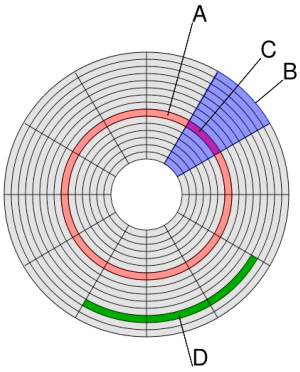
Design of Geographic Information Systems class 8

Thomas Bittner
bittner3@buffalo.edu

Recap: Data storage and access

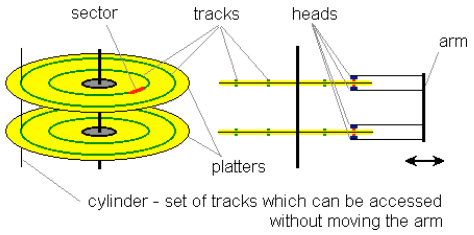


Disk structure:

(A) track
(B) geometrical sector
(C) track sector
(D) **cluster**

A cluster need not be physically contiguous on the disk; it may span more than one track or, if sector interleaving is used, may even be discontiguous within a track.

Storing the data

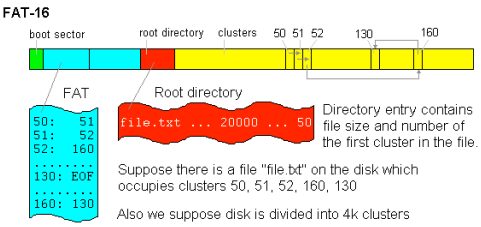


The lecture notes are based on slides by Shashi Shekhar

Secondary Storage Hardware: Disk Drives

- Disk hardware and organization
  - Circular platters with magnetic storage medium
  - Multiple platters are mounted on a spindle
  - Platters are divided into concentric tracks
  - A cylinder is a collection of tracks across platters with common radium
  - Tracks are divided into sectors
  - A sector size may a few kilo-Bytes
- Writing data on a disc
  - Disk heads to read and write
  - There is disk head for each platter (recording surface)
  - A head assembly moves all the heads together in radial direction
  - Spindle rotates at a high speed, e.g. thousands revolution per minute
- Accessing a sector has three major steps:
  - Seek: Move head assembly to relevant track
  - Latency: Wait for spindle to rotate relevant sector under disk head
  - Transfer: Read or write the sector
  - Other steps involve communication between disk controller and CPU

File storage



The FAT table has one entry for each cluster in the partition - it contains the number of the next cluster in the chain.

There is one chain of clusters for each file.

The number of the first cluster of the file is stored in the directory entry for that file, along with the file size, some attributes, and the last modification date.

Space for the directories other than `root` is allocated among the data clusters, just as if they were ordinary files.

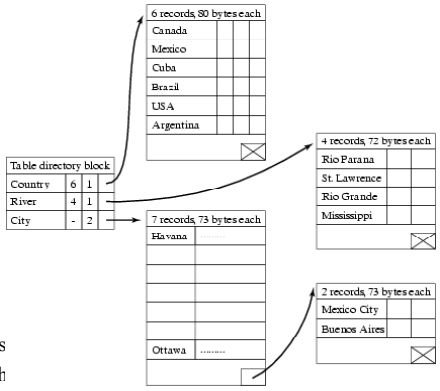
Only the `root` directory has a special location.

Using Disk Hardware Efficiently

- Disk access cost are affected by
  - Placement of data one the disk
  - Fact than seek cost > latency cost > transfer (See Table 4.2, pp. 86)
  - A few common observations follow
- Size of sectors
  - Larger sector provide faster transfer of large data sets
  - But waste storage space inside sectors for small data sets
- Placement of most frequently accessed data items
  - On middle tracks rather than innermost or outermost tracks
  - Reason: minimize average seek time
- Placement of items in a large data set requiring many sectors
  - Choose sectors from a single cylinder
  - Reason: Minimize seek cost in scanning the entire data set.

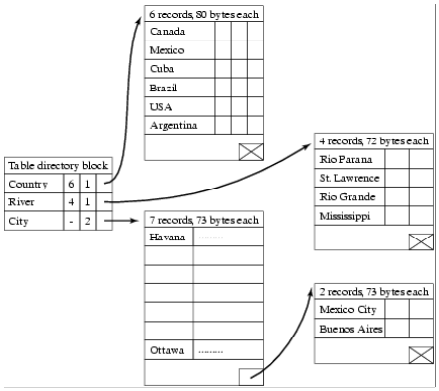
Mapping Records and files to Disk

- Records
  - Often smaller than a sector
  - Many records in a sector
- Files with many records
  - Many sectors per file
- File system
  - Collection of files
  - Organized into directories
- Mapping tables to disk
  - City table takes 2 sectors
  - Others take 1 sector each



File operations (1)

- Find: key value --> record matching key values
- Findnext --> Return next record after find if records were sorted
- Insert --> Add a new record to file without changing file-structure
- Nearest neighbor of a object in a spatial dataset

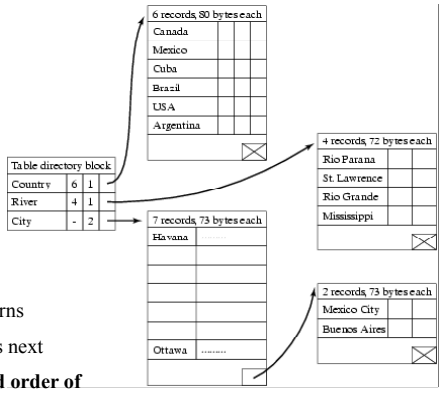


File operations (2)

find(Name = Canada) on Country table returns record about Canada

findnext() on Country table returns ???

findnext() on Country table returns record about Cuba since Cuba is next value after Canada **in the sorted order of Name**

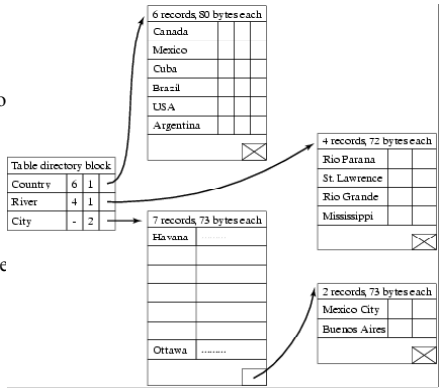


File operations (3)

insert(record about Panama) into Country table

adds a new record

location of record in Country file depends on file-structure

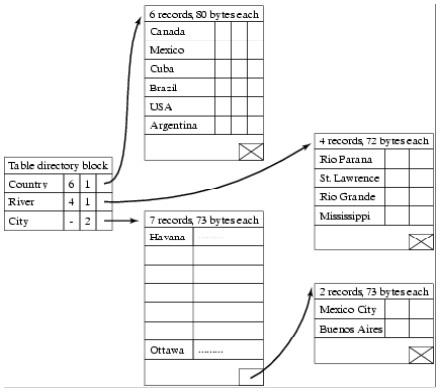


Common File Structures and how they support file operations

- Common file structures
  - Heap or unordered or unstructured
  - Ordered
  - Hashed
  - Clustered
- Basic Comparison of Common File Structures
  - Heap file is efficient for inserts and used for logfiles
    - But find, findnext, etc. are very slow
  - Hashed files are efficient for find, insert, delete etc.
    - But findnext is very slow
  - Ordered file organization are very fast for findnext
    - and pretty competent for find, insert, etc.

4.1.4 File Structures: Heap

- Heap
  - Records are in no particular order
  - insert can simple add record to the last sector
  - find, findnext, nearest neighbor scan the entire files



File Structure : Hash

- Components of a Hash file structure
  - A set of buckets (sectors)
  - Hash function : key value --> bucket
  - Hash directory: bucket --> sector

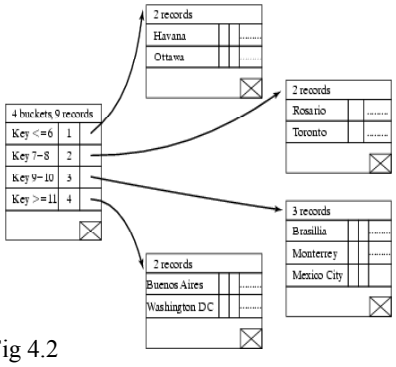


Fig 4.2

File Structures: Ordered

- Ordered
  - Records are sorted by a selected field (Example Fig. 4.3 below)
  - findnext can simply pick up physically next record
  - find, insert, delete may use binary search, is is very efficient

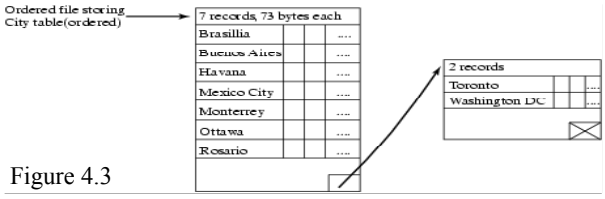


Figure 4.3

Spatial File Structures: Clustering

- Motivation:
  - Ordered files are not natural for spatial data
  - Clustering records in sector by space filling curve is an alternative
  - In general, clustering groups records
    - accessed by common queries
    - into common disk sectors
    - to reduce I/O costs for selected queries
- Clustering using Space filling curves
  - Z-curve
  - Hilbert-curve

Z-Curve

- What is a Z-curve?
  - A space filling curve
  - Connecting points by z-order
    - curves look like combinations of Ns or Zs

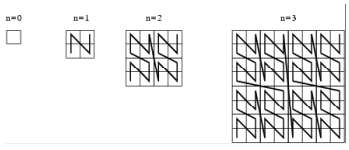
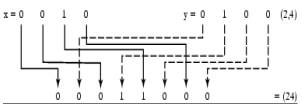


Fig 4.4

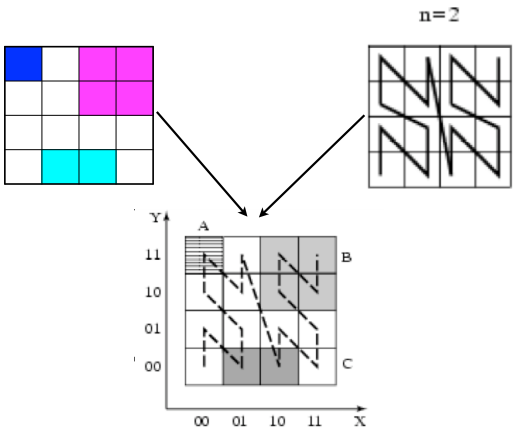
Z-Curve

binary rep. of 2: 0010
binary rep. of 4: 0100

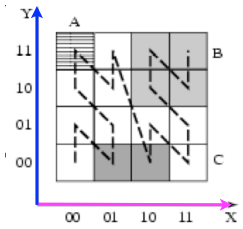
a Z-curve
Generated from interleaving bits
binary representations
of x and y coordinates



Example of Z-values



Example of Z-values



2^3	2^2	2^1	2^0	Obj	Z
0	1	0	1	A	5
0	0	1	0	C1	2
1	0	0	0	C2	8
1	1	0	0	B1	12
…	…	…	…	…	…

How to use indexes

Classifying indexes

- Classification criteria
  - Data-file-structure
  - Key data type
  - assumption of total order on values of indexed attributes

- Primary index
  - Data file ordered by indexed attribute
  - 1 index record per data sector
  - Example: Fig. 4.11
- Secondary index
  - Heap data file
  - 1 index record per data record
  - Example Fig. 4.10

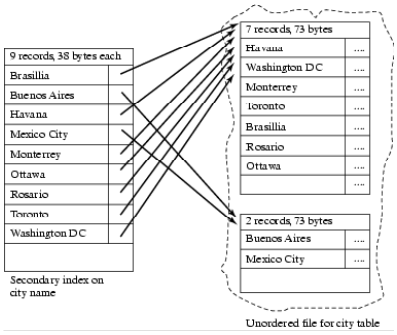
What is a secondary index?

- Concept of a secondary index
  - Heap data file
  - 1 index record per data record
  - auxiliary file to search a data file
  - Example: Fig. 4.10

- index records have
  - key value
  - address of relevant data sector
    - see arrows in Fig. 4.10

- Index records are ordered
  - find, findnext, insert are fast

Fig 4.10

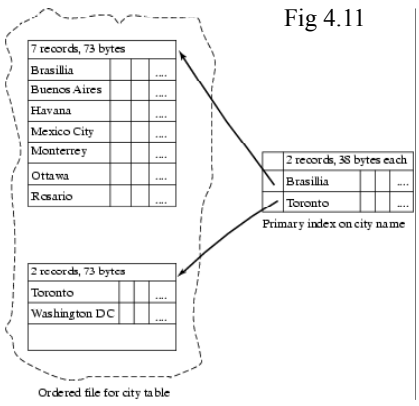


What is a primary index?

- Primary index
  - Data file ordered by indexed attribute
  - 1 index record per data sector
  - Example: Fig. 4.11

- Q? A table can have at most one primary index. Why?

Fig 4.11



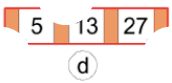
B-tree (ordering based on a total order) Insert Operations

- every node fits into one sector
- every node has at most three data entries (numbers) and three leaf pointers (orange blocks)

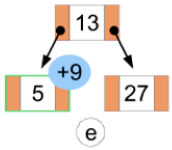
B-tree (ordering based on a total order) Insert Operations

- every node fits into one sector
- every node has at most three data entries (numbers) and three leaf pointers (orange blocks)

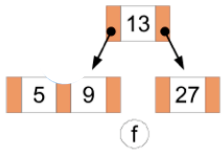
B-tree (ordering based on a total order) Insert Operations

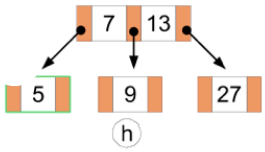


1.Split node
2.distribute data into all three nodes
3.insert new data entry

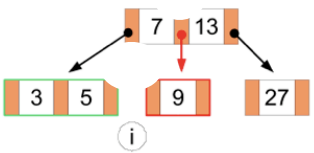


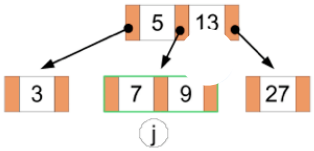
B-tree (ordering based on a total order) Insert Operations



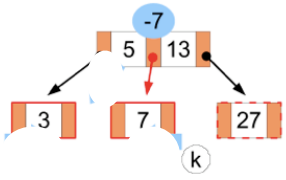


B-tree (ordering based on a total order) Delete Operations





B-tree (ordering based on a total order) Delete Operations



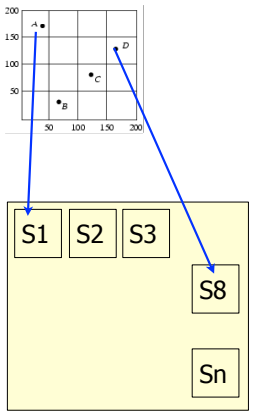
Attribute data types and Indices

- Index file structure depends on data type of indexed attribute
  - Attributes with total order
    - •Example, numbers, points ordered by space filling curves
    - •B-tree is a popular index organization
  - Spatial objects (e.g. polygons)
    - •Spatial organization are more efficient
    - •Hundreds of organizations are proposed in literature
    - •Two main families are Grid Files and R-trees

Ideas behind Grid Files

•Basic idea- Divide space into cells by a grid

- •Store data in each cell in distinct disk sector
- •Efficient for find, insert, nearest neighbor
- •But may have wastage of disk storage space
  - •non-uniform data distribution over space



R-trees

George Kollios

- An external memory tree
- Index nodes and data (leaf) nodes
- All leaf nodes appear on the same level
- Every node contains between m and M entries
- The root node has at least 2 entries (children)
- (only deal with Minimum Bounding Rectangles - **MBR**s)

Example

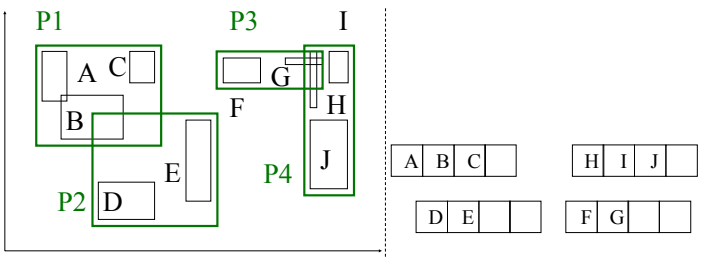
George Kollios

- 4 entries per page (sector); group nearby rectangles to parent MBRs; each group -> disk page (sector)

George Kollios

Example

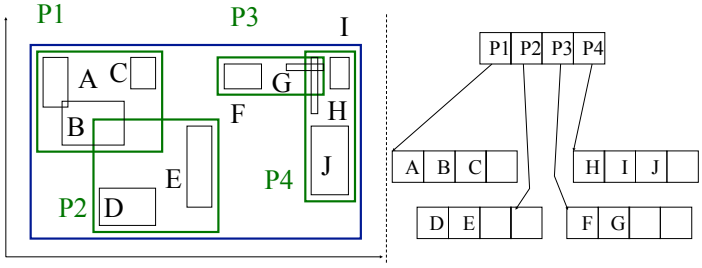
- F=4



George Kollios

Example

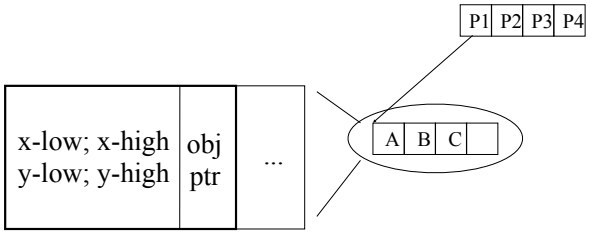
- F=4



R-trees - format of nodes

George Kollios

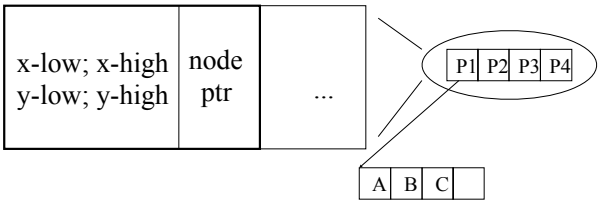
- {(MBR; obj_ptr)} for leaf nodes



R-trees - format of nodes

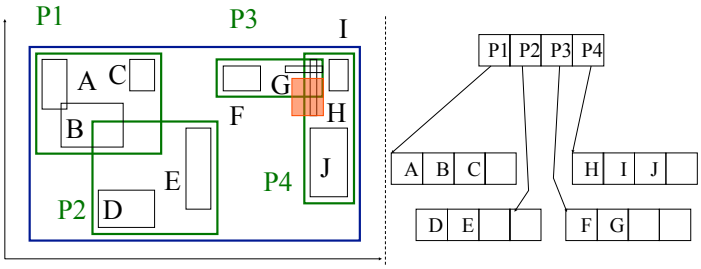
George Kollios

- {(MBR; node_ptr)} for non-leaf nodes



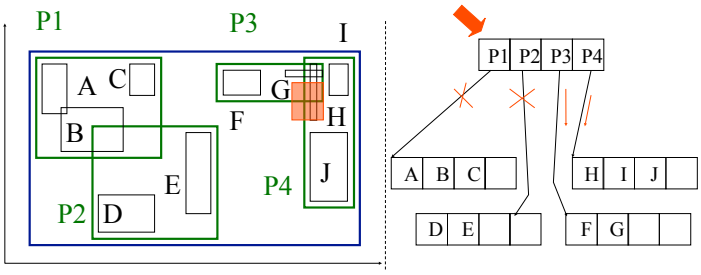
R-trees:Search

George Kollios



R-trees:Search

George Kollios



R-trees:Search

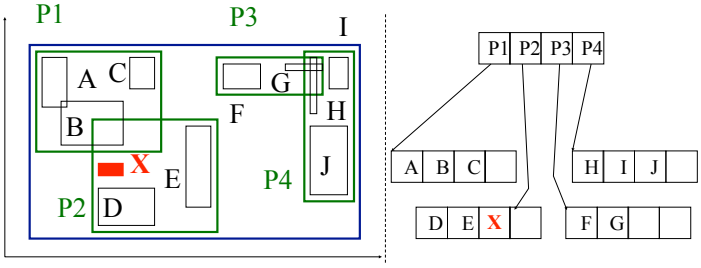
George Kollios

- Main points:
  - every parent node completely covers its 'children'
  - a child MBR may be covered by more than one parent - it is stored under ONLY ONE of them. (ie., no need for duplicate elimination)
  - a point query may follow multiple branches.
  - everything works for **any(?)** dimensionality

R-trees:Insertion

George Kollios

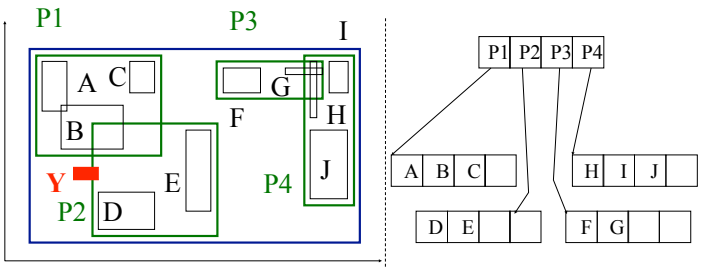
Insert X



George Kollios

R-trees:Insertion

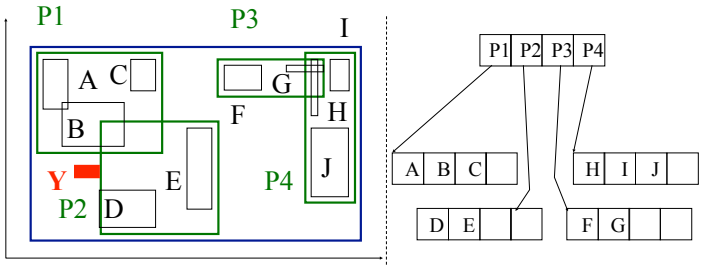
Insert Y



George Kollios

R-trees:Insertion

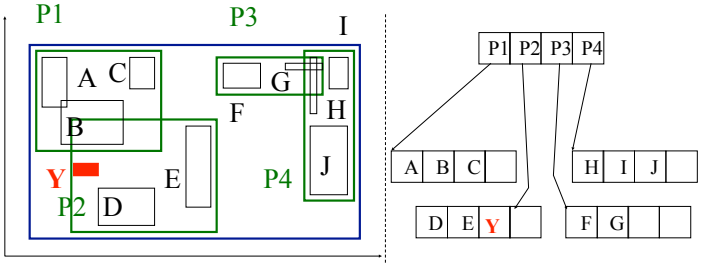
- Extend the parent MBR



George Kollios

R-trees:Insertion

- Extend the parent MBR



George Kollios

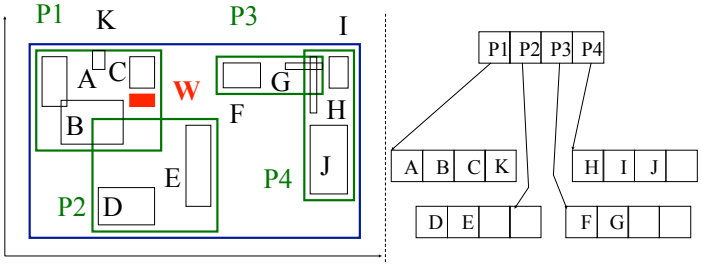
R-trees:Insertion

- How to find the next node to insert the new object?
  - Find the entry that needs the least enlargement to include Y.
  - Other methods ...

George Kollios

R-trees:Insertion

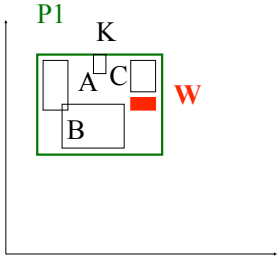
- If node is full then Split : example Insert w



George Kollios

R-trees:Split

- Split node P1: partition the MBRs into two groups.

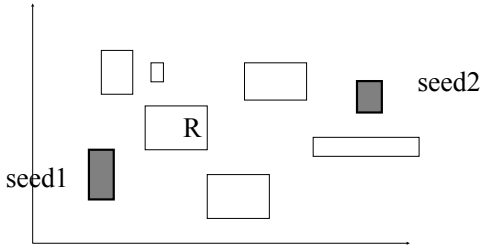


- (A1: plane sweep, until 50% of rectangles)
- A2: 'linear' split
- A3: quadratic split
- A4: exponential split: 2^{M-1} choices

George Kollios

R-trees:Split

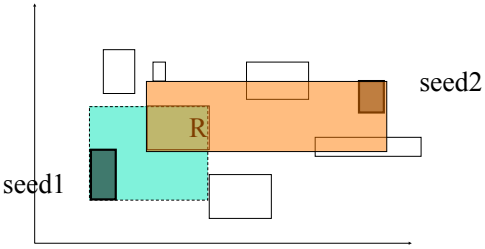
- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'



George Kollios

R-trees:Split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed':
- 'closest': the smallest increase in area



George Kollios

R-trees:Insertion

- Use the **ChooseLeaf** to find the leaf node to insert an entry E
- If leaf node is full, then **Split,** otherwise insert there
  - Propagate the split upwards, if necessary
- Adjust parent nodes

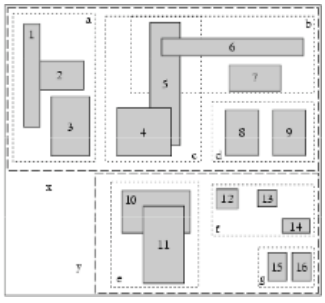
George Kollios

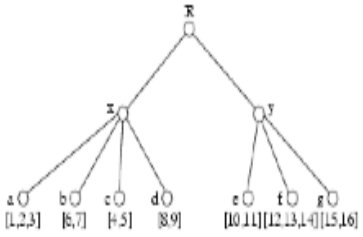
R-Trees:Deletion

- Find the leaf node that contains the entry E
- Remove E from this node
- If underflow:
  - Eliminate the node by removing the node entries and the parent entry
  - Reinsert the orphaned (other entries) into the tree using **Insert**

The lecture notes are based on slides by Shashi Shekhar

Spatial Objects with R-Tree





The lecture notes are based on slides by Shashi Shekhar

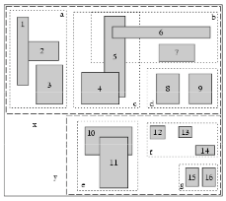
Spatial Objects with R-Tree

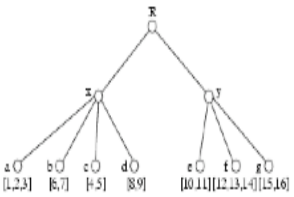
•Properties of R-trees

- •Balanced
- • Nodes are rectangle
  - • child's rectangle within parent's
  - • possible overlap among rectangles!

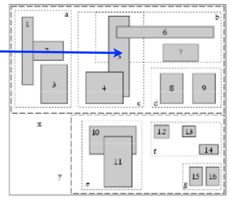
•Implementation of find operation

- •Search root to identify relevant children
- •Search selected children recursively
- •

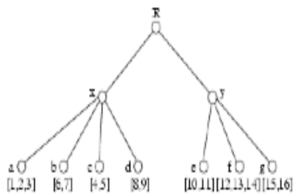




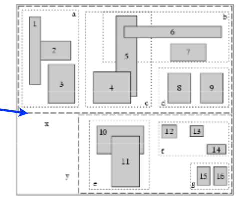
Spatial Objects with R-Tree



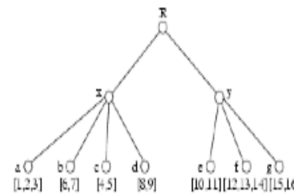
- Example: find record for rectangle 5
  - Root search identifies child x
  - Search of x identifies children b and c
  - Search of b does not find object 5
  - Search of c find object 5



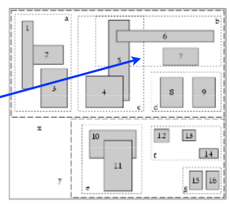
Spatial Objects with R-Tree



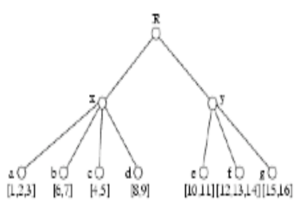
- Example: find record for rectangle 5
  - Root search identifies child x
  - Search of x identifies children b and c
  - Search of b does not find object 5
  - Search of c find object 5



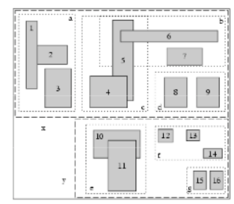
Spatial Objects with R-Tree



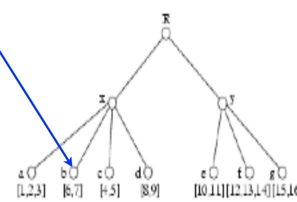
- Example: find record for rectangle 5
  - Root search identifies child x
  - Search of x identifies children b and c
  - Search of b does not find object 5
  - Search of c find object 5



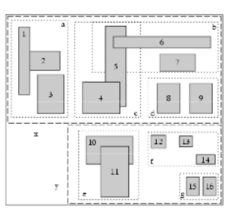
Spatial Objects with R-Tree



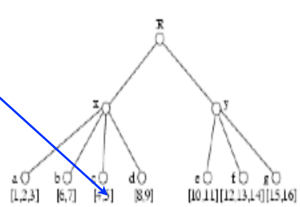
- Example: find record for rectangle 5
  - Root search identifies child x
  - Search of x identifies children b and c
  - Search of b does not find object 5
  - Search of c find object 5

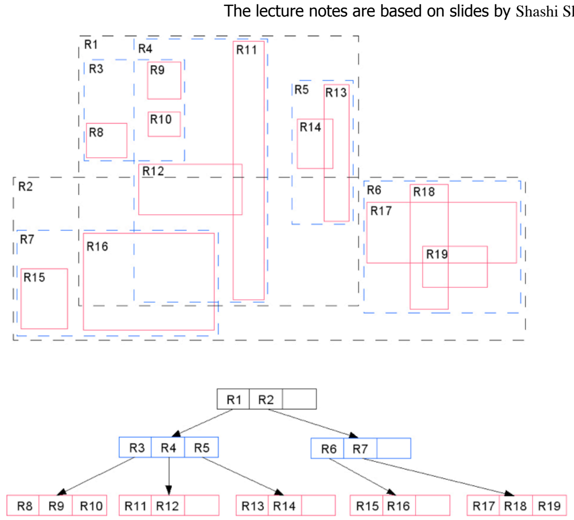


Spatial Objects with R-Tree



- Example: find record for rectangle 5
  - Root search identifies child x
  - Search of x identifies children b and c
  - Search of b does not find object 5
  - Search of c find object 5

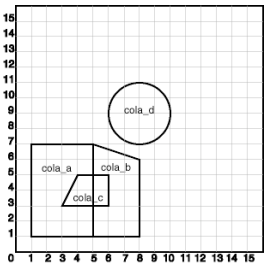




Indexing in Oracle Spatial

Example: Oracle commands for the following operations

- Create a table (COLA_MARKETS) to hold the spatial data
- Insert rows for four areas of interest (`cola_a`, `cola_b`, `cola_c`, `cola_d`)
- Create a spatial index (COLA_SPATIAL_IDX)
- Perform some spatial queries



the areas of interest for four colas

Indexing of Spatial Data

A spatial index, like any other index, provides a mechanism to limit searches, but in this case the mechanism is based on spatial criteria such as intersection and containment on minimal bounding boxes stored in the index.

A spatial index is needed to:

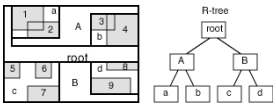
- Find objects within an indexed data space that interact with a given point or area of interest (window query)
- Find pairs of objects from within two indexed data spaces that interact spatially with each other (spatial join)

Oracle Spatial lets you use R-tree indexing (the default) or quadtree indexing, or both. However, the use of quadtree indexes is discouraged, and you are strongly encouraged to use R-tree indexing.

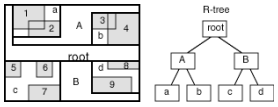
R-Tree Indexing

A spatial R-tree index can index spatial data of up to four dimensions. An R-tree index approximates each geometry by a single rectangle that minimally encloses the geometry (called the minimum bounding rectangle, or MBR)

For a layer of geometries, an R-tree index consists of a hierarchical index on the MBRs of the geometries in the layer.



R-Tree Indexing



An R-tree index is stored in the spatial index table (SDO_INDEX_TABLE)

Geometry Metadata Views

The geometry metadata describing the dimensions, lower and upper bounds, and tolerance in each dimension is stored in a global table owned by MDSYS (which users should never directly update).

Each Spatial user has the following views available in the schema associated with that user:

- USER_SDO_GEOM_METADATA contains metadata information for all spatial tables owned by the user (schema). This is the only view that you can update, and it is the one in which Spatial users must insert metadata related to spatial tables.
- ALL_SDO_GEOM_METADATA contains metadata information for all spatial tables on which the user has SELECT permission.

In **SQL**, a **view** is a virtual table based on the result-set of an **SQL** statement. A **view** contains rows and columns, just like a real table. The fields in a **view** are fields from one or more real tables in the database.

Geometry Metadata Views

The geometry metadata describing the dimensions, lower and upper bounds, and tolerance in each dimension is stored in a global table owned by MDSYS (which users should never directly update).

Each Spatial user has the following views available in the schema associated with that user:

- USER_SDO_GEOM_METADATA contains metadata information for all spatial tables owned by the user (schema). This is the only view that you can update, and it is the one in which Spatial users must insert metadata related to spatial tables.
- ALL_SDO_GEOM_METADATA contains metadata information for all spatial tables on which the user has SELECT permission.

Spatial users are responsible for populating these views. For each spatial column, you must insert an appropriate row into the USER_SDO_GEOM_METADATA view. Oracle Spatial ensures that the ALL_SDO_GEOM_METADATA view is also updated to reflect the rows that you insert into USER_SDO_GEOM_METADATA.

Geometry Metadata Views

- USER_SDO_GEOM_METADATA contains metadata information for all spatial tables owned by the user (schema). This is the only view that you can update, and it is the one in which Spatial users must insert metadata related to spatial tables.

Each metadata view has the following definition:

```
(
  TABLE_NAME    VARCHAR2(32),
  COLUMN_NAME   VARCHAR2(32),
  DIMINFO       SDO_DIM_ARRAY,
  SRID          NUMBER
);
```

The TABLE_NAME column contains the name of a feature table, such as COLA_MARKETS, that has a column of type SDO_GEOMETRY.

Geometry Metadata Views

- USER_SDO_GEOM_METADATA contains metadata information for all spatial tables owned by the user (schema). This is the only view that you can update, and it is the one in which Spatial users must insert metadata related to spatial tables.

Each metadata view has the following definition:

```
(
  TABLE_NAME    VARCHAR2(32),
  COLUMN_NAME   VARCHAR2(32),
  DIMINFO       SDO_DIM_ARRAY,
  SRID          NUMBER
);
```

The COLUMN_NAME column contains the name of the column of type SDO_GEOMETRY. For the COLA_MARKETS table, this column is called SHAPE.

Geometry Metadata Views

- USER_SDO_GEOM_METADATA contains metadata information for all spatial tables owned by the user (schema). This is the only view that you can update, and it is the one in which Spatial users must insert metadata related to spatial tables.

Each metadata view has the following definition:

```
(
  TABLE_NAME    VARCHAR2(32),
  COLUMN_NAME   VARCHAR2(32),
  DIMINFO       SDO_DIM_ARRAY,
  SRID          NUMBER
);
```

The DIMINFO column is a varying length array of an object type, ordered by dimension, and has one entry for each dimension.

DIMINFO

The DIMINFO column is a varying length array of an object type, ordered by dimension, and has one entry for each dimension. The SDO_DIM_ARRAY type is defined as follows:

```
Create Type SDO_DIM_ARRAY as VARRAY(4) of SDO_DIM_ELEMENT;
```

The SDO_DIM_ELEMENT type is defined as:

```
Create Type SDO_DIM_ELEMENT as OBJECT (
  SDO_DIMNAME VARCHAR2(64),
  SDO_LB NUMBER,
  SDO_UB NUMBER,
  SDO_TOLERANCE NUMBER);
```

The SDO_DIM_ARRAY instance is of size *n* if there are *n* dimensions.

DIMINFO contains 2 SDO_DIM_ELEMENT instances for two-dimensional geometries,

3 instances for three-dimensional geometries, and

4 instances for four-dimensional geometries.

DIMINFO

The DIMINFO column is a varying length array of an object type, ordered by dimension, and has one entry for each dimension. The SDO_DIM_ARRAY type is defined as follows:

```
Create Type SDO_DIM_ARRAY as VARRAY(4) of SDO_DIM_ELEMENT;
```

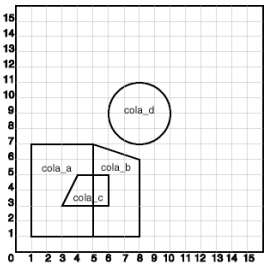
The SDO_DIM_ELEMENT type is defined as:

```
Create Type SDO_DIM_ELEMENT as OBJECT (
  SDO_DIMNAME VARCHAR2(64),
  SDO_LB NUMBER,
  SDO_UB NUMBER,
  SDO_TOLERANCE NUMBER);
```

Each SDO_DIM_ELEMENT instance in the array must have valid (not null) values for the SDO_LB, SDO_UB, and SDO_TOLERANCE attributes.

Creating the Geometry Metadata View for the cola data table

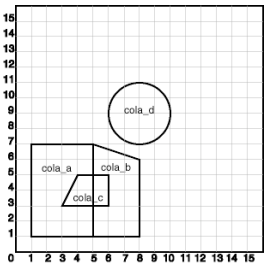
```
INSERT INTO USER_SDO_GEOM_METADATA
  VALUES (
  'cola_markets',
  'shape',
  SDO_DIM_ARRAY(   -- 20X20 grid
    SDO_DIM_ELEMENT('X', 0, 20, 0.005),
    SDO_DIM_ELEMENT('Y', 0, 20, 0.005)
     ),
  NULL   -- SRID
);
```



the areas of interest for four colas

Creating the Spatial Index for the cola data table

```
CREATE INDEX cola_spatial_idx
   ON cola_markets(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```



the areas of interest for four colas

Exercise:

- fill the cola_markets table
- insert the required meta data into the table: USER_SDO_GEOM_METADATA
- create the index on the table cola_markets
- inspect the created tables and understand what they do

Spatial queries

based on slides by Shashi Shekhar

Classifying Spatial objects

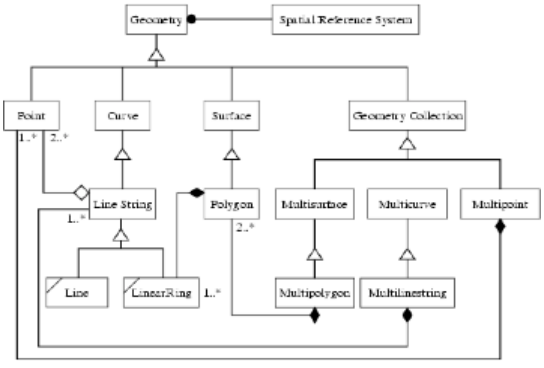
- Spatial objets are spatial attributes of general objects
- Spatial objects are of many types
  - •Simple
    - •0- dimensional (points), 1 dimensional (curves), 2 dimensional (surfaces)
    - •Example given at the bottom of this slide
  - •Collections
    - •Polygon collection (e.g. boundary of Japan or Hawaii), …
    - •See more complete list in Figure 2.2

Spatial Object Types	Example Object	Dimension of representation
Point	City	0
Curve	River	1
Surface	Country	2

based on slides by Shashi Shekhar

Spatial Object Types in OGIS Data Model

Fig 2.2: Each rectangle shows a distinct spatial object type



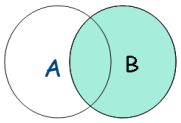
Set operations and spatial relations

based on slides by
Shashi Shekhar

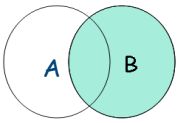
Classifying Operations and Relations on spatial objects in Object Model

- Classifying operations and relations
  - Set based: 2-dimensional spatial objects (e.g. polygons) are sets of points
    - a set operation (e.g. intersection) of 2 polygons produce another polygon

Set Operators

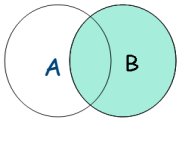


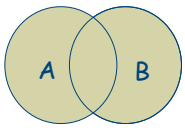
Set Operators



A or B

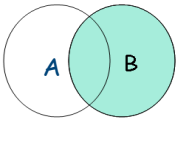
Set Operators

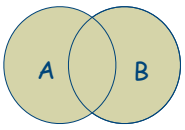




A or B

Set Operators

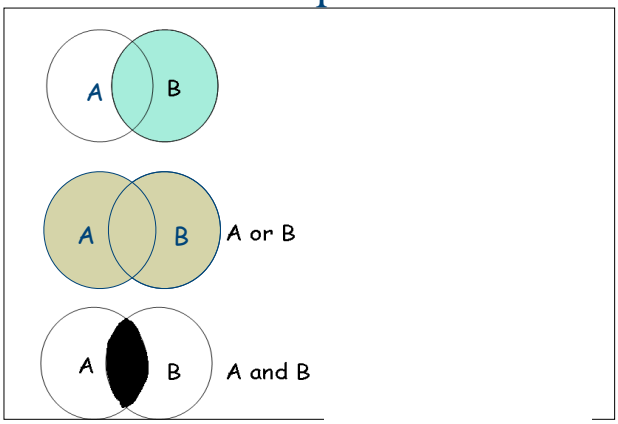




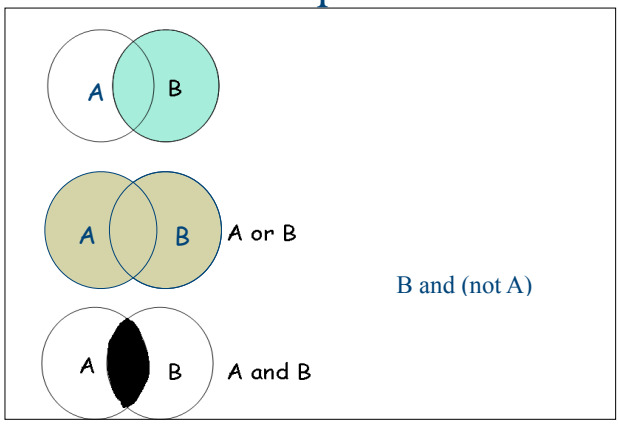
A or B

A and B

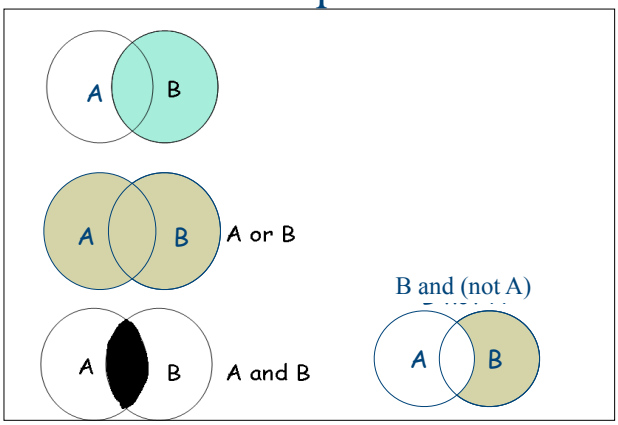
Set Operators



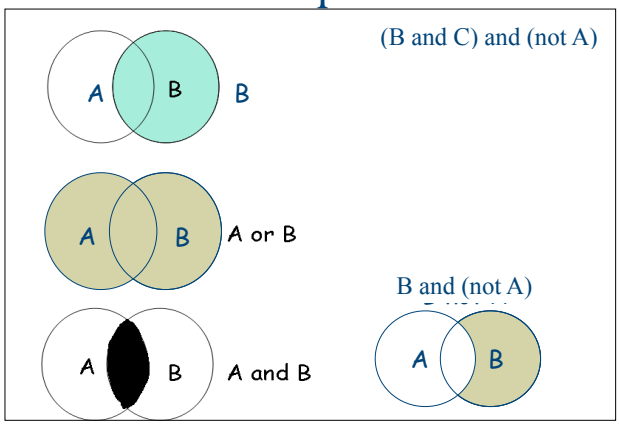
Set Operators



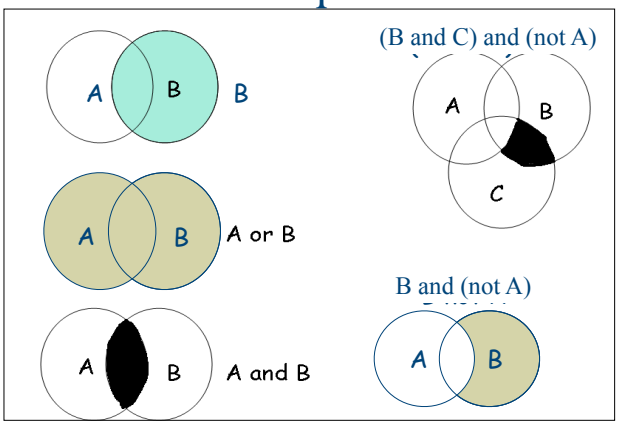
Set Operators



Set Operators



Set Operators



based on slides by
Shashi Shekhar

Classifying Operations and Relations on spatial objects

•Classifying operations and relations
- Set based: 2-dimensional spatial objects (e.g. polygons) are sets of points
  - a set operation (e.g. intersection) of 2 polygons produce another polygon

•Topological relations: Boundary of USA touches boundary of Canada
•Directional relations: New York city is to east of Chicago
•Metric relations: Chicago is about 700 miles from New York city.

Set based operations	Union, Intersection, complement,
Toplogical relations	Touches, Disjoint, Overlap, etc.
Directional relations	East,North-West, etc.
Metric relations	Distance

based on slides by
Shashi Shekhar

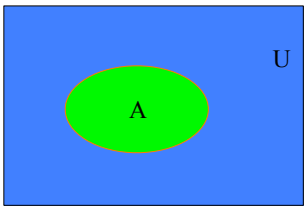
Topological Relationships

- Topological Relationships
  - invariant under elastic deformation (without tear, merge).
  - Two countries which touch each other in a planar paper map will continue to do so in spherical globe maps.
- Topology is the study of topological relationships
- Example queries with topological operations
  - What is the topological relationship between two objects A and B ?
  - Find all objects which have a given topological relationship to object A ?

based on slides by
Shashi Shekhar

Topological Concepts

- Interior, boundary, exterior
  - Let A be an object in a "Universe" U.



Green is A's interior (A^o)

Red is boundary of A (∂A)

Blue –(Green + Red) is A's exterior (A^-)

based on slides by
Shashi Shekhar

Nine-Intersection Model of Topological Relationships

- Many toplogical Relationship between A and B can be specified using 9 intersection model

based on slides by
Shashi Shekhar

Nine-Intersection Model of Topological Relationships

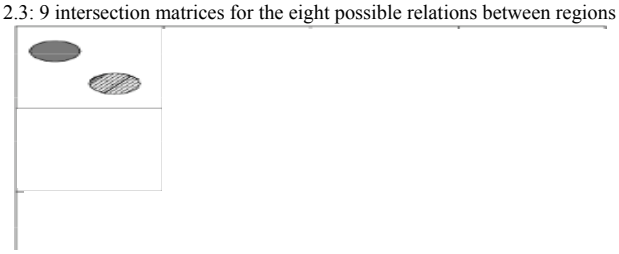
- Many toplogical Relationship between A and B can be specified using 9 intersection model
- Nine intersections
  - A and B are spatial objects in a two dimensional plane.
  - intersections between interior, boundary, exterior of A and B
  - Can be arranged as a 3 by 3 matrix
  - Matrix element take a value of 0 (false) or 1 (true).

$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

Fig 2.3: 9 intersection matrices for the eight possible relations between regions

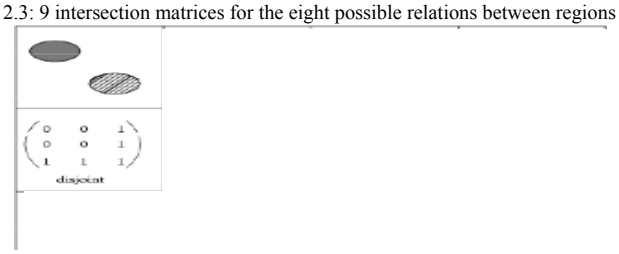


$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

Fig 2.3: 9 intersection matrices for the eight possible relations between regions

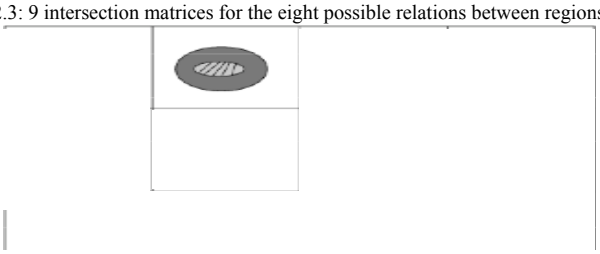


$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

Fig 2.3: 9 intersection matrices for the eight possible relations between regions

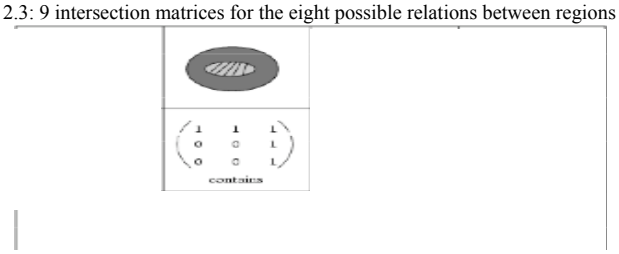


$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

Fig 2.3: 9 intersection matrices for the eight possible relations between regions

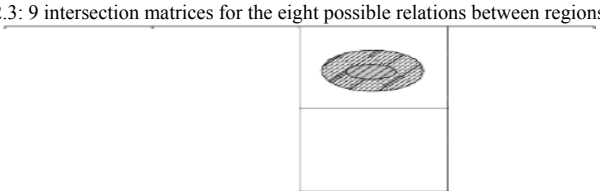


$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

Fig 2.3: 9 intersection matrices for the eight possible relations between regions

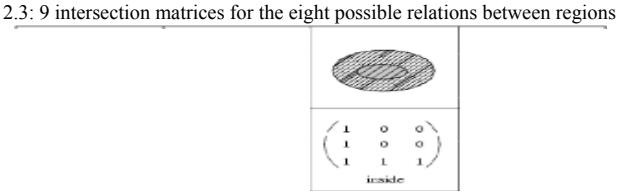


$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

Fig 2.3: 9 intersection matrices for the eight possible relations between regions



$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

Fig 2.3: 9 intersection matrices for the eight possible relations between regions

$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

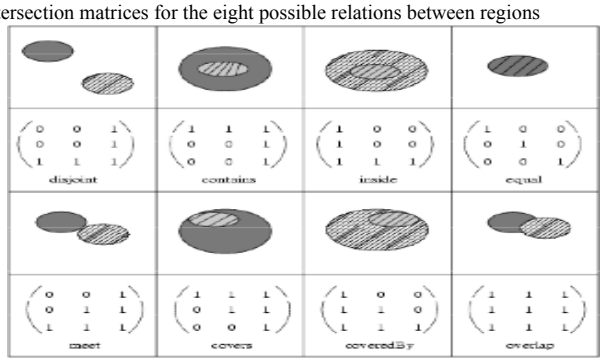
Fig 2.3: 9 intersection matrices for the eight possible relations between regions

$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

based on slides by
Shashi Shekhar

Specifying topological operation in 9-Intersection Model

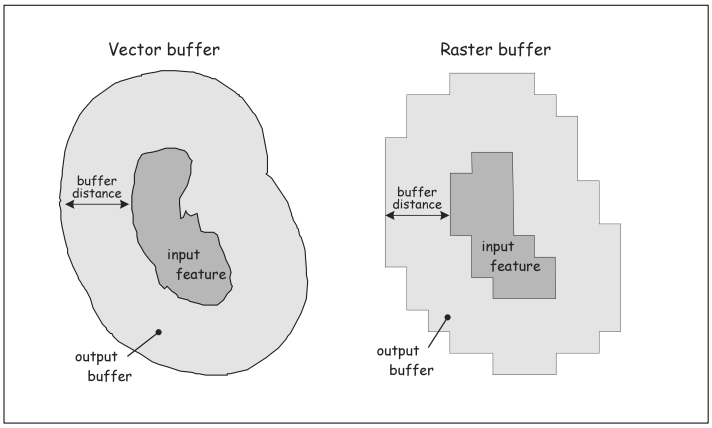
9 intersection matrices for the eight possible relations between regions



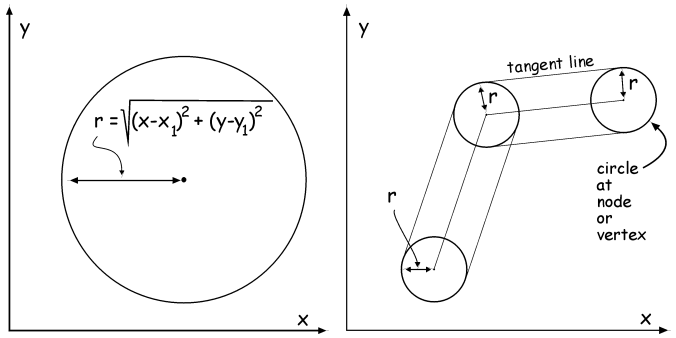
Question: Can this model specify topological operation between a polygon and a curve?

Buffer operations

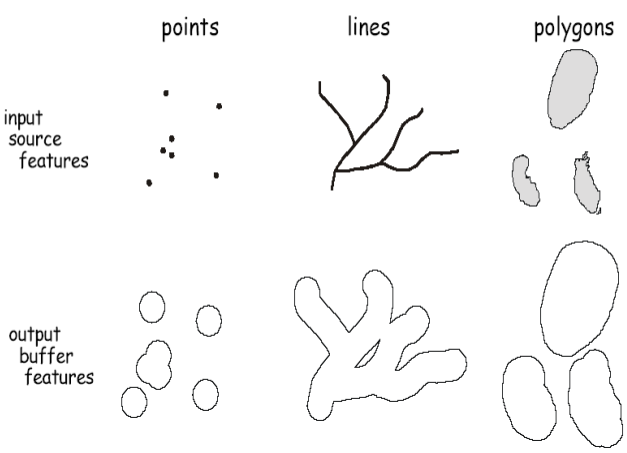
Buffering and other Proximity Functions



Mechanics of Point and Line Buffering



Vector Buffers



Variable-distance buffer: a line buffer is shown with a variable buffer distance, 100 km from main stem of the Mississippi River, 75 km from larger tributaries, and 50 km from remaining tributaries.

river_identifier	buffdist
mississippi	100
missouri	50
arkansas	50
ohio	75
tennessee	75