

Spatial Informatics
Prof. Soumya K. Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

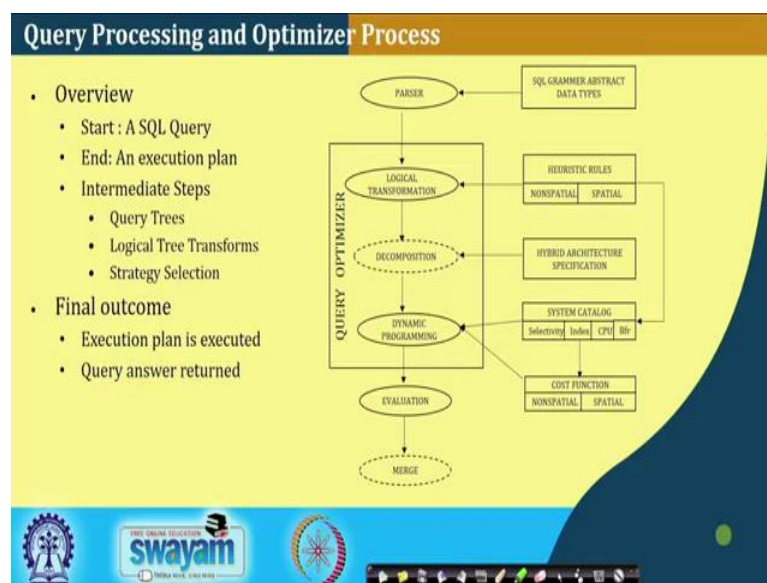
Lecture - 15
Spatial Query Processing / SQL (4)

Hello, so we will continue our discussion on Spatial Query Processing rather more on query optimization, query processing and optimization. As we; as we are discussing in last couple of lectures the spatial queries that or spatial database parse has a more has something different from our normal or traditional database with respect to the thing. It is not only IO intensive it is also data intensive compute both IO and compute intensive.

So, these two things has to be taken care when we do right. Rather what we have seen there are several spatial operations where for which there may be several strategies right. So, like I have a overlap operations so there may be different strategies of overlap when we use different type of operations like overlap between two polygon may have couple of algorithms, overlap between polygon and line can have different algorithms, overlap between or intersect between two lines and two lines may be having different things.

So, polygon line polygon polygon, polygon point etcetera all these topological relationship between them, when we have different type of algorithms and approaches for that right. For that it becomes more complex to handle these type of queries right and that is why we as we discussed earlier there is there are need of optimization in this case is more stringent it is there in a traditional database also.

(Refer Slide Time: 01:59)

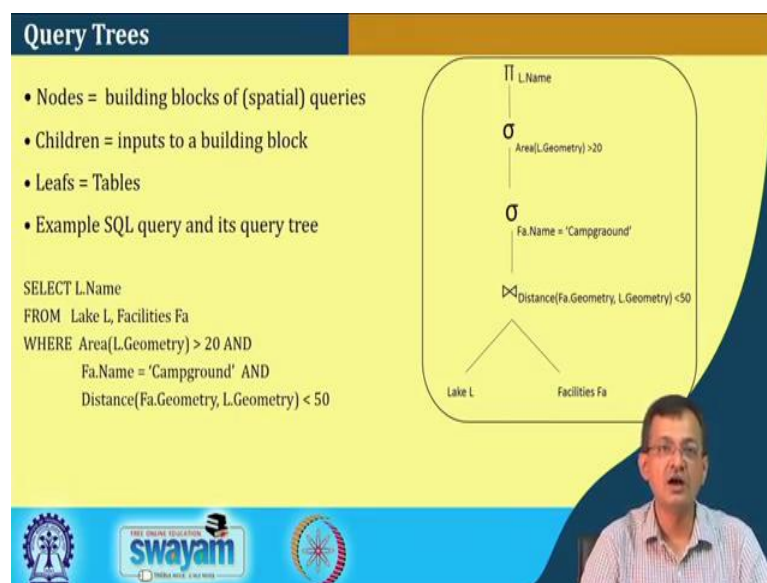


So, this slide we discussed last in our last lecture or last discussion also refer to our Shashi Shekhar's book. So, we have a SQL query, so what we want to given a SQL query what we want to do is a execution plan right. So, I have a SQL query and I need to have a execution plan for. There are intermediate steps as query trees, logical transformation of the tree so that the results are not different, but I can have different type of computational complexities. There are strategy selection for every operations like I want to do a spatial join, so out of that couple of strategies what strategy I want to take up right.

And final outcome is a execution plan which is executed and query is answered and such so this is the the execution plan is the final outcome of the things right. So, if we conceptually or look at the processes. So, I have a SQL grammar abstract data types.

So, which has a parser logical transformation, decomposition, dynamic programming, evaluation and merging of the results value. So, here we have different heuristic rules both for spatial and non spatial data, there are hybrid architectural specification where you want to decompose these queries into different part. There are dynamic programming which selectivity index, CPU buffering etcetera system catalogue and also it has also has a different cos function or for both your spatial and non spatial data right.

(Refer Slide Time: 03:43)



So, like let us look at the a query execution tree right. So, the query is that select L name from lake L facility some FA where area of the L geometry is greater than 20 and FA equal to campground and distance FA geometry and L geometries less than 50. If this my query so that should be executed in some fashion. So, what we have to do we want find out those our objective is find to the lake particular lake name of the lakes which has a area more than some twenty square units and so, a facility called campground and the campground from the lake distance should be less than 50.

So that means, we want to find out those lakes which are within the 50 units of the campground and having a total area something like that right. So, underlining we have these tables, so these lake tables, facility tables etcetera. So, at the I have these two relations right lake and facility. So, if you go by this we need to have a distance of a spatial joint between these two and finding out the distance where less than 50. Over that f a campground is there facility name is FA campground and the area of the is greater than finding those lakes out of that where whose area is 20 those are projected and listed in the things.

Now, if you; if you look at this execution tree, so this is the typical one execution tree see one thing immediately may come to you think this FA name is a campground is a non spatial query right it does not has a geometry class into it right. So, in other sense

there may be several facilities out of that some there are if there are five facilities we can expect that around one fifth of the record is the campground it may not be also.

But so what is there if I could have taken this facility filtered with respect to the facility it could be and so happened that I can have a the number of cardinality of this facility filter database could have been less s, this my this joint could be less loaded. And there are other aspects like there are select, project join, spatial join this type of operators are there right or for these all these things we may have different type of strategies. Like spatial join can have couple of strategies not only that there are there is that predicate for distance right distance is one of the predicate.

So, it has a different there can be a different strategies. Now not only that based on that the a spatial join may be more costly than a spatial select and then project right. So, this matters right with which I want to do.

So I can have different manifestation of this execution query execution tree or query execution plan which will end up in the same result, if the result is coming different then there is mess right. We can have different strategies or different plan to execute the thing, but end of the day the results should be same. But there may be a difference in the overall execution time or overall efficiency of these resolving these queries right, so this is important.

(Refer Slide Time: 07:35)

Logical Transformation Query Trees

- Motivation**
 - Transformation do not change the answer of the query
 - But can reduce computational cost by
 - Reducing data produced by sub-queries
 - Reducing computation needs of parent node
- Example Transformation**
 - Push down select operation below join
 - Reduces size of table for join operation
- Other common transformations**
 - Push project down
 - Reorder join operations

```

graph TD
    L[Lake L] --> J((⋈))
    J --> S1[σ Fa.Name = 'Campground']
    S1 --> Fa[Facilities Fa]
    J --> S2[σ Area(L.Geometry) > 20]
    S2 --> P[π L.Name]
    J --- D[Distance(Fa.Geometry, L.Geometry) < 50]
          
```

So, a variant of these execution plan may be as we are discussing may be this one. So, motivation transformation do not change the answer of the query right. So, I need to transform this one execution tree or exhibition plan to another, so there should not be any change in the final answer of the query right. So, but can reduce the computational cost by reducing the data produced by the sub queries, like in this case if I can filter this non spatial data called campground then I can basically filter out this sub queries right, reduces the size of the table for join operation right. It reduces the size of the table for join operation that is also possible right.

So, common transformations are push, project, down, reorder join operation and type of things, so these are common transformation. So, these selection projection etcetera are push down and so that things it is likely that cardinality will be there. Where we may think that they we may it may be considered that the join is a more costlier than a select. So, doing a select and reduces the overall cardinality of the relation or the table content of the table maybe useful in joining where you do not have to have lot of joins into the things. So, one transformation one was this one from directly reduced from the query I can have transformed form like this, that first the facility equal to campground has been extracted then we have distance that joining this and area and so and so forth right. So, if overall computational cost of this execution plan is there then I can say that I could achieve some sort of efficient implementation or optimization of the things right.

(Refer Slide Time: 09:31)

Logical Transformation and Spatial Queries

- Traditional logical transform rules
 - For relational queries with simple data types and operations
 - CPU costs are much smaller and I/O costs
 - Need to be reviewed for spatial queries
 - Complex data types, operations
 - CPU cost is higher
- Example:
 - Push down spatial selection below join
 - May not decrease cost if
 - area() is costlier than distance()

```

graph TD
    L[Lake L] --> S1["σ Area(L.Geometry) > 20"]
    Fa[Facilities Fa] --> S2["σ Fa.Name = 'Campground'"]
    S1 --> J["⋈ Distance(Fa.Geometry, L.Geometry) < 50"]
    S2 --> J
    J --> P["π L.Name"]
            
```

There may be other way things as alright like traditional logical transformation rule for relational queries with simple data types and operation, CPU cost is much smaller than IR IO cost smaller and IO cost right.

So that means, if is a if the relation cost with simple data types and operations, so it is there is no joining etcetera the CPU cost is much costly. Like here if you; if you look at it that here the area calculation now I bring down out here right. So, the I only select those lakes which has a area 20 out here more than 20 square units right. Now select only those facilities which has a campground facility of type campground or name campground right, so that means after this to selecting I have the output of these two are now much lesser than what the join is to face for this type of things right. It has the all total lake table, total facility table full then they has to join and then find out this.

Now, you see this is has little bit lesser by taking only the campground this is more lesser. So, if the join is more costlier than the select then I can achieve some computational efficiency here right. So, there are two things one is that overall on an average join is more costlier than the select operation or project operation and etcetera and then I can have computational efficiency and changing this there should not be second thing is there should not be any change in the results right.

Another aspect now I can have different strategies to do this right, like for this particular join operation I can have three strategies and one of them I pick up. Even for this area calculation I can have different type of strategies. I can pick up maybe for the non spatial that is the FA name equal to campground, there may not be many strategies I can have a only select operation standard database select operation into the thing. But here we can have different type of algorithm and strategies to do that I can select those strategies which give me a same result right.

So, need to review for spatial queries like we have to see that complex data types operation CPU cost is much higher so we need to look into the things. Like pushdown spatial selection below join may is a one of a good approach to reduce the computational cost, may not decrease if the area is costlier than the distance function there is a catch right. Now this what we are telling that select is less costlier in operation than join right. So, on that basis I push this select which was much at a upper thing rather in the both the case both on the upper than the join below this join right.

So, especially for this if the area calculation is much costlier than the distance calculation, now this particular select now I have to handle all the; all the tuples all the; all the data of this lake. Now this particular select may not help much if area calculation the algorithm for area calculation is much costlier than this distance calculation. But if it is not like that or if there is something parity in that then this can be efficient.

So, there are different ifs and buts. If you if you look at our traditional database execution queries etcetera this things are not that stringent right. So, as with spatial data not only that type of data loads, type of the table may be lot of dictates like. If there are only 5 or 10 entries in the table I do not require so many reorder organization etcetera right. The QPO processing or QPO engine itself be more costlier than reordering right, but if there are 1000 or 10000 tuples in the things then it may be much reducing the cardinality and then going for join etcetera may be much what we say needed stuff needed things.

(Refer Slide Time: 14:11)

Execution Plans

- An execution plan has 3 components
 - A query tree
 - A strategy selected for each non-leaf node
 - An ordering of evaluation of non-leaf nodes
- Example
 - Strategies for Query tree in Fig
 - Use scan for $\text{Area}(\text{L.Geometry}) > 20$
 - Use index for $\text{Fa.Name} = \text{'Campground'}$
 - Use space-partitioning join for
 - $\text{Distance}(\text{Fa}, \text{L}) < 50$
 - Use on-the-fly for projection
 - Ordering
 - As listed above


So, an execution plan has three component parse query tree which is which you executed a strategy select for each non leaf node right. So, these are the leaf node which directly connect to the underlining database, this lake data base, facility data base and type of thing. So, for all non leaf node we should have some strategy or what we say some algorithms or process to follow and ordering of the evaluation of the non-leaf nodes, so at which order that will be evaluated right.

But never the less whatever the different transformed trees or query trees are there all will have the same result, if I am not having the same result so there is; there is other problem to handle with right that is that is not the same query you are processing then not the same. So for example, strategy for query tree in this particular figure use scan area L dot geometry greater than 20 use index for FA name campground to find out those facilities use space partitioning join for distance FA and L less than partitioning join for FA L equal to 50.

So, this my these are may be as for scanning area that is the may be the strategy index using the index of FA name equal to campground may be a strategy for this, using space partitioning join for distance FA and L less than particular 50 units maybe the execution. And use on the fly projection for the project L dot name to find the things right standard projection things.

So, this may be the ordering of doing the things right. So, so I have a execution plan which has a execution tree, a strategy for all leaf node non leaf node sorry and an ordering an evaluation of the leaf nodes which way I will evaluate. So, that my with changing this ordering I can have different instances of the plans nevertheless the same results will be there.

(Refer Slide Time: 16:29)



Choosing strategies for building blocks

- A priority scheme
 - Check applicability of each strategies given file-structures and indices
 - Choose highest priority strategy
 - This procedure is fast, Used for complex queries
- Rule based approach
 - System has a set of rules mapping situations to strategy choices
 - Example: Use scan for range query if result size > 10 % of data file
- Cost based approach

Choosing strategies for building blocks there are different approaches right. So, one is the priority scheme check availability of each strategy given file structure and indices

right. So, that is one sort of a priority scheme choose the highest priority strategy right. So, I have a predefined or a priority scheme is there, so this is this strategy is this is the priority scheme for execution. This procedure is fast used for complex queries.

So, I do not have to I have sort of a lookup table right I see that this one I required these strategies right. It can have different I can have different inputs from the underlying database too then number cardinality of the database and number of columns etcetera the overall data load and etcetera can be lot of other inputs. Nevertheless it is fast because you do not have to again do run another algorithm to find out which strategy is based on this these inputs right. Rule based approach system has a set of rules mapping situation to the strategy choices some rules, like I can say that if there is a join and project I will do the project first. If I join and select I will do a select first, if there is a distance and area calculation that I do a distance if there are there is a same type of things are there, so there may be different type of system level rules which are there.

You scan of the range query if the result size is greater than 10 percent of the data value right. So, if the result size is 10 percent greater than the 10 percent then you do a scan so there is no harm. So, there are several cost based approach giving cost to different strategies, cost to different type of operations spatial operation to topological operations and that cost finding a cost which tree is less costly higher costly and means making a strategy for that. So, that can be one way of looking at it.

(Refer Slide Time: 18:41)

Choosing strategies for building blocks (contd.)

- Cost model based approach
 - Single building block
 - Use formulas to estimate cost of each strategy, given table size etc.
 - Choose the strategy with least cost
 - A query tree
 - Least cost combination of strategy choices for non-leaf nodes
 - Dynamic programming algorithm
- Commercial practice
 - RDBMS use cost based approach for relational building blocks
 - But cost models for spatial strategies are not mature
 - Rule based approach is often used for spatial strategies


The slide features a yellow background with a blue curved border on the right. At the bottom left, there are two circular logos: one of the Indian Institute of Space Science and Technology (IIST) and another of a research institution. In the bottom right corner, there is a small video inset showing a man with glasses and a light blue shirt speaking.

So, cost model based approach which is one of the important thing. So, for a single building block use formulas to estimate cost of each strategies given the table size etcetera right. Choose the strategy with the least cost. So, if I can get the table size then based on the table size I use formulas to estimate the cost of each strategies right, given and choose the strategy with the least cost as the thing right so that is that may be the cost based.

In a query tree least cost combination of the strategy choice is for non leaf nodes. dynamic programming algorithms, these are the different approaches which we which are commonly followed or which I followed. There are several commercial practices like RDBMS use, cost based approach for relational building blocks. But cost models for spatial strategies are not mature we do not know that not do not know that it is important, that if the cost models are not standardize then the way a particular package or particular vendor calculate cost may be different for different type of things right.

So, that is the thing that what sort of cost model and type of things are not that very mature stage. Rule based approach is often used for spatial strategies. So, I can have rule based approaches right. So, a given set of rules that if this happens then you do this, if this is the thing then you do this and type of things so rule based approach.

(Refer Slide Time: 20:27)



Spatial Join

- Traditional join methods such as hash join or sort/merge join are not applicable.
- Filtering cartesian product is expensive.
- Two general classes:
 1. Grid approximation/bounding box
 2. None/one/both operands are presented in a spatial index structure
- Grid approximations and overlap predicate:
 - A parallel scan of two sets of z-elements corresponding to two sets of spatial objects is performed
 - Too fine a grid, too many z-elements per object (inefficient)
 - Too coarse a grid, too many "false hits" in a spatial join

The slide features a yellow background with a blue header and footer. A video inset in the bottom right corner shows a man with glasses speaking. Logos of institutions are visible in the bottom left corner.

So, one of the important aspect or one of the what we say a one of the major costly operation. So, there is a implication of the cost is the traditional join method such as hash

join or sort merge join are not applicable right. So, we cannot have these traditional join methods in here right. Filtering Cartesian product is expensive right, so this is very expensive that Cartesian product two general method grid based approach or bounding box based approach none, one or both operands are present in the spatial index structures right. So, that is important. So, grid approximation and overlap predicate a parallel scan of two set two sets of z elements correspond to two sets of spatial objects is performed.

So, that z order etcetera we will; we will look into the things. So, that is a space filling curve with that indexing in that particular fashion. Too fine a grid too many z elements per objects inefficient and too coarse a grid too many false hits in a spatial joins is also a challenge.

(Refer Slide Time: 21:51)

Spatial Join

- **Bounding boxes:** for two sets of rectangles R, S all pairs (r, s) , r in R , s in S , such that r intersects s :
 - **No spatial index on R and S :** *bb_join* which uses a computational geometry algorithm to detect rectangle intersection, similar to external merge sorting
 - **Spatial index on either R or S :** *index join* scan the non-indexed operand and for each object, the bounding box of its SDT attribute is used as a search argument on the indexed operand (only efficient if non-indexed operand is not too big or else *bb_join* might be better)
 - **Both R and S are indexed:** synchronized traversal of both structures so that pairs of cells of their respective partitions covering the same part of space are encountered together.



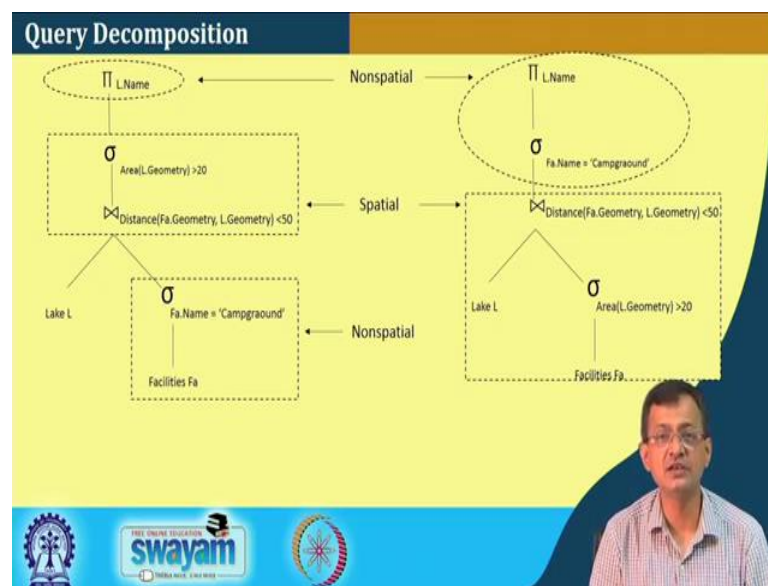



So, bounding boxes two or two sets of rectangle R and S all pairs r, s in the are in space R r in capital R s in capital S such that this such that r intersects s . So, that three things we can say that spatial index on R and S . So, I can know spatial index on R and S *bb_join* which uses a computational geometry algorithm to detect rectangle intersection, similar to the external merge sorting merge sort and type of things right. So, that *bb_join* will do a computation to the rectangular rectangle intersection something of the with the bounding box and type of things and do as a what we say initial stage of initially to filter out those candidates which are likely to be participating in the join. Spatial index on either S R or S *index join* scan the non index operand from each object bounding box this

SDT attribute is used to search argument in the indexed operand right. So, I can have the index either on R or S I can have the index both on R and S right.

So that means both R and S are index synchronized traversal of both strategies. So, that the pairs of cells of the respective partitions covering the same part of the space are encountered together right. So, this type of things are done to ensure that you get the quickly that joining element. So, one is that no indexing one is index on R and S another is index on the both R and S.

(Refer Slide Time: 23:45)



Now, with the; with the things what we can see, so that I what we require a query decomposition right. So, this is non spatial L name and FA equal to campground is a non-spatial query, whereas rest of the things are the spatial again this one is a non-spatial query right. So, what we see say suppose I have this type of strategy right, so here this is my non spatial, this is a spatial and this is a non spatial query right. So, and if I this FA ground is above this and so then what we are telling that the non spatial thing is brought at the top right. So, those are non spatial and rest is this is involve geometric class that is your queries right.

So, this can be the query can be decomposed and things I can but usually the general what we say a motivation or general way of approaches is that we have this we will have this non spatial queries should we are pushed as down the tree as possible right. So, that it helps in reducing the overall cardinality of the data of the that relations that output

relations at as we go up the tree from the bottom all right. And as the non spatial queries are strategies are not very varying and they have a some sort of a costing in a particular range. So, that is something helpful in executing spatial operations at a much higher level of the tree, so that maybe a general trend in doing so.

(Refer Slide Time: 25:59)



The slide is titled "Evaluation of Spatial Operation" in a blue header. The main content area is yellow and contains a bulleted list. The bottom of the slide features a blue footer with two circular logos on the left and a small video inset of a man on the right.

- Spatial Database vs Relational Database [Query Processing perspective]
 - Spatial database does not have any fixed set of operators for building blocks
 - Spatial database deal with extremely large volumes of complex objects, which have spatial extensions
 - Computation of algorithms for spatial predicates are costly.
 - Assumption that I/O cost dominate CPU (processing) cost is not valid

Now, so given this approaches, we have now need to see that how do I evaluate these queries or what are the things evaluating. So, just to again combine them together just to see that the basic again as number of processing from the query processing perspective, the major difference between spatial data base and relational databases are spatial database does not have any fix set of operators for building block right.

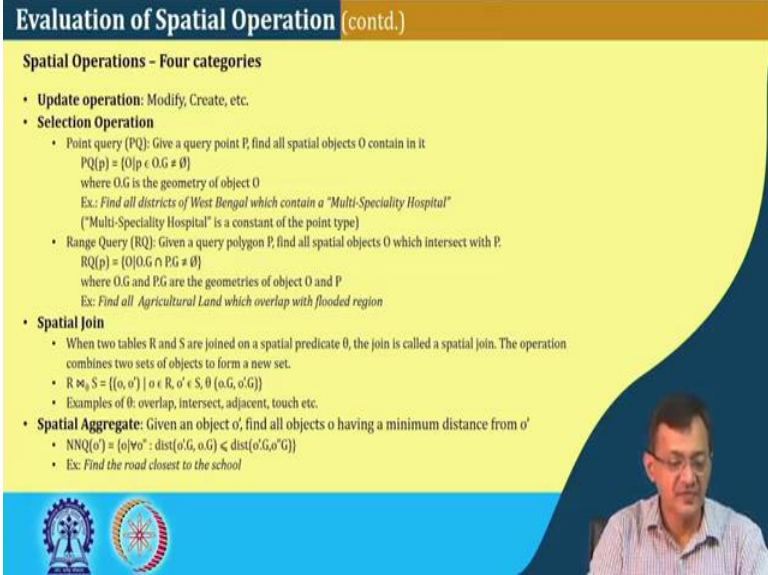
So, if there are building blocks of as we have seen that different building blocks of select project spatial select spatial project and type of things, there is no fix set of operators for the building block. So, it has different type of strategies not only that there can be different type of predicates which are attached to that. Spatial data base deals with extremely large volumes of complex objects which have spatial extensions, so spatial database deals with large volumes of spatial complex objects which has a spatial extension.

So, we can we have a large volume is the data size is pretty large right with a spatial extension. Computation of algorithm so spatial predicates are pretty costly. So, calculation a computation of algorithms in spatial predicate are pretty costly, assumption

that IO cost dominate the CPU processing is no longer valid here all right. So, typically that is a standard way of looking at it that IO cost is much much costlier than the your CPU processing, but it is not valid in case of a spatial operation.

So, for all these reasons these are the four reasons there will be different other complications with when we go with little deeper. But these four things a these four major points segregate between the spatial and relational database. Though the database here also we are using a some sort of relational data base only, though it may be some other way of handling. But those are also relational database which works on the spatial data base right.

(Refer Slide Time: 28:37)



Evaluation of Spatial Operation (contd.)

Spatial Operations - Four categories

- **Update operation:** Modify, Create, etc.
- **Selection Operation**
 - **Point query (PQ):** Give a query point P , find all spatial objects O contain in it
 $PQ(p) = \{O | p \in O.G \neq \emptyset\}$
where $O.G$ is the geometry of object O
Ex: Find all districts of West Bengal which contain a "Multi-Speciality Hospital"
("Multi-Speciality Hospital" is a constant of the point type)
 - **Range Query (RQ):** Given a query polygon P , find all spatial objects O which intersect with P .
 $RQ(p) = \{O | O.G \cap P.G \neq \emptyset\}$
where $O.G$ and $P.G$ are the geometries of object O and P
Ex: Find all Agricultural Land which overlap with flooded region
- **Spatial Join**
 - When two tables R and S are joined on a spatial predicate θ , the join is called a spatial join. The operation combines two sets of objects to form a new set.
 - $R \bowtie_{\theta} S = \{(o, o') \mid o \in R, o' \in S, \theta(o.G, o'.G)\}$
 - Examples of θ : overlap, intersect, adjacent, touch etc.
- **Spatial Aggregate:** Given an object o' , find all objects o having a minimum distance from o'
 - $NNQ(o') = \{o | \forall o' : \text{dist}(o.G, o'.G) \leq \text{dist}(o'.G, o''.G)\}$
 - Ex: Find the road closest to the school

So, in doing so there are what we see that there are serious catches. Now if you look at that spatial operations parse, so there are typically four categories right. So, one is update operation it is like standard database like create, modify create and so and so forth. So, that is the update option. There is a selection option so selection option has two types one is a point query given a polygon P . So, sorry given a query point P find all spatial object O contained in it; contain in it right. That means, say where we say that find all districts of West Bengal which contains a multispecialty hospital right, if there is a there is my query I want to find all district where it a there is a multispecialty. So, multispecialty hospital is a constant of this particular point type right. So, here also if you see that that

PQ of P equal to there is a object O over P belongs to O dot G geometry is not null right, so that we try to represent.

So, this is the point query and there is range query or regional type of region it handles. So, like a given a query polygon P find all spatial object which intersect with P right. Like I say that agricultural land which overlapped with flooded region, so I have a flood region in a particular scenario. So, I want calculate how much area of the agricultural land is affected by this flood situations right. So, I have the agricultural polygon and polygon or polygons and the flood regions and then I overlap and see that how much agricultural things are there. So, there is a functional overlap things are there. I may want to try to see that if there is a road expansion like we discussed that how much agricultural field will be needs to be destroyed. So, road is a line feature so if there is expansion I create a buffer, so it becomes a polygon then over lay these two things together.

So, that is the; that is the way what we look at that range query. So, the again here another option, so what we have the update operation, selection operation and then spatial join. When two tables R and S are joined in a predicate O or predicate theta the join is called spatial join this is true for our normal like say two relation R and S they are joined but there on a on a predicate theta right.

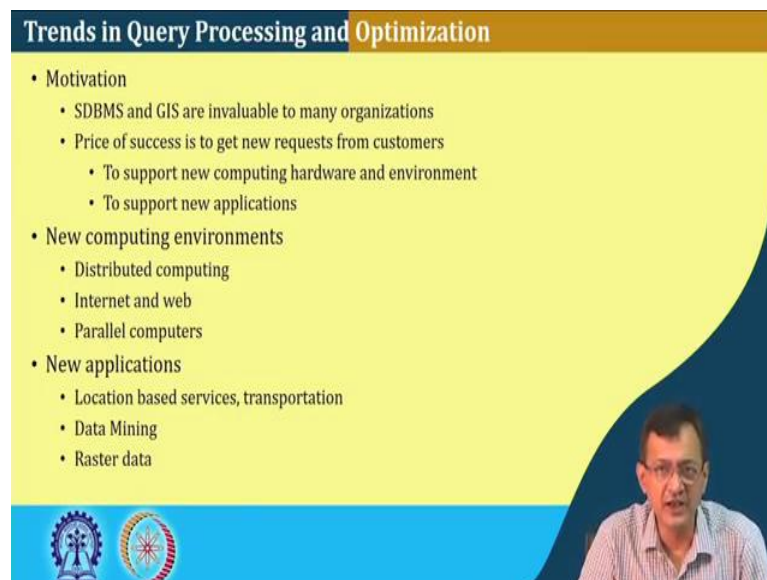
The operation combines two set of object to form a new set, similarly same join operation if you do in a non spatial then system also it will be new set thing will be there. So, like represented by R this join over theta with S then o and o dash at o o belong to R object to o dash belongs to S and theta o dot G that geometry types two geometry type this theta is I can have a operation on the theta. Like typical theta as you have seen overlap, distance, intersect, adjacent, touch these are the different features for different examples of theta like I over lap the distance between on one example we are seeing and so and so forth. So, this is it plays a important role right, so and finally we have a spatial aggregate operation right.

So, given an object o dash find all objects O having a distance from a particular minimum distance from o dash. Like I want to find out find a like here is there find the close road closest to a road way to a school. So, I have school I want to find out the closest road from the school and maybe I have want to set up a factory and I want to find out the closest water supply or pond or river nearest to the things right.

So, there can be different type of this, so these are spatial if I have a spatial aggregate of operation. So, given an object o find all object objects o having a minimum distance from the o right that is the thing we are trying to do that. Like o for all o distance $o \cdot G$ and $o \cdot o \cdot G$ and $o \cdot G$ is less than equal to distance $o \cdot G$ and $o \cdot o$. There is a something what we say nesting the things to find out that which is the minimum right that is a aggregate query what we try to do.

So, I can have different operations and as we understand this has different costing also right. So, with some cardinality of the database we can have different costing.

(Refer Slide Time: 33:57)



Trends in Query Processing and Optimization

- Motivation
 - SDBMS and GIS are invaluable to many organizations
 - Price of success is to get new requests from customers
 - To support new computing hardware and environment
 - To support new applications
- New computing environments
 - Distributed computing
 - Internet and web
 - Parallel computers
- New applications
 - Location based services, transportation
 - Data Mining
 - Raster data

So finally, if you look at that what is the query processing and optimization and what are the recent is, one is motivation is spatial database management system and GIS are invaluable for many organization. It basically gives you a tool to take a decision. Price of success is to new request from the say customer to support new computer hardware environment support new application and type of things.

So, new computing environment like distributed computing, IOT, internet and web parallel computing and this different type of thing different type of what we say development are there on the which can be exploited. There are several new applications like location based services is one of the major application, there is a huge use in the transportation application data mining handling raster data or analyzing raster data there are different type of applications as such are there right.

So, what if we look at if we try to summarize. So, what we see that this spatial query processing itself is a has a different challenge, because which is first of all both IO and compute intensive right. The data sets are pretty large sometimes may not be able to put all the things on the main memory to execute the things. So, that is a serious challenge and not only that as we use for different building blocks different strategies right based on different topological operation etcetera so that means, I need to evolve a appropriate execution plan of the queries right. Given a query I need to do a appropriate execution plan.

Now it all depends on the database properties that cardinality of the database then size of the database because that dictates and in doing so that should not be there that their query processing engine takes more time or is becomes more costlier than the processing the query itself right. So, there are several consideration and these queries are pretty complex and applies different algorithms involving computational geometry and different other aspects of CS. So, it is a more costly and need to be handled appropriately with proper optimization techniques.

So, let us conclude our discussion today we will continue in our future class with looking at different other aspects of spatial informatics. In some point of time we would like to show you a demo of that of these different SQL or query processing that how that queries are processed with that data, that more on spatial SQL or spatial query processing, how things work and showing some results will try to show a demo some time in during this course. So, with these let us conclude our discussion today we will continue our discussion in the next class.

Thank you.