Realm update (cont'd)



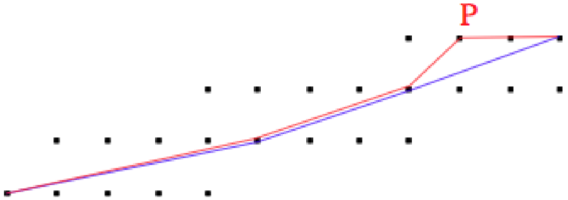
Realm update (cont'd)



Segments can move! Point x is now on the wrong side of A!

Realm update (cont'd)

Solution



- Concept of Greene & Yao (1986): **Redraw** segments within their **envelope**
- Segments are "captured" within their envelope; can never cross a grid point

Overview

Models

SDT & Algebras

Querying & Storage

Access Methods

Spatial Data Type with Algebra

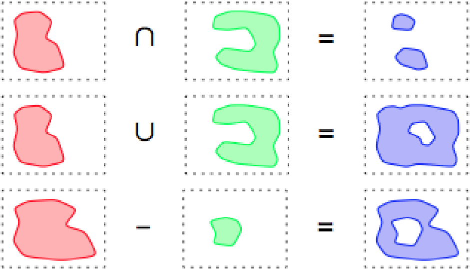
1. Requirements
2. Infinite Point Set Model
3. Finite Point Set Model
4. Topological Model

SDT Requirements

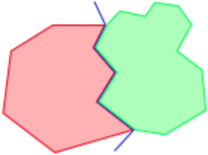
1. **Closure** under set operations over the underlying point sets
2. **Formal definition** of SDT values and functions
3. Definition in terms of **finite precision** arithmetics
4. Support for **geometric consistency** of spatially related objects
5. **Independent** of particular DBMS data model, but cooperating with any

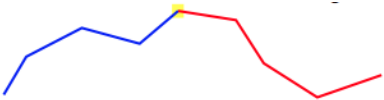
Closure Property and Geometric Consistency

- Complex geometries in real-life application
- Formal requirement



- Constraints over spatially related objects
  adjacent regions meeting lines





Geo-Relational Algebra, Güting (1988)

- **Point set theory**: spatial object is an—infinite—set of points
- Relational algebra viewed as a **many-sorted algebra**
- Relations + (Atomic) Data Types
- **Sorts**: Rel, Num, String, Bool,
  Geo (Point, Ext (Line, Reg (Pgon, Area)))
- Area for partitions
- Example:
  City(*cname*:String, *cextent*:Area, *cpop*:Num)

Operations

- Geometric **predicate**: $=$, $\neq$, intersects?, inside?, neighbor?
- Geometric **set-oriented op.**: intersection, overlay, vertices, voronoi
- Spatial object **constructor**: convex hull, center
- **Scalar operation**: distance, diameter, length, area

...Compared to Requirements

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

Pros & Cons

- Pros:
  - conceptually simple geometric object
  - clear formal definitions
  - implementation: basic data structures + algorithms
- Cons:
  - Simple polygons only
  - No set difference of regions (otherwise, polygon with holes)
  - Intersection op. is embedded into set-oriented op.
  - Numerically critical operations are not managed

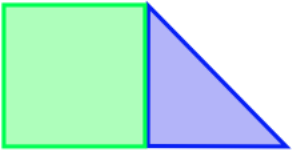
Linear Constraint Approach

- Point set theoretic: satisfaction of FO formulas

$$\{(x, y) \mid y - x \leq 0\}$$

Definition (Convex polygon)

Intersection of a finite set of **half planes**



$$\{(x, y) \mid x \leq 1 \wedge x \geq -1 \wedge y \leq 1 \wedge y \geq -1\}$$

$$\{(x, y) \mid x \geq -1 \wedge y \geq -1 \wedge x + y - 2 \leq 0\}$$

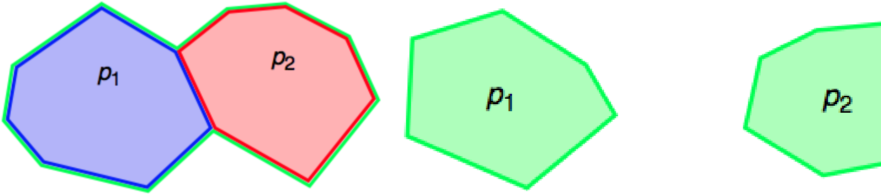
Linear Constraint Approach (cont'd)

Definition (Disjunctive Normal Form)

A non-convex polygon is the union of a finite set of convex polygons

Example

DNF representation: $p_1 \vee p_2$



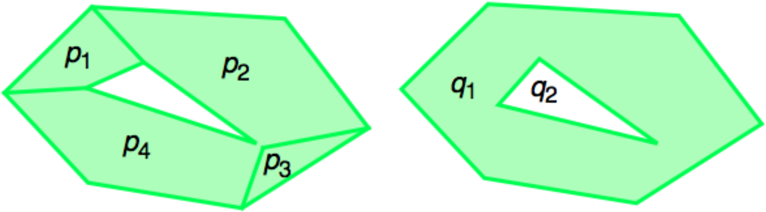
Linear Constraint Approach (cont'd)

Convex with Holes Normal Form (CHNF)

Generalization of DNF

Example

DNF representation: $p_1 \vee p_2 \vee p_3 \vee p_4$
CHNF representation: $q_1 - q_2$



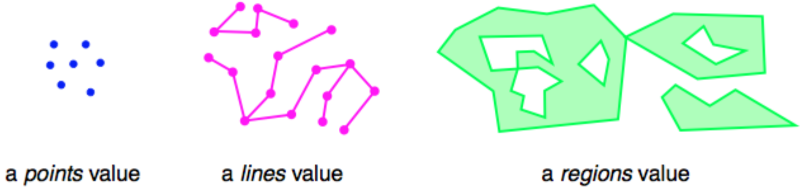
Compared to Requirements

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

RObust Spatial Extension, Güting & Schneider (1995)

ROSE Algebra

- A system of **realm-based** spatial data types
- Objects composed from realm elements
- **Types**: Geo (Points, Ext (Lines, Regions))



ROSE

Complex structure of objects must be handled in definitions

Example

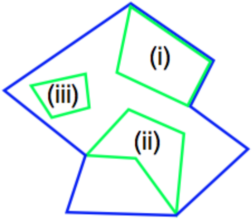
Definition of a **Regions** object

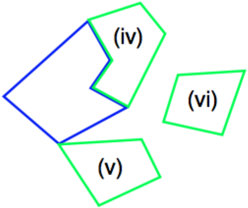
c_2 is ___ c_1

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

c_2 an c_1 are ___

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



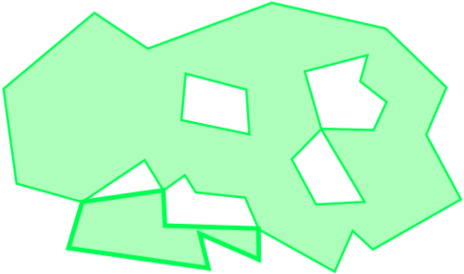


ROSE Example

Definition (R-face)

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

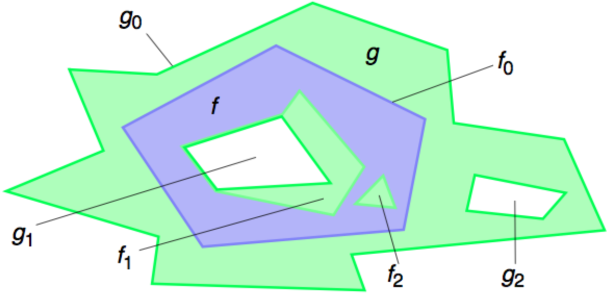
1. $\forall i \in \{1..m\}$, h_i edge-inside c
2. $\forall i, j \in \{1..m\}$, $i \neq j$, h_i and h_j are edge-disjoint
3. "no other cycle can be formed from the segments of f"



ROSE Example (cont'd)

Definition (R-face area-inside predicate)

Let $f = (f_0, F)$ and $g = (g_0, G)$ be two R-faces; f (area-)inside g iff f_0 area-inside g_0 and for each $x \in G$, x area-disjoint f_0 or there exists $y \in F$ such that x area-inside y



ROSE Example (cont'd)

Definition (Regions value)

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

What about operations on Regions?

Definition (Regions inside predicate)

Let F, G be two regions values; F (area-)inside G iff for each $f \in F$, there exists $g \in G$ such that f area-inside g

ROSE Algebra

A collection of well-defined operators on **Geo** objects

- Contains numerically critical operations
  - on_border_of: Points $\times$ Ext $\rightarrow$ Bool
  - border_in_common: Ext $\times$ Regions $\rightarrow$ Bool
- $\cup$ $\cap$ $-$ could be performed due to general structure of values

Compared to Requirements

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

Nice theoretical Model. . .

So what?!

ROSE Cons

- No way to create new geometries (leave the realm closure), e.g. voronoï, center, convex hull, etc.
- Integrating realms into database systems somewhat difficult. Updates of the realm must be propagated to realm-based attribute values in objects

. . . Practically not so satisfying

General Issues with SDT and Algebras

- Soundness and Completeness
- Extensibility
- One or more types (base type "geometry")?
- Operations on sets of DB objects

Spatial Relations

Boolean predicates are the most important operations of spatial algebra

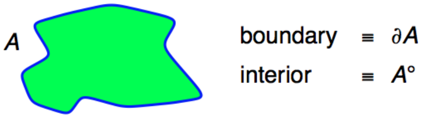
> *Find all objects that satisfy a given relation with a query object*

The Three Categories:

- **Topological**: inside, intersect, adjacent, . . .
  invariant under rotation, translation, scaling
- **Directional**: above, below, north_of, . . .
- **Metric**: distance-based predicate

Topological relations

- At the heart of any algebra
- Completeness criteria?
  - Yes! Egenhofer (1989) and subsequent work
  - Originally for two simple regions only (no holes, connected)



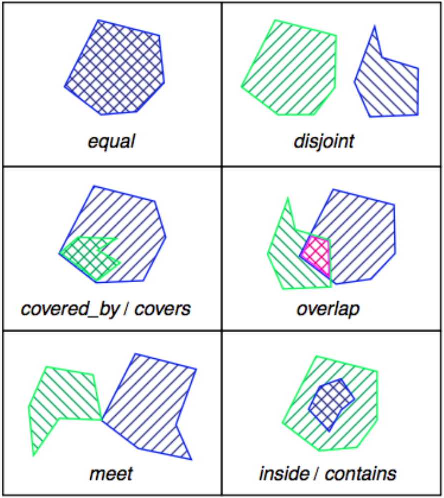
Four intersection sets for two objects A and B

The 4-intersection Model

$\partial A \cap \partial B$	$\partial A \cap B^\circ$	$A^\circ \cap \partial B$	$A^\circ \cap B^\circ$	relationship name
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	*A* and *B* are *disjoint*
$\emptyset$	$\emptyset$	$\emptyset$	$\neq\emptyset$	
$\emptyset$	$\emptyset$	$\neq\emptyset$	$\emptyset$	
$\emptyset$	$\emptyset$	$\neq\emptyset$	$\neq\emptyset$	*A contains B* / *B inside A*
$\emptyset$	$\neq\emptyset$	$\emptyset$	$\emptyset$	
$\emptyset$	$\neq\emptyset$	$\emptyset$	$\neq\emptyset$	*A inside B* / *B contains A*
$\emptyset$	$\neq\emptyset$	$\neq\emptyset$	$\emptyset$	
$\emptyset$	$\neq\emptyset$	$\neq\emptyset$	$\neq\emptyset$	
$\neq\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	*A* and *B meet*
$\neq\emptyset$	$\emptyset$	$\emptyset$	$\neq\emptyset$	*A* and *B* are *equal*
$\neq\emptyset$	$\emptyset$	$\neq\emptyset$	$\emptyset$	
$\neq\emptyset$	$\emptyset$	$\neq\emptyset$	$\neq\emptyset$	*A covers B* / *B covered_by A*
$\neq\emptyset$	$\neq\emptyset$	$\emptyset$	$\emptyset$	
$\neq\emptyset$	$\neq\emptyset$	$\emptyset$	$\neq\emptyset$	*A covered_by B* / *B covers A*
$\neq\emptyset$	$\neq\emptyset$	$\neq\emptyset$	$\emptyset$	
$\neq\emptyset$	$\neq\emptyset$	$\neq\emptyset$	$\neq\emptyset$	*A* and *B overlap*

$4^2 = 16$ combinations, 8 are valid

The 4-intersection Model (cont'd)



Extensions of the 4-intersection Model

- The **9-intersection Model**, Egenhofer (1991):

  Consider also intersections of δA and $\mathring{A}$ with the exterior/complement $\bar{A}$
  - $9^2 = 81$ combinations
  - 8 are valid under the co-dimension constraint (region-region)
  - Many others are valid within line-region
- **Dimension extended method**, Clementini et al. (1993): Consider dimension of the intersection ($\emptyset$, 0D, 1D, 2D in 2D space)
  - $4^4 = 256$ combinations
  - 52 are valid

Alternative Algebras

- For **partitions**:
  - Scholl & Voisard (1983)
  - Erwig & Schneider (1997)
- **Simplex-based** model: Frank et al. (1986)
- **Logic**—axiomatic—model: Cui, Cohn & Randell (1993)

Overview

Models

SDT & Algebras

Querying & Storage

Access Methods

Main challenge

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

Issues

1. Fundamental operations (algebra) for manipulating sets of database objects
2. Graphical input and output
3. Extending query languages

Fundamental Operations (Algebra)

Four classes of operations

- Spatial **selection**
- Spatial **join**
- Spatial function
- Set-oriented operation

Spatial selection

Selection based on a spatial predicate

- Find all cities in Loire Atlantique

```
SELECT cname FROM Cities
WHERE area inside Loire Atlantique
```

- Find all rivers intersecting a query window

```
SELECT rname FROM Rivers
WHERE route intersects Window
```

- Find all big cities no more than 100 km from Nantes

```
SELECT cname FROM Cities
WHERE dist(center, Nantes) <= 100
AND cpop > 500 000
```

Spatial join

Join based on a predicate matching spatial attribute values

- Combine cities with their states

```
SELECT C.cname, S.sname FROM Cities C JOIN States S
USING C.carea inside S.sarea
```

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

```
SELECT R.rname, C.cname FROM Rivers R JOIN Cities C
USING dist(C.center, R.route) < 50
```

Spatial function

- How can we use operations of a spatial algebra computing new SDT values?
  - E.g. intersection: regions $\times$ lines $\rightarrow$ lines
  - In the `SELECT` clause
- For each river going through Loire Atlantique, return the name, the part inside LA and the length of that part

```
SELECT R.rname,
       intersection(R.route, LA) AS part,
       length(part) AS length
FROM Rivers
WHERE R.route intersects LA
```

Set-oriented operation

Manipulate **whole sets of spatial objects** in a dedicated way

- Operation is a **conceptual unit**
- Separation of DBMS set-based engine from SDT set-oriented operations of spatial algebra is—most often—not possible
- For example: overlay, fusion, voronoi
- Here, interfacing the spatial algebra with the DBMS is a mess

Geometry into declarative query language

Three facets

1. Denoting SDT values / input for “constant” values
2. Expressing the four classes of fundamental operations
3. Describing the presentation of query results

Denoting SDT values

Raw geometric description:

```
SELECT cname FROM Cities
WHERE carea intersects 'POLYGON((0,0), (1,0), (1,1), (0,1))'
```

Interactive query:

```
SELECT cname FROM Cities WHERE carea intersects #PICK
```

Variables and named entities:

```
Nantes := 'POLYGON((0,0), (1,0), (1,1), (0,1))'
Loire-Atlantique := #PICK
PdL := SELECT R.area FROM Regions R
         WHERE R.rname='Pays de Loire'
```

The four classes of operations

Straightforward translation

- Spatial Selection
- Spatial Join

Possible translation

- Spatial Function

Do not fit in S-F-W statement

- Set-oriented Operation

Also require to mix **spatial and theme content** in the queries

Presentation of query results

Main issues:

- Render spatial attributes within themes
- Combination—overlay—of query results

Other directions

- **Deductive database** approach, e.g. Abdelmoty, Williams & Paton (1993)
- **Visual querying**: draw a sketch of spatial configurations of interest in a query, e.g. Maingenaud & Portier (1990), Meyer (1992)
- **Virtual reality** exploration: fast navigation through large topographic scenes, e.g. Pajarola et al. (1998)
- Query by **spatial structure**: find all *n*-tuples of objects fulfilling a set of specified relations. Can be viewed as a generalization of spatial join, e.g. Papadias, Mamoulis & Delis (1998)

SDT implementation

SDT with algebra requires DB query engine compliance for

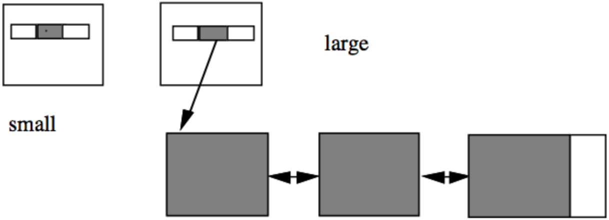
- data structures and
- standalone operations, i.e. do two regions overlap?

Other hot topics

- Use of predicates in—set-based—query processing
  - spatial selection
  - spatial join
- Algorithms for set-oriented operations of spatial algebra

SDT values under the DBMS view

Paged data structure



Example: Polygon data structure as a pair (i, g) of references to

- **Info** part: small, constant size
  ⟨ bbox: `rectangle`, perimeter: `real`, no_vertices: `int` ⟩
- **Exact geometry** part: large, varying size
  ⟨ vertices: `array[1..1,000,000] of points` ⟩

Implementation of SDT operations

2+1 steps process

1. **Filter**: Pre-checking on approximations
2. Looking up stored function values
3. **Refinement**: Use *plane sweep*

Plane sweep

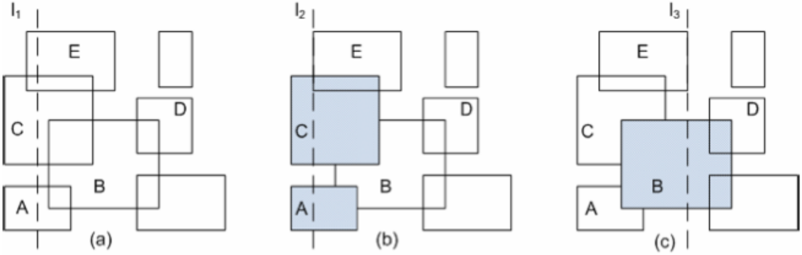
- Family of sweep line—or plane sweep—algorithms
- Key technique from Computational Geometry

Basic idea

A conceptual line is swept across the plane and geometric operations are restricted to local objects to a finite set of *stop points*

Plane sweep example

Given a set of rectangles S; find all pairwise intersections

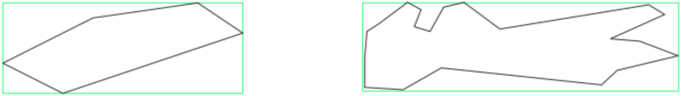


- Example of the active list at ℓ_1: $S_{\ell_1} = \{ACE\}$
- Key observations:
  - Overlap of y-axis projections suffices
  - $2 \cdot |S|$ stop points as left/right vert. edges of rectangles in S
- $O(n \cdot \log n + k)$, k being the number of intersections

SDT support for operations

Internal representation of SDT values contains:

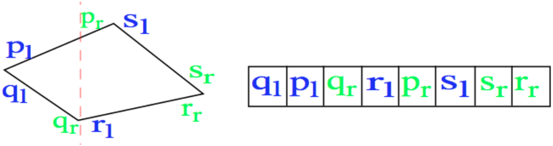
1. Approximation



2. Stored unary function values

   area, length, etc. computed at compile-time or runtime

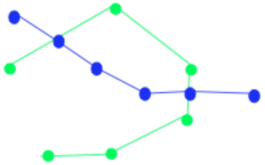
3. Plane sweep sequence

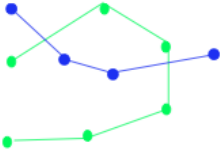


Realm-based SDT

intersection: lines $\times$ lines $\longrightarrow$ points

- Parallel scan of 2 SDT values—half-segments—in $O(n+k)$





- ...vs. Usual plane sweep

- Plane sweep also simplified
  - Only static sweep-event structure needed
  - Güting, de Ridder & Schneider (1995)

Spatial indexing

Definition (Spatial index)

External data structure that organizes space and/or spatial objects to support

- spatial selection
- spatial join
- other spatial operations

Two flavors

1. Dedicated external data structures
2. Map spatial objects into 1D space and use a standard access method (B-tree)

Spatial indexing (cont'd)

Stored objects

- a set of points—**Point Access Methods** (PAM)
- a set of rectangles—**Spatial Access Methods** (SAM)

Operations

insert, delete, member + following query operations

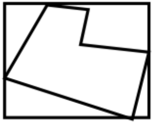
- point query: $x \ni$ point
- range query: $x \cap$ range $\neq \emptyset$
- nearest neighbor: argmin dist(x, obj)
- overlap query: $x \cap y \neq \emptyset$
- containment query: $x \subseteq y$

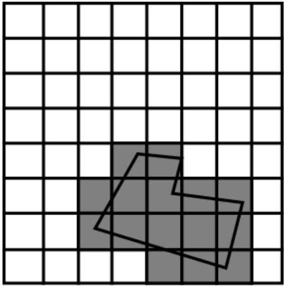
Spatial indexing (cont'd)

Prerequisite

Use of **approximations** in the **Filter** step

- Continuous approximation
- Grid approximation

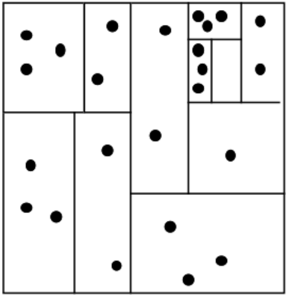




Spatial indexing (cont'd)

Key feature

Space decomposed into **buckets** with associated bucket **regions**



Hundreds of proposals in the literature!

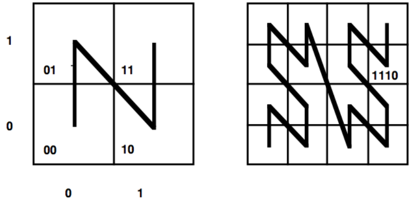
Fractal decomposition of the space

Basic idea

- **Linear order** for the cells of the grid preserving proximity
- Define this order recursively for a grid corresponding to a hierarchical subdivision of space

Example: *Z*-order

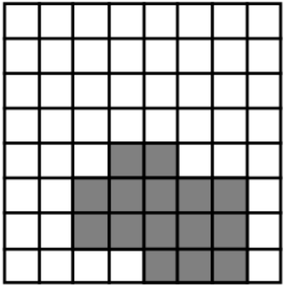
- Morton code (1966), bit interleaving
- **Lexicographical order** of the bit strings

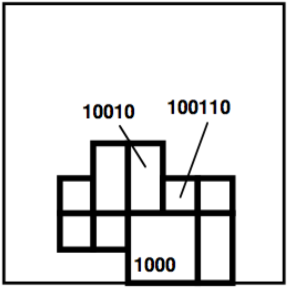


Z-order

Indexing

1. Represent any shape by a set of bit strings, called z-**elements**
2. Put z-elements as spatial keys into a B-**tree**





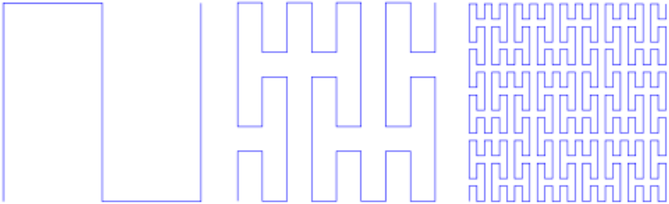
Z-order (cont'd)

Containment query with rectangular window r

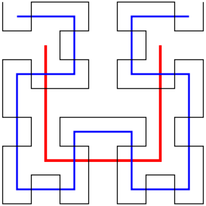
1. Determine Z-elements for r
2. For each Z-element e scan a part of the leaf sequence of the B-tree having e as a **prefix**
3. Check theses candidates for actual containment, avoid duplicate reports

Famous alternative space filling curves

Peano curve (1890)



Hilbert curve (1891)



Point Access Methods

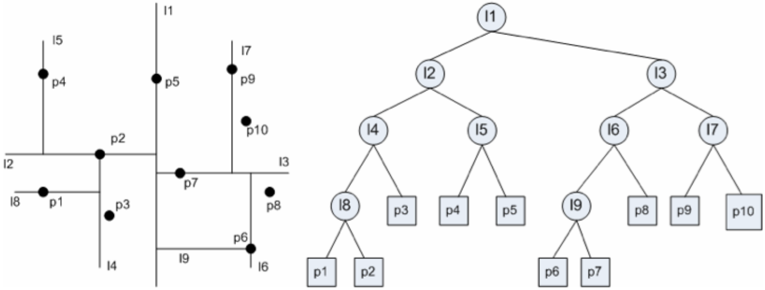
- Long tradition as structures for multi-criteria retrieval
- Tuple $t = (x_1, \ldots, x_k)$ is a point in k-dimensional space

The three categories with—so few—instances:

1. **Main memory** structures
   *k*d-tree, quadtree, multidim. hashing, BD-tree, BSP tree
2. **Grid-based paged** structures
   Fixed grid, Grid file, EXCELL, BANG file
3. **Tree-based paged** structures
   *k*d-B-tree, LSD-tree, *h*B-tree, BV-tree

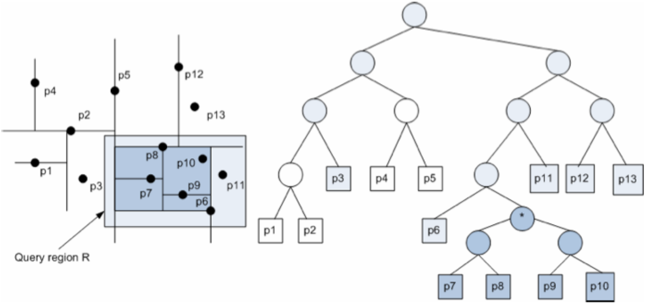
kd-tree —Bentley (1975)

- Basically a binary search tree in k-dimensional space
- Recursive splitting into two regions by means of hyperplanes
- $O(n)$ space and $O(n \cdot \log n)$ time for construction



Operations within *k*d-tree

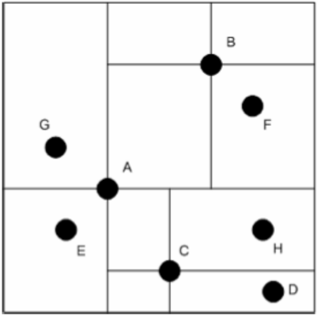
- Insertion and search made easy, deletion is not

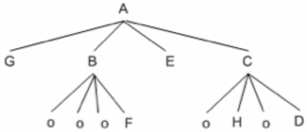


- Extension to adaptive *k*d-tree from Gaede & Günter (1998)
  - Split lines are not required to contain data points and
  - They may have non alternating directions

Quadtree —Finkel & Bentley (1974)

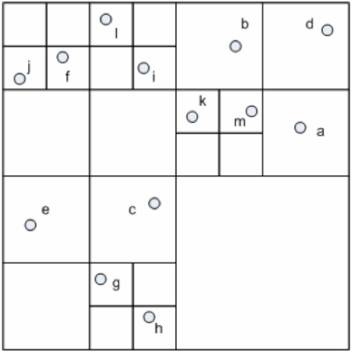
- Closely related to the kd-tree
- Subdivision into 2^k regions
- Large **branching factor** in high dimensional space
- Four quadrants in 2D space: NW, NE, SW, SE





Point-Region quadtree —Samet (1984)

- Regular subdivision into 4 equal-sized squares
- **Unbalanced** structure



PR-quadtree

Results

- Depth d at most $\log \frac{\ell}{a} + \frac{3}{2}$, where
  - ℓ is the side length of initial square
  - a is the smallest distance between any two points
- $O((d+1) \cdot n)$ nodes and $O((d+1) \cdot n)$ construction time

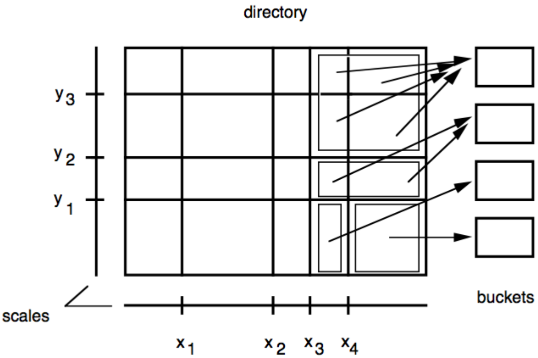
Operations on a quadtree

- Insertion is trivial, deletion is easy
- No sub-linear bound on range searching but performs well
- Specific operation: **Neighbor retrieval** in $O(d+1)$ time
  - Given a node x and a direction δ;
  - Find the node y such that (i) $y \delta x$ and (ii) $d(y) = d(x)$ or y is the deepest δ-neighbor node of x
- Balanced quadtree: any two leaves whose squares are neighbors can differ at most by 1 in depth
- Packaging of a quadtree into a B-tree b.t.w. of Z-order

Grid file

Nievergelt, Hinterberger & Sevcik (1984)

- Requirements: **Directory** and dimensional **scales**
- Paged structure: Capacity of cells are bounded by **page size**



Grid file (cont'd)

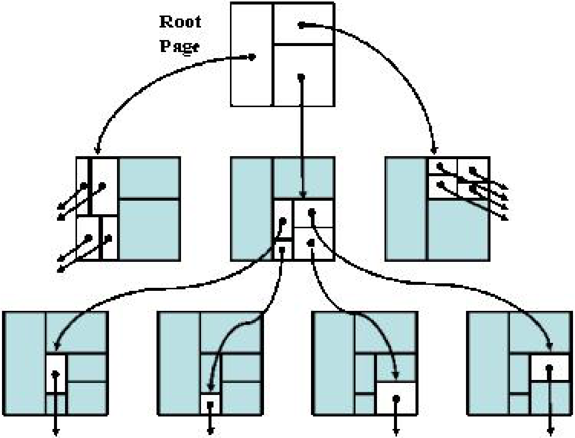
- Fixed grid improvement: Cell splitting whenever page overflow
- Insert brings three cases:
  - no **cell** split
  - cell split and no **directory** split
  - cell split and directory split
- Point query in $O(1)$ and window query in $O(B_w)$, number of overlapping pages

*k*d-B-tree

Robinson (1981)

- Combines adaptive *k*d-tree within *B*-tree features
- Space subdivision as an adaptive *k*d-tree
- Nodes fit disk pages
  - Inner nodes store index entries (dr, pid), with dr a directory rectangle
  - Leaf nodes store data points
- Perfectly balanced structure
- No minimum filling rate for disk pages

*k*d-B-tree (cont'd)



kd-B-tree (cont'd)

Results

- kd-B-tree can be constructed in $O(n/B \log_B n)$ I/Os
- Queries:
  - $O(\sqrt{n/B} + k/B)$ I/Os, a paged version of kd-tree algorithm
- Insertions in $O(\log_B^2 n)$ I/Os
  - First, perform point search to locate matching partition
  - If not full, insert
  - Otherwise split and move half of the entries to a new node
  - Split can propagate up to root
- Deletions are straightforward but may require merging of sibling nodes

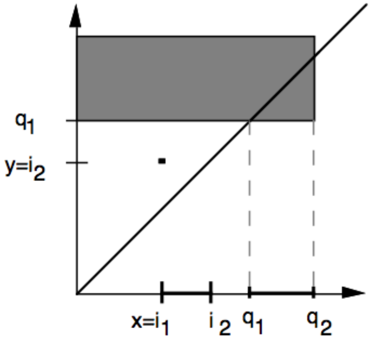
Spatial Index Structures for Rectangles

- Rectangles more difficult than points
- Do not fall into a single cell of a bucket partition
- Three strategies:
  1. Transformation approach
  2. Overlapping bucket regions
  3. Clipping

Transformation approach

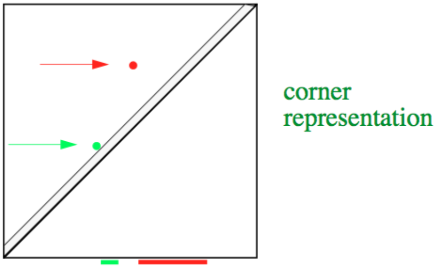
Hinrichs (1985), Seeger & Kriegel (1988)

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



Transformation approach (cont'd)

- Skewed distributions of points

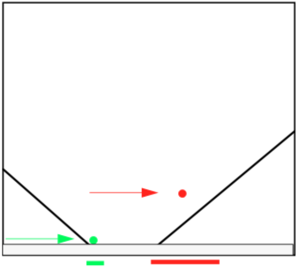


- LSD-tree designed to adapt to such skewed distributions
  - A paged *kd*-tree variant
  - Abandon strict cycling through dimensions
  - Clever paging algorithm keeps external path length balanced

Transformation approach (cont'd)

Center representation

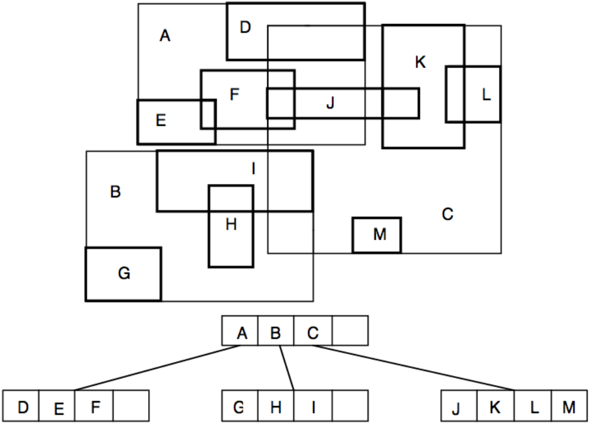
- Rectangle represented by (x, y, x-ext, y-ext)



- For intervals: (x, x-ext)
- Cone-shaped query regions

Overlapping bucket regions

- Prime example: the R-tree, Guttmann (1984)
- Main extension: R^*-tree, Beckmann et al. (1990)



The R-tree

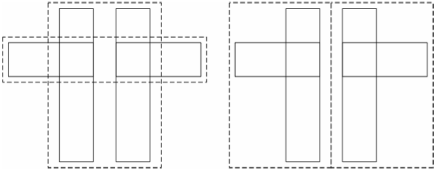
Insertion

- Node selection: minimal directory rectangle enlargment
- Node overflow: binary splitting
  - $m + i$ entries in one node and $M + 1 - m - i$ in the other node, with $0 \leq i \leq M - 2m + 1$, is acceptable
- Worst case: propagation up to the root

The *R*-tree (cont'd)

Splitting strategy

- Requirements:
  - Minimize the total area of the two nodes
  - Minimize the overlapping of the two nodes

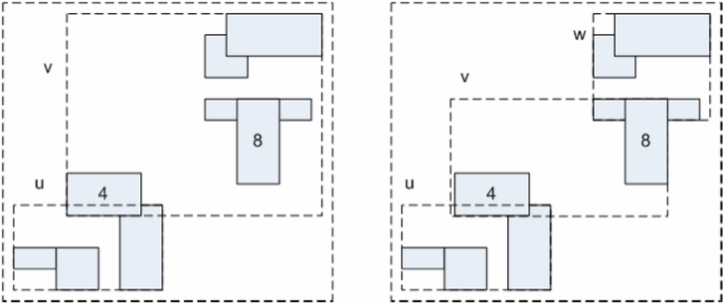


- Algorithms
  - Two seeds + assignment of remaining entries
  - Quadratic splitting maximizes the difference of dead space in both groups
  - Linear splitting minimizes group enlargment

The *R*-tree (cont'd)

*R**-tree extension

- Splitting performed along an axis
- Forced reinsertion
  - Prevent from splitting
  - Reinsert rectangles with the largest dead space in the node



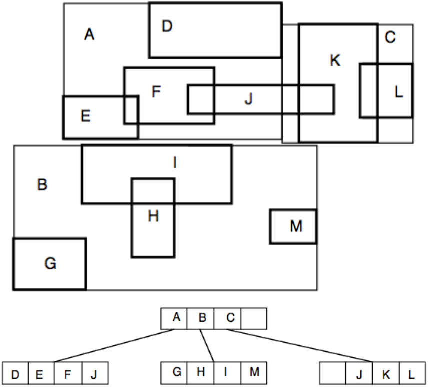
The R-tree (cont'd)

- **Pros**: Spatial object—or key—in a single bucket
- **Cons**: Multiple search paths due to overlapping bucket regions

Clipping

The R^+-tree —Sellis, Rossopoulos & Faloutsos (1987)

- Bucket regions are **disjoint**
- Data rectangles cut into several pieces, when needed



The R^+-tree

- **Pros**: less branching in search
- **Cons**: multiple entries for a single spatial object (not good as a clustering index)