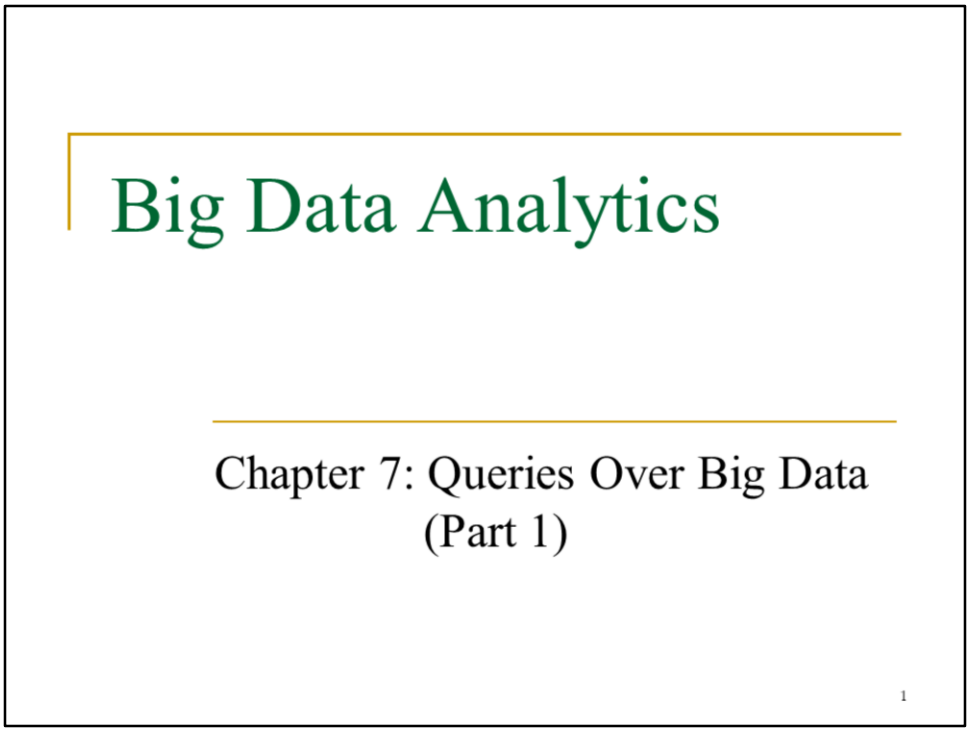
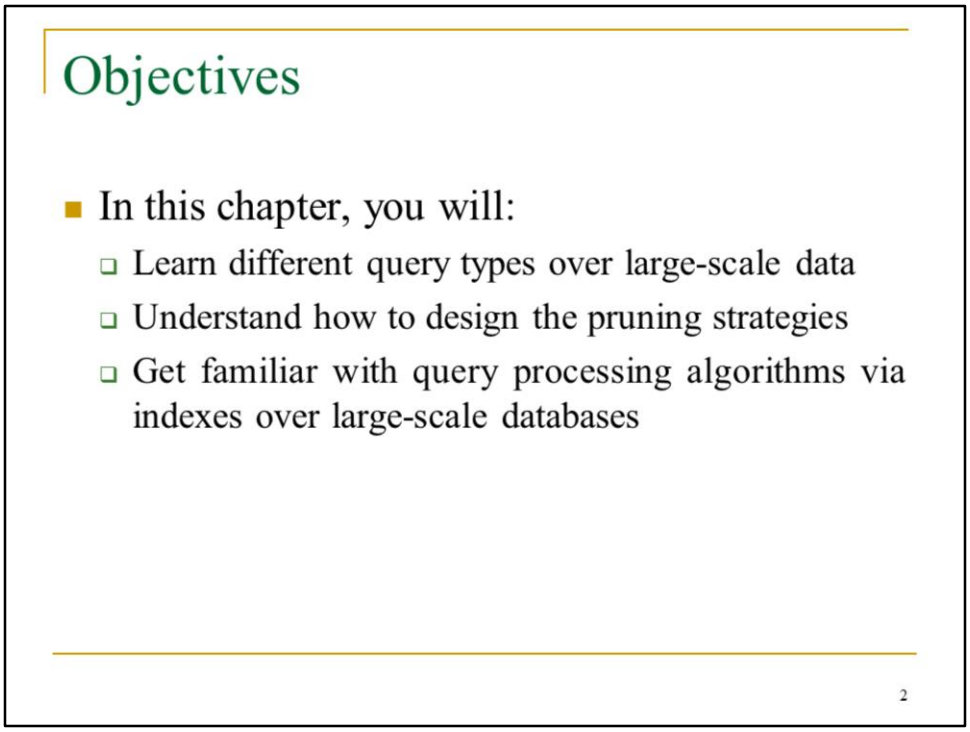


Chapter 7: Queries Over Big Data (Part 1)



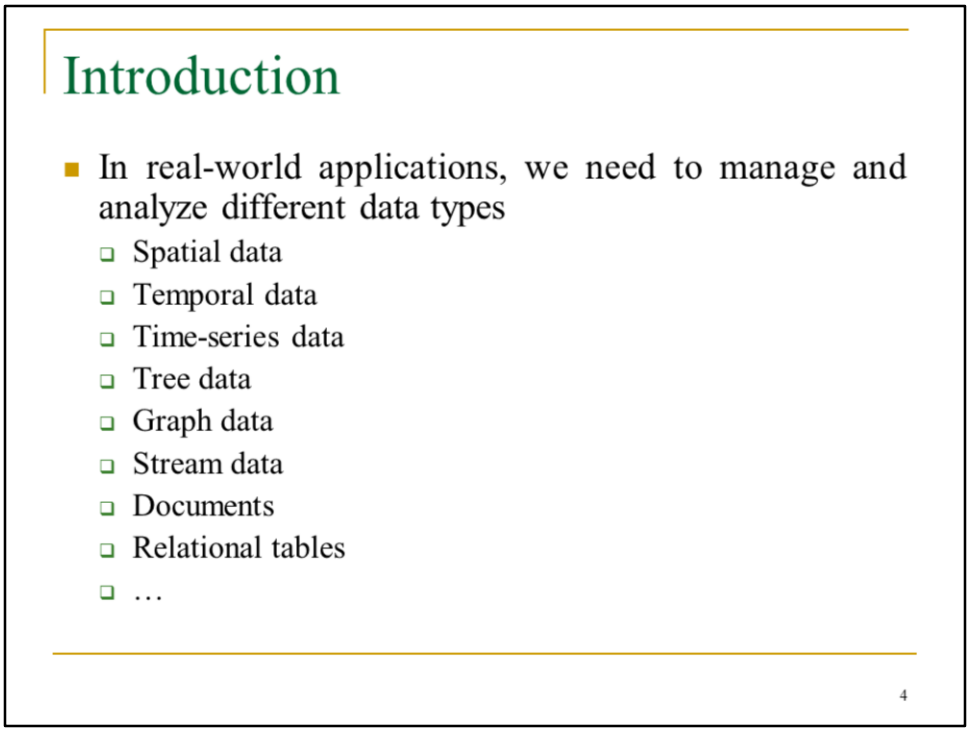
In this chapter, we will learn different query types on big data, and understand how to design pruning strategies and indexes to facilitate efficient query processing over big data.

Outline

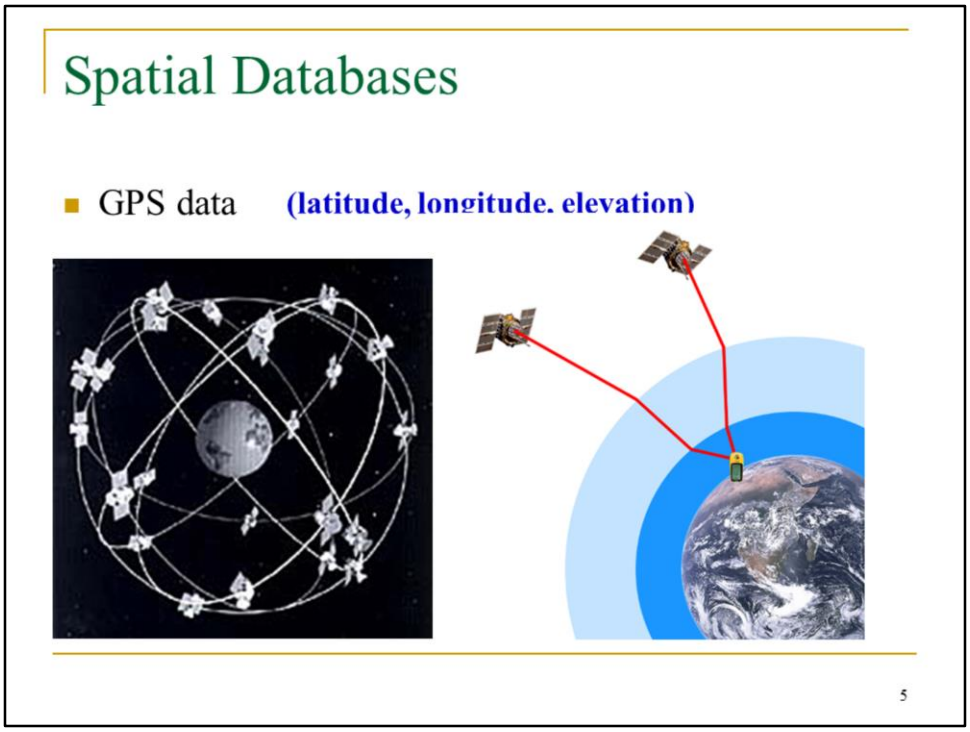
- Introduction of Real-World Application Data
- Query Types
  - Range Query
  - Nearest Neighbor (NN) Query
  - *k*-Nearest Neighbor (*k*NN) Query
  - Group Nearest Neighbor (GNN) Query
  - Reverse Nearest Neighbor (RNN) Query
  - …

3

We will first introduce various real-world application data. Then, we will discuss queries over spatial databases one by one, starting from classic range query and nearest neighbor query.



In real applications, we may encounter many data types that need to be managed and analyzed, including, but not limited to, spatio-temporal data, time-series data, tree data, graph data, stream data, text documents, relational tables, and so on.

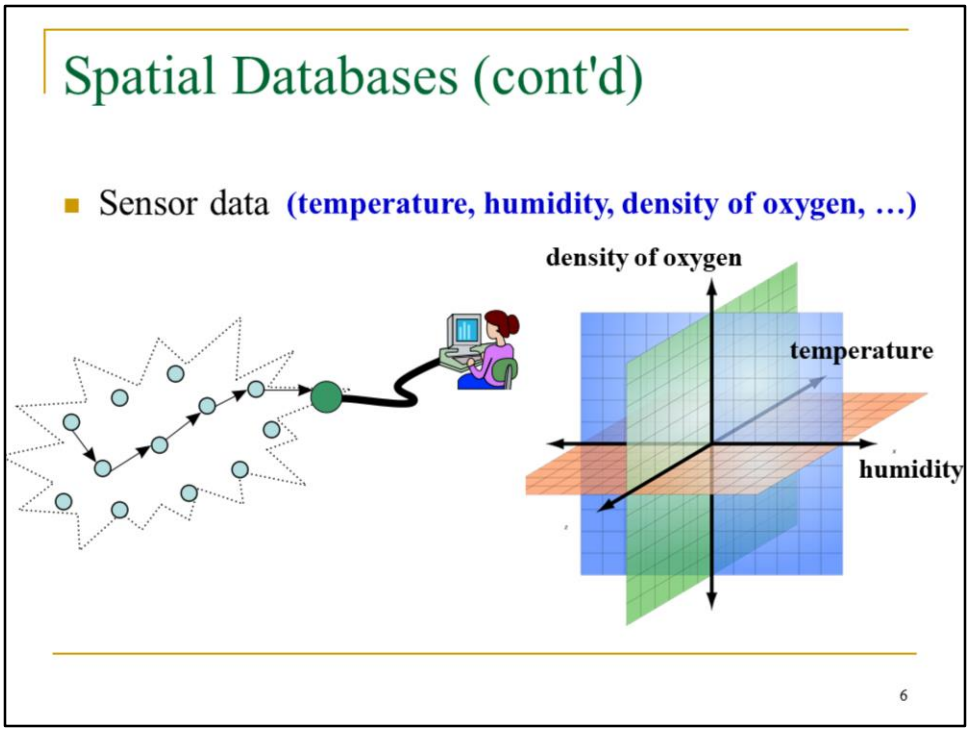


References

NOAA. *The Orbiting Constellation of GPS Satellites.* [object HTMLTableCellElement]. http://www.oceanservice.noaa.gov/topics/navops/positioning/welcome.html, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:GPS_satellite_constellation.jpg. Changes were not made.

Javiersanp, Gps-atmospheric-efects png: *The Earth seen from Apollo 17 jpg: NASANavstar-2. jpg: NASAGPS tracking satellites jpg: Vaughan Weatherderivative work: *Español: Efecto de La Atmósfera En La Propagación de La Señal GPS. English: Effect of the Atmosphere in the GPS Signal Propagation.* 18 Apr. 2011. Gps-atmospheric-efects.png, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Sygnal_GPS-opoznienie_atmosferyczne.png. Changes were not made.

In spatial databases, we may encounter GPS data with attributes such as latitude, longitude, and elevation.

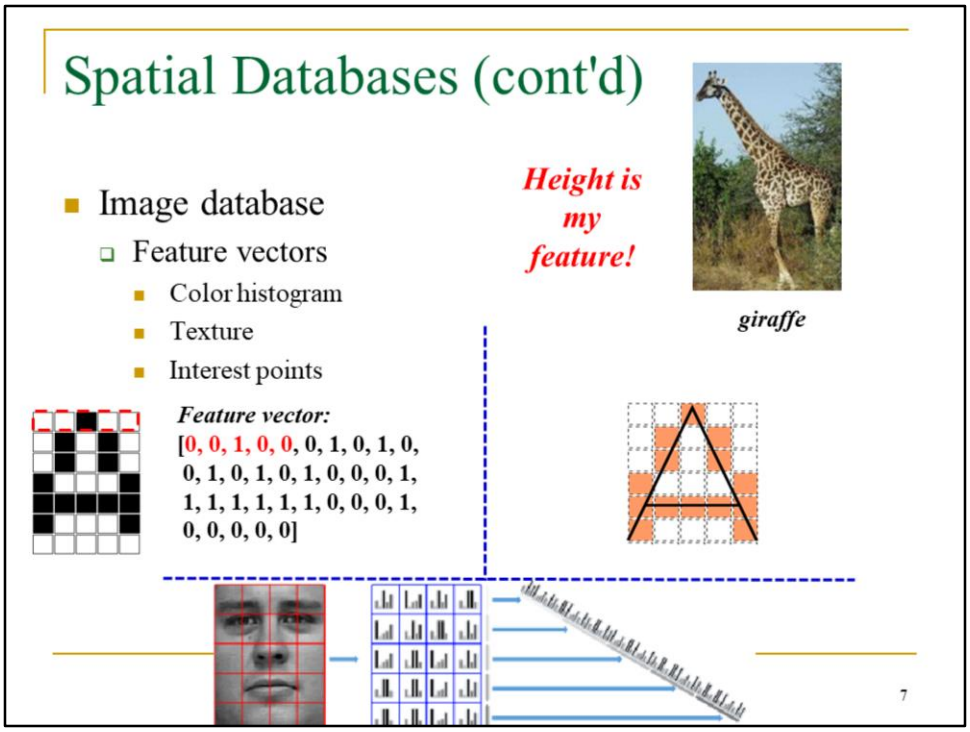


References

V, WSN svg: Original by Adi Mallikarjuna Reddy. *Wireless Sensor Network - French Version*. 13 Jan. 2010. WSN.svg, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:WSN-french.svg. Changes were not made.

assumed, No machine-readable author provided Sakurambo~commonswiki. *A Right-Handed Three-Dimensional Cartesian Coordinate System with the +z Axis Pointing towards the Viewer. Own Work, Produced as a Replacement for Image:3D Cartesian Coordinates.PNG*. 26 July 2007. No machine-readable source provided. Own work assumed (based on copyright claims)., *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:3D_coordinate_system.svg. Changes were made.

In sensor networks, we may deploy sensors to different sites (such as forests), and collect sensory data such as temperature, humidity, density of oxygen, and so on.



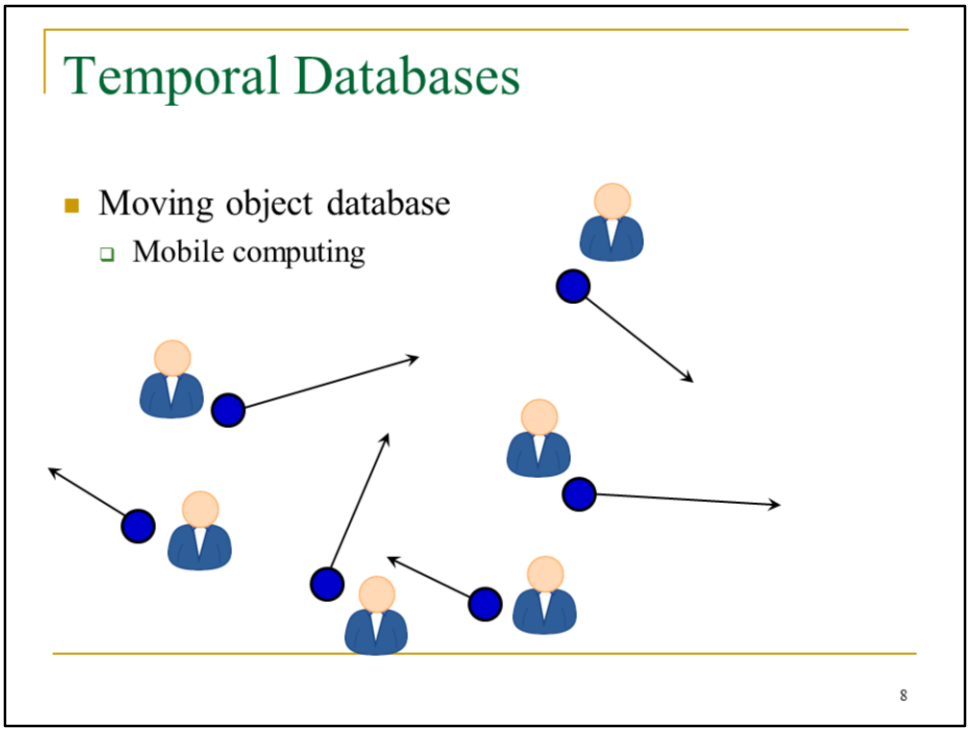
References

File:Giraffe.Jpg - Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Giraffe.jpg. Accessed 26 Feb. 2019. Changes were not made.

Wgabrie, Bitmap vs vector png: Fbjderivative work: *Deutsch: Vergleichende Darstellung Des Buchstaben „A“ Als Bitmap-Darstellung Und Als Vektor-Darstellung (Bitmap-Font vs. Vektorfont)*. 19 Feb. 2010. Bitmap vs vector.png, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Bitmap_vs_vector.svg. Changes were not made.

Karolina Nurzynska, Bogdan Smolka: Facial Displays Description Schemas for Smiling vs. Neutral Emotion Recognition. ICAISC (1) 2015: 594-605

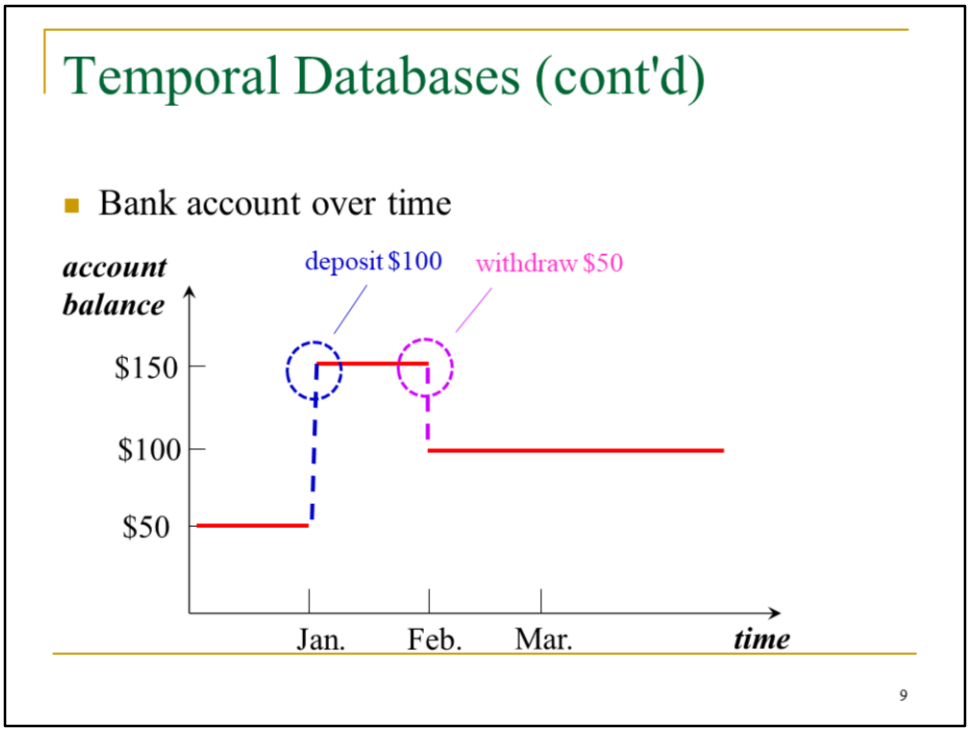
In image databases, we may extract feature vectors from images, for example, color histogram, texture, interest points, and so on. For example, in an image with black-and-white pixels, it can be transformed to a sequence of 0’s and 1’s, which can be considered as one feature vector. Similarly, we can also extract colors from pixels of images, and obtain color feature vector. Or we can divide an image into multiple partitions, and concatenate color histograms of all partitions as one feature vector.



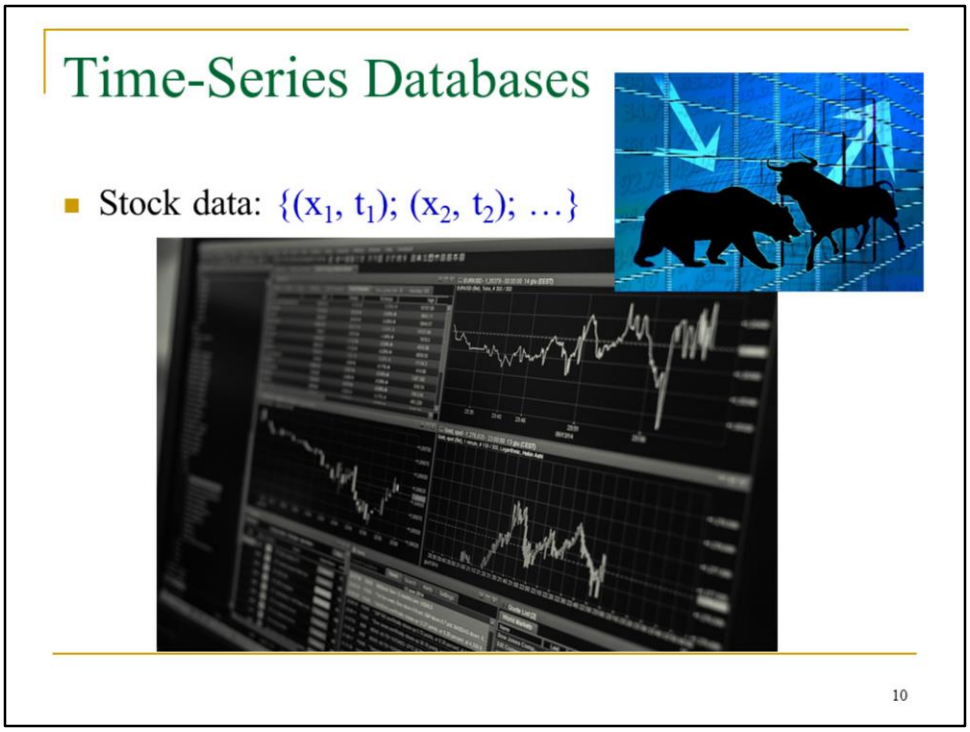
References

Clipart - User 1. https://openclipart.org/detail/171432/user-1. Accessed 26 Feb. 2019. Changes were not made.

For mobile computing, mobile users periodically send their GPS locations to central servers, which form a temporal moving object database.



For the bank account balance over time, we can also store the balance data of each account as a temporal object, and form a temporal database. As shown in the figure, initially the account balance is $50. Then, in January, the account owner deposited $100, so that the curve jumps to $150 in total. Next, in February, the account owner withdrew $50, so that the curve goes down to $100 balance. The temporal database records the changes of such an account balance over time.

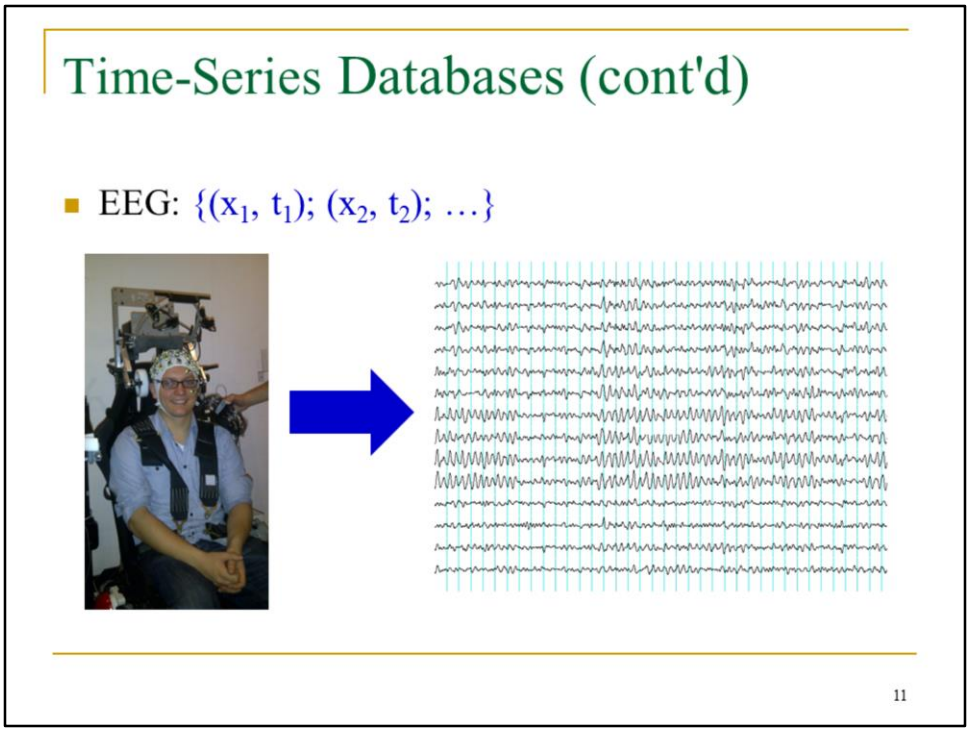


References

Free Image on Pixabay - Stock, Trading, Monitor, Business. https://pixabay.com/photos/stock-trading-monitor-business-1863880/. Accessed 26 Feb. 2019. Changes were not made.

Free Image on Pixabay - Stock Exchange, Bull, Bear. https://pixabay.com/illustrations/stock-exchange-bull-bear-securities-642896/. Accessed 26 Feb. 2019. Changes were not made.

Another important database is the time-series database in real applications such as stock market analysis, sensor networks, and trajectory data analysis, where each time series is an ordered sequence of values. The figure shows the time series of stock prices over time.



References

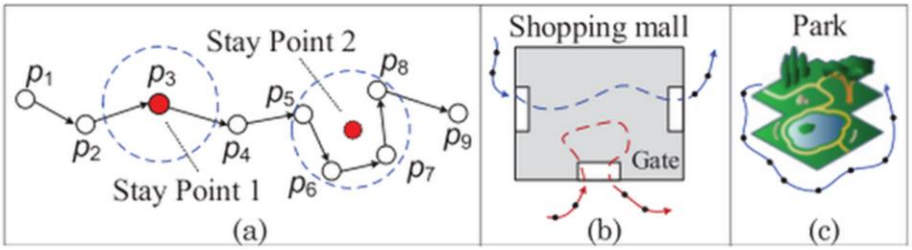
Alexhvl15. *English: An Experimental Subject Is Partaking in an EEG Experiment.* 3 Dec. 2012. Own work, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:EEG.png. Changes were not made.

File:ElectroEncephalogram.Png - Wikimedia Commons. https://commons.wikimedia.org/wiki/File:ElectroEncephalogram.png. Accessed 26 Feb. 2019. Changes were not made.

Electroencephalogram (EEG) data are also in the form of time series, which collects sequences of signals from sensors.

Time-Series Databases (cont'd)

- Trajectory data: $\{(x_1, y_1, t_1); (x_2, y_2, t_2); \ldots\}$



12

References

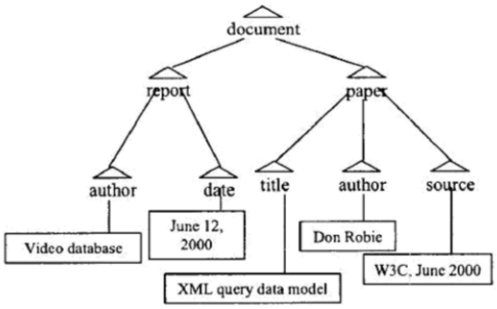
Yu Zheng: Trajectory Data Mining: An Overview. ACM TIST 6(3): 29:1-29:41 (2015)

The trajectory data is also an ordered sequence of locations, which can be considered as time series. The analysis of trajectories is important for location-based services, study of customer behaviors in the shopping malls or parks, and detection of abnormal behaviors for the transportation system.

Tree Data

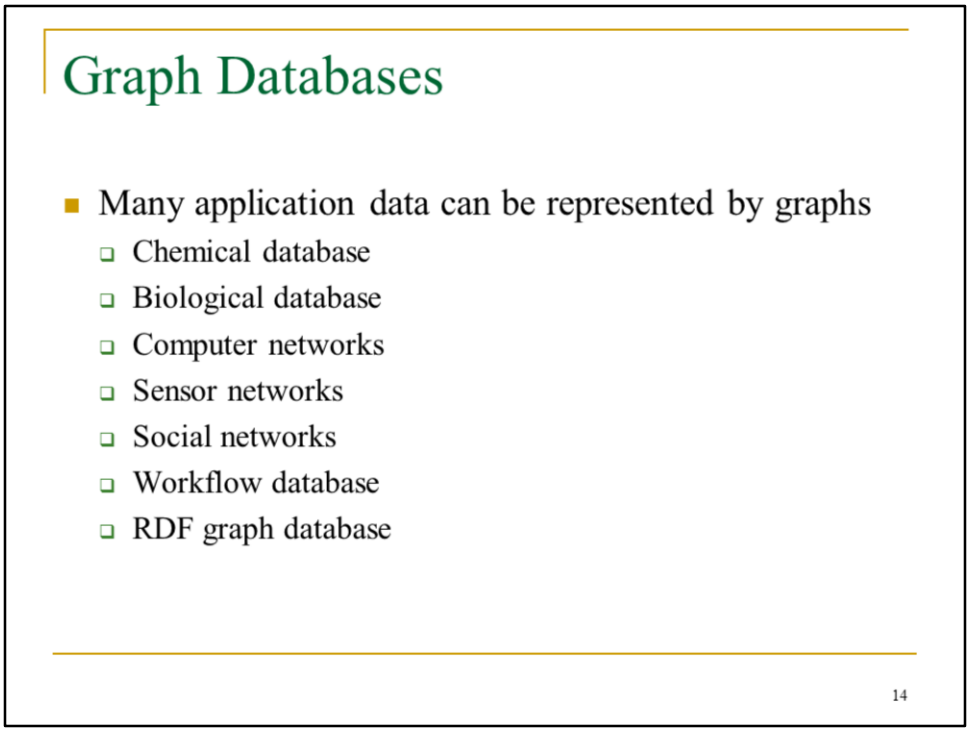
- XML documents
  - Semi-structured data

```
<document>
  <report>
    <author>Video database</author >
    <date>June 12, 2000</date>
  </report >
  <paper>
    <title>XML query data model</title>
    <author>Don Robie</author>
    <source>W3C, June 2000</source>
  </paper>
</document>
```

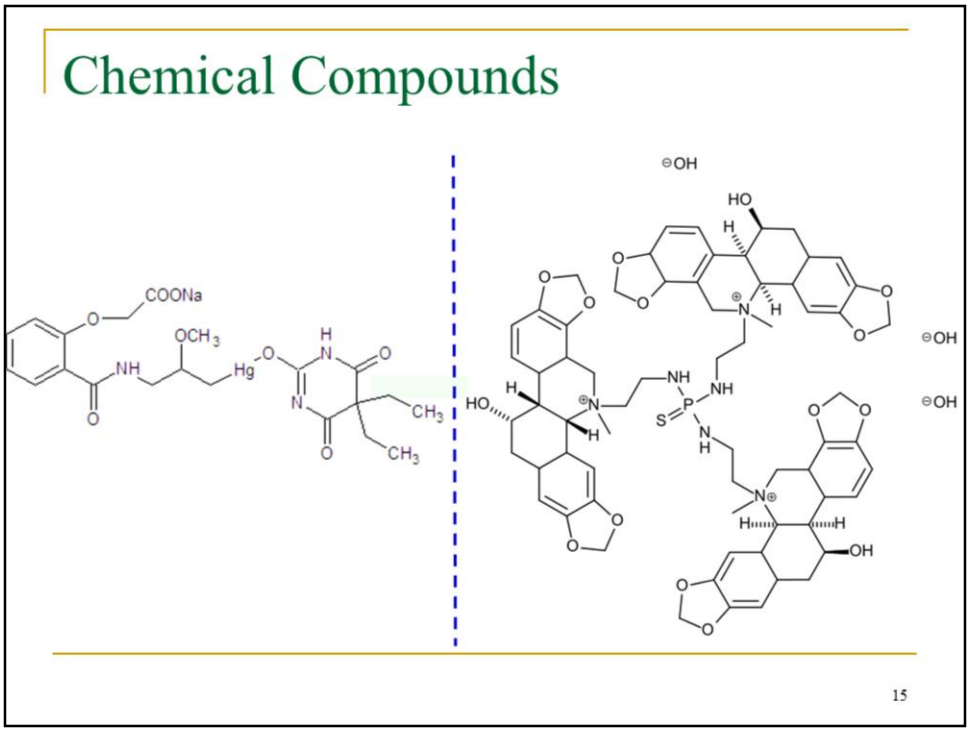


13

XML document is semi-structured data, represented by a tree structure, which can be used for the data exchange among programs. As shown in the figure, the “document” tag is a root, under which there are two tags “report” and “paper”. The “report” tag contains “author” and “date” tags, which are two children under “report” in the XML tree.



There are also many applications that deal with graph databases such as chemical database, biological database, computer networks, sensor networks, social media, workflow database, RDF graph database, and so on.

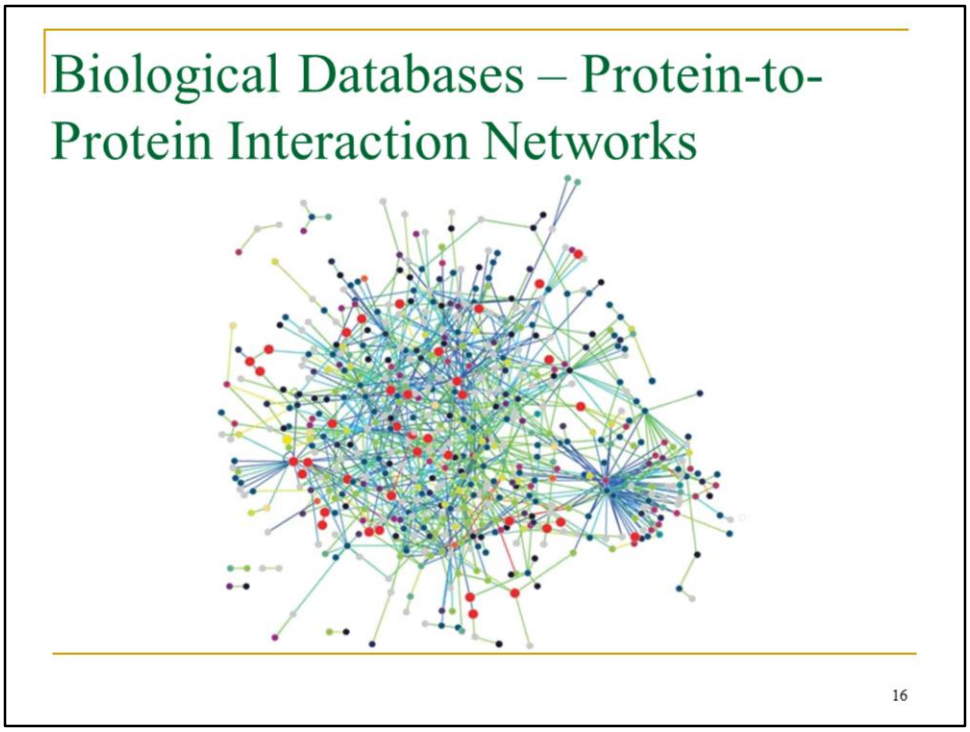


References

Zhilin, Denis. *English: Ctructural Formula of Mersalin (Allyrgal, Diluvit, Dimed, Diuregan, Diurette, Diursal, Fernmersyl, Gortulina, Hg-Phyllin, Hydrargan, Irgrosin, Mercuprocyl, Mercurgan, Mercurosalyl, Mercusalum, Mercuzal, Merphyllin, Mersaline, Mersalpin, Mersalyn, Merthipyl, Neomersyl, Neptal, Salirgan, Salurin, Salyrgan, Sibilen, Syralgan, Thesalyl, Theurin, Uragan, Uragon)*. 3 July 2012. Own work, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Merkusal.png. Changes were not made.

Hennersdorf, Felix. *English: Chemical Structure of Ukrain*. 19 Feb. 2008. Own work, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Alleged_molecular_structure_of_ukrain.svg. Changes were not made.

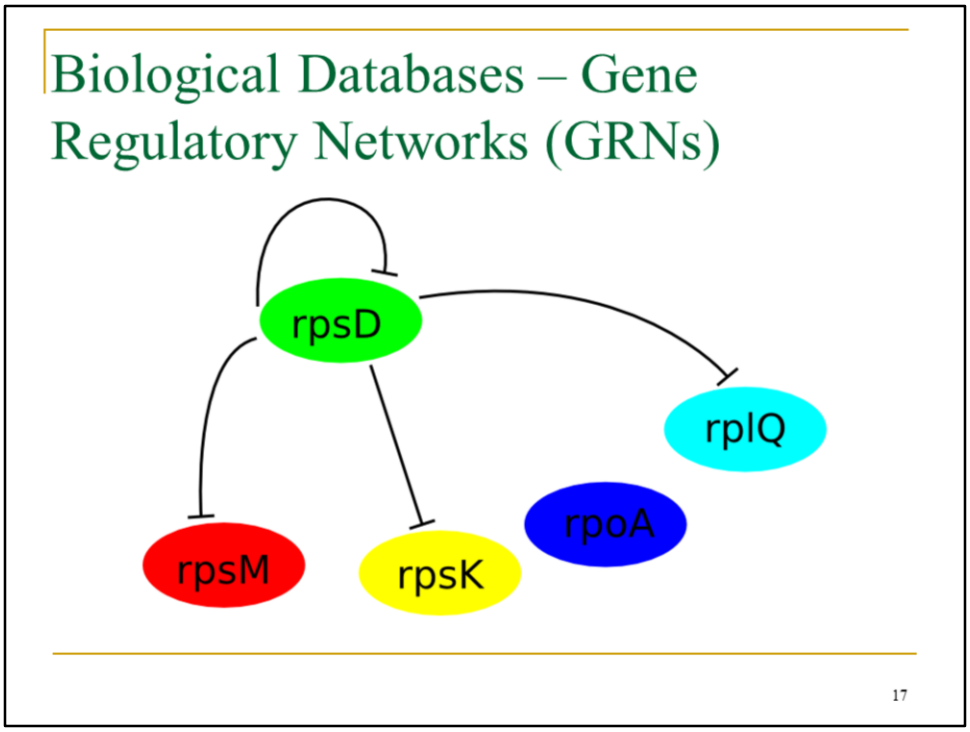
For example, chemical compounds can be represented by graph structures, where nodes are atoms and edges are relationships between two atoms.



References

al, Häuser et. *English: The Protein Interaction Network of T. Pallidum Including 576 Proteins and 991 Interactions. Nodes Are Color-Coded According to TIGR Main Roles. Proteins Involved in DNA Metabolism Are Shown as Enlarged Red Circles.*[object HTMLTableCellElement]. Titz B, Rajagopala SV, Goll J, Hä¨user R, McKevitt MT, et al. (2008) The Binary Protein Interactome of Treponema pallidum – The Syphilis Spirochete. PLoS ONE 3(5): e2292. doi:10.1371/journal.pone.0002292, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:The_protein_interaction_network_of_Treponema_pallidum.png. Changes were not made.

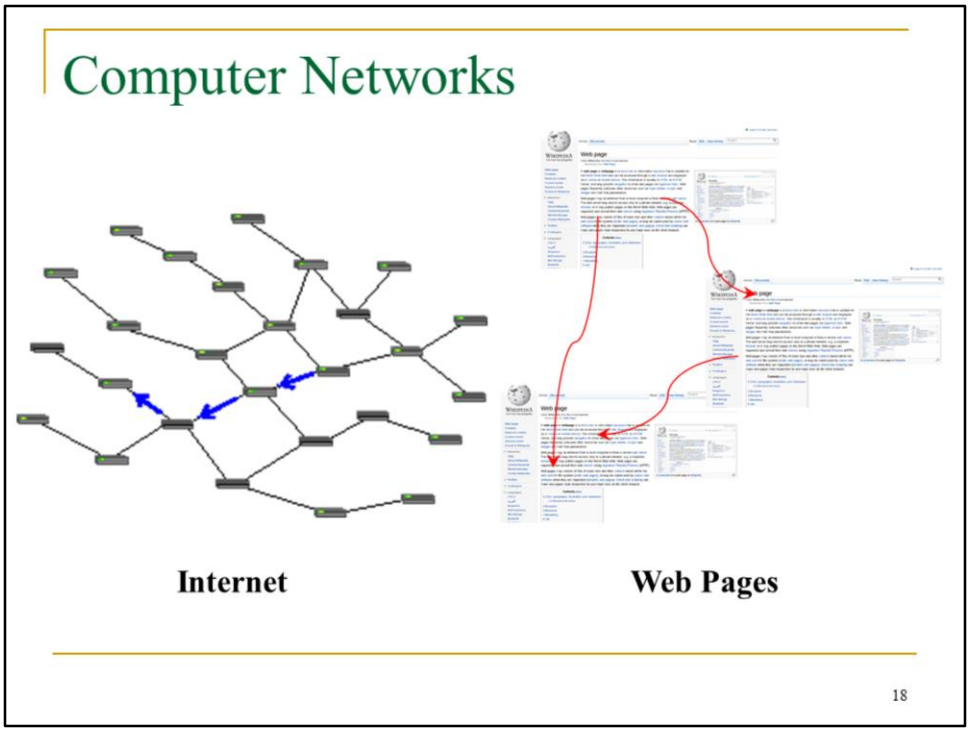
In biological databases, protein-to-protein interaction networks (PPI) can be represented by a graph, where each vertex is a protein, and each edge represents the interaction relationship between two proteins.



References

Aravindabharathi. *Alpha Operon Gene Regulatory Network Diagram*. 27 Sept. 2013. Own work, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Alpha_operon_gene_regulatory_network.png. Changes were not made.

Similarly, gene regulatory network (GRN) is also a graph with genes as vertices and regulatory relationships between any two genes as edges.

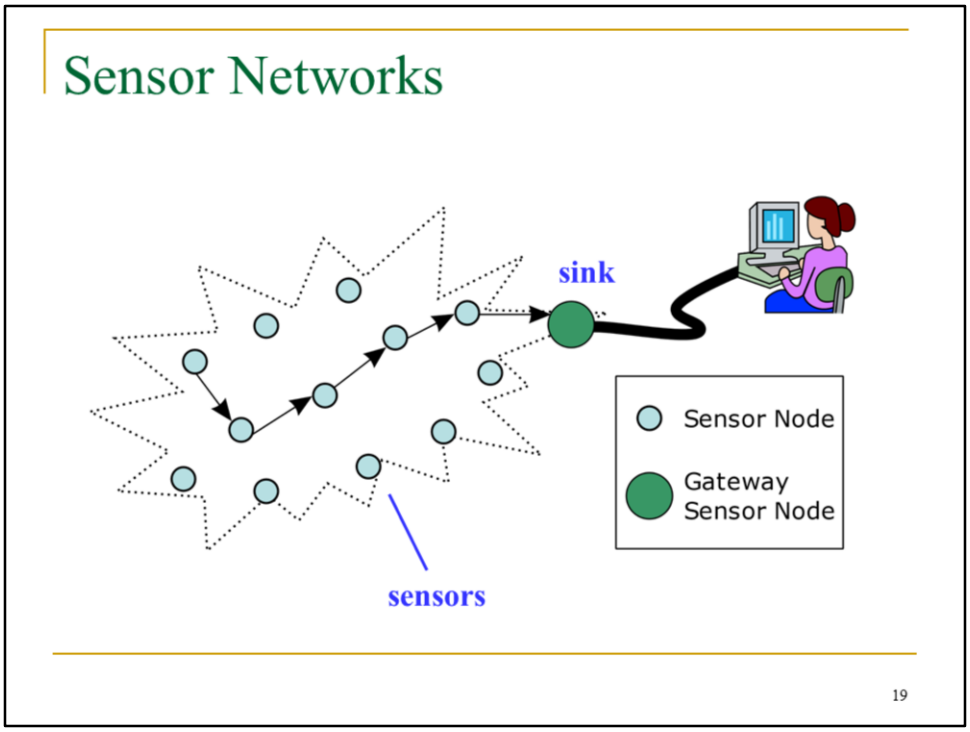


References

assumed, No machine-readable author provided Baccala@freesoft org~commonswiki. *A Unicast Forwarding Pattern in a Computer Network*. 27 Sept. 2005. No machine-readable source provided. Own work assumed (based on copyright claims)., *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Unicast_forwarding.png. Changes were not made.

en.wikipedia, Lulzmango at. *English: A Screenshot of a Web Page. Русский: Скриншот Веб-Страницы.* 21 Sept. 2010. en:Web_page, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Web_Page.png. Changes were not made.

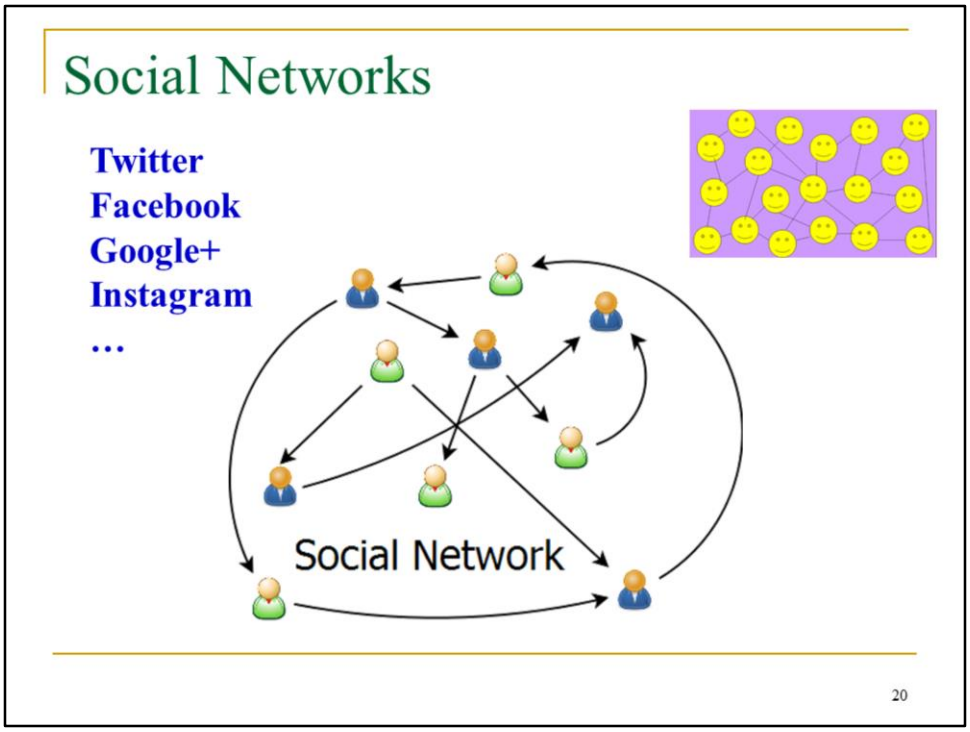
We also encounter other graphs such as computer networks that connects multiple servers/routers, and Web pages with hyperlinks linking to one another.



References

V, Original by Adi Mallikarjuna Reddy. *Wireless Sensor Network*. 11 June 2007. Own work, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:WSN.svg. Changes were not made.

Sensor networks is also in the form of graphs, where sensors can be considered as nodes, and the communication between any two sensors can be considered as an edge.

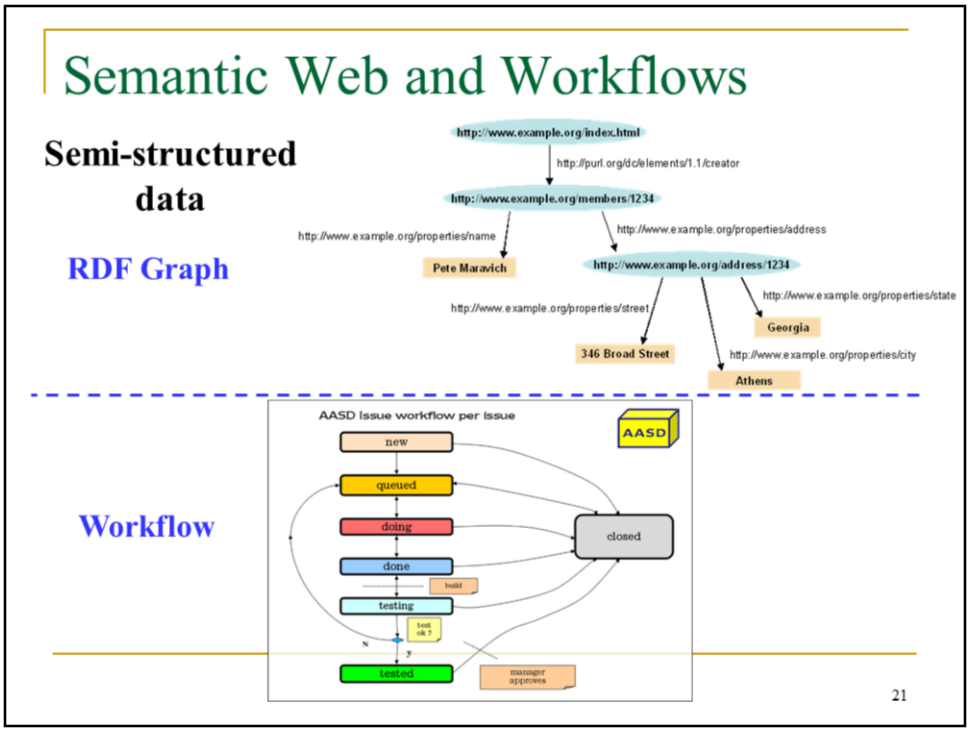


References

Thor4bp, Zigomitros Athanasios-. *English: Social Network*. 27 Feb. 2012. Own work, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Social_Network.png. Changes were not made.

Wikibooks, Koreshky at English. *English: Description: Social Networking Source: Own Work Author: Koreshky Date: 12/10/2007*. 11 Dec. 2007. Transferred from en.wikibooks to Commons by Adrignola using CommonsHelper., *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Social_Networking.png. Changes were not made.

In social networks such as Twitter, Facebook, Google+, and Instagram, users have their own profiles and follow or be followed by their friends. In this case, social networks are also graphs with users as vertices and friendships as edges.

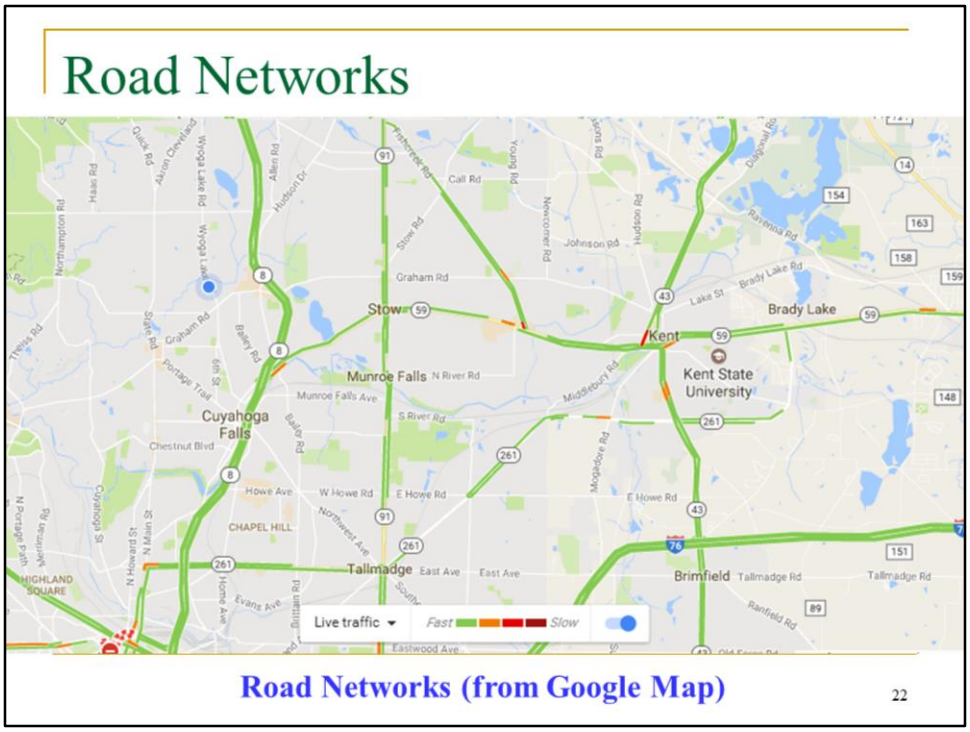


References

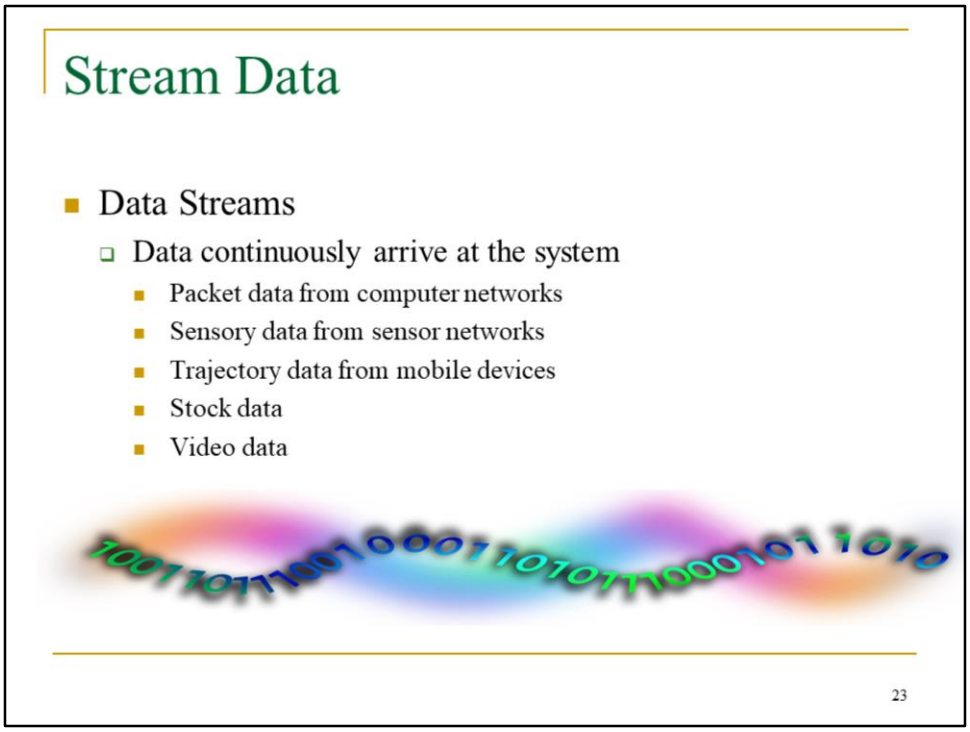
File:Rdf-Graph2.Png - Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Rdf-graph2.png. Accessed 26 Feb. 2019. Changes were not made.

en.wikipedia, Sbvb at. *English: Version 2 of Aasd Workflow*. 7 Sept. 2010. Transferred from SreeBot to Commons by SreeBot., *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Aasd_workflow_v2.png. Changes were not made.

There are other examples of graphs such as workflow data and RDF graphs in the Semantic Web.



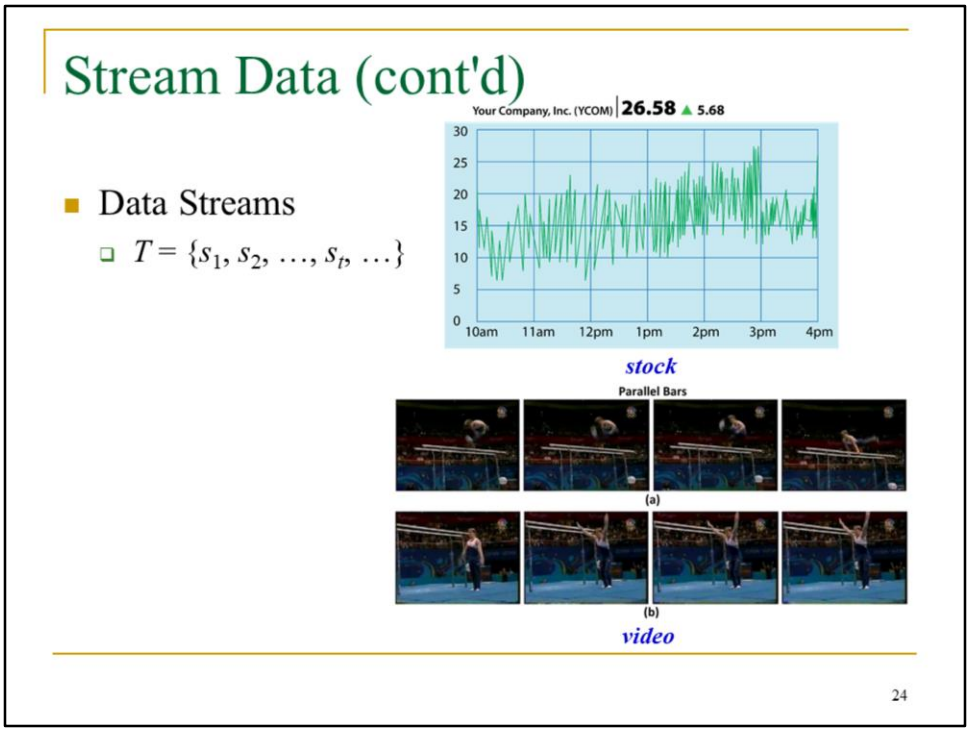
Moreover, spatial road networks can be also modeled by a graph, where edges are road segments and vertices are intersection points of roads.



References

Clipart - Binary Data Stream 1. https://openclipart.org/detail/167104/binary-data-stream-1. Accessed 26 Feb. 2019. Changes were not made.

Another type of application data is the stream data, which continuously arrive at the system, such as packet data from the Internet, sensory data from sensor networks, trajectories from taxi or mobile users, stock data, or video/audio data.



References

Clipart - Stock Quote Graph. https://openclipart.org/detail/170148/stock-quote-graph. Accessed 26 Feb. 2019. Changes were not made.

Bikingdog. *English: Video Frames of the Parallel Bars Action Category in the UCF-101 Dataset[1] (a) The Highest Ranking Four Frames in Video Temporal Attention Weights, in Which the Athlete Is Performing on the Parallel Bars; (b) The Lowest Ranking Four Frames in Video Temporal Attention Weights, in Which the Athlete Is Standing on the Ground. All Weights Are Predicted by the ATW CNN Algorithm[2]. The Highly Weighted Video Frames Generally Captures the Most Distinctive Movements Relevant to the Action Category.* 12 Sept. 2018. Own work, *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Video_frames_of_the_Parallel_Bars_action_category.png. Changes were not made.

Here are examples for stock data stream and video data (streaming frames).

Documents

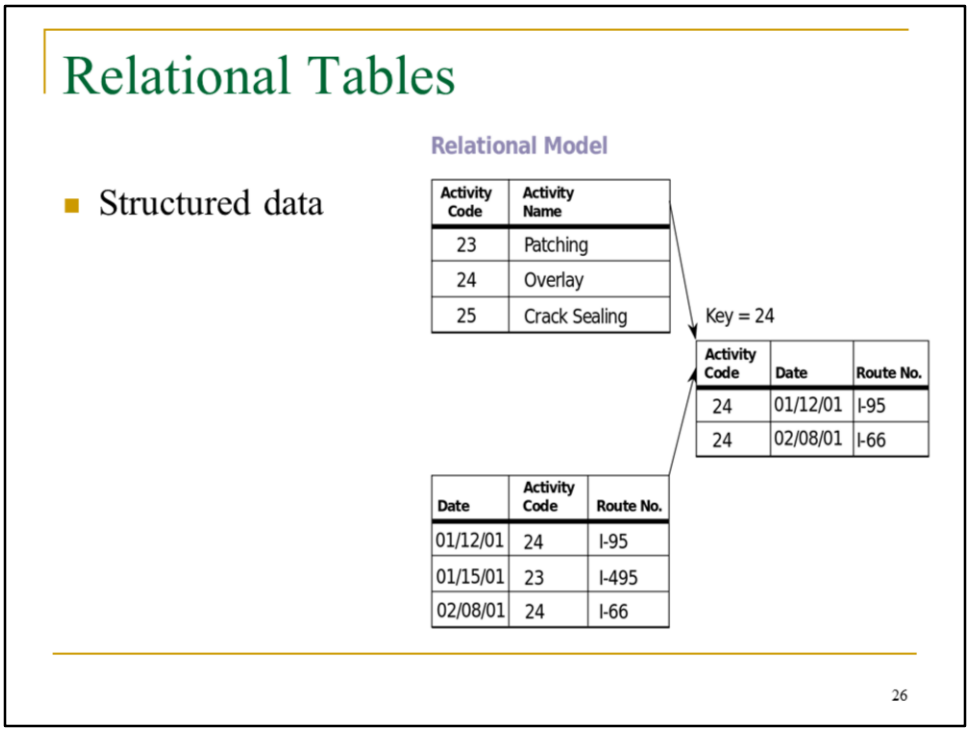
- Text data
  - Unstructured data
  - Strings

25

References

Clipart - TXT Document. https://openclipart.org/detail/212868/txt-document. Accessed 26 Feb. 2019. Changes were not made.

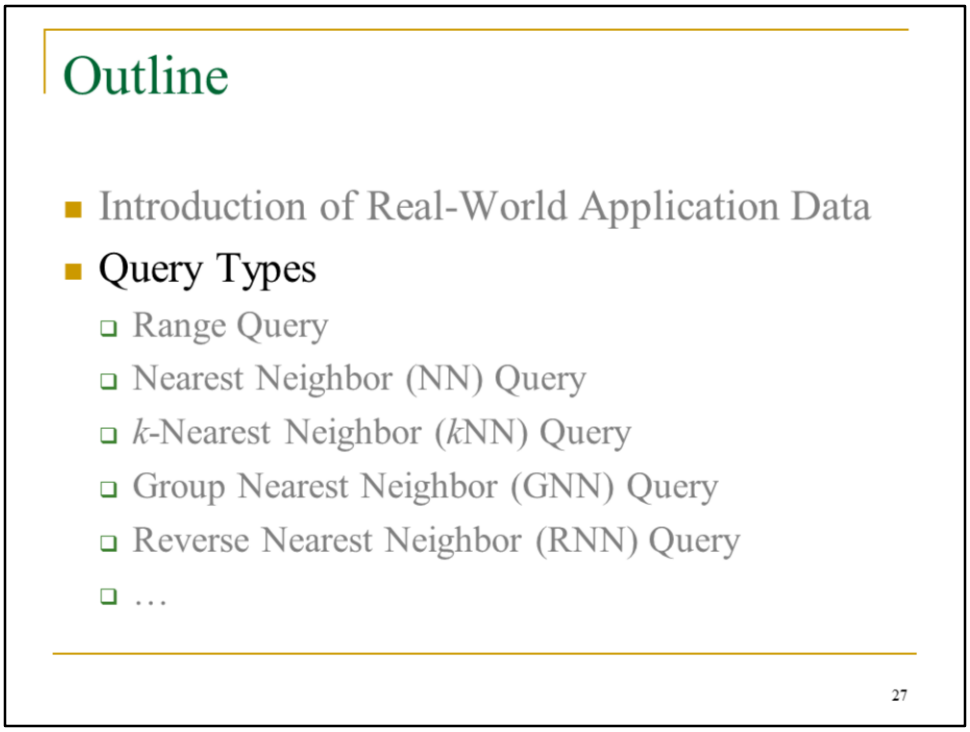
In documents, there are many text data in unstructured format.



References

work, U. S. Department of Transportationvectorization: Own. *Relational Model*. Aug. 2001. Data Integration Glossary., *Wikimedia Commons*, https://commons.wikimedia.org/wiki/File:Relational_Model.svg. Changes were not made.

In traditional relational databases, there are many structured tables.



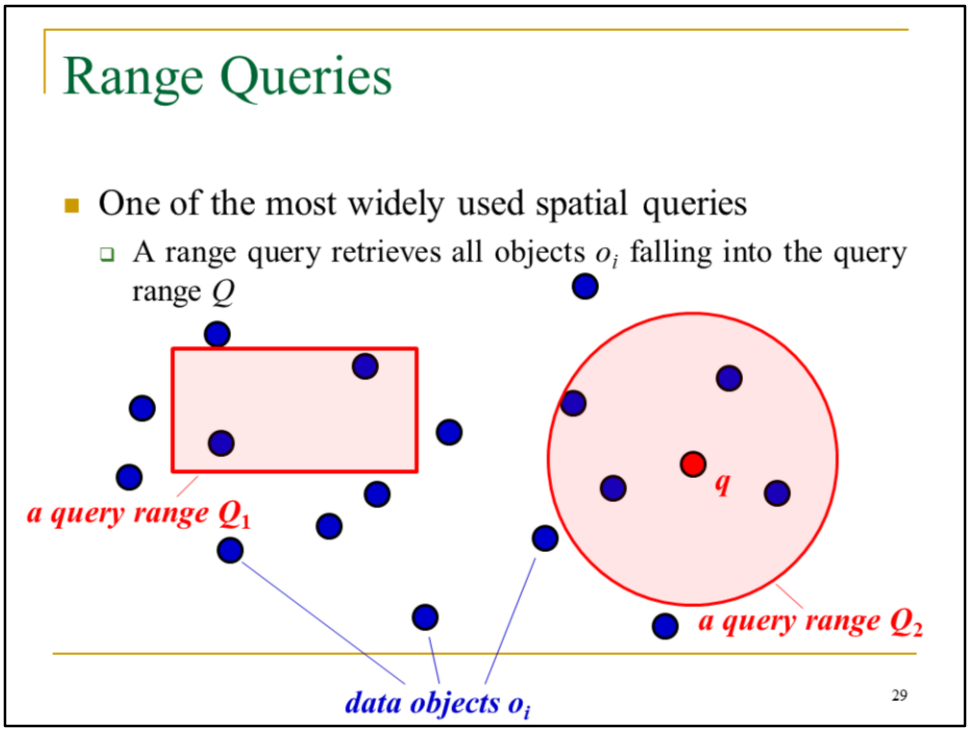
Next, we will talk about different query types over data, which are very useful in many real applications.

Queries Over Big Data

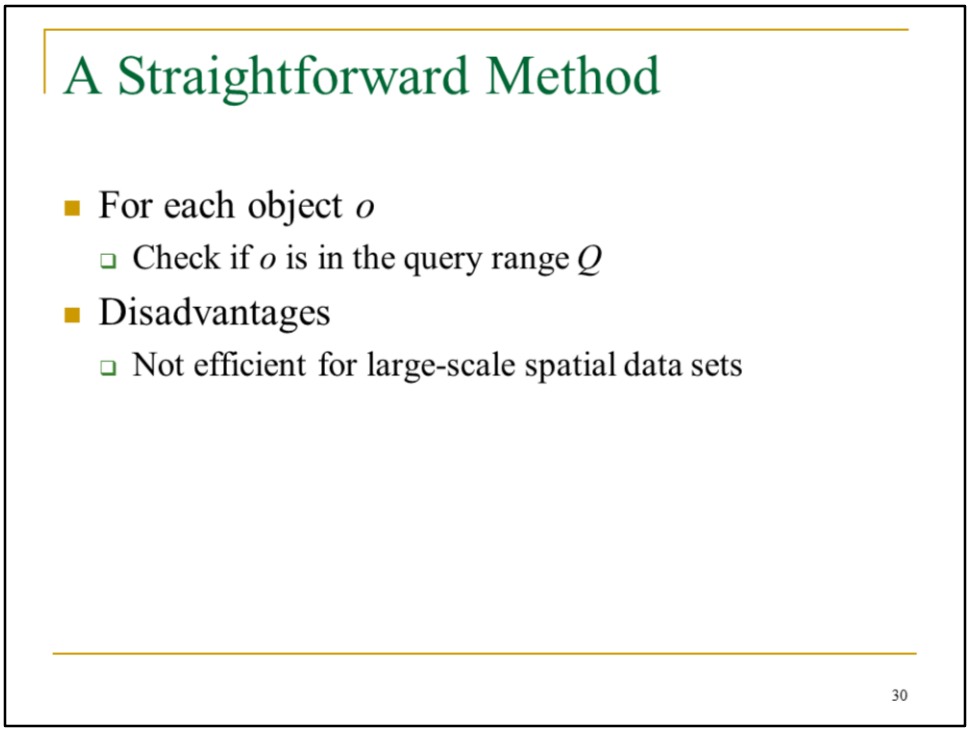
- Range Query
- Nearest Neighbor (NN) Query
- *k*-Nearest Neighbor (*k*NN) Query
- Group Nearest Neighbor (GNN) Query
- Reverse Nearest Neighbor (RNN) Query
- Top-*k* Query
- Skyline Query
- Top-*k* Dominating Query
- Reverse Skyline Query
- Inverse Ranking Query
- Aggregate Queries
- Keyword Search Query
- Graph Queries

28

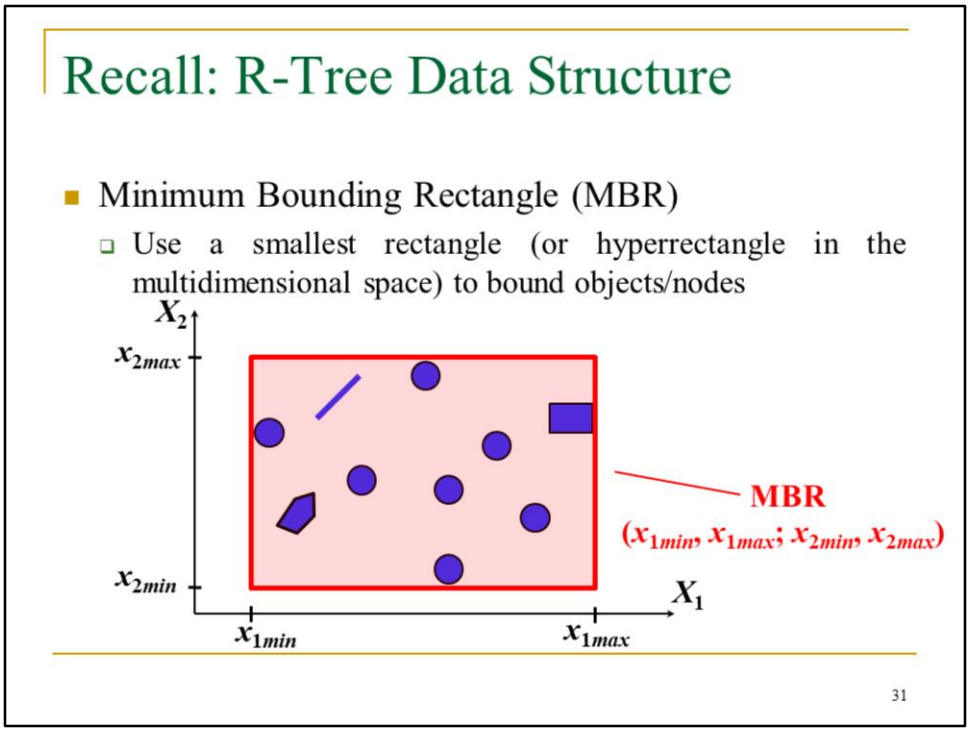
Here is a short list of spatial queries over big data. There are many other queries and their variants which are not covered in the slides.



We will start from the classic range query, which retrieves all objects falling into a given query range Q. The shape of the query range can be hyperrectangle or hypersphere. The range query is very useful in applications like location-based services, map services, and so on. For example, the range query can be used to find all restaurants within a city region, or all hotels within 15 miles away from your current location (query point q).



The straightforward method is to sequentially scan the entire database, and check if each object is inside the query range Q. This is however not efficient, especially for databases of large scale.

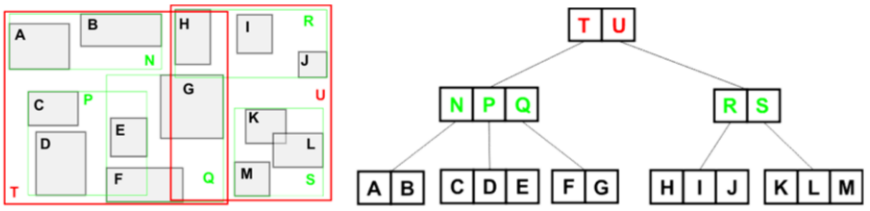


Recall that we talked about the R-tree index over spatial objects, which can use MBRs to summarize a number of spatial objects.

Recall: R-Tree Data Structure (cont'd)

- R-tree is a *height-balanced* **multi-way external memory** tree over n multidimensional data objects (height: $log_F(n)$)
  - **Non-leaf node (or intermediate node):** contains a number of entries (MBRs) that minimally bound their child nodes, as well as pointers pointing to child nodes
  - **Leaf node:** contains spatial objects

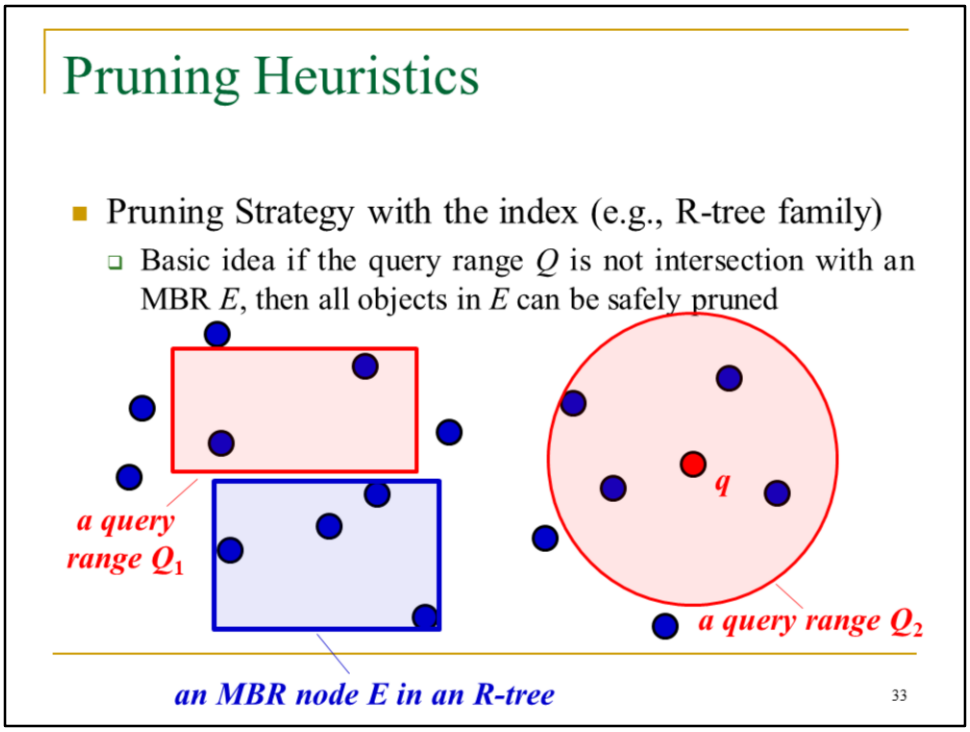
node *fanout*: $F \in [m, M]$, where $m \leq M/2$



32

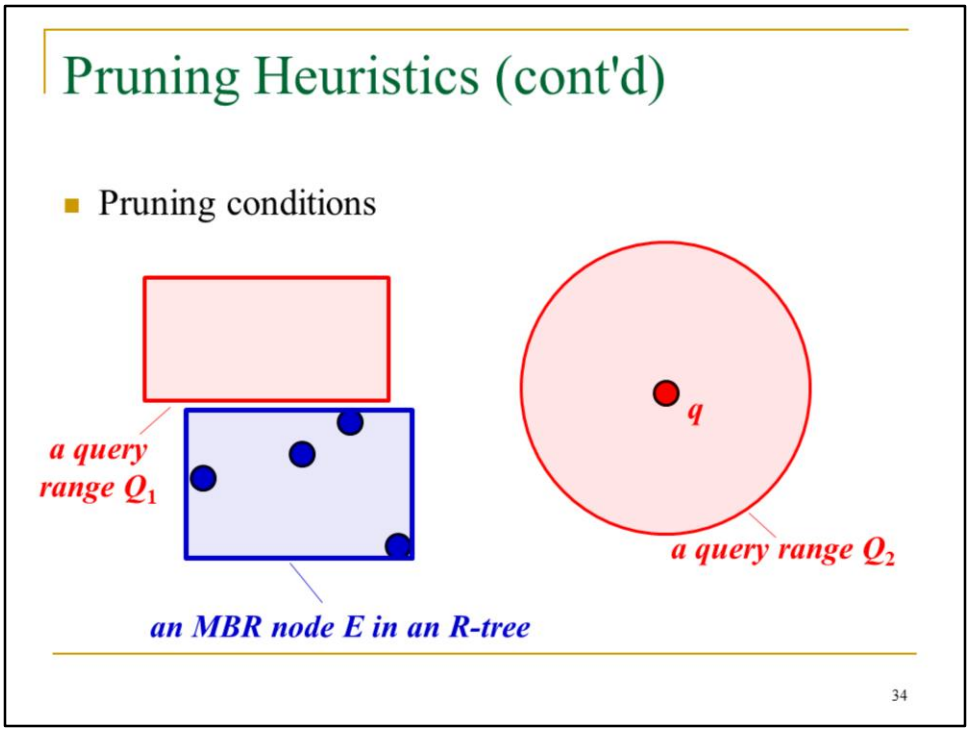
To answer the range query, instead of checking objects one by one, we can check whether MBRs are intersecting with the query range, with the help of the R-tree.

By using the concept of the MBR, we can save the computation cost for checking a group of objects in an MBR (if this MBR does not intersect with the query range at all).



To enable efficient range query processing, our goal is to derive some pruning conditions w.r.t. MBRs for the range query. If an MBR node in the R-tree can be safely pruned, then we do not have to access all objects inside this MBR node, which can save a lot of computation cost.

For example, the blue rectangle is an MBR node E in an R-tree. If this node E is completely outside the query range, then we can see that none of objects under node E can be the query answer. We thus can safely prune the node E without accessing its underlying objects.



Therefore, for range query Q_1, we only need to decide if two rectangles are intersecting or not. For range query Q_2, we only need to decide whether or not the rectangle E is outside the query circle.

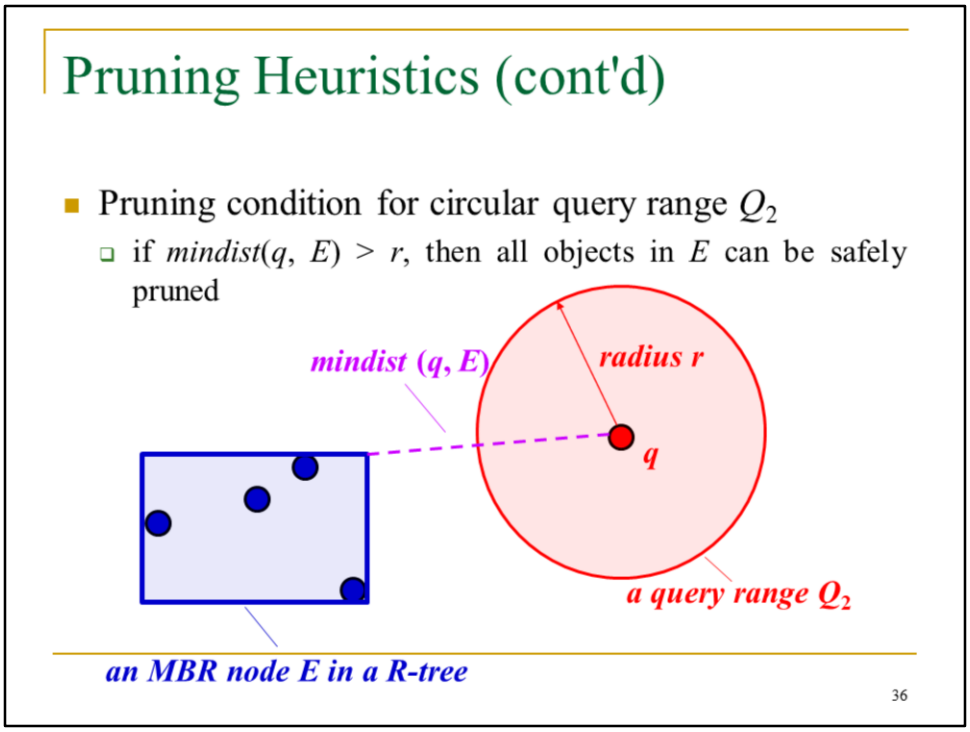
Pruning Heuristics (cont'd)

Input: an MBR node $E = (x_{1min}, x_{1max}; x_{2min}, x_{2max}; \ldots; x_{dmin}, x_{dmax})$ and a rectangular query range $Q_1 = (q_{1min}, q_{1max}; q_{2min}, q_{2max}; \ldots; q_{dmin}, q_{dmax})$

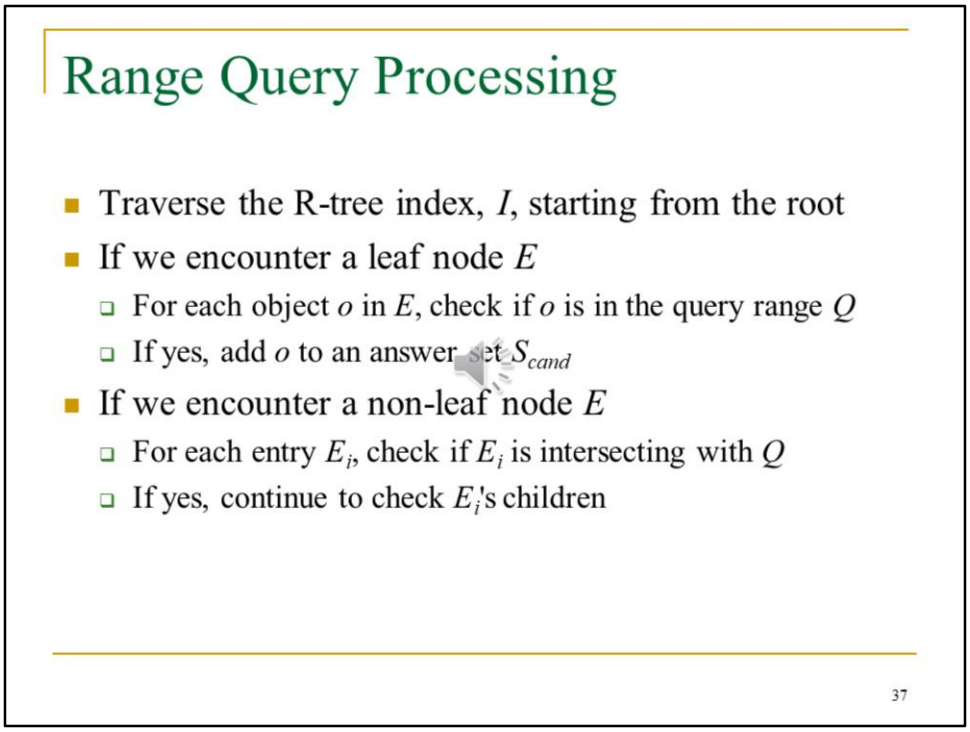
```
// basic idea: if MBR E and query range Q1 overlap on all dimensions
// (i.e., [ximin, ximax] and [qimin, qimax] on all dimensions i), two nodes overlap;
// otherwise, two nodes are disjoint
bool overlap = true;
for each dimension i
{
        if ([ximin, ximax] and [qimin, qimax] do not overlap with each other)
        {
                overlap = false;
                break;
        }
}
return overlap;
```

35

Here is the pseudo code to decide if two hyperrectangles are overlapping. Intuitively, if the projected intervals of two rectangles have overlaps on all dimensions, then these two rectangles overlap in the d-dimensional space. If there exists one dimension such that the projected intervals of two rectangles are disjoint, then these two rectangles cannot overlap.



For the range query Q_2 with circular query range, we need to compute the minimum distance, mindist(q, E), from query point q to node E. If this mindist() has distance larger than radius r, then MBR node E must be outside the query range, and can be safely pruned.



After discussing the pruning heuristics, we present how to process range queries via R-tree index. Specifically, we traverse the R-tree index starting from the root. If we encounter a leaf node, we then add objects in the leaf node that fall into the query range Q to the answer set S_{cand}. When we encounter a non-leaf node E, we will find those entries E_i under E that are intersecting with Q, and continue to check these child entries.

Range Query Processing Algorithm

```
Queue H=empty;
Insert root(index I) into H;
while (H is not empty)
{
        Pop out a node E from H;
        If (E is a leaf node)
        {
                for each object o in E
                        if o is in Q
                                add o to candidate set S_cand
        }
        else // non-leaf nodes
        {
                for each entry, E_i, in E
                        if E_i intersects with Q
                                insert E_i into H;
        }
}
```

38

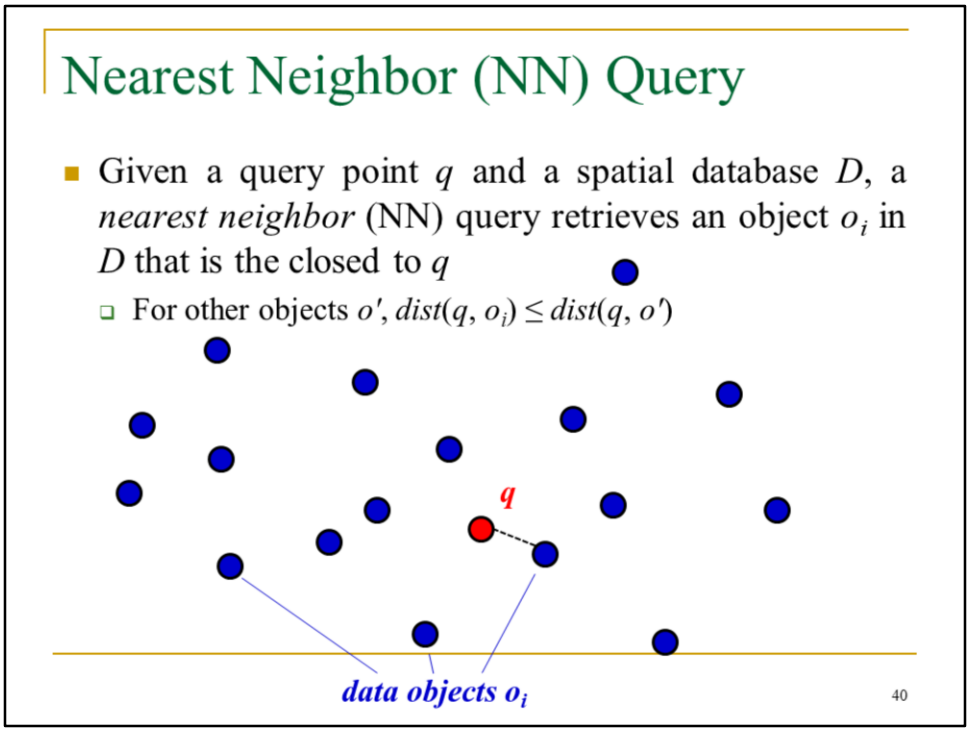
This is the pseudo code of range query processing over the R-tree index. We use a queue H to facilitate the traversal of the R-tree, which contains MBR nodes that intersect with the query range Q.

Outline

- Introduction of Real-World Application Data
- Query Types
  - Range Query
  - Nearest Neighbor (NN) Query
  - *k*-Nearest Neighbor (*k*NN) Query
  - Group Nearest Neighbor (GNN) Query
  - Reverse Nearest Neighbor (RNN) Query
  - …

39

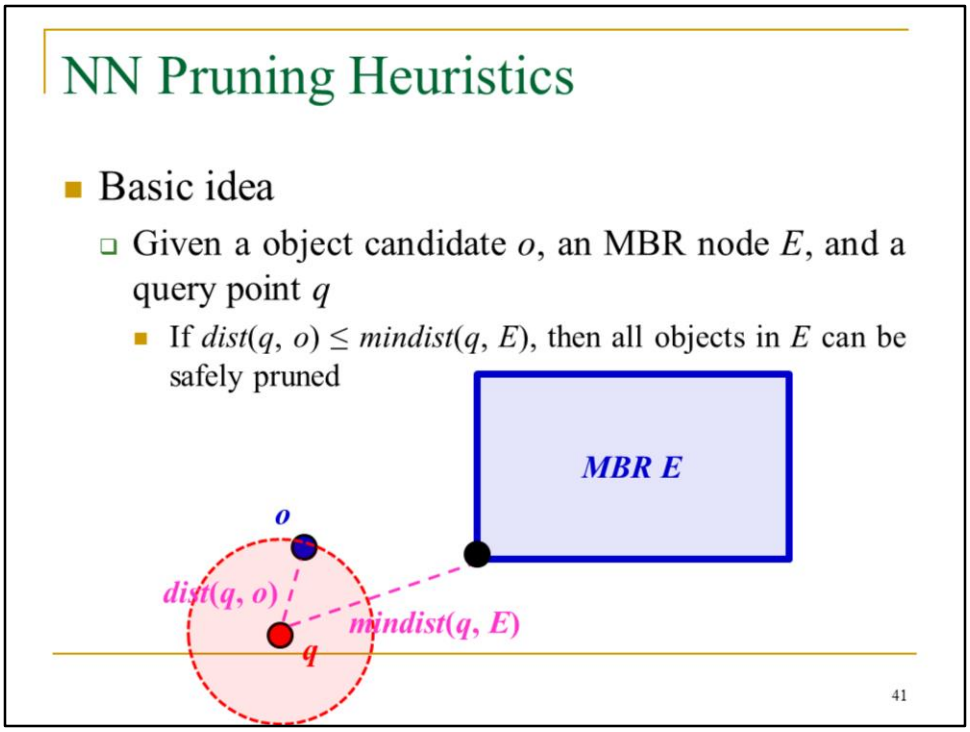
Next, we will talk about nearest neighbor query (or in the general case, k-nearest neighbor query).



Assume that you want to go to a restaurant that is closest to your current location, and have your dinner there. In this case, you need to issue a nearest neighbor (or NN) query over all restaurants in the city, and find the closest one to your current location (which is a query point).

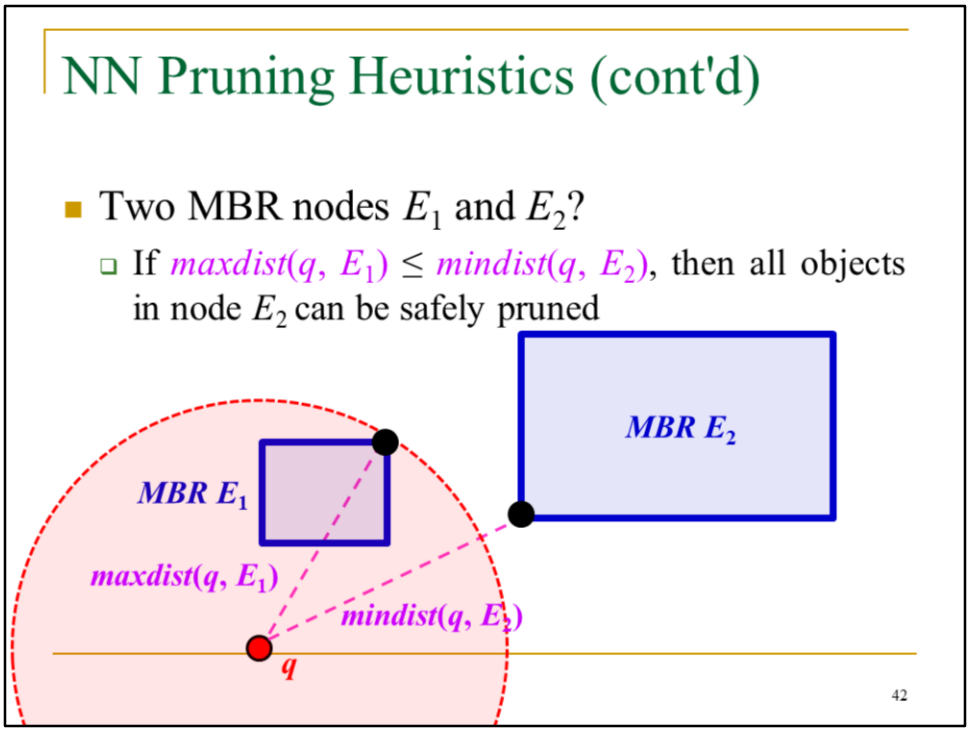
Formally, given a query point q, and a spatial database containing data objects o_i, a nearest neighbor query retrieves an object o_i that is the closest to query point q.

We can definitely enumerate all objects in the database, compute their distances to q, and obtain the one with the smallest distance. However, this is not efficient for large-scale database.



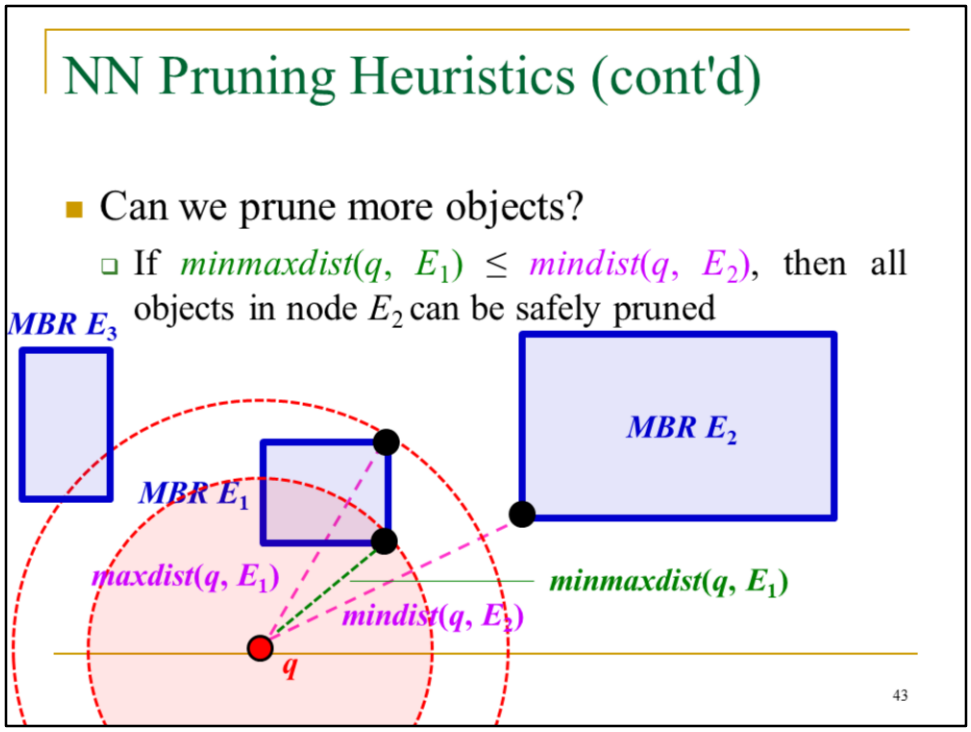
Therefore, we need to design some pruning heuristics to quickly filter out false alarms of query answers and reduce the problem search space. Our basic idea is also to utilize MBRs in the R-tree index to fast eliminate a group of objects in each MBR (without even checking them).

In particular, assume that we have encountered an object o. If we find that the minimum distance from query point q to MBR E is greater than or equal to the distance from q to object o, then any object inside MBR E can never become the nearest neighbor of q (due to the existence of object o). In this case, MBR node E can be safely pruned.



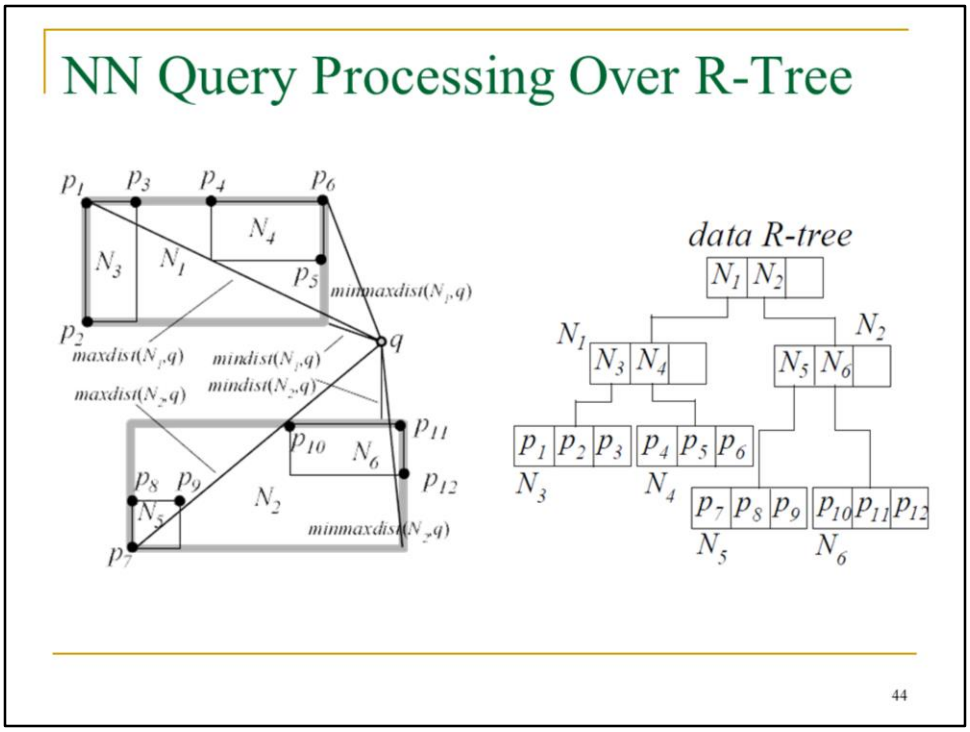
Previously, we say that we have encountered an object o, which is used for pruning the MBR. However, during the index traversal, what if we are still traversing the nodes of the R-tree and have not accessed the objects. Can we still prune MBR nodes? The answer is yes.

As shown in the figure, we have two MBR nodes E_1 and E_2. Assume that we have encountered MBR E_1. In this case, we can still use MBR E_1 to prune E_2, if the maximum distance from q to E_1 is smaller than or equal to the minimum distance from q to E_2. In other words, we can draw a circle centered at q with radius maxdist(q, E_1). If MBR E_2 is completely outside the circle, then we can safely prune MBR node E_2. Intuitively, the distance from q to any object in E_2 is always greater than or equal to that from q to that in E_1, so objects in E_2 can never be NN answers. We thus can safely prune MBR node E_2.

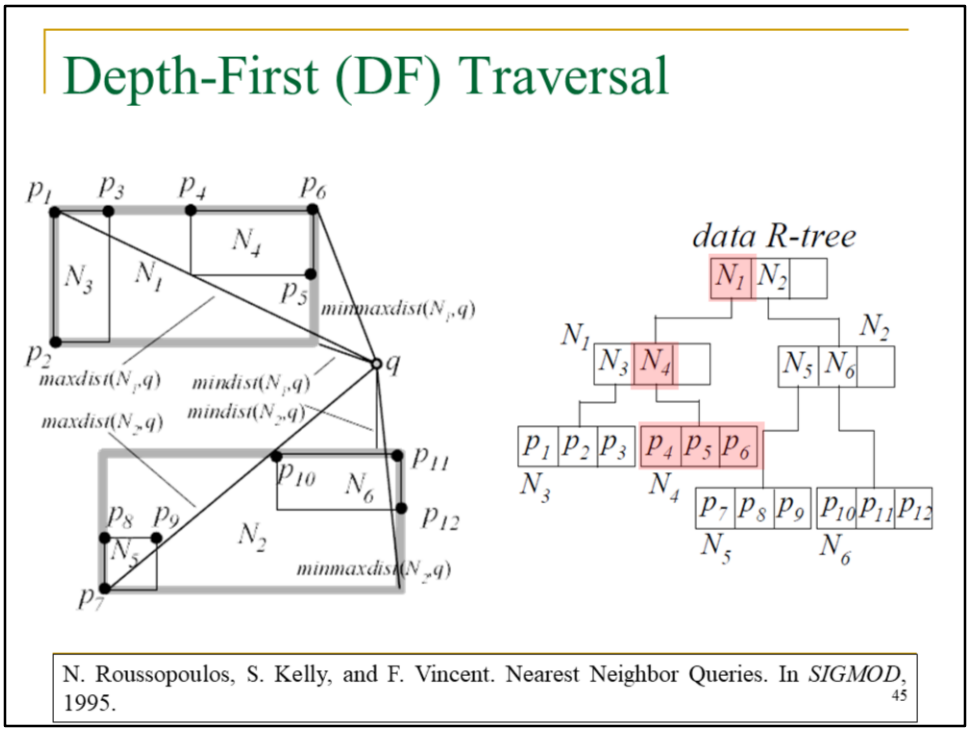


One more question, can we achieve even higher pruning power? The answer is yes.

According to the definition of MBR, MBR is the minimum bounding rectangle of all objects inside MBR. Therefore, for each edge of the MBR, there must exist at least one object on the boundary. Instead of using maximum distance from q to E_1, we now can use the minmaxdist(q, E_1), which is the smallest maximum distance from q to each edge/boundary of MBR E_1. We can see that, the minmaxdist() is much smaller than maxdist(). Similarly, we can draw a circle centered at query point q, and with radius minmaxdist(q, E_1). In this case, we can obtain a smaller circle (compared with the one with maxdist()), which will lead to higher pruning power. As shown in the figure, MBR E_3 cannot be pruned by using maxdist(), but can be pruned by minmaxdist().



Next, we will discuss how to perform NN query processing over the R-tree index. Given a query point q, we would like to traverse the R-tree such that the nearest neighbor of q can be efficiently retrieved. There are different approaches to traverse the R-tree structure. Existing works for NN considered depth-first and best-first index traversal.

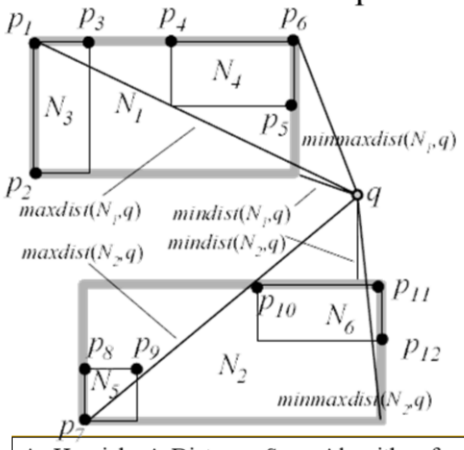


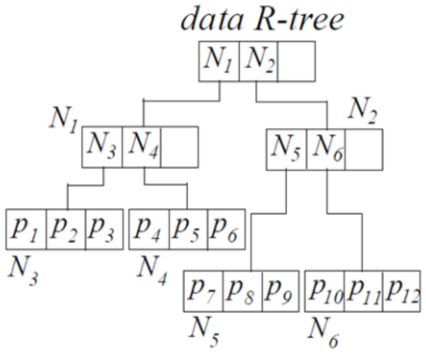
For the depth-first (DF) traversal, we start from the root of the R-tree, and always select one node entry N with the smallest mindist(q, N) to descend. Here, we select entry with the smallest mindist(), since objects in this entry might have closer distance than other entries. Each time we encounter a node, we will apply our pruning method to filter out those MBRs that can never be NN answers (i.e., far away from query point q).

As shown in this figure, we traverse the R-tree by accessing nodes N_1, N_4, and objects p_4~p_6. Once we reach the leaf node, we will recursively trace back to the parent node and traverse other children.

Best-First (BF) Traversal

- Use a minimum heap H to traverse the R-tree

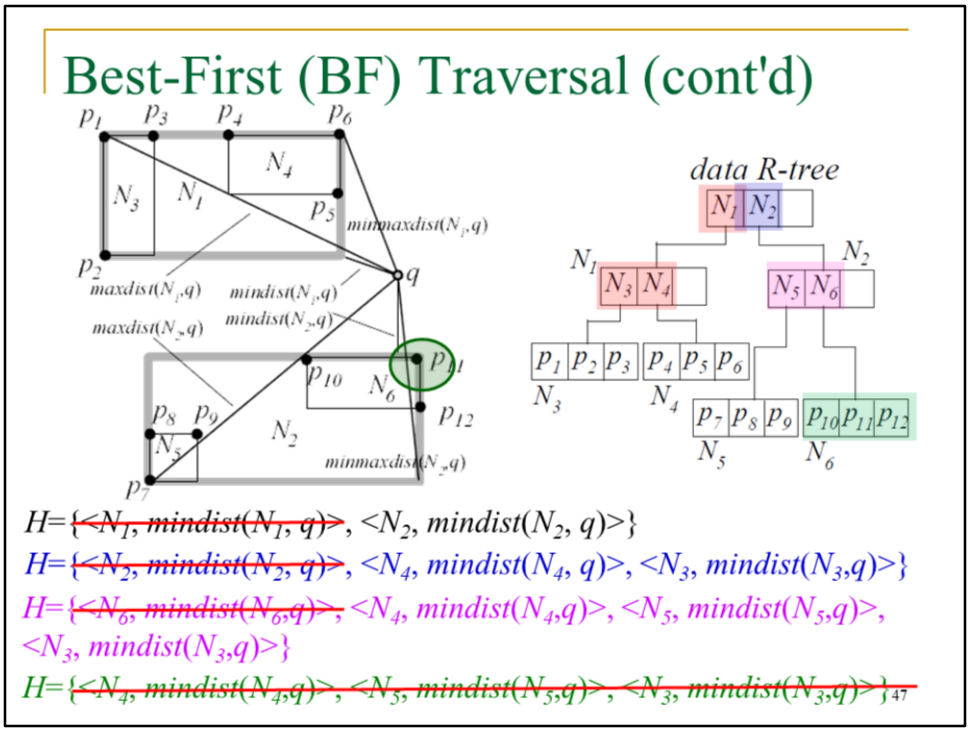




A. Henrich. A Distance Scan Algorithm for Spatial Access Structures. In *ACM GIS*, 1994.

46

Another traversal style is the best-first (BF) traversal. It uses a minimum heap to traverse nodes in the R-tree with the best order. Each heap entry has the form (N, key), where N is the MBR node, and key defines the node traversal order, given by the minimum distance from query point q to MBR N.



As shown in this figure, initially, we insert two entries N_1 and N_2 of the R-tree root into the heap. Then, the minimum heap always pops out an entry with the smallest key, in this example, N_1 is popped out and its children N_3 and N_4 are accessed. Since N_3 and N_4 cannot be pruned, we will add them to the heap.

Next, heap entry N_2 with the minimum key is popped out, and its children N_5 and N_6 are added to the heap. Finally, we pop out N_6 and access its children, objects p_10, p_11, and p_12. Object p_11 has the smallest distance to q, so we keep it as a candidate.

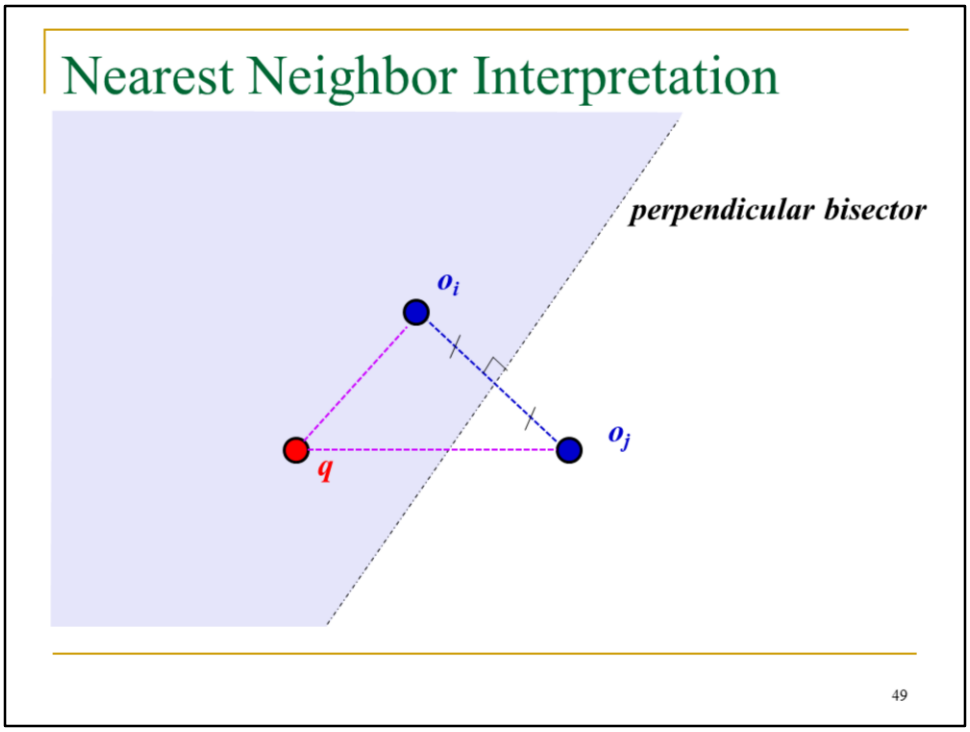
We find that all entries in the heap have the keys greater than the distance from q to p_11. Therefore, we do not have to access other entries in the heap, and can return p_11 as the NN answer.

Best-First Algorithm

```
best_dist_so_far = +∞; S_cand = Φ;
Initialize a minimum heap H = Φ;
Insert <root(I), mindist(q, root(I))> into heap H
while (H is not empty)
{
        (E, d_min) = pop-out (H)
        if (d_min > best_dist_so_far), then terminate;
        if E is a leaf node
          for each object o in E
            compute dist(q, o)
            if (dist(q, o) < best_dist_so_far)
              S_cand = {o}; best_dist_so_far = dist(q, o);
        else // non-leaf node
          for each entry E_i in E
            compute mindist(q, E_i)
            if (mindist(q, E_i) < best_dist_so_far)      // pruning
              insert (E_i, mindist(q, E_i)) into heap H
              update best_dist_so_far with minmaxdist(q, E_i)
}
```

48

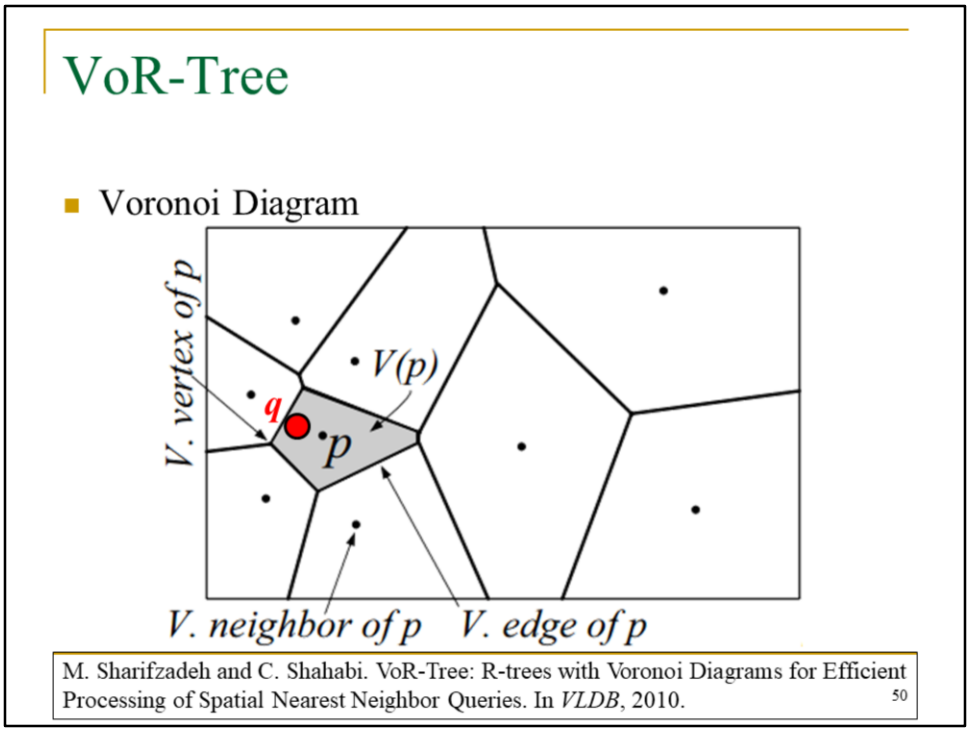
Here is the pseudo code of the best-first algorithm. It is a more efficient algorithm than the DF traversal algorithm.



Next, we will consider another interpretation of the nearest neighbor query.

Given two objects o_i and o_j, we can draw a perpendicular bisector between o_i and o_j, which divides the data space into two halfplanes. If a query point q falls into a halfplane containing o_i (shaded one in the figure), then o_j can never be the nearest neighbor of q. The reason is that, for any point within the shaded area, its distance to o_i is always smaller than that to o_j (due to the property of the perpendicular bisector).

Inspired by this interpretation, Voronoi diagram is used for a newly designed tree index for NN queries.



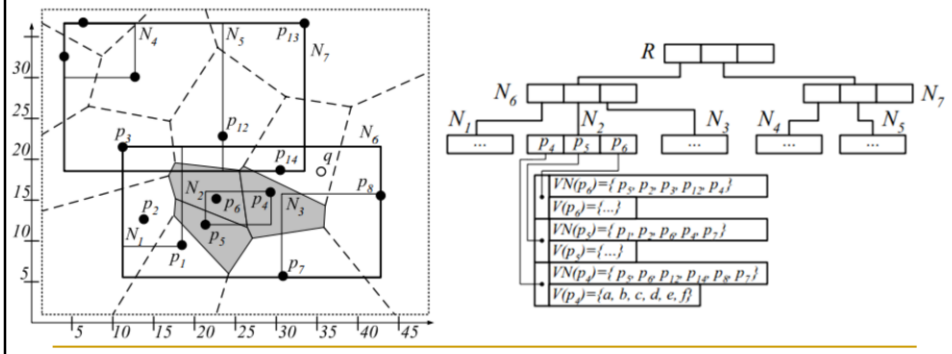
http://infolab.usc.edu/papers/VorTree.pdf

In particular, the figure shows an example of a Voronoi Diagram. Given any data point p in the space, we draw perpendicular bisectors between p and its spatially close neighbors, so that a Voronoi cell V(p) can be obtained, as shown in the shaded region. We can prove that if a query point q falls into this Voronoi cell V(p), then object p must be the nearest neighbor of q. Other objects cannot be NN answers.

Observing this property, in the literature, VoR-tree is proposed to offline pre-compute Voronoi cells of objects in the data space, and then use R-tree index to store these Voronoi cells.

VoR-Tree Structure

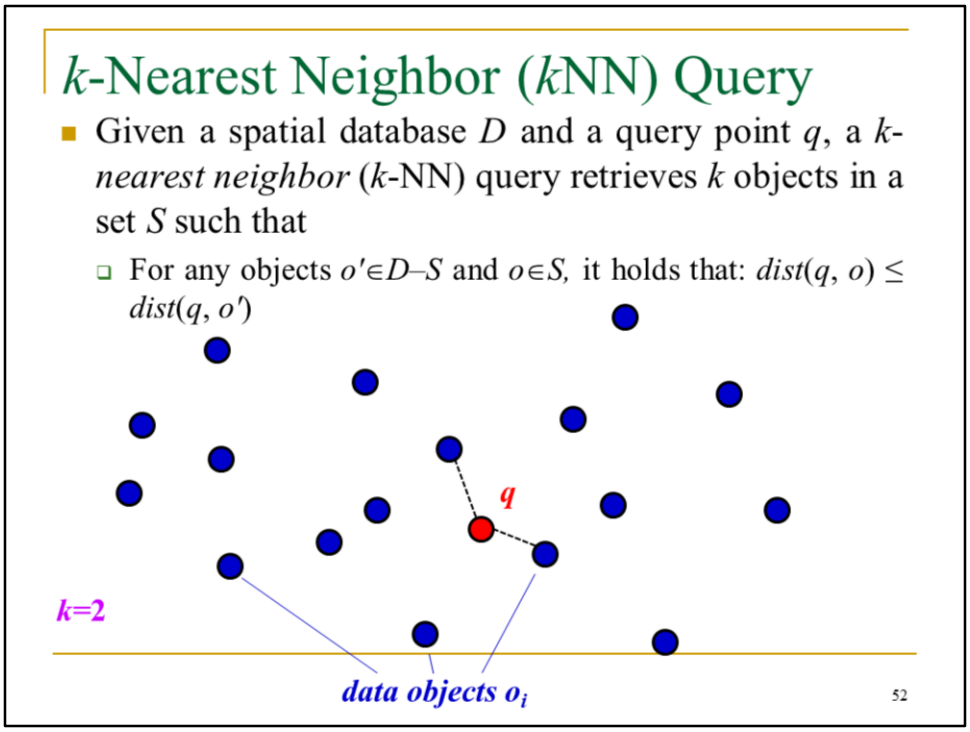
- Leaf nodes: data objects with their Voronoi Diagrams



51

Here are the details of the VoR-tree structure. We compute Voronoi cells of objects, and insert them into an R-tree, which results in the VoR-tree.

Then, for any NN query, we only need to search VoR-tree to obtain a Voronoi cell V(p) containing the query point q, and return the corresponding object p as the NN answer.



Nearest neighbor query is a special case of k-nearest neighbor (kNN) query, where k=1. For the general kNN query, we want to retrieve k objects from the database such that they are the closest to the query point q in the database.

In this figure, if k=2, then the kNN query returns the closest two objects of q as the kNN answer.

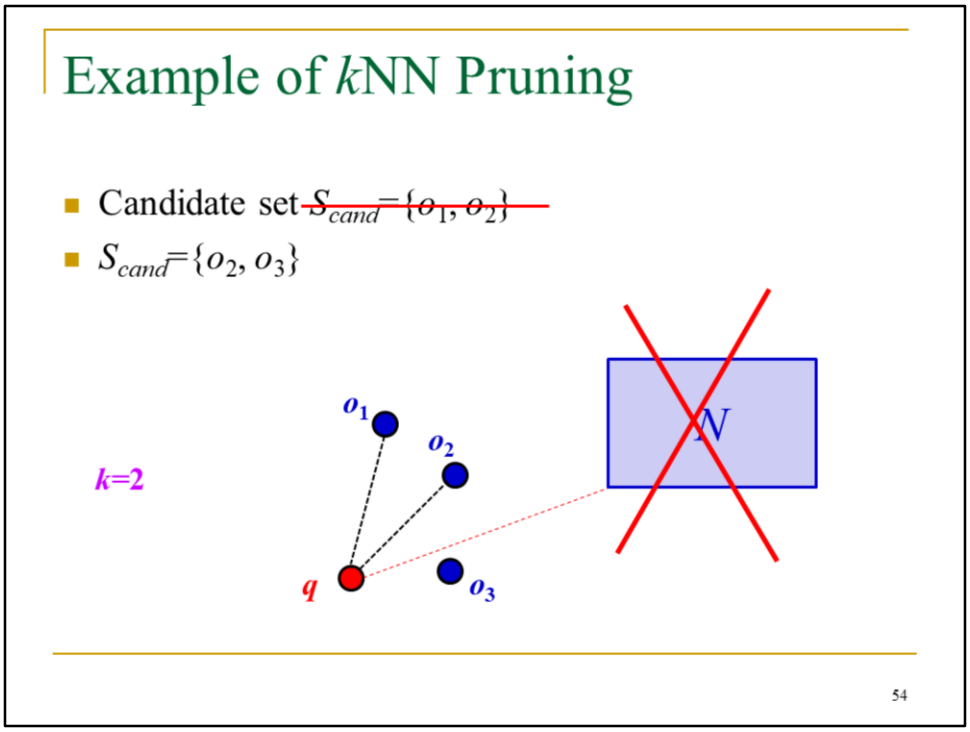
*k*NN Pruning Heuristics

- Let *best_so_far* be the *k*-th smallest distance from *q* to candidates we have seen so far
- For any object *o*, if it holds that ***dist(q, o)*** ≥ ***best_so_far***, then object *o* can be safely pruned (Why?);
- Otherwise, we need to update the candidate set S_{cand} (adding *o* and removing the one with the maximum distance), and set a new *best_so_far* value

53

Similar to NN query, it is not efficient to scan the entire spatial database to retrieve kNN answers, when the database is of large scale. Therefore, pruning methods are necessary to reduce the problem search space. Assume that best_so_far is the k-th smallest distance from query point q to all the candidates we have seen so far. Then, then for any object o, if it holds that the distance from q to o is greater than or equal best_so_far, then object o can be safely pruned. This is because, object o is farther than at least k candidates from q, and can never become a kNN answer.

If object o cannot be pruned, we need to add it to the candidate set S_{cand}, evict an object with the maximum distance from S_{cand}, and update the best_so_far value.



For the pruning on the level of the MBR node, assume that we have accessed candidate objects o_1, o_2, and o_3. Since k=2, the candidate set S_{cand} maintains two candidates o_2 and o_3 with the smallest distances to q we have seen so far. Then, we can set the best_so_far to the second nearest candidate in S_{cand}, that is, dist(q, o_2).

If we have an MBR node N whose mindist(q, N) is greater than best_so_far, it indicates that all objects in N cannot be 2NN answers (due to the existence of objects o_2 and o_3). Thus, MBR node N can be safely pruned.

Outline

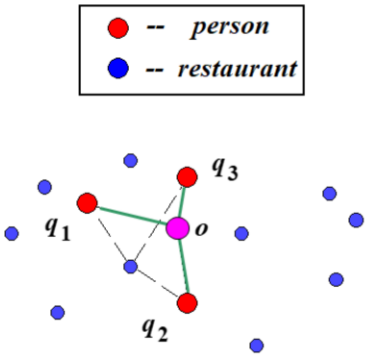
- Introduction of Real-World Application Data
- Query Types
  - Range Query
  - Nearest Neighbor (NN) Query
  - *k*-Nearest Neighbor (*k*NN) Query
  - Group Nearest Neighbor (GNN) Query
  - Reverse Nearest Neighbor (RNN) Query
  - …

55

Next, we will talk about group nearest neighbor (GNN) query.

A Motivation Example of Group Nearest Neighbor Query

- In a city, there are many restaurants
- Three people want to have lunch together at a restaurant
- Problem:
  - To find a restaurant in the city that minimizes its total distances to these 3 people



D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group Nearest Neighbor Queries. In *ICDE*, 2004.

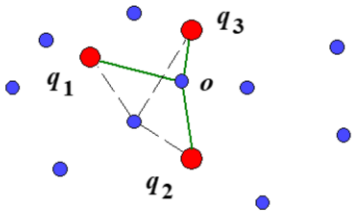
56

The group nearest neighbor (GNN) query was proposed in an ICDE 2004 paper. We first give the motivation example of this GNN query.

Consider a group of friends who want to select a close restaurant to have lunch together. Each person is located at different positions in the city, and we want to minimize the total traveling cost (e.g., gas fees) for this group of friends. In this case, the group nearest neighbor (GNN) query can be used for retrieving the best restaurant that suits their needs.

Other GNN Applications

- Image Retrieval
  - Find an image in the database that is similar to a group of user-specified query images
- Geographic information system
- Mobile computing applications



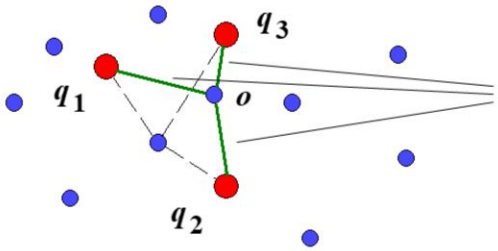
57

There are many other applications such as image retrieval. Given feature vectors extracted from images in image database, we may want to find an image that are similar to multiple query images. In this case, we can also issue the GNN query.

GNN queries can be used for other applications like GIS system, mobile computing services, and so on.

Group Nearest Neighbor (GNN) Query

- Group Nearest Neighbor (GNN) Search [ICDE04]
  - Given a database D and a set, Q, of query objects q_1, q_2, …, and q_n, a GNN query retrieves an object $o \in D$ that has the smallest *summed distance* to Q, i.e. $\Sigma_{i=1\sim n}\, dist(q_i, o)$



find a data object o that minimizes $\Sigma_{i=1\sim 3}\, dist(q_i, o)$

D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group Nearest Neighbor Queries. In *ICDE*, 2004.

58

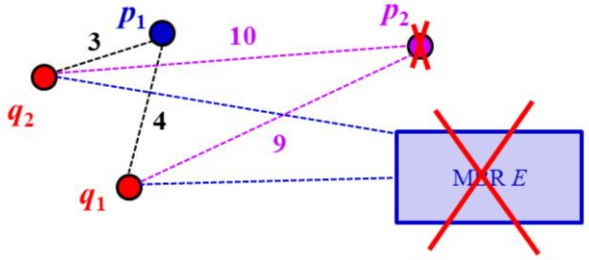
Formally, a GNN query retrieves an object that has the smallest summed distance to a set of query points.

As shown in the figure, we have 3 query points q_1, q_2, and q_3. The GNN query finds object o that minimizes the total summed distance from o to all query points q_i.

In order to solve this GNN query, one straightforward method is to scan the entire database, and for each object compute the summed distance from object to the query set. Finally, we return the object with the smallest summed distance as the GNN answer. This straightforward method is not efficient for large-scale spatial database. Thus, we need to think of pruning techniques to reduce the problem search space, that is, we only need to access a very small subset of candidates, rather than the entire database.

Basic Pruning Idea for GNN Query

- Maintain *an upper bound threshold of the smallest summed distance*, $\Sigma_{i=1\sim n}\, dist(q_i, o)$, for pruning other objects/MBRs
  - Upper bound in the example = ?
- If the upper bound is smaller than $\Sigma_{i=1\sim n}\, mindist(q_i, E)$, then all objects in E can be safely pruned

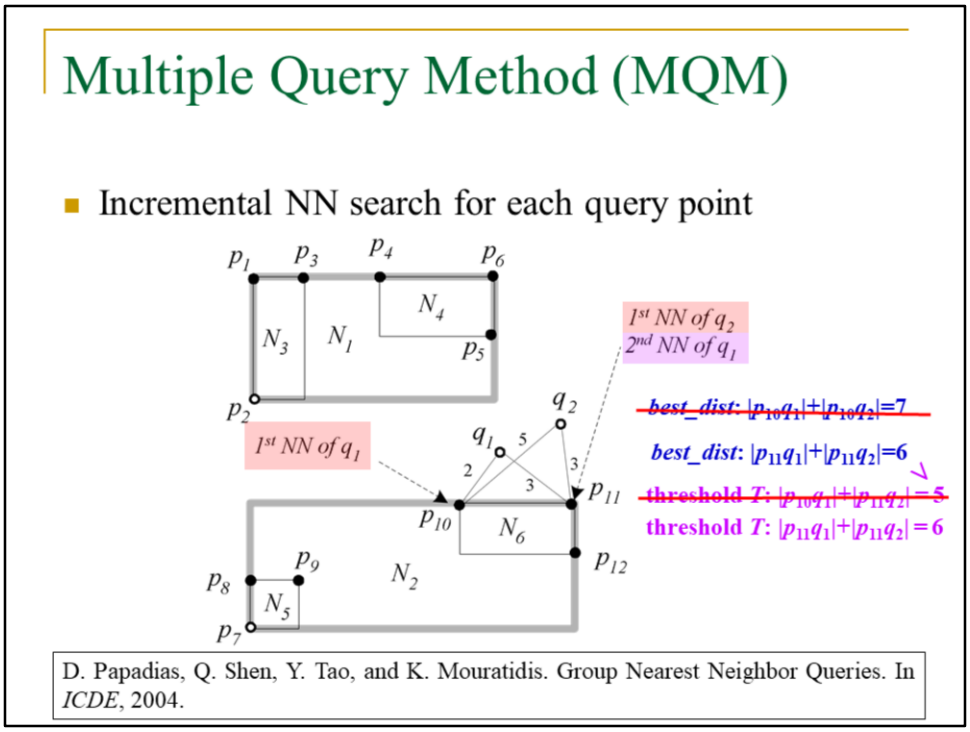


59

The basic pruning idea for the GNN query is as follows. If we can obtain an upper bound of the smallest summed distance from a candidate to the query set, then we can use it to prune other objects or MBRs with larger summed distances (intuitively, those objects/MBRs can never become the GNN answers).

Now one question is how to obtain such an upper bound. In fact, as long as we obtain any object o, and calculate the summed distance from o to Q, then this summed distance can be used as the upper bound.

As an example in the figure, if we obtain the summed distance of object p_1, which is given by 7 (=3+4), then we can use it as a threshold to prune other objects/MBRs. For instance, object p_2 have the summed distance 19 (= 10 + 9) farther than threshold 7, and can be safely pruned. Moreover, MBR E have the minimum summed distance to q_1 and q_2 greater than 7, and thus can be also pruned.



To tackle the GNN problem, existing works proposed three different approaches. The first approach is called multiple query method (or MQM), which incrementally computes the NN of each query point, and use the retrieved NNs to prune other nodes/objects until no candidates can be found.

As an example in the figure, assume that we have two query points q_1 and q_2. We first find the NN of q_1, which is p_10, and obtain the best summed distance so far, best_dist = 7. Then, we find the NN of q_2, which is p_11, and has the summed distance 6 (less than that of p_10). Thus, we update the best_dist to 6. Since we access NNs of q_1 and q_2 in increasing order of distances, objects accessed later cannot have distances greater than threshold T= |p_10q_1| + |p_11q_2| = 5. Since best_dist > threshold 5, we continue to search the second NN of q_1, which is p_11 and has been accessed before. Now the threshold T becomes 6, which is exactly equal to best_dist. Therefore, we cannot find any more objects with the summed distance smaller than 6, and p_11 is our GNN answer.

Single Point Method (SPM)

- MQM has the problem of multiple NN queries accessing the same objects
- Instead, SPM computes with a centroid q of the query set Q
  - A centroid $q(x, y)$ minimizing $dist(q,Q) = \sum_{i=1}^{n} \sqrt{(x - x_i)^2 + (y - y_i)^2}$

$$\begin{cases} \frac{\partial\, dist(q,Q)}{\partial\, x} = \sum_{i=1}^{n} \frac{x - x_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2}} = 0 \\ \frac{\partial\, dist(q,Q)}{\partial\, y} = \sum_{i=1}^{n} \frac{y - y_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2}} = 0 \end{cases}$$

$$x = x - \eta \frac{\partial\, dist(q,Q)}{\partial\, x} \text{ and } y = y - \eta \frac{\partial\, dist(q,Q)}{\partial\, y}$$

where η is a step size.

61

The MQM method needs to incrementally compute the NNs of multiple query points, and the same objects may be accessed multiple times, which is not efficient. Therefore, the second method, single point method (SPM), computes with a single centroid of the query set Q, instead of the query set Q directly.

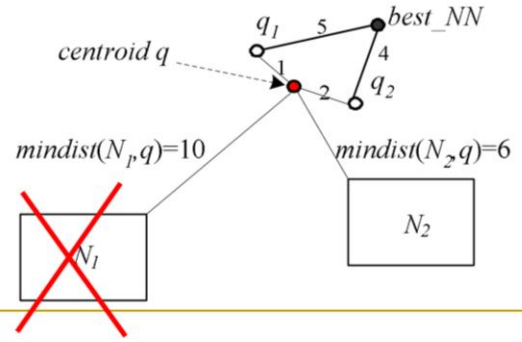
The coordinates of the centroid are given by the formulae, as given in the slide.

Pruning Conditions for SPM

- An MBR node N can be pruned, if it holds that:

$$mindist(N,q) \geq \frac{best\_dist+dist(q,Q)}{n}$$

  - where *best_dist* is the distance of the best GNN found so far



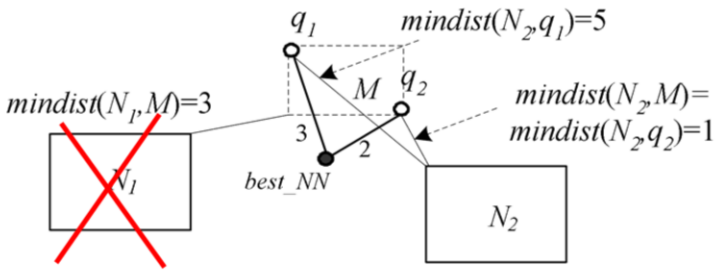
62

Once we have the centroid of the query set Q, we can utilize it to derive the SPM pruning condition. Any MBR node N can be pruned if the minimum distance from q to N is greater than or equal to the RHS of the inequality. To prove this, it needs to use the triangle inequality to derive the inequality. Intuitively, RHS of the inequality is an upper bound of the average distance from object to a query point.

In the figure, RHS of the inequality is given by ((5+4)+(1+2))/2 = 6, which is smaller than mindist(N_1, q) = 10. In this case, MBR node N_1 can be safely pruned.

Minimum Bounding Method (MBM)

- MBM uses the minimum bounding rectangle M of Q (instead of the centroid q) to prune the search space

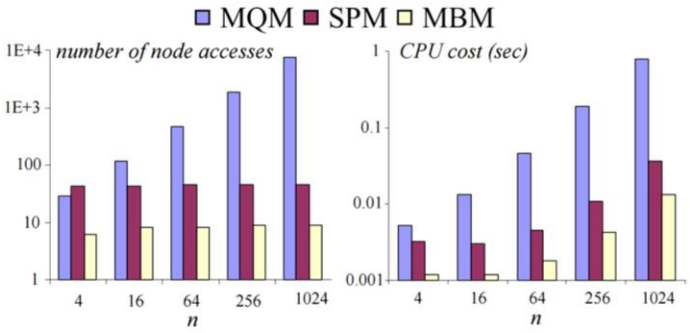


63

The SPM uses centroid of query points to enable the pruning via the triangle inequality, however, the derived distance bounds may result in low pruning power. Therefore, authors proposed another method, called minimum bounding method (MBM). The basic idea is to use an MBR, M, to bound all query points in Q. As shown in the figure, we only need to compute the mindist(N_1, M), which is the minimum possible distance from N_1 to any query point in Q. If mindist(N_1, M)*2 is greater than or equal to best_dist, then we can safely prune node N_1.

Experimental Results for GNN

- Comparisons of three approaches
  - PP data sets: 24,493 populated places in North America



(a) NA vs. n (PP dataset) (b) CPU vs. n (PP dataset)

D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group Nearest Neighbor Queries. In *ICDE*, 2004.

Here are the experimental results, comparing the 3 approaches, over PP data sets, which contains about 24K places in North America. From the experimental figures, we can see that MQM takes higher I/O and CPU costs than SPM, and the MBM approach performs the best among the 3 approaches.

Outline

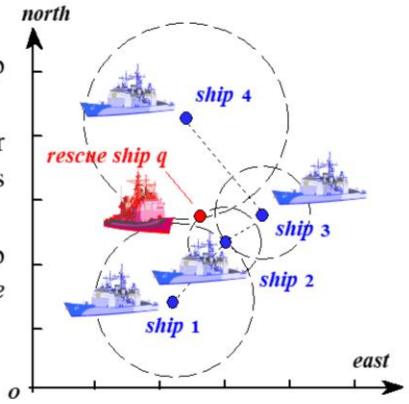
- Introduction of Real-World Application Data
- Query Types
  - Range Query
  - Nearest Neighbor (NN) Query
  - *k*-Nearest Neighbor (*k*NN) Query
  - Group Nearest Neighbor (GNN) Query
  - Reverse Nearest Neighbor (RNN) Query
  - …

65

Next, we will talk about reverse nearest neighbor query.

Reverse Nearest Neighbor Query (RNN)

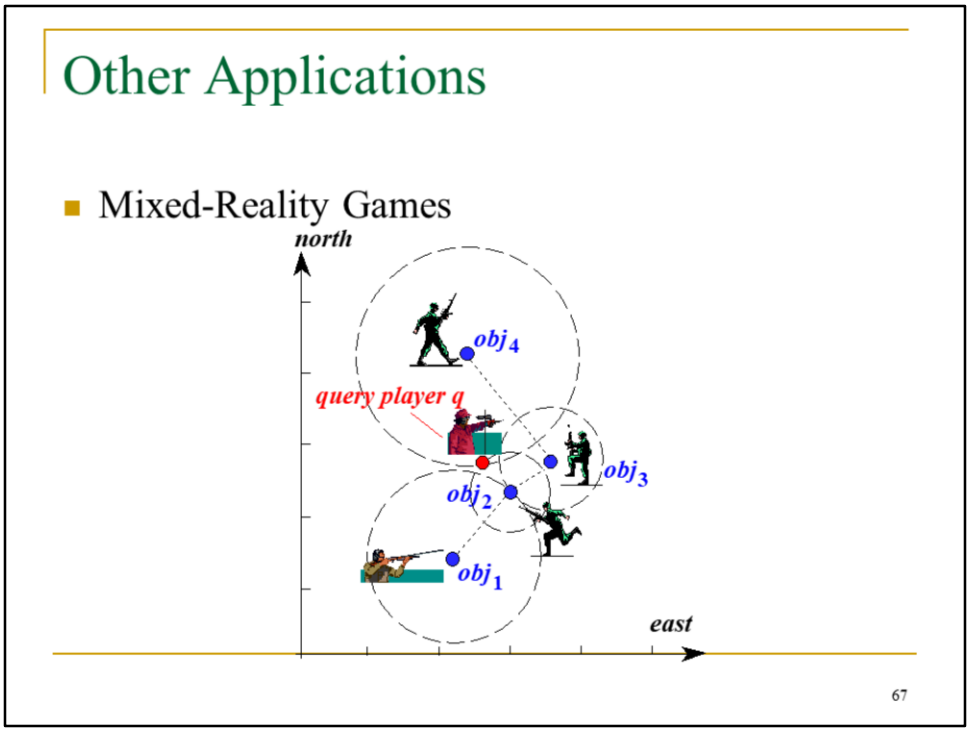
- Rescue tasks in oceans
  - In the case of emergency, a ship will ask its nearest ship for help
  - A rescue ship needs to monitor those ships that have itself as their nearest neighbors
  - In other words, the rescue ship needs to obtain its *reverse nearest neighbors* (RNNs)



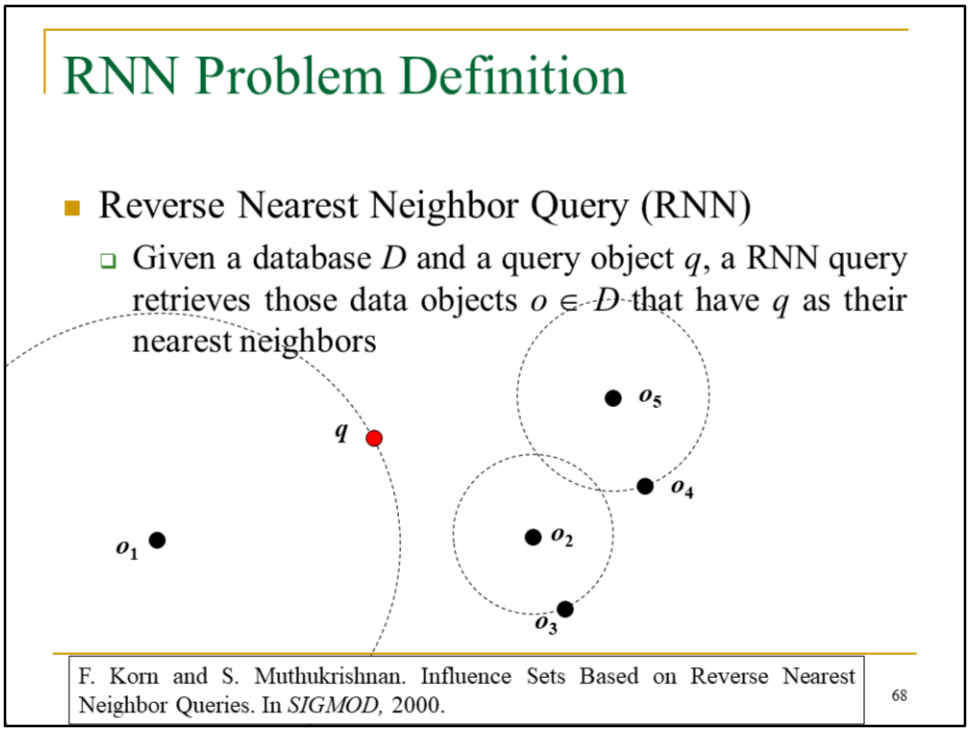
66

We first introduce the motivation example of the reverse nearest neighbor (RNN) query. In the application of rescue tasks in oceans, if a ship is in danger, it tends to ask for help from its nearest ship. Therefore, a rescue ship needs to keep track of those ships that have itself as their nearest neighbors.

As shown in the figure, q is a rescue ship, and the nearest neighbor of Ship #2 is exactly the rescue ship q. We say that, Ship #2 is the reverse nearest neighbor (or RNN) of q.



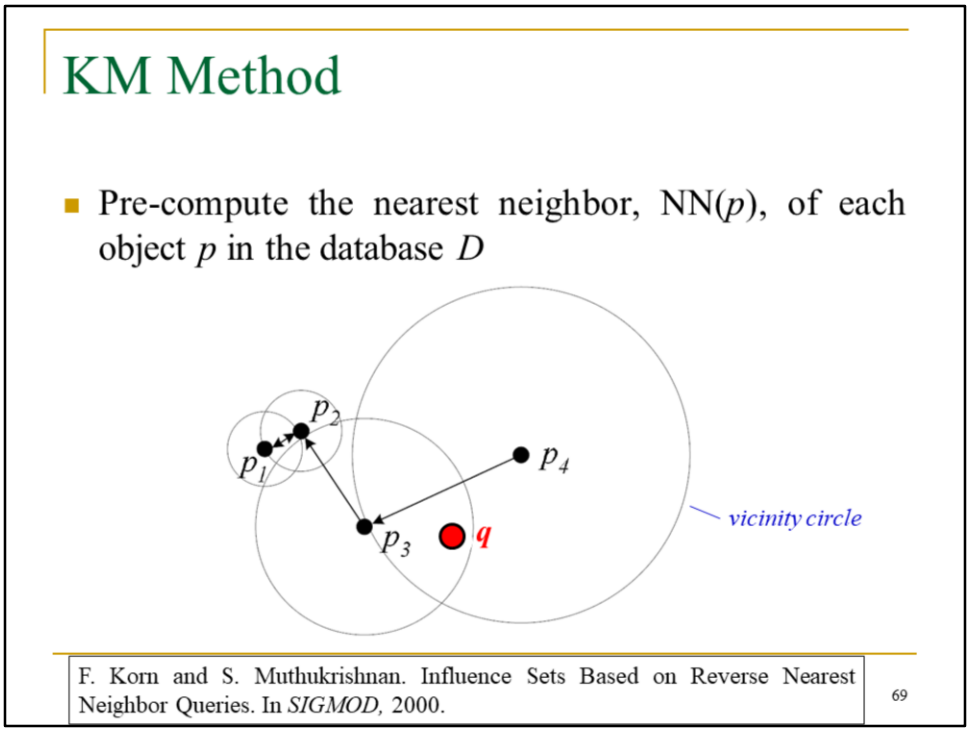
Similarly, in the mixed-reality games, each player tends to shoot one's nearest player. Therefore, if I am a query player q, I need to know who tend to shoot me, in other words, who have me as their nearest neighbors. Or I would like to find out my reverse nearest neighbor (RNN) of my current location q.



Formally, an RNN query retrieves those objects from the database such that they have a given query point q as their nearest neighbors.

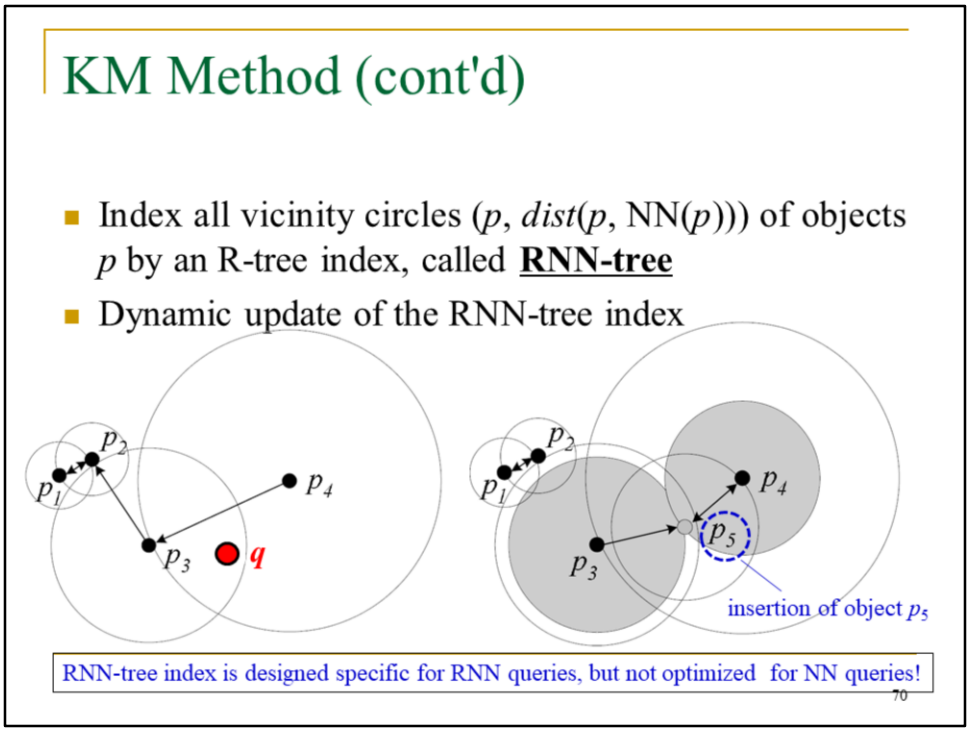
As shown in the example, query point q is the nearest neighbor (NN) of object o_1, thus, o_1 is the reverse nearest neighbor (RNN) of q.

Note that, NN of q and RNN of q are two different sets. As shown in the example, object o_1 is the RNN of q, whereas object o_2 is the NN of q.



To tackle the RNN problem, KM method precomputes the nearest neighbor of each object in the database. For example, in the figure, the nearest neighbor of p_4 is p_3. Therefore, we can draw a circle, called vicinity circle, centered at p_4 with radius equal to dist(p_4, p_3). If a query point q falls inside the vicinity circle, then that means q becomes the nearest neighbor of p_4, in other words, p_4 is the reverse nearest neighbor of q. This way, we can directly retrieve those objects whose vicinity circles contain the query point q.

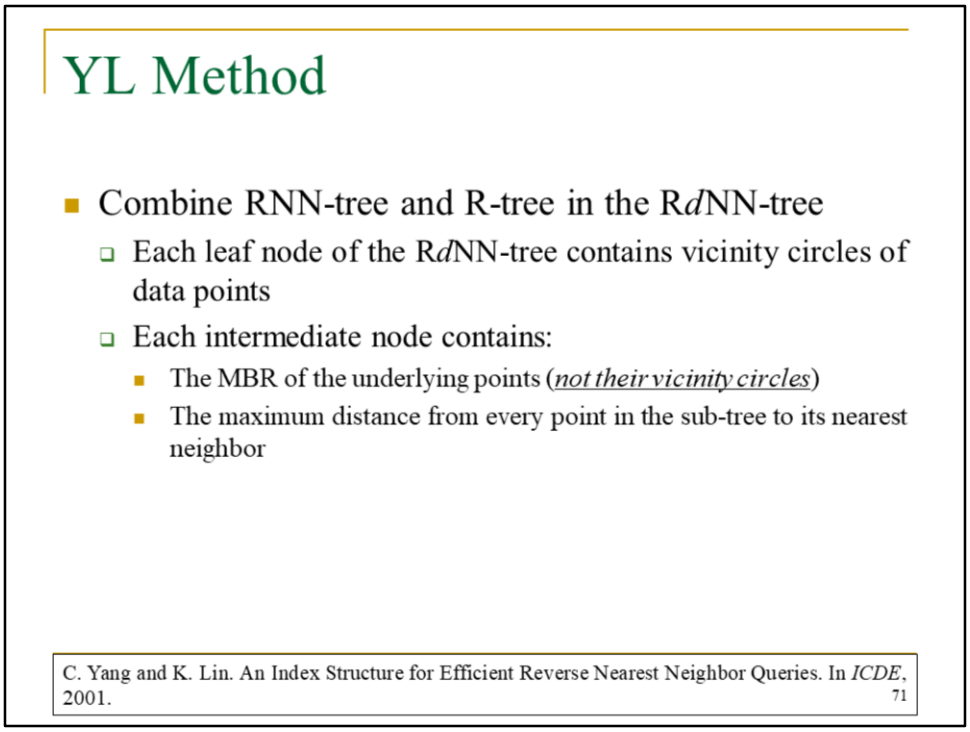
In this example, p_3 is also the RNN of q, since q falls into the vicinity circle of p_3.



As discussed in the example, it is important to efficiently check if a query point q falls into vicinity circles. Therefore, KM method indexes all vicinity circles of objects in the database by an R-tree index, called RNN-tree, and answers RNN queries by traversing the RNN–tree.

To dynamically update the RNN-tree index, for example, to insert a new object p_5, we first find the vicinity circles that contain p_5, that is, the ones with p_3 and p_4. Then, we will shrink the vicinity circles of p_3 and p_4 (since p_5 becomes their new nearest neighbors). Moreover, we will also find the nearest neighbor of new object p_5 in the database, and set its own vicinity circle.

However, the problem for RNN-tree is that, RNN-tree index is specifically designed for RNN queries, and cannot optimally support other queries like NN query.

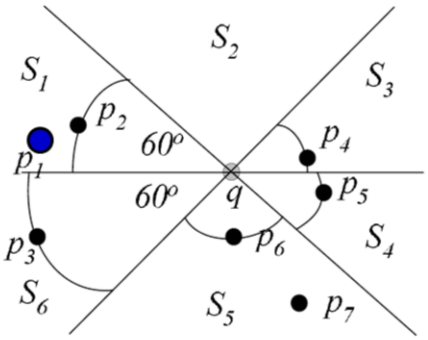


Therefore, later on YL method was proposed in ICDE 2001, which combines the RNN-tree with R-tree, resulting in a so-called RdNN-tree.

In this RdNN-tree index, each leaf node stores vicinity circles of data points, and each non-leaf node contains the MBR over all data points (not vicinity circles) under this node, as well as an aggregate which is the maximum radius of vicinity circles under this node.

SAA Method

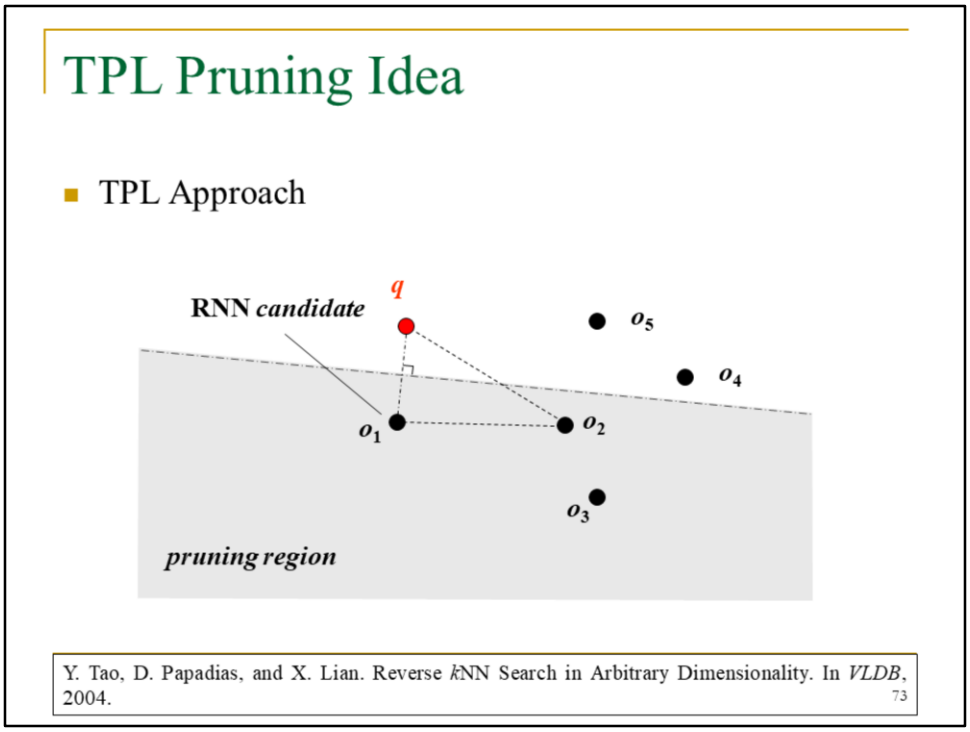
- SAA only applies to RNN in the 2D space



I. Stanoi, D. Agrawal, and A. Abbadi. Reverse Nearest Neighbor Queries for Dynamic Databases. In *SIGMOD* Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000. 72

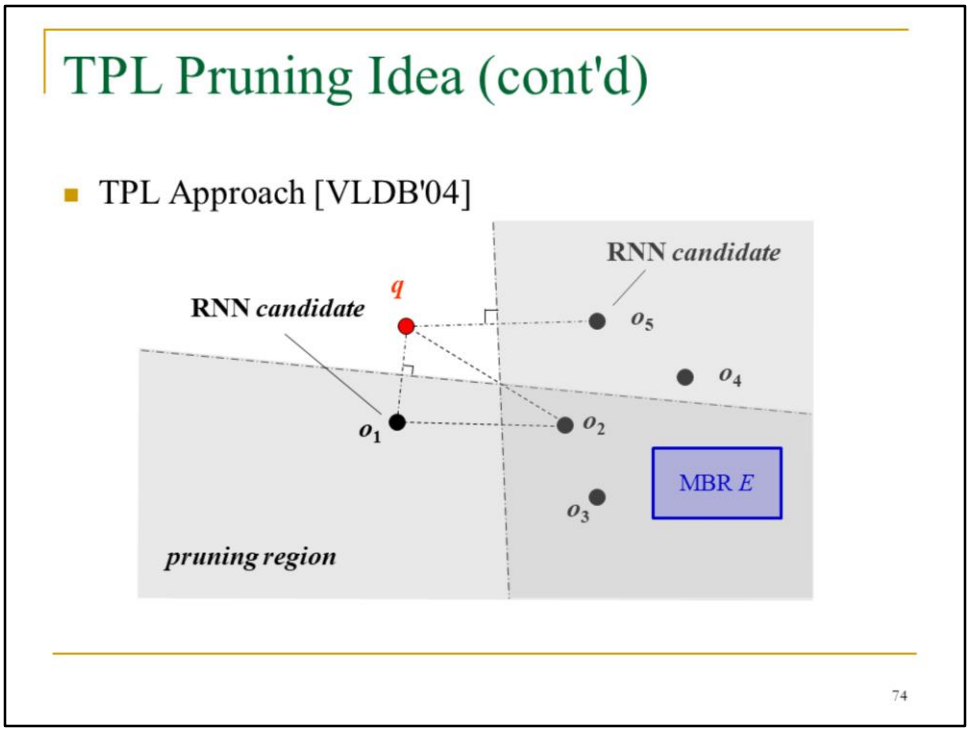
Another SAA method was proposed to solve the RNN problem in a 2D data space. It can be proved that in a 2D space, at most 6 RNNs can be obtained for a given query point. The basic idea of the SAA method is to divide the 2D data space into 6 pie regions, S_1 ~ S_6, w.r.t. q of the same size (with 60 degree angles). Then, within each constrained pie region, we conduct a nearest neighbor search, resulting in totally 6 NNs which are candidate RNN answers.

For example, in the constrained pie region S_1, we can obtain the nearest neighbor p_2 of q. For any other objects, say p_1, in region S_1, they can never be RNNs. Taking p_1 as an example, due to the existence of p_2, the nearest neighbor of p_1 cannot be q (since dist(p_1, p_2) <= dist(p_1, q)).

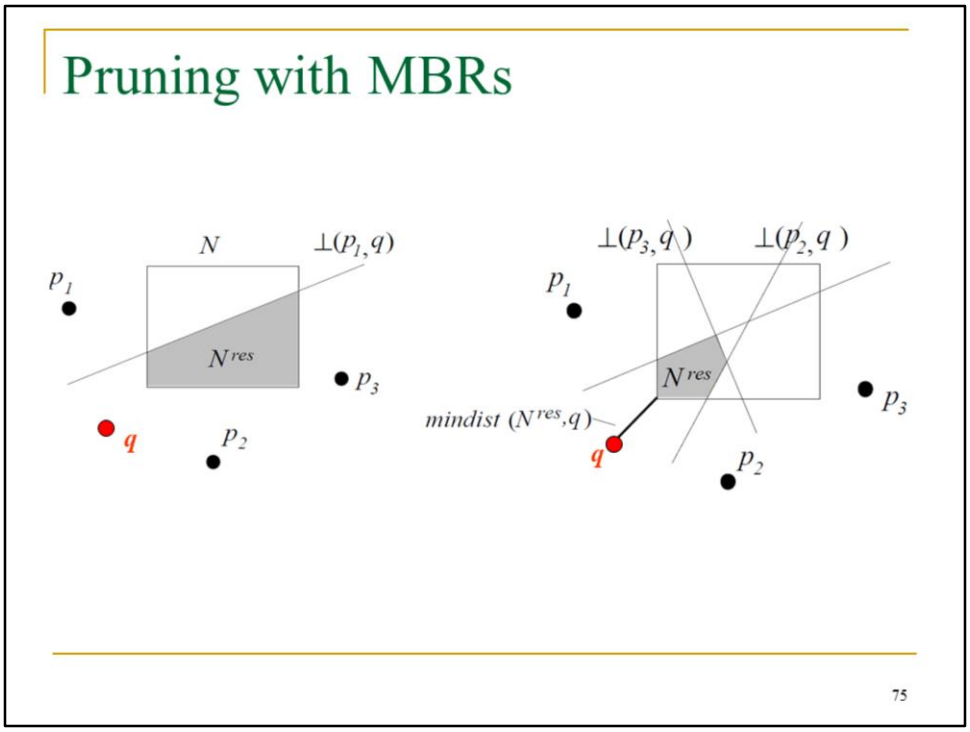


Another RNN approach, called TPL, designs a pruning region for filtering out false alarms of RNN candidates. We illustrate the basic idea of TPL using the example in the figure. Given an RNN candidate o_1 and a query point q, we can draw a perpendicular bisector between o_1 and q, which divides the data space into two halfplanes. Then, the halfplane containing the RNN candidate o_1 can be defined as the pruning region.

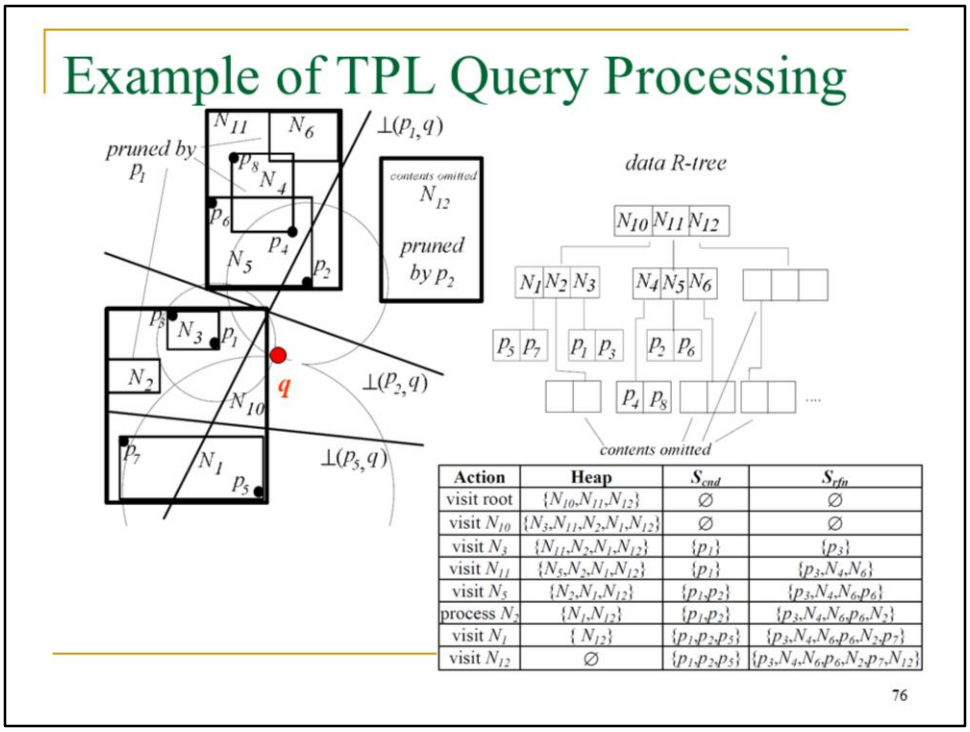
It can be proved that any object, for example, o_2, inside the pruning region can never be RNN and can be safely pruned. The reason is that the distance from o_2 to RNN candidate o_1 is guaranteed to be smaller than that from o_2 to q. Then, q cannot be the nearest neighbor of o_2, due to the existence of o_1, and in turn o_2 inside the pruning region cannot be the reverse nearest neighbor of q.



In practice, we have more than one candidate (e.g., candidates o_1 and o_5), and can define multiple pruning regions. This way, we can prune all objects or MBRs E inside any of the pruning regions.



If an MBR is partially inside the pruning region, TPL approach will only consider the residual MBR (as shown in the shaded region of the figure), that is, only access those children of the node intersecting with the residual MBR.



Action	Heap	S_{cnd}	S_{rfn}
visit root	$\{N_{10},N_{11},N_{12}\}$	$\varnothing$	$\varnothing$
visit N_{10}	$\{N_3,N_{11},N_2,N_1,N_{12}\}$	$\varnothing$	$\varnothing$
visit N_3	$\{N_{11},N_2,N_1,N_{12}\}$	$\{p_1\}$	$\{p_3\}$
visit N_{11}	$\{N_5,N_2,N_1,N_{12}\}$	$\{p_1\}$	$\{p_3,N_4,N_6\}$
visit N_5	$\{N_2,N_1,N_{12}\}$	$\{p_1,p_2\}$	$\{p_3,N_4,N_6,p_6\}$
process N_2	$\{N_1,N_{12}\}$	$\{p_1,p_2\}$	$\{p_3,N_4,N_6,p_6,N_2\}$
visit N_1	$\{N_{12}\}$	$\{p_1,p_2,p_5\}$	$\{p_3,N_4,N_6,p_6,N_2,p_7\}$
visit N_{12}	$\varnothing$	$\{p_1,p_2,p_5\}$	$\{p_3,N_4,N_6,p_6,N_2,p_7,N_{12}\}$

The TPL approach uses the R*-tree to index data objects in the spatial database, and traverse the index in a best-first manner to retrieve the RNN candidates by applying the pruning methods.

In particular, the RNN processing procedure is as follows. We maintain a minimum heap to traverse the R*-tree index, which stores nodes that may contain RNN answers. Each time we pop out a node with the smallest mindist() to query point q. Intuitively, those nodes close to query point may have higher pruning power to prune other objects/nodes.

Initially, we visit the root of the R*-tree, and add entries N_10, N_11, and N_12 to the heap. Then, node N_10 is first popped out from the heap, and its children N_1, N_2, and N_3 are entered into the heap. Next, node N_3 is popped out, and its child objects p_1 and p_3 are accessed. In this case, p_1 is a candidate, and the pruning region between p_1 and q can rule out p_3, which is moved to a refinement set.

For the next iteration, we pop out N_5, and obtain objects p_2 and p_6. While p_6 can be pruned by p_1 and moved to refinement set S_rfn, object p_2 cannot be pruned and is added to the candidate set S_cand. Next, N_2 can be pruned by the pruning region of p_1, and moved to the refinement set S_rfn. Then, N_1 is accessed, p_7 is pruned, and p_5 is added to the candidate set S_cand. Finally, node N_12 can be pruned by candidate p_2, and heap becomes empty.

As a result, objects p_1, p_2, and p_5 are RNN candidates, and we refine them by calculating their actual nearest neighbors for objects/nodes in the refinement set.

TPL Query Processing Algorithm

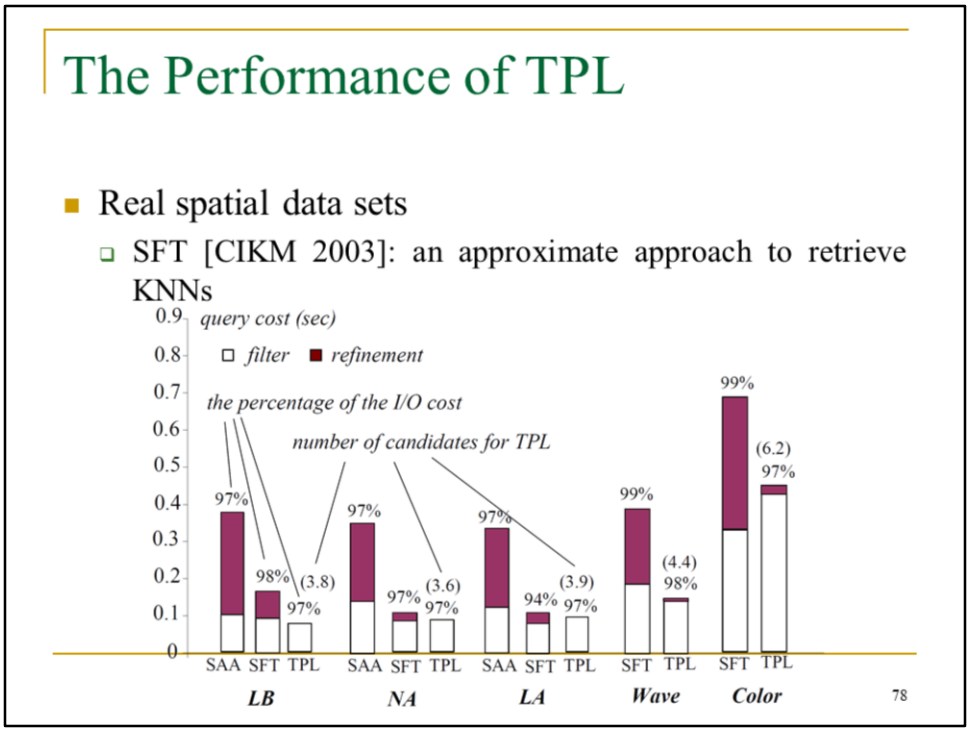
- Maintain a minimum heap H

Algorithm TPL-filter(q) /* q is the query point */
1. initialize a min-heap H accepting entries of the form (e, key)
2. initialize sets S_{cnd}=∅, S_{rfn}=∅
3. insert (R-tree root, 0) to H
4. while H is not empty
5. (e, key)=de-heap H
6. if (**trim**(q, S_{cnd}, e)=∞) then S_{rfn}=S_{rfn}∪{e}
7. else // entry may be or contain a candidate
8. if e is data point p
9. S_{cnd}=S_{cnd}∪{p}
10. else if e points to a leaf node N
11. for each point p in N (sorted on $dist(p,q)$)
12. if (**trim**(q,S_{cand},p)≠∞) then insert (p,$dist(p,q)$) in H
13. else S_{rfn}=S_{rfn}∪{p}
14. else // e points to an intermediate node N
15. for each entry N_i in N
16. $mindist(N_i^{resM}, q)$=**trim**(q, S_{cnd}, N_i)
17. if ($mindist(N_i^{resM}, q)$=∞) then S_{rfn}=S_{rfn}∪{N_i}
18. else insert (N_i, $mindist(N_i^{resM}, q)$) in H

End TPL-filter

77

Here is the pseudo code of the TPL query processing algorithm. If you are interested in this algorithm, please check the detailed descriptions in the TPL paper.



Singh, A., Ferhatosmanoglu, H., Tosun, A. High Dimensional Reverse Nearest Neighbor Queries. *CIKM*, 2003.

Here is the experimental result by comparing TPL with other competitors such as SAA and SFT (which is an approximate solution by retrieving a number of NNs of query point q as RNN candidates). From the experimental results, we can see that TPL approach can achieve the lowest query cost compared with the other two approaches on 5 real data sets, including LB, NA, LA, Wave, and Color histogram data. For the TPL approach, the number of candidates is around 3.6-6.2, which shows good pruning power of the pruning methods.