Spatial Database Concepts

April 23, 2002 Michael Minock

Spatial Database Concepts

Providing constructs to store and query objects (or phenomena) in multi-dimensional space

- Cartographic (map-based) data is often 2-dimensional.
- CAD (Computer Assisted Design) and Virtual World data is often three dimensional and consists of wire frames with associated surface properties, etc.
- Weather (scientific data) is often 3 (or 4) dimensional and might have non-Cartesian (spherical, relativistic) coordinate systems.
- Image Understanding systems often represent the image content in a layering of forms. Pixels, spectral coefficients, line segments (edges), segmented regions, 3-D models.

Discrete vs. Continuous Representation

- Discrete
  - pixel (or grid, or raster) based
  - An *interpolation* (or curve fit) maps to a continuous representation
- Continuous
  - vector-based (or function-based)
  - *discretization* or (function solving) maps to a discrete representation

In the limit, discrete goes to continuous, and, on an actual machine, continuous is handled with discrete types.

Often a single system will describe objects in the same space with continuous and discrete representations.

2-dimension vector-based Spatial data

Geometric primitives:

- points, lines, line segments, paths,...
- squares, rectangles, polygons,...
- circles, arcs, ellipses, splines,...

Operations:

- distance, area, overlap, ...
- rotations, translations, scaling,...

Queries over 2-dimension vector-based Spatial data

- Range Queries:
  Find objects of a given type within a specific geographical area or distance from a particular location.

- Nearest Neighbor Query:
  Finds an object of a particular type that is closest to a given location.

- Spatial Joins or overlays
  Joins objects of two types based on a spatial condition

Example

Never underestimate the power of simple SQL.

`Location(id, north, east)`
`Structure(id, name, type)`

- Range Query:
  Give the houses within 3km of Sparrow Hospital.

- Nearest Neighbor:
  Give the closest hospital to the Johnson's residence.

- Spatial Join:
  Give all the houses within 2 km of a hospital.

Range Query

```
SELECT House.id, House.name
FROM Location AS HLocation,
     Location AS SLocation,
     Structure AS House,
     Structure AS Sparow
WHERE
     Sparow.name = 'Sparrow' AND
     Sparow.id = SLocation.id AND
     House.type = 'House' AND
     House.id = HLocation.id AND
     distance(SLocation.north, SLocation.east,
              HLocation.north, HLocation.east)
      < 3.0;
```


Nearest Neighbor

```
SELECT DISTINCT Hospital.name
FROM Location AS HLocation,
  Location AS SLocation,
  Structure AS House,
  Structure AS Hospital
WHERE
  House.name = 'Johnson\'s residence' AND
  House.id = HLocation.id AND
  Hospital.type = 'Hospital' AND
  Hospital.id = SLocation.id AND
    NOT EXISTS
     (SELECT Hospital2.id
      FROM Structure AS Hospital2,
        Location AS SLocation2
      WHERE
        Hospital2.type = 'Hospital' AND
        Hospital2.id = SLocation2.id AND
        Distance(SLocation2.north, SLocation2.east,
               HLocation.north, HLocation.east)
        <
        Distance(SLocation.north, SLocation.east,
                HLocation.north, HLocation.east));
```

Spatial Join

```
SELECT House.id, House.name
FROM Location AS HLocation,
     Location AS SLocation,
     Structure AS House,
     Structure AS Hospital
WHERE
     Hospital.type = 'Hospital' AND
     Hospital.id = SLocation.id AND
     House.type = 'House' AND
     House.id = HLocation.id AND
     distance(SLocation.north, SLocation.east,
              HLocation.north, HLocation.east)
      < 2.0;
```


Spatial Processor

Some problems may be dealt with through direct representation of an underlying graph. Usually this involves the calculation of the recursive transitive closures. We shall consider this in the upcoming Deductive Database lectures.

Certainly it would be nice to have a set of spatial primitives and polymorphic methods that compute area, intersects, etc. over these vector-based primitives.

And Clearly for scalability we need some special purpose spatial access mechanism.

And the query optimizer must be smart enough to use special indices for spatial access, while maintaining fast B+-tree access to non-spatial data.

In GIS the data model often stores separate layers in separate relations. E.g. one relation for hospitals, another relation for houses. Or one relation for precipitation, another for temperatures...

Indexing

The indexing techniques we saw in the first course were single dimensional.

Most spatial indices use the notion of *spatial occupancy*.

Special Spatial Indices

- Grid Files: Buckets - vector-based
- Area (volume) data: R-Trees and their variants - vector-based
- Regions: Quad-trees (Klinger) - pixel based

Non-hierarchical approached

Fixed-Grid (or cell) method

Fixed-Grid (or cell) method

Divide the region into a grid. And then for all points lying within a grid region, form a linked list (or other structure) of the object addresses.

Large (or boundary) objects will be added to multiple lists.

Good for uniform, non-sparse, static data.

R-Trees

Objects are grouped in close spatial proximity

A hierarchical data structure derived from the B+-tree.

Each node in the tree corresponds to the minimal d-dimensional rectangle that encloses its descendent objects. (our examples will be dimension 2.)

All leaf nodes appear at the same level.

Nodes correspond to disk pages.

The leaf nodes contain pointers that point to the actual geometric objects.

Node entries

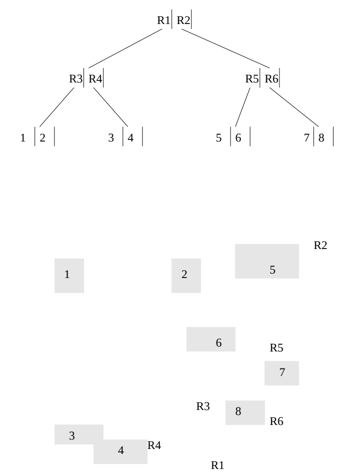
Each entry in a leaf node is of the form (R,O) R is the rectangle that contains the object O. And O is the pointer to the object.

Each entry in an interior node is a 2-tuple of form (R,P) R is the smallest rectangle that spatially contains the children in the Descendents of the node pointed to by P.

Aside from the root, all nodes must have more than some minimum number m entries. An R-tree is said to be of order (m,M) where $m \leq \lceil M/2 \rceil$.

In the following example we have a 2-3 R-Tree over 8-objects.

Example



Insertion, Deletion, and Access

Insertion is made by traversing down the tree, picking the path that requires the minimum amount of enlargement (in terms of area) of the rectangles in each internal node.

If there is overfill of a leaf, node then we require node splitting.

Node splitting is complex in R-Trees, but we in general wish to minimize the total area of the covering rectangles - expensive to actually compute.

Deletion cause an object to be removed from a leaf node

Under-fill precipitates the re-insertion of the objects

Accessing points and regions is straight-forward. But the search may have to follow multiple paths.

Other 'R-Tree like' Options

Packed R-Trees:

For static arrangements of objects

Build the R-tree bottom up, relying on close spatial proximity

R+ trees:

In these overlap is not allowed in the bounding rectangles of interior nodes.

And a rectangle that spans two interior node rectangles will have a leaf node entry under both internal nodes.

Height of tree grows quite a bit.

Keeping nodes greater than 1/2 full is costly.

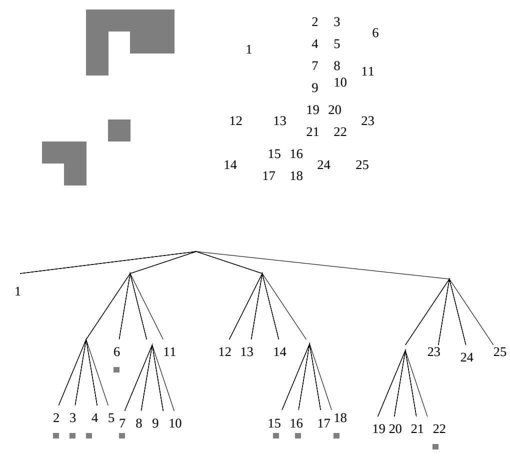
Region Quad-Trees

Successive sub-division of image array into four equal sized quadrants

If quadrants are not homogeneous, then sub-divide recursively.

Stop when the 1 x 1 blocks in the granularity of the grid is reached.

Example



(0 ((1110) 1 (1000) 0) (000 (1101)) ((0001)000)

Comments

Variable resolution method

Very useful for land categories.

Very useful for performing set operations (e.g. Intersect areas of settlement with wild animal territories)

Quad trees are origin sensitive.

Quad Tree complexity theorem: *The average number of nodes in the quad tree representation is* $O(p+q)$ *for a* $2^q x 2^q$ *image with perimeter* p*.*

In our example above $q = 3$ and $p = 28$ (?). So we should have on average 31 nodes. So I drew an easy example to save time :-).

Think about a 1024x1024 grid. A bit map gives 1MG pixels. A quad-tree gives (on average) a 1023 * 4 + 10 nodes.

Spatial Manipulation Through Affine Transformations

Give the set of two-dimensional points in our description of an object, we may apply affine transformations to scale, translate, or rotate the object.

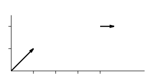
$$Translation: \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$Scaling: \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Rotation: \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example

A line segment, defined by endpoints (0,0) and (1,1) shrinks by 50%, rotates minus 45deg, and moves 4 units in the X direction and 2 units in the Y direction.



The endpoint (0,0) is transformed:

$$\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} .71 & .71 & 0 \\ -.71 & .71 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} .355 & .355 & 4 \\ -.355 & .355 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix}$$

The endpoint (1,1) is transformed:

$$\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} .71 & .71 & 0 \\ -.71 & .71 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} .355 & .355 & 4 \\ -.355 & .355 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4.71 \\ 2 \\ 1 \end{bmatrix}$$

Using Affine Transformations...

Operations are often carried out relative to the centroid of the object.

- Find centroid of object.
- Translate each point defining the object such that centroid is on the origin.
- Apply scaling and rotation operations to each point defining the object.
- Apply translation operations to each point defining the object.
- Re-translate object back to original position of centroid.

Continuous Phenomena

Sometimes we have a continuous function that has a value for each point in space (E.g. Thermospheric Model or Heart models)

In such a situation we often have a set of basis functions paired with a set of coefficients (the data).

The coefficients have been found through one technique or another. Fits to simulated or empirical data.

The query is the solution of the (coefficient weighted) basis functions at a particular point.

Some basis functions:

- Fourier Series
- Wavelets
- Legendre Polynomials
- ...

Truncation schemes save time and space.

Conclusions

Spatial Queries -

- Range queries through R-trees
- Nearest Neighbors through sequences (if necessary) of Range Queries
- Spatial Joins - very expensive, but layering helps with Quad-Trees, and access through R-Trees help.

Operations -

- We showed how to rotate, translate, scale vector-based spatial objects.

Continuous -

- Brief mention of alternative, coefficient-based modeling of physical phenomena.

Graph-based -

- to be dealt with later in Deductive Database lectures.