SYNONYMS
Range Query, Window Query

DEFINITION
Spatial range queries are queries that inquire about certain spatial objects related to other spatial objects within a certain distance. The distance predicate of these queries can be abstracted into the following form
\[ D_{\text{min}} < \text{distance}(A, B) \leq D_{\text{max}}. \]
where \( D_{\text{min}} \) and \( D_{\text{max}} \) are given distance variables and distance is a function to find distance between two objects A and B. The case distance \((A, B) \leq D_{\text{max}}\) can be considered a special case where \( D_{\text{min}} = 0 \).

In a range query at least one attribute is specified as a range instead of a value. An example of a simple range query is finding all cities within 50 miles of Minneapolis.

A more complex query would be finding all lakes within 50 miles of any big city in a state. Here first we have to find all big cities. Then find all the lakes near those cities.

Another form of range query is a window query. Here we are interested in finding objects within a rectangular region (window) in space. This form is particularly useful when we are trying to view or print a section of image. Only objects falling inside a rectangular window are displayed. As the user moves the window area in space, the objects to be displayed change. Some objects are no longer visible, while others become visible.

HISTORICAL BACKGROUND
Spatial data is different from traditional commercial data. Spatial data has a complex structure. A spatial data object may be composed of a single point or several thousands of polygons distributed across space. It is usually not possible to store collections of such objects in a single relational table with a fixed tuple size. Spatial data is also dynamic and require lot of storage space. Spatial operators may not be closed and are more expensive. Retrieval and update of spatial data is usually based on the spatial location of the data object. So a retrieval query on a spatial database often requires fast execution of geometric search based on point or region query. To support such search operations, multidimensional access methods are needed.

Traditional business data is usually zero or one dimensional. But spatial data can be 2 to multi-dimensional. So storing and querying of spatial data needs some modifications to traditional RDBMS. Many approaches have been proposed on how to implement spatial data in traditional RDBMS. Extensions to SQL as well as many special spatial query languages have been proposed to efficiently query the spatial data. The SQL extensions add additional types and operators to deal with spatial data. Spatial data also needs
special indexes like R-trees or Quad trees. Currently most of the commercial databases like Oracle, DB2 etc, support storing of spatial and retrieval of spatial data using some extensions or cartridges. The OpenGIS specifications specify various functions, operators etc for spatial data.

Storing of spatial data is also handled differently. Since it is not efficient to store all the points belonging to a certain object, spatial databases store an approximation of the object, using circles, polygons and other curves that need only a few points for their representation. Minimum Bounding Rectangles (MBRs) are commonly used to approximate spatial data. R-tree and its variants are based on MBRs and are used for providing efficient indexes on spatial data.

**SCIENTIFIC FUNDAMENTALS**

Searching methods in spatial databases can be divided into two main categories, *point access methods* (PAMs) and *spatial access methods* (SAMs). Point access methods have primarily been designed to perform spatial searches on point databases. These points may be embedded in two more dimensions, but don’t have a spatial extension like shape. Spatial access methods on the other hand are able to manage extended objects, such as lines, polygons etc. To index the spatial object, first the object is approximated to object of simpler shape like rectangle or sphere. These uniform objects are then inserted into index. Another way of obtaining simple index entries is to represent the shape of each data object as the geometric union of simpler shapes. This approach is called *decomposition*. R-tree using MBR is the most popular spatial access method.

In spatial queries we are interested in finding objects which satisfy certain relations. Spatial relations are divided into 3 main categories

- Topological – eg disjoint, meet, overlap, covered by
- Directional – eg north, south, south east
- Distance – eg near, far

Distance relations directly translate to spatial range queries. But the other relations topological and directional can also be mapped to range queries.

In order to define distance relations between objects we will start from distances between points.

The distance between two points \( pp\text{-dist}(p_i,q_j) \) \( (p_i \in p) \) and \( q_j \) \( (q_j \in q) \) is defined according to the Euclidean metric: \( pp\text{-dist}(p_i,q_j) = \sqrt{((p_i_x - q_j_x)^2 + (p_i_y - q_j_y)^2)} \) (where \( p_i_x \) is the x-coordinate of point \( p_i \), \( p_i_y \) is the y-coordinate of point \( p_i \), and so on).

Next step is to define the distance between point \( p_i \) and object \( q \) \( (po\text{-dist}) \) as the minimum distance of \( p_i \) from any point of \( q \): \( po\text{-dist}(p_i,q) = \min(pp\text{-dist}(p_i,q_j), \forall q_j \in q) \).

Finally the distance from object \( p \) to object \( q \) is defined as the maximum of all \( po\text{-dist} \): \( oo\text{-dist}(p,q) = \max(po\text{-dist}(p_i,q), \forall p_i \in p) \).

Using the above definitions of distances we define the qualitative relation *near* as: \( near(p,q,k) \equiv oo\text{-dist}(p,q) \leq k \), that is, *all* points of \( p \) must be within \( k \) distance from some point of object \( q \) (Figure 1).
Conjunctions of spatial relations can be considered in a similar manner. Figure 2 shows configuration corresponding to relation northeast \((p,q) \land \text{near}(p,q,k)\).

All these spatial relations can be implemented using search algorithms on MBRs. Table 1 shows how spatial relations between objects can be mapped into constraints of MBR coordinates.
Table 2, shows what constraints are needed for the intermediate nodes when searching an R-tree to find matching MBRs.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Constraints on the p'₁₁₋ₓ, p'₁₋ₓ, p'ₚ₋ₓ, p'ₚ₋ₓ parameters with respect to the reference MBR q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>east(p,q)</td>
<td>p'₁₋ₓ ≥ q'ₓₓ</td>
</tr>
<tr>
<td>northeast(p,q)</td>
<td>(p'₁₋ₓ ≥ q'ₓₓ) ∧ (p'₁₋ₓ ≥ q'ₓₓ)</td>
</tr>
<tr>
<td>meet(p,q)</td>
<td>(p'₁₋ₓ ≤ q'ₓₓ) ∧ (p'₁₋ₓ ≥ q'ₓₓ) ∧ (q'₁₋ₓ ≤ q'ₓₓ) ∧ (p'₁₋ₓ ≥ q'₁₋ₓ)</td>
</tr>
<tr>
<td>inside(p,q)</td>
<td>(q'₁₋ₓ &lt; p'₁₋ₓ) ∧ (p'₁₋ₓ &lt; q'₁₋ₓ) ∧ (q'₁₋ₓ &lt; p'₁₋ₓ) ∧ (q'₁₋ₓ &lt; q'₁₋ₓ)</td>
</tr>
<tr>
<td></td>
<td>(q'₁₋ₓ &lt; p'₁₋ₓ) ∧ (q'₁₋ₓ &lt; q'₁₋ₓ) ∧ (q'₁₋ₓ &lt; q'₁₋ₓ) ∧ (q'₁₋ₓ &lt; q'₁₋ₓ)</td>
</tr>
<tr>
<td>near(p,q,k)</td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td>about(p,q,k)</td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td>near(p,q,k) ∧ northeast(p,q)</td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (q'₁₋ₓ,k ≤ p'₁₋ₓ ≤ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td>meet(p,q) ∧ east(p,q)</td>
<td>(p'₁₋ₓ ≤ q'₁₋ₓ) ∧ (p'₁₋ₓ ≤ q'₁₋ₓ) ∧ (p'₁₋ₓ ≤ q'₁₋ₓ)</td>
</tr>
</tbody>
</table>

Table 2 Constraints for the retrieval of spatial relations using MBRs.

Table 2, shows what constraints are needed for the intermediate nodes when searching an R-tree to find matching MBRs.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Constraints for the intermediate Nodes P to be Searched with respect to the reference MBR q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>east(p,q)</td>
<td>p'₁₋ₓ ≥ q'ₓₓ</td>
</tr>
<tr>
<td>northeast(p,q)</td>
<td>(p'₁₋ₓ ≥ q'ₓₓ) ∧ (Pₓₓ ≥ q'ₓₓ)</td>
</tr>
<tr>
<td>meet(p,q)</td>
<td>(P₁₋ₓ ≤ q'₁₋ₓ) ∧ (Pₓₓ ≥ q'₁₋ₓ) ∧ (Pₓₓ ≤ q'₁₋ₓ) ∧ (Pₓₓ ≥ q'₁₋ₓ)</td>
</tr>
<tr>
<td>inside(p,q)</td>
<td>(P₁₋ₓ &lt; q'₁₋ₓ) ∧ (Pₓₓ &gt; q'₁₋ₓ) ∧ (Pₓₓ &lt; q'₁₋ₓ) ∧ (Pₓₓ &gt; q'₁₋ₓ)</td>
</tr>
<tr>
<td></td>
<td>(P₁₋ₓ &lt; q'₁₋ₓ) ∧ (Pₓₓ &gt; q'₁₋ₓ) ∧ (Pₓₓ &lt; q'₁₋ₓ) ∧ (Pₓₓ &gt; q'₁₋ₓ)</td>
</tr>
<tr>
<td></td>
<td>(P₁₋ₓ &lt; q'₁₋ₓ) ∧ (Pₓₓ &gt; q'₁₋ₓ) ∧ (Pₓₓ &lt; q'₁₋ₓ) ∧ (Pₓₓ &gt; q'₁₋ₓ)</td>
</tr>
<tr>
<td>about(p,q)</td>
<td>(P₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k) ∧ (Pₓₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(P₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k) ∧ (Pₓₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(P₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k) ∧ (Pₓₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td>near(p,q,k)</td>
<td>(P₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k) ∧ (Pₓₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(P₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k) ∧ (Pₓₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td></td>
<td>(P₁₋ₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k) ∧ (Pₓₓ ≤ q'₁₋ₓ,k) ∧ (Pₓₓ ≥ q'₁₋ₓ,k)</td>
</tr>
<tr>
<td>meet(p,q) ∧ east(p,q)</td>
<td>(P₁₋ₓ ≤ q'₁₋ₓ) ∧ (Pₓₓ &gt; q'₁₋ₓ) ∧ (Pₓₓ ≥ q'₁₋ₓ)</td>
</tr>
</tbody>
</table>

Table 2 Constraints for intermediate nodes of R-trees.

But since MBR are an approximation to the object not actual object we might have the case that the MBR satisfies one or more of the spatial relations while the actual object doesn’t. Fig 3 shows an example where the MBRs meet the criteria ‘not near (p, q, k)” but the actual object doesn’t.
For this reason spatial queries use an additional refinement step where each MBR candidate that matched is examined using computation geometry techniques and then false hits are eliminated.

In general, the processing of a query of the form "find all objects p that satisfy a given spatial relation with respect to object q" using R-trees involves the following steps:

1. Starting from the top node, exclude the intermediate nodes P which could not enclose MBRs that satisfy the spatial relation and recursively search the remaining nodes. This procedure involves Table 2.

2. Among the leaf nodes retrieved, select the ones that satisfy the spatial relation. This procedure involves Table 1.

3. If necessary, follow a refinement step for the selected MBRs.

Other access methods for spatial data include

- **K-d-tree**: k-d-tree is a binary search tree that represents a recursive division of the universe into subspaces by means of (d-1)-dimensional hyperplanes. The hyperplanes are iso-oriented and their direction alternates between the d possibilities. K-d-tree can only handle points, so polygons are represented by their centroids.

- **Quadtree**: The quadtree with its many variants is very similar to k-d-tree. The term *quadtrees* usually refers to the two-dimensional variant but the basic idea applies to arbitrary dimensions. Like k-d-tree the quadtree decomposes the universe by means of iso-oriented hyperplanes. But quadtrees are not binary trees. Interior nodes of a quadtree have $2^d$ dimension, each corresponding to an interval-shaped partition of subspace. For d=2, for example, each interior node has four descendants, each corresponding to a rectangle. These rectangles are usually referred to as NW, NE, SW and SE quadrants.

Searching in a quadtree is similar to searching in an ordinary binary search tree. At each level, one has to decide which of the four subtrees need to be included in the future search. In case of a point query, typically only one subtree satisfies, whereas for range queries there are often several.

**KEY APPLICATIONS**
Spatial databases have applications in the area of Geographic Information Systems (GIS), Computer Aided Design (CAD), and Computer Vision. Spatial range queries are needed in each of these fields to identify objects.

Example of a GIS system is keeping land records for a city or county. A use case in such setting is as follows. A new private school is to be built in an area. In such case a hearing is needed to get approval for building the school. All the property owners within a certain distance (say 2 miles) of the development area needs to be notified about this hearing.

This is a fairly complex spatial query. First the boundary of land (object p) to be developed needs to be identified. Then all properties (land parcels) within 2 miles of object p need to be found. Then the owners and addresses of owners for these properties need to be found and letters sent to these addresses.

Other examples of spatial database are – satellite imagery database, medical imaging databases.

**FUTURE DIRECTIONS**

Since spatial databases are becoming increasingly common and also large and complex, search is always on to find more efficiently store and query the spatial data. Much work is being done in area of spatial query processing in location dependent data (LDD). With popularity of mobile units there is also focus on improving performance of continuous spatial range queries. An example of such query is a user trying to find nearest restaurants when he is traveling by car. Since the location is changing the database has to be queried continuously to find the nearest restaurants.

Newer spatial access methods are always being proposed. Some of these are modifications to existing methods like quadtrees. One such example is a *skip quadtree*, which is a variant of compressed quadtree and allows for faster inserts and updates to octree.

**CROSS REFERENCES**

- Minimum Bounding Rectangles (MBR)
- R-Tree
- Quadtree

**RECOMMENDED READING**


