Spatial Informatics

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

Lecture – 23
Spatial Networks – III

Hello. We were discussing on Spatial Networks. More specifically on how to traverse how to query on the spatial networks right under this our spatial informatics course. Now, as we discuss this spatial network is specifically has a little bit of different dealing with an our own our other type of spatial data basis right.

In case of network it is more of a graph of a structure, there are nodes and the edges which connect to the things it can be directed graph it can be undirected graph right. You can have a both way motion or a one way type of things. And it varies on in different aspects right like we can have a transport network, road network or rail network or combination of different transport network.

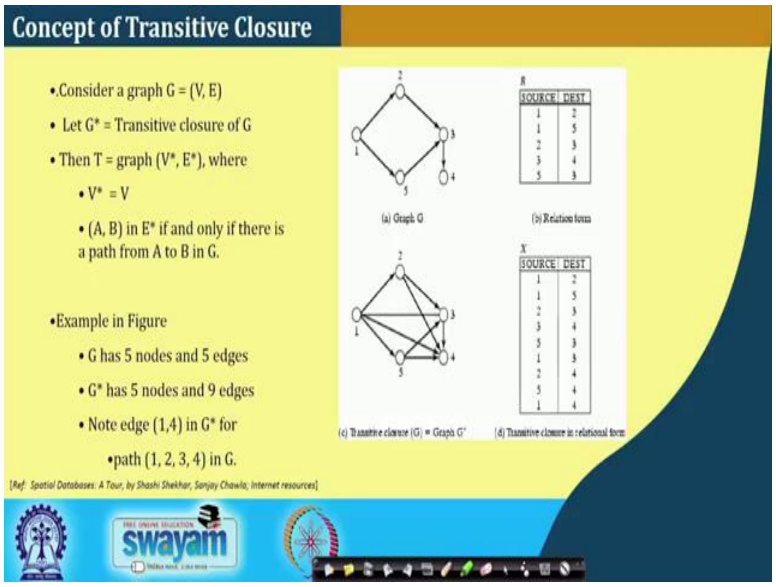
Or even I can look at about a drainage network or right say river network where tributaries distributaries of a river of a river. Or what we say sometimes in hydrological term there is a watershed or something what they refer to as a number of streams which has a thing and that has a direct directed flow right. A river cannot flow on the other direction it has a particular direction of the flow on the things.

Nevertheless, the way we deal that is different right. Rather, when we say a point a and point b on a particular road network it follows the road right. The point a and point b may be very close like may be 100 meters away, but the actual road maybe 500 meters right in between there may be a boundary wall and you need to go through the gate and come back to the things.

So, one side of the boundary wall or other side of the boundary wall may be Euclidean space. Or if I say land use wise very close, but when we want to move connect these two it can be a again a different path right, based on the road network availability. So, the dealing of this spatial networks need to be little different than our standard other spatial data basis on a spatial objects type of things right.

Mostly these are primarily these are all line objects or poly line objects type of things. And another aspects is that I need to have some sort of a this graph algorithms we can put into play right there is another advantage of the things. So, continuing of the discussion we will see some of the different sort of queries on this type of network in today's discussion and subsequently some other aspects of the thing.

(Refer Slide Time: 02:58)



Concept of Transitive Closure

- •Consider a graph G = (V, E)
- Let G* = Transitive closure of G
- Then T = graph (V*, E*), where
  - V* = V
  - (A, B) in E* if and only if there is a path from A to B in G.
- •Example in Figure
  - G has 5 nodes and 5 edges
  - G* has 5 nodes and 9 edges
  - Note edge (1,4) in G* for
    - •path (1, 2, 3, 4) in G.

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

2
1
3
4
5
(a) Graph G
2
1
3
4
5
(c) Transitive closure (G) = Graph G*

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form

X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4
1	4

(d) Transitive closure in relational form

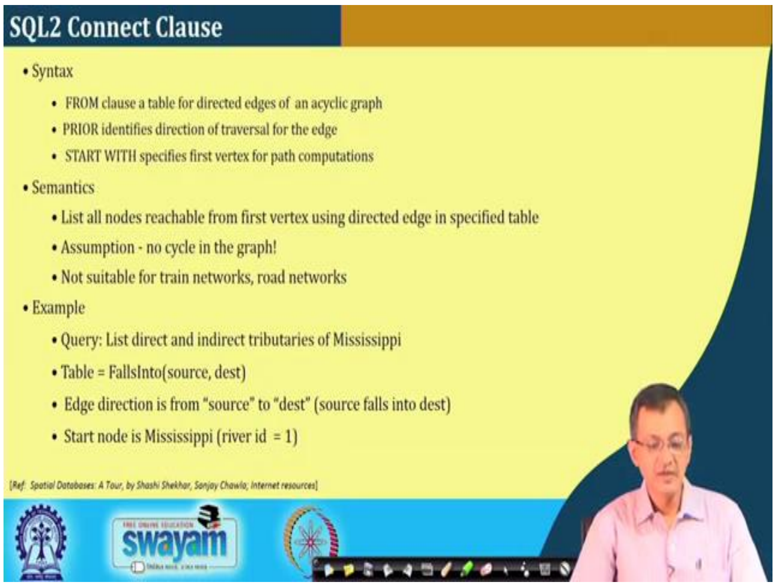
Like this we discussed in our previous lecture also. But primarily we consider it is as a graph and one important thing is a transitive closure like way; that means, if a implies b b implies c then I have a transitive closure a implies c right. Like in other sense, if I can go to 1 to 3 by 1 to 2 and 2 to 3, then there is a path there between 1 to 3 right.

So, a given a network we want to find out a transitive closure right, because see where I want to use you want see that. I whether I can I have a say from IIT Kharagpur, whether I can go to IIT say Bhubaneswar right. So, there are different path initially within the IIT Kharagpur this is some of the road networks, which are within the Kharagpur region within the IIT Kharagpur control or the private. So, called coat un coat private road of IIT Kharagpur. Then you go to this through this district control road then you go met avail train go to Bhubaneswar station and etcetera etcetera.

So, there are I need to know that if I want to go to IIT Kharagpur to IIT Bhubaneswar. So, there should be a to b b to c c to d etcetera now if the if my transitive closure is a starting from a is to the registration d if there is a path then I say things are there. So, given a graph we always try to find out the transitive closure of the things like in this case if this is the graph G. So, this is my transitive closure graph G star where it has all the things like a is connected 2 a is 5 a 1 to 5. Then I have 1 to 3 which is not explicitly out here then I have 1 to 4 and so and so forth.

So, this is an important aspects. Now this need to be handle by a spatial data base right. So, if the that is not handled if there is no like if the if your spatial database SQL does not handle this then we have a problem of finding a path, between a source and destination where there are multiple hops.

(Refer Slide Time: 05:12)



SQL2 Connect Clause

- Syntax
  - FROM clause a table for directed edges of an acyclic graph
  - PRIOR identifies direction of traversal for the edge
  - START WITH specifies first vertex for path computations
- Semantics
  - List all nodes reachable from first vertex using directed edge in specified table
  - Assumption - no cycle in the graph!
  - Not suitable for train networks, road networks
- Example
  - Query: List direct and indirect tributaries of Mississippi
  - Table = FallsInto(source, dest)
  - Edge direction is from "source" to "dest" (source falls into dest)
  - Start node is Mississippi (river id = 1)

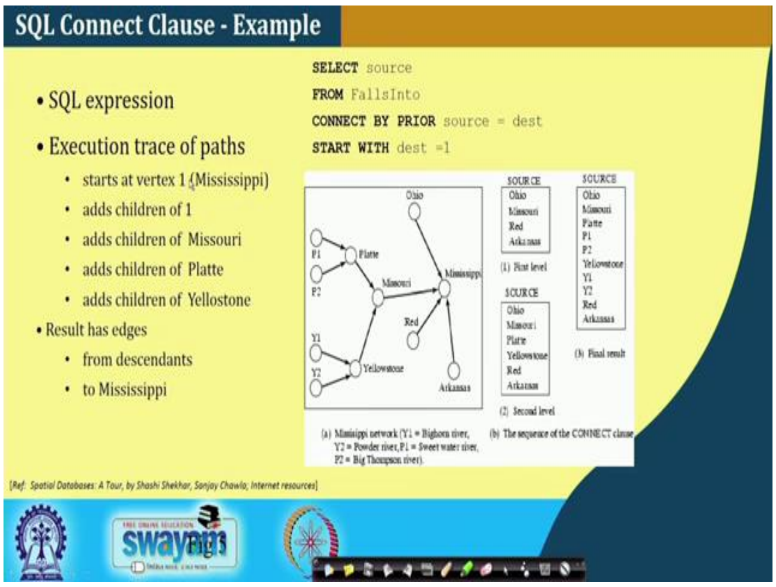
[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, in case of SQL 2; SQL version 2, we have a connect clause right. From prior start with so this is the syntax. And semantically list all nodes reachable from the first vertex using the directed graph in a specified table right. So, in the table it is specified right, which is what 1 is to 2, 1 to 5, 2 to 3, 3 to 4, 5 to 3 there is a which specifies the graph right. So, assumption there is no cycle in the graph and not always suitable for train network road network. So, whether there is implicitly there may be cycle in the graph right. So, there can be acyclic.

So, query in this direct and indirect tributaries of Mississippi; if you remember that our river network. From the processor secretes that book what that example what we have taken is there is a direct and Mississippi right. So, there is a table call falls into source to destination. And a source to destination source falls into edges and my start node equal to say Mississippi which may be a say for in this particular example river id a.

So, I have a river id 1 and then I want to look at this falls into table we said source to destination and see that whether the what are the different direct and indirect tributaries of Mississippi.

(Refer Slide Time: 06:42)



SQL Connect Clause - Example

- SQL expression
- Execution trace of paths
  - starts at vertex 1 (Mississippi)
  - adds children of 1
  - adds children of Missouri
  - adds children of Platte
  - adds children of Yellostone
- Result has edges
  - from descendants
  - to Mississippi

```
SELECT source
FROM FallsInto
CONNECT BY PRIOR source = dest
START WITH dest =1
```

(a) Mississippi network (Y1 = Bighorn river, Y2 = Powder river, P1 = Sweet water river, P2 = Big Thompson river).

SOURCE
Ohio
Missouri
Red
Arkansas

(1) First level

SOURCE
Ohio
Missouri
Platte
Yellowstone
Red
Arkansas

(2) Second level

SOURCE
Ohio
Missouri
Platte
P1
P2
Yellowstone
Y1
Y2
Red
Arkansas

(3) Final result

(b) The sequence of the CONNECT clause

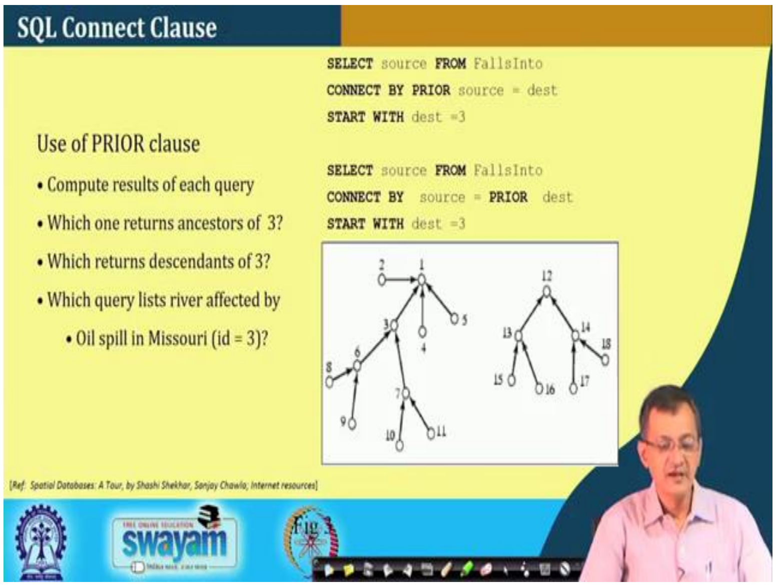
[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, if this is my just to the same thing what we have seen this is the river network. Where, there is a Mississippi that Ohio that its immediate this chains are Ohio Missouri red and Arkansas then we have Platte any yellow river and P 1, P 2; Y 1, Y 2 other things. So, it is a resource this is destination and so, if I go on that vertex is Mississippi at child 1 children of 1; then children of Missouri Platte and so on right.

So, my query is source falls into from the table and source to destination what the falls into specify that the source to destination right. That means, Platte falls into Missouri; Missouri falls into Mississippi P 1 falls into Platte. So, this is specified into the thing, but that, but I have to need a transitive closure P 2 to Platte to Missouri to Mississippi so that need to be found out. So, this connect clause by prior source to destination.

So, source is the, then I go on prior source into destination and then find out that where up to where so long, the it is not exhaustive right. So, this way we go on and we will see interestingly the use of prior allows me to move on the backward or in the forward directions.

(Refer Slide Time: 08:12)



SQL Connect Clause

Use of PRIOR clause

- Compute results of each query
- Which one returns ancestors of 3?
- Which returns descendants of 3?
- Which query lists river affected by
  - Oil spill in Missouri (id = 3)?

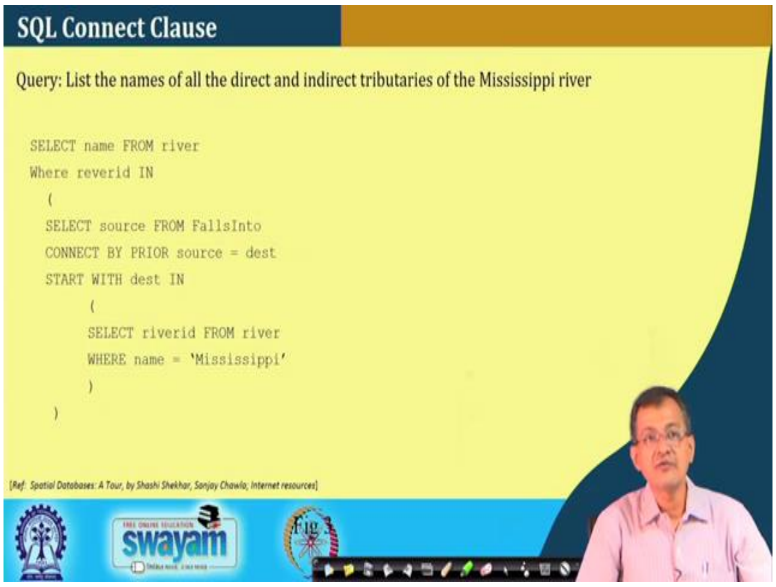
```
SELECT source FROM FallsInto
CONNECT BY PRIOR source = dest
START WITH dest =3

SELECT source FROM FallsInto
CONNECT BY  source = PRIOR  dest
START WITH dest =3
```

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, use of prior clause compute results of each query right, which one returns ancestor of 3 which returns are descends of 3; which query list the affect by the oil spill in Missouri. Like i d equal to 3, if there is a spill in Missouri river then which are the things will be affected right? So, that may be the thing so select source from falls into that is source to destination that is the table maintain; connect by prior source to destination start with destination equal to 3. So, I have to find out where are the things right and it is interesting that the direction will reverse, if I have this prior after that connect by source equal to prior destination right.

So, I am going on a reverse direction or then the top one. So, the way of moment out here searching for the ancestors, which becames the ancestor and descendants will be two different thing by using this prior clause. So, this is a support of the a this is a SQL 3, SQL sorry SQL 2 connect clause right.

(Refer Slide Time: 09:29)



SQL Connect Clause

Query: List the names of all the direct and indirect tributaries of the Mississippi river

```
SELECT name FROM river
Where reverid IN
  (
  SELECT source FROM FallsInto
  CONNECT BY PRIOR source = dest
  START WITH dest IN
        (
        SELECT riverid FROM river
        WHERE name = 'Mississippi'
        )
   )
```

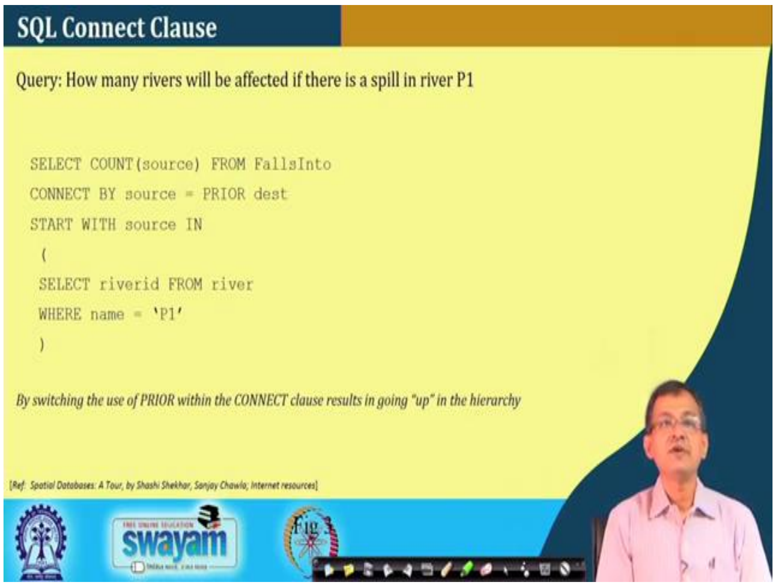
[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla, Internet resources]

So, like there are few more examples we will see like list the names of all direct indirect tributaries of the river Mississippi. Mississippi river SELECT name FROM river where river id IN; SELECT source FROM FallsInto CONNECT BY PRIOR source equal to destination because the falls into, table may be having the river id falls into which river id right. So, from the river table which is having the river id into the and the river name.

So, connect by prior source to destination START WITH destination IN FROM river id source to destination WHERE name equal to Mississippi. So, I pick up the river id from this where the river name from this river table where the Mississippi is there get that river id right. Use that river id, to find out go use that river id into the things.

So, if I see this query it was the destination equal to 1; because I directly try to map that river id from the river table refer to this one. In this case, the same thing we are picking up that river id the same way of giving in the things during the things right.

(Refer Slide Time: 10:47)



SQL Connect Clause

Query: How many rivers will be affected if there is a spill in river P1

```
SELECT COUNT(source) FROM FallsInto
CONNECT BY source = PRIOR dest
START WITH source IN
 (
 SELECT riverid FROM river
 WHERE name = 'P1'
 )
```

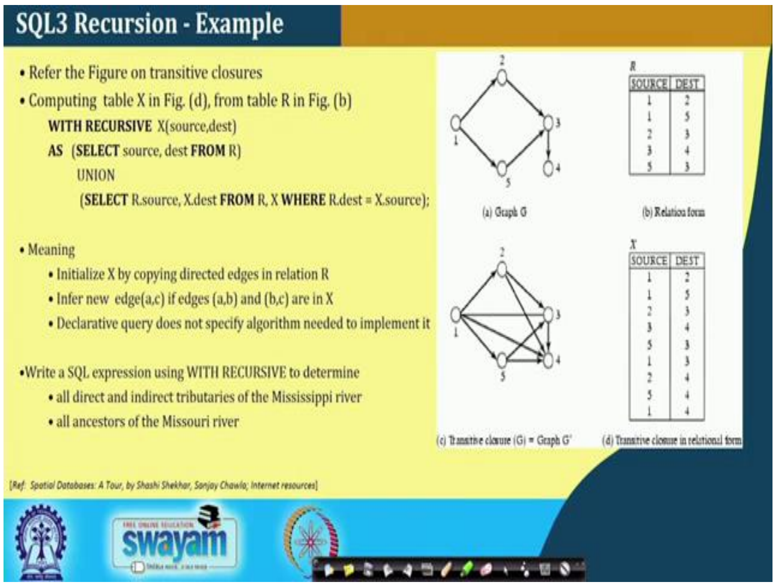
By switching the use of PRIOR within the CONNECT clause results in going "up" in the hierarchy

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

Similarly, if I how many rivers will be affected if there is a spill in P 1 right. So, now, it is in the other way direction right, now it was I am not going back I am going on the other way direction. If there is a spill, so its descendents will be affected right. So, again count source. So, how many rivers? So, I want to find out the count source from falls into by source equal to prior destination start with source in, select river id from the river where river name equal to P 1.

So, that particular source is or particular id is selected and then I use the prior in the other direction right in other way. So, if you take that example scenario and go on a on handwriting things it will be clear that how things are moving. So, it is interesting to see that to note that this switching the use of prior, with connect clause results in up in the hierarchy or down in the hierarchy right. So, where you use the prior clause it allows you to moving the direction of the things ok.

(Refer Slide Time: 11:55)

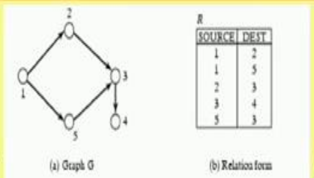


SQL3 Recursion - Example

- Refer the Figure on transitive closures
- Computing table X in Fig. (d), from table R in Fig. (b)

```
    WITH RECURSIVE X(source,dest)
    AS  (SELECT source, dest FROM R)
          UNION
         (SELECT R.source, X.dest FROM R, X WHERE R.dest = X.source);
```

- Meaning
  - Initialize X by copying directed edges in relation R
  - Infer new edge(a,c) if edges (a,b) and (b,c) are in X
  - Declarative query does not specify algorithm needed to implement it
- Write a SQL expression using WITH RECURSIVE to determine
  - all direct and indirect tributaries of the Mississippi river
  - all ancestors of the Missouri river

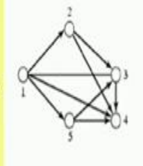


(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form



(c) Transitive closure (G) = Graph G'

X

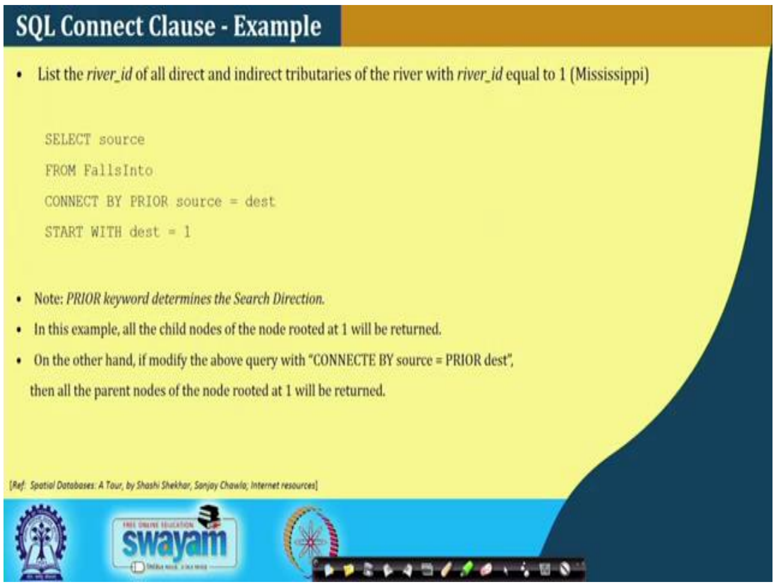
SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4
1	4

(d) Transitive closure in relational form

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, this is one is the by the connect clause. So, there is a another way of handling this river id so, where what we are doing.

(Refer Slide Time: 12:13)



SQL Connect Clause - Example

- List the *river_id* of all direct and indirect tributaries of the river with *river_id* equal to 1 (Mississippi)

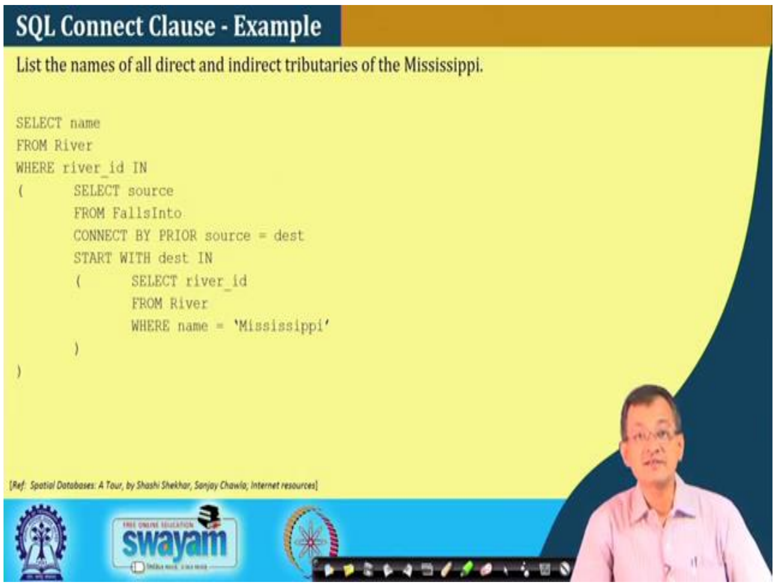
```
SELECT source
FROM FallsInto
CONNECT BY PRIOR source = dest
START WITH dest = 1
```

- Note: *PRIOR keyword determines the Search Direction.*
- In this example, all the child nodes of the node rooted at 1 will be returned.
- On the other hand, if modify the above query with "CONNECTE BY source = PRIOR dest", then all the parent nodes of the node rooted at 1 will be returned.

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

Here it is I am using this prior intelligently and then go on like select one next prior to the things. And select for all the things and then select everyone and go on doing the things right doing the same type of queries, but there is in SQL there is a support call recursion so, recursively looking to the things right.

(Refer Slide Time: 12:31)



SQL Connect Clause - Example

List the names of all direct and indirect tributaries of the Mississippi.

```
SELECT name
FROM River
WHERE river_id IN
(       SELECT source
        FROM FallsInto
        CONNECT BY PRIOR source = dest
        START WITH dest IN
        (       SELECT river_id
                FROM River
                WHERE name = 'Mississippi'
        )
)
```

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, those I think most of you are accustomed with our generic notion of a in recursion. Like in say if we look at in C programming terms or any other programming terms. So, recursion is a thing that where function calls itself right, if a recursive function what you say so; that means, it will go on calling itself.

The problem of the things is that I need to have a appropriate stop condition right otherwise there will be combinatorial expression it will go on and does not know where to end. Now, here list the name of all direct and indirect tributary of Mississippi. The same queries if I use the recursive clause then so refer to the figures for the transitive closure.

So, the syntax of the Mississippi from table x; figure d from table R figure b. So, I have a with recursive X source to destination as select source from destination R union select source destination X dot destination from R where X equal to this. That means, what I am trying to do I recursively calling the queries and building of those things.

So, once I done the things I get a set of nodes which are the member nodes then I go on using that for all those things and again do a recursive queries. So, initialize like it means the initialize X by copying the directed edges of the relational R right. In this case it is what about the directed edges out there. So, initialize x by copying directed edges.

Now, new infer next edge a c, if there is exist a to b and b to c. So, there is the edge form a to c is invert which is in the table. So, initially it is 1 to 2, 1, 5, 2, 3, 3, 4, 5, 3 that for everyone I want to see that if 1 to 2 is there; then if there is a 2 to 3 there will be a 1 to 3 entry. Now if there is a 2 to 3 and 3 to 4 there will be a 3 to 4 entry right. So, declarative query does not specify algorithm needs to be implement here.

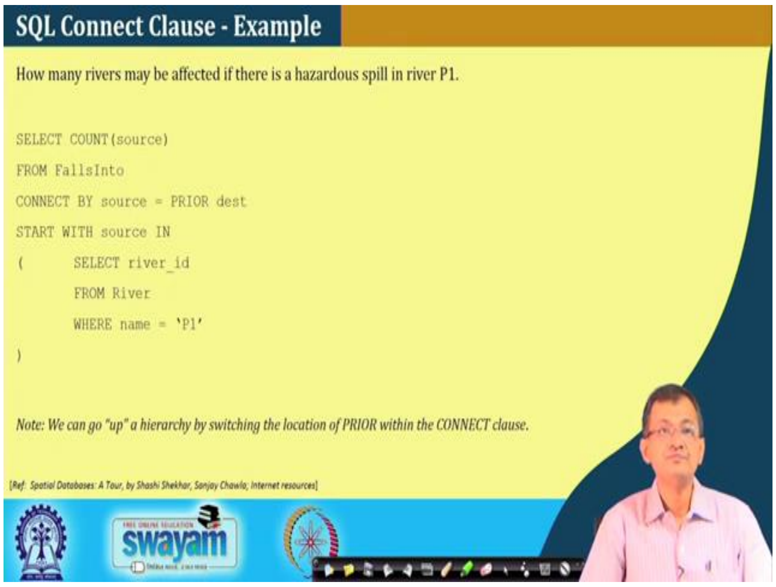
So, I do not we do not looking at that algorithm. So, this is a query where the inherently it is running the algorithm at the background right. The appropriate algorithms were doing those operations for as such our this query we do not have to specify that which what is the underlying algo which is following itself. So, write a SQL expression using recursive determine all direct tributaries of the things all ancestor of Mississippi river. So, we can have use this recursive queries ok.

So, there are few more example first on the connect clause right like list the river id of all direct and indirect tributaries of the river with river id equal to 1 right. So, direct and indirect tributary already we have seen right list direct tribute direct and indirect names of the things then I have to find out these names of the river. So, there may be at least the direct tributaries of another river say Colorado River right.

So, direct means; that means, it should be one level only right, so not that subsequent level. So, here if you see you say that select source from falls into right WHERE level is less than equal to 1 right. So, there is no next level of the things connect by prior source is going to destination and destination is this is basically finding out the destination id from the river id.

Now, as this dictates level is less than equal to 1 it dictate that it goes up to that only the direct queries a direct tributaries. So, not the indirect tributaries in other sense, we are not querying again those taking those tributaries into the table and querying in the next level right. So, there is the way we want to look at the direct queries a direct tributaries of the rivers right. So, what we try to see that we have different construct to handle this type of scenarios.

(Refer Slide Time: 16:58)



SQL Connect Clause - Example

How many rivers may be affected if there is a hazardous spill in river P1.

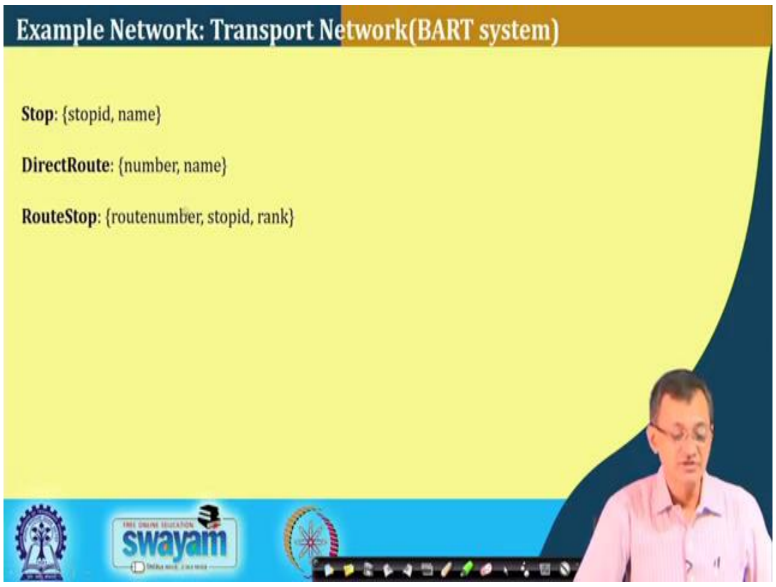
```
SELECT COUNT(source)
FROM FallsInto
CONNECT BY source = PRIOR dest
START WITH source IN
(       SELECT river_id
        FROM River
        WHERE name = 'P1'
)
```

Note: We can go "up" a hierarchy by switching the location of PRIOR within the CONNECT clause.

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, now there may be other queries like how many rivers may be affected if there is a hazardous spill if we already we have seen that right. So, I have to find out the p id and then use that.

(Refer Slide Time: 17:11)



Example Network: Transport Network(BART system)

Stop: {stopid, name}

DirectRoute: {number, name}

RouteStop: {routenumber, stopid, rank}

Now, we want to look at the recursive way that is SQL 3. Now we have consider the network which is considered by in the again the Shashishekhar's book is a Transport Network or what we say that BART or bay area rapid transit system. So, that its a transportation network, say it can be it is a say we consider that is a metro network or something some network where things are there.

So, we have primarily three tables; one is that stops which has a stop id and the name of the stop like stop id one name these and type of things. And there is another table call direct route that is the number and the name like where is the where we have a direct route or these are the route.

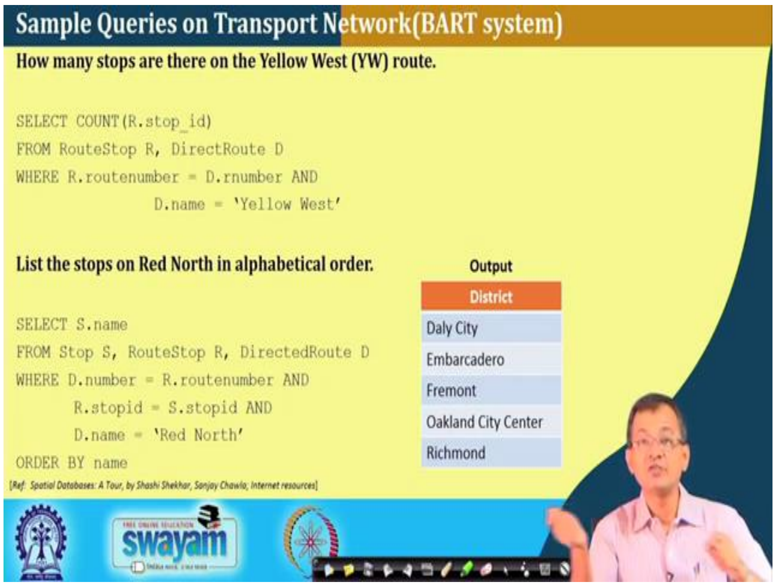
So, route number and the route name. And then the route stop route number stop id rank like I say route number 6 has stop ids, 4, 2, 6, a 4, 2, 6, 1 that is a stop id and this is route id 6. And then I have a rank like 4, 2, 6, 1 is in a sequence; that means, I say rank 4 is in first things and etcetera it is something like if I have; if I have 4 stop ids.

So, it maybe I stop in like 4 6 2 1 why I put that sequence the first of is the fourth stop id 6 and 4 then 6 then 2 and 1 or something like that right. So, what we try to see that there are list of top ids right, which has a stop number and the name which has been given based on some way of handling. The things there is a route number and a route name right any standard transportation network has a particular route id. And route is consist of a set of stops right that is the third table.

And if I it is a set of stop the that every stop want to say because there is a directed right. I want to go from a to b then as to d then a s; then I have to go stop some stop then stop series of stops right so, that are given by the rank like. So, 1, 2, 3, 4 or whatever the rank is it dictates that which stops come after whom and type of things this is the 3 so, to say three tables which are maintained right.

And this tables you see this tables are somewhat maintained by the transport authority or that local authorities which is changes. When new routes comes up new stops comes up or some stops are deleted or a new route comes up and then things goes on. So, rather it so to say if I have this two the third will come up based on this 3 only.

(Refer Slide Time: 20:18)



Sample Queries on Transport Network(BART system)

How many stops are there on the Yellow West (YW) route.

```
SELECT COUNT(R.stop_id)
FROM RouteStop R, DirectRoute D
WHERE R.routenumber = D.rnumber AND
                  D.name = 'Yellow West'
```

List the stops on Red North in alphabetical order.

```
SELECT S.name
FROM Stop S, RouteStop R, DirectedRoute D
WHERE D.number = R.routenumber AND
        R.stopid = S.stopid AND
        D.name = 'Red North'
ORDER BY name
```

Output

District
Daly City
Embarcadero
Fremont
Oakland City Center
Richmond

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla, Internet resources]

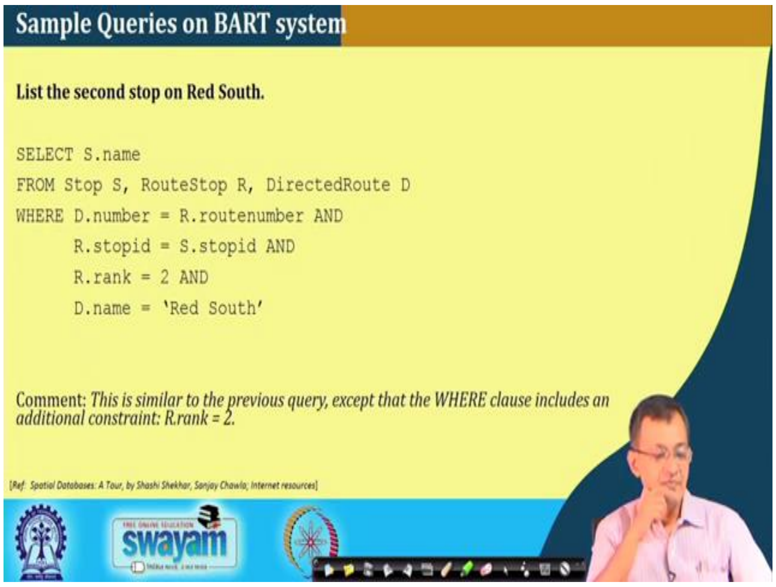
So, there is no new things. Now, if the sample query how many stops are there in the something yellow white line of in the BART systems. So, SELECT COUNT this from stop id R stop route id R; if the route number equal to D number and D name equal to yellow west and things are like this right. So, what you in the how many stops are there in the yellows route right.

So, I want to find out the select count from R dot stop id from route stop R this route stop R and direct route D. So, this is the number of that route D and then where I have the route number equal to R number D dot R number and the D name equal to yellow west. So, I select those routes and find out this count of that a those stop ids and that count of the count the number of stops there are so, I can find out the stops. Then if I have a list the stops on red NORTH on a alphabetically order right. So, this is not the usually the stops are in the rank is in the order of the stops which encounters in the route.

So, now, I can have a alphabetical order right. So, SELECT S dot name from stop S route R and direct route R and then find out that where D; D number equal to route number stop id equal to S stop id and things. So, find out all those stops and then order them by name. So, it comes as a alphabetically alphabetical order right. So, it comes in this typically output for this particular data sets which is which is there you can refer that book or then you have a the.

These are the different stops so, that comes in, but these are not the sequence in the may not be sequence in the route name or the particular route the stop may be may not be in the sequence in the things right. So, it would be based on that that path which follows.

(Refer Slide Time: 22:31)



Sample Queries on BART system

List the second stop on Red South.

```
SELECT S.name
FROM Stop S, RouteStop R, DirectedRoute D
WHERE D.number = R.routenumber AND
      R.stopid = S.stopid AND
      R.rank = 2 AND
      D.name = 'Red South'
```

Comment: *This is similar to the previous query, except that the WHERE clause includes an additional constraint: R.rank = 2.*

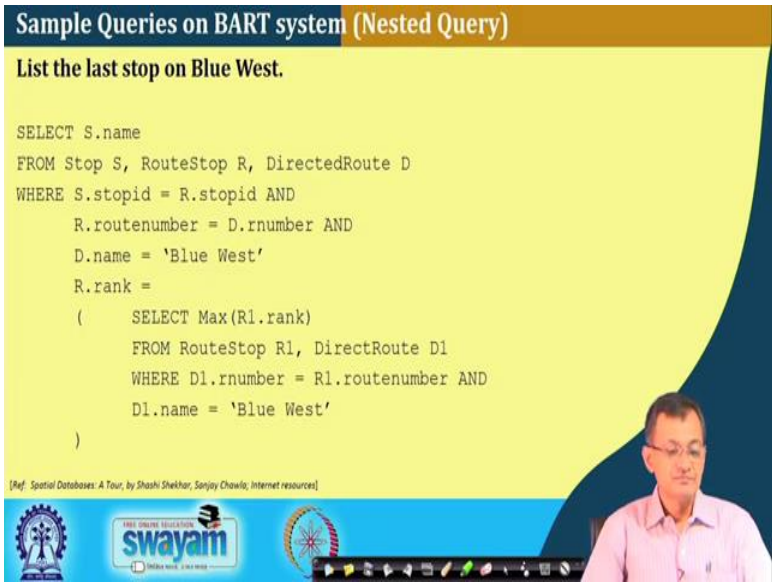
[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, similarly list the second stop in the something Red South. So, that we have the list S name from stop S route stop R directed route D where D dot number equal to R dot numbers stop id equal to stop id rank equal to 1 AND Red South. So, you see that here also we are able to handle that with respect to the this matching that particular D dot route number with the route number. And stop ids and rank and that particular Red South or whatever the route name.

So, red south is one of the route name what we can see that this is similar to our previous query. If I except the where clause includes an additional rank of the R dot rank equal to 2. If you recollect the rank what we are telling that what is the stop number within that stop within that particular route. So, the there maybe route stop id 1, 2 3 4. So, what is that particular number of that stop.

So, what we want to find out what is the second stop in this particular route. So, that is why we are having that R dot rank equal to 2 right. So, what we see that these are the different ways I can have different queries, where the network is primarily a the network. The that overall representation of the spatial network can be handled as a graphical network. In some of the cases in these cases I have not exploiting that graphical phenomena what we are trying what we are doing is only queries querying on the tables what we are trying to look at.

(Refer Slide Time: 24:34)



Sample Queries on BART system (Nested Query)

List the last stop on Blue West.

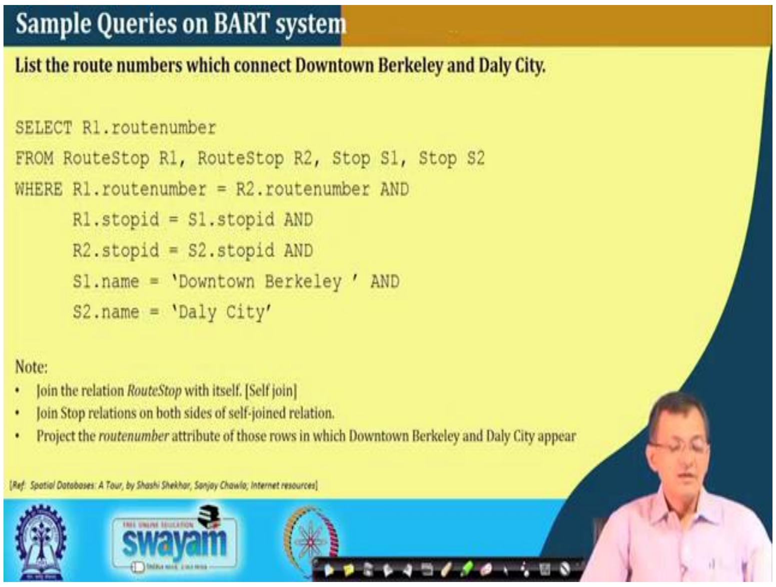
```
SELECT S.name
FROM Stop S, RouteStop R, DirectedRoute D
WHERE S.stopid = R.stopid AND
      R.routenumber = D.rnumber AND
      D.name = 'Blue West'
      R.rank =
      (     SELECT Max(R1.rank)
            FROM RouteStop R1, DirectRoute D1
            WHERE D1.rnumber = R1.routenumber AND
            D1.name = 'Blue West'
      )
```

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, this is find the last stop on the Blue West thing. So, SELECT S name FROM stop S route R directed route D WHERE, S dot stop id equal to R dot stop id routenumber equal to S dot number AND routenumber equal to blue west right. So, this is my selection of the things and then what we get a last stop is the a rank where the rank is maximum. How we ranked it? We ranked it at that number of stops the last rank will be the last stop then the destination comes right.

So, max of R dot rank route R 1 and D dot route R 2 WHERE D 1 dot route number equal to R dot route number AND D name equal to Blue West. So, that the what we have the rank at the destination id that maximum rank. So, if I have a particular route number route name of Blue West and then I have a sequence of numbers into the thing right; that rank is the number of stop the highest rank stop is the last stop in the thing. So, I can have that using this query manipulation using that table I can have, that table has to be a transitively close table right; I know that all those things.

(Refer Slide Time: 26:05)



Sample Queries on BART system

List the route numbers which connect Downtown Berkeley and Daly City.

```
SELECT R1.routenumber
FROM RouteStop R1, RouteStop R2, Stop S1, Stop S2
WHERE R1.routenumber = R2.routenumber AND
      R1.stopid = S1.stopid AND
      R2.stopid = S2.stopid AND
      S1.name = 'Downtown Berkeley ' AND
      S2.name = 'Daly City'
```

Note:

- Join the relation *RouteStop* with itself. [Self join]
- Join Stop relations on both sides of self-joined relation.
- Project the *routenumber* attribute of those rows in which Downtown Berkeley and Daly City appear

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

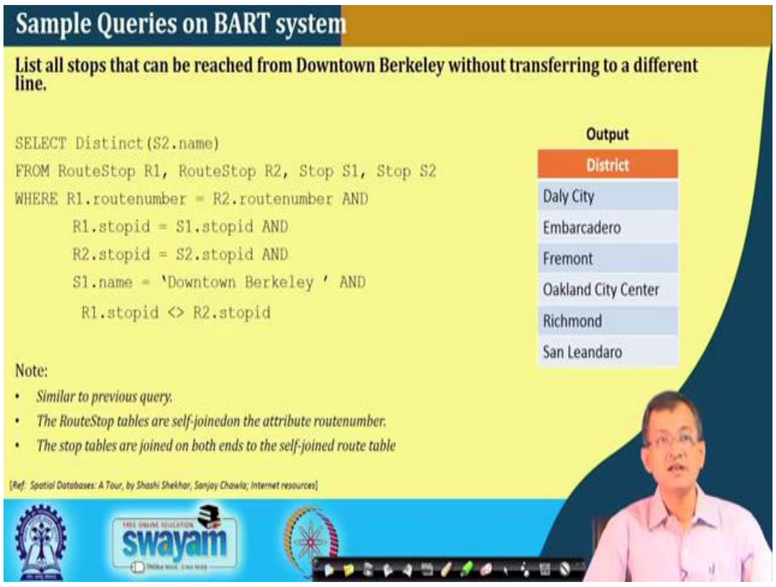
Now, list the route number which connects something Berkeley to Daly City downtown; Berkeley to Daly cities from source to destination right. So, I want to find out the which are the transport say which connects these two things? So, R dot routenumber right. So, the set of route numbers and route stop id R 1 route stop id R 2 stop S 1 stop S 2 right.

WHERE R 1 routenumber equal to R 2 routenumber and R 1 stop id equal to S 1 stop id right AND R 2 stop id equal to S 2 stop id right. So, I am trying to find out that all possible route of the things and your S 1 name is this downtown Berkeley and S 2 name is the thing. So, all I want to find out that within this route where this two stops are there.

So, it can be number of routes where this two stops are there and it find outs that all possible routes. So, note join the relational route stop with itself self joining right. So, I am I need to do a self joining of the thing join stop relation with both sides of the self joined relation and project the routenumber attribute those rows which are the which have this Berkeley and Daly city where these two things appear.

I want to project these two particular those routes which are where these two things are there right. So, we can have different type of queries which are applicable out in this type of scenarios right.

(Refer Slide Time: 27:47)



Sample Queries on BART system

List all stops that can be reached from Downtown Berkeley without transferring to a different line.

```
SELECT Distinct(S2.name)
FROM RouteStop R1, RouteStop R2, Stop S1, Stop S2
WHERE R1.routenumber = R2.routenumber AND
        R1.stopid = S1.stopid AND
        R2.stopid = S2.stopid AND
        S1.name = 'Downtown Berkeley ' AND
         R1.stopid <> R2.stopid
```

Output

District
Daly City
Embarcadero
Fremont
Oakland City Center
Richmond
San Leandaro

Note:

- *Similar to previous query.*
- *The RouteStop tables are self-joinedon the attribute routenumber.*
- *The stop tables are joined on both ends to the self-joined route table*

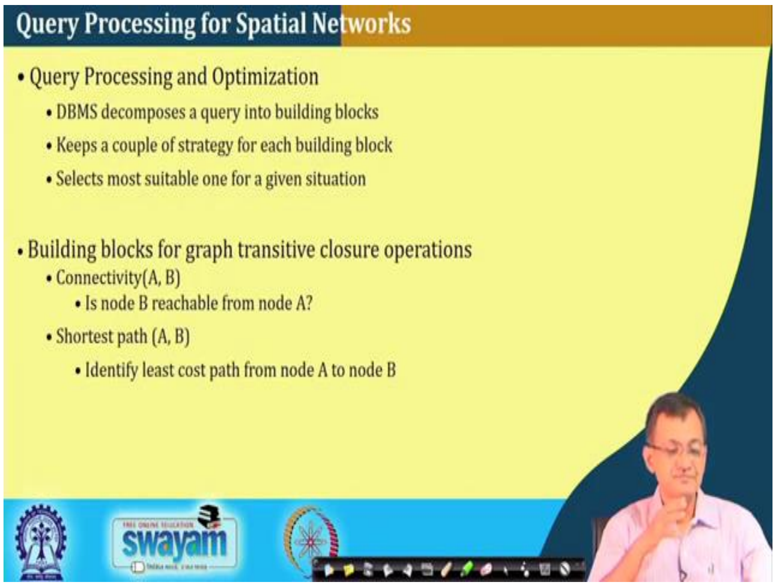
[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, this is list all stop that can be reach from particular source without transferring into a different line right. So, it that they refer to this Bart then what you say bay area at rapid transit network. So, without transferring to the different lines find all stops that can be reach from particular city area right.

So, what we have to say that distinct S 2 name right and from route id R 1 route R 2 stop S 1 stop S 2 AND then root number R 1 root number equal to R 2 root number R 1 stop id equal to S 1 stop id. WHERE, I take from that particular stop id set AND R 2 stop id from that stop id and then my source of these S 1 is downtown Berkeley and R 1 stop ids not equal to R 2 stop id right.

So, these two things cannot be a different route stop or the different routes right. So, similar to previous query the route stop tables are self join with attribute and here the stop tables are joined both ends to the self join route table right. So, we find out that all those things where all those stops where which can be reach from the downtown Berkeley without shift into other things. So, if you see these are all different dealing with different type of SQL things right SQL query construct and using these type of tabular ids.

(Refer Slide Time: 29:29)



Query Processing for Spatial Networks

- Query Processing and Optimization
  - DBMS decomposes a query into building blocks
  - Keeps a couple of strategy for each building block
  - Selects most suitable one for a given situation
- Building blocks for graph transitive closure operations
  - Connectivity(A, B)
    - Is node B reachable from node A?
  - Shortest path (A, B)
    - Identify least cost path from node A to node B

Now, next thing what we like to look into having these where are the different possibilities of query processing and optimizations right. So, and another thing is the building block for the graph transitive closure operations and type of things right. So, first of all, in what are the typical this query processing DBMS decomposes a query into building blocks right.

So, whenever I process the query into building blocks keep a couple of strategies for each building block when if you remember when we did query optimization etcetera. So, we have building blocks of different type of building blocks right like even that spatial, join, select, project all are these things are building blocks right.

So, there can be different algorithms or strategies for this building, block and select the most suitable strategies to do the things right. And that is; obviously, based one of the major factor is that cardinality of the overall things are how much I am need to deal or the basically meta data from the databases right. And there are there can be for if you look at that specifically for spatial networks then graph transitive closure operation is one of the major operation right that is graph transitive.

Once the closure is there then I can do lot of other queries into the things right. So, connectivity A to B is node B reachable to A or not this is one of the may be one of the building blocks like to whether A can be reachable from B using this transitive closure thing. Or shortest path between A to B right it may be minimum hop or it may be based on other things. So, these are different aspects of the things, we will continue our discussion on those aspects.

So, what we see try to look at today is different type of queries, which are possible on this spatial networks. And also one of the major aspects one of the major feature of the things is what we found is the finding a transitive closure of the things. So, I have a initially a connected graph a basically a graph and it can be directed or it can be undirected based on the type of networks having.

If it is directed then I need to find out this what are the different transitive closure. There are two construct, when SQL 2 is a connect and in SQL 3, there is a recursive method of handling this. Once I have those tables then I can have different type of queries into the for the particular spatial networks. So, we will continue our discussion. Let us conclude our discussion today for particular discussion now.

Thank you.