GEOGRAPHIC INFORMATION SYSTEMS

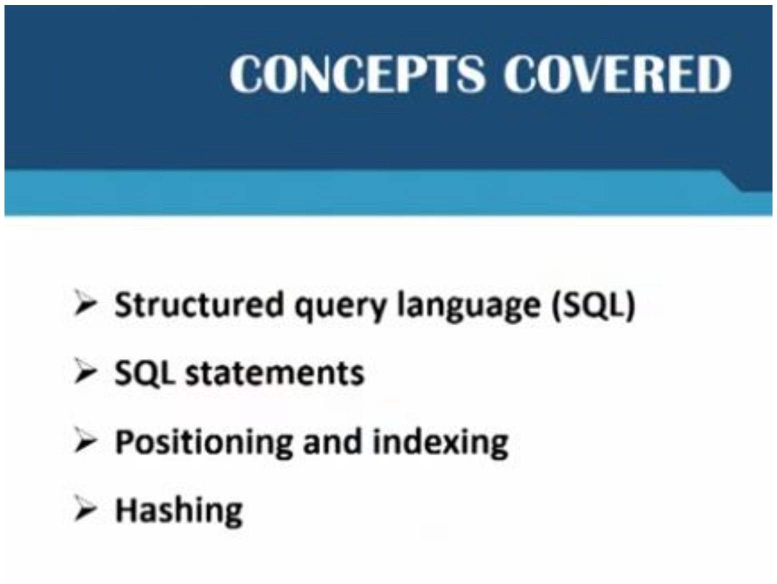
Prof. Bharath H Aithal
Ranbir and Chitra Gupta School of Infrastructure Design and Management Indian Institute of Technology - Kharagpur

Module No # 08
Lecture No # 37
Spatial Query using SQL – Introduction

Hello Namaste welcome back to the course on geographic information system we were in the module 8. Now let us look at the lecture 2 which is based on spatial queries so now we have learnt how do you create a database? How do you maintain a database? Now let us look at what do you mean by querying how do you how does SQL or the SQL help you in querying the data from the database whatever the information that you need from the database.

So let us look at this I will just these basically what I am trying to do is that give you just a flavor of how it works. If you want to really get into advanced mode I would each of this for example when you are looking at database management system itself is a course and similarly the querying using sequel or using post GIS all of these are different courses. So if you want to look at the advanced user I would suggest taking up these courses and understanding it okay. So maybe a I will just give you flavor and if we can take it forward it would be extremely helpful for you as an GIS user.

(Refer Slide Time: 01:33)

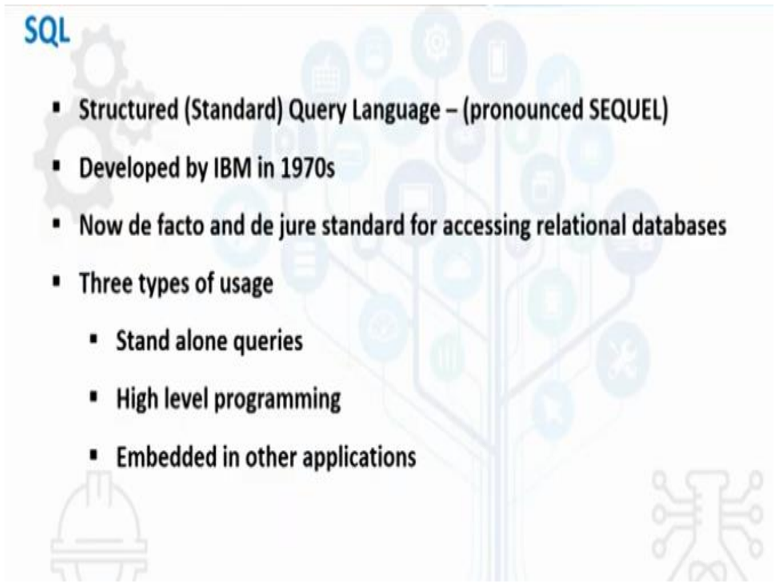


CONCEPTS COVERED

- Structured query language (SQL)
- SQL statements
- Positioning and indexing
- Hashing

So today we would look at what you mean by SQL or structured query language commonly known as sequel or SQL. So we look at some of the SQL statements we look at how the positing and indexing is important and also certain things called plus hashing but other than this we would look at certain other concepts of how the searching also happens we look at R-tree Quad tree etc., So these are some of the concepts that I would be covering in this today's class.

(Refer Slide Time: 02:05)



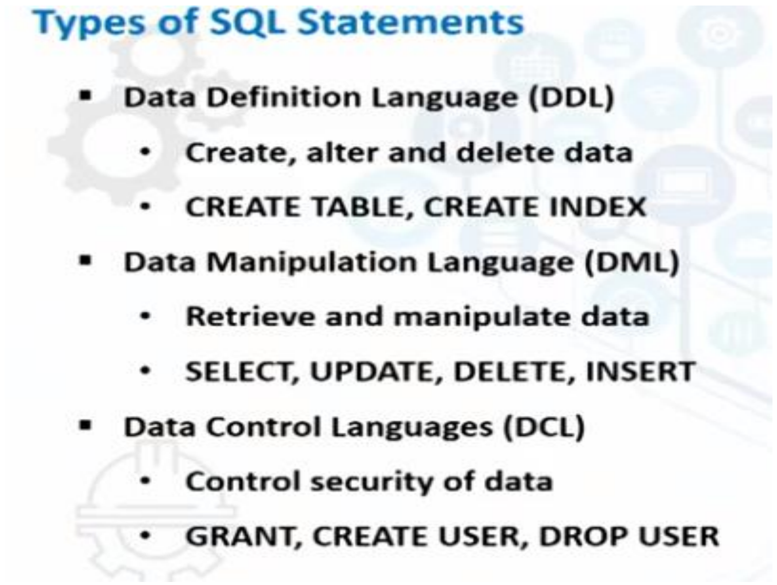
SQL

- Structured (Standard) Query Language – (pronounced SEQUEL)
- Developed by IBM in 1970s
- Now de facto and de jure standard for accessing relational databases
- Three types of usage
  - Stand alone queries
  - High level programming
  - Embedded in other applications

So when you look at SQL is structured query language it is called as standard query language okay normally pronounced as sequel so it is called SQL normally developed by IBM in 1970's now it is a de facto standard for relational databases. If you are looking at relational databases by default the query language is sequel okay or now the way it is the latest one is my sequel. So looking at that if you are trying to understand sequel it would be very good in terms of learning sequel as a advanced user then your database query will be extremely useful.

There are 3 types of usage one is standard standalone queries if you just have a small queries or standalone it can be high level programming you can look at programming for extraction of information. Most important thing that is happening in today's world is embedding it in other applications so that it extracts information from users very similar to those API's that you use okay. So you embedded it into the applications so then the information is sort.

(Refer Slide Time: 03:22)

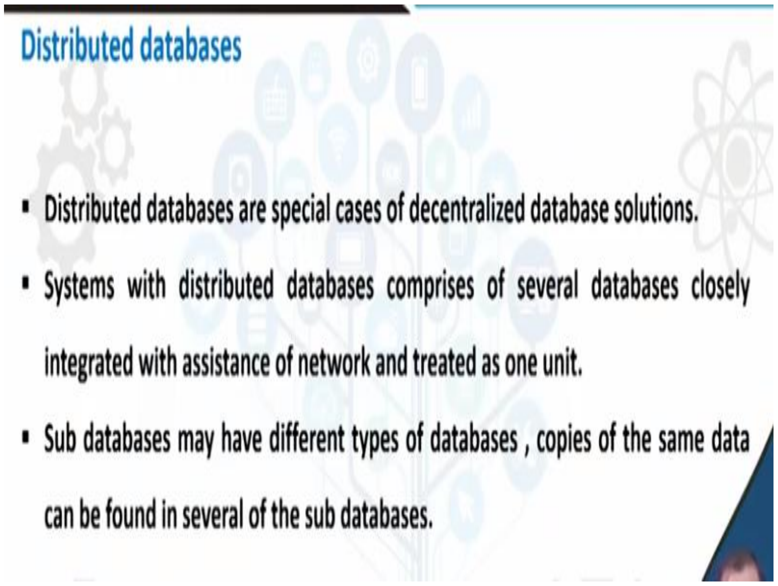


Types of SQL Statements

- Data Definition Language (DDL)
  - Create, alter and delete data
  - CREATE TABLE, CREATE INDEX
- Data Manipulation Language (DML)
  - Retrieve and manipulate data
  - SELECT, UPDATE, DELETE, INSERT
- Data Control Languages (DCL)
  - Control security of data
  - GRANT, CREATE USER, DROP USER

So when you are looking at sequel SQL statement there are 3 kinds of statements one is the data definition language so it creates alters and deletes data okay. Then data manipulation language it as for retrieving and manipulating data if you want to retrieve a particular data and manipulate it then use data manipulation language and when you are looking at control of data in terms of security etc., then you have datas control language which as again called as DCL. So these are 3 types of SQL statement that are used in terms of using a sequel.

(Refer Slide Time: 04:04)



Distributed databases

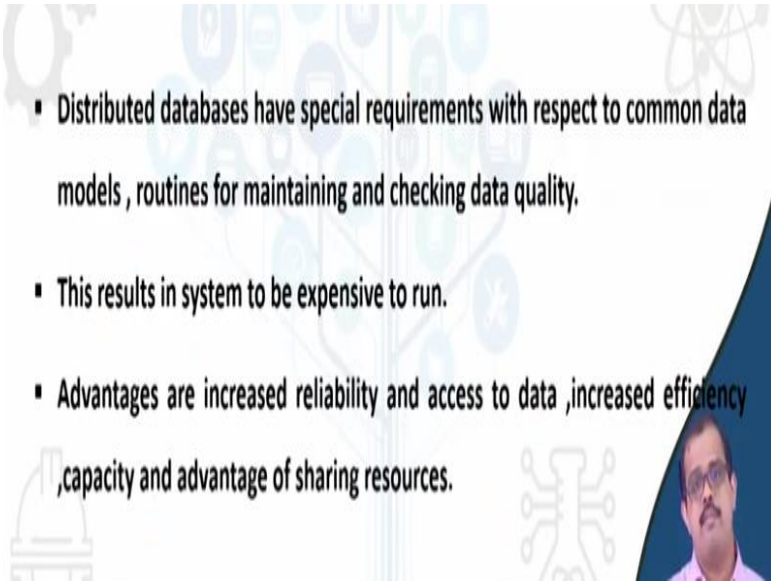
- Distributed databases are special cases of decentralized database solutions.
- Systems with distributed databases comprises of several databases closely integrated with assistance of network and treated as one unit.
- Sub databases may have different types of databases , copies of the same data can be found in several of the sub databases.

So there is a concepts distributed databases when I say distributed databases these are special case of decentralized database solutions which means to say that all databases does not really set in a particular machine in a particular workbook or particular place these are the measure these are from various places if for example in today's scenario someone wants to look at web pages so these are very good examples of a decentralized database solutions you have different web pages from different loop and corners of the country each having its own database looking at its own profiles.

So these are different things there are so many websites which combine all of these together and you see a single different things there are so many website which combine all of these together and gives you a single comparative issue. That is again a distribute databases you have websites to in that issues so now the web handling is more of the thing has been growing in GIS is similarly with the other applications in the computer science .

So systems where distributed database where comprises of several databases that are closely integrated with assistance of network and it is treated as 1 unit okay. So sub databases have different types of databases copies of same data can be found in several of these databases. So only thing is that data redundancy in case not need it so that data redundancy is part where you have to maintain it in a much better way which relational databases normally handles in a very effective way.

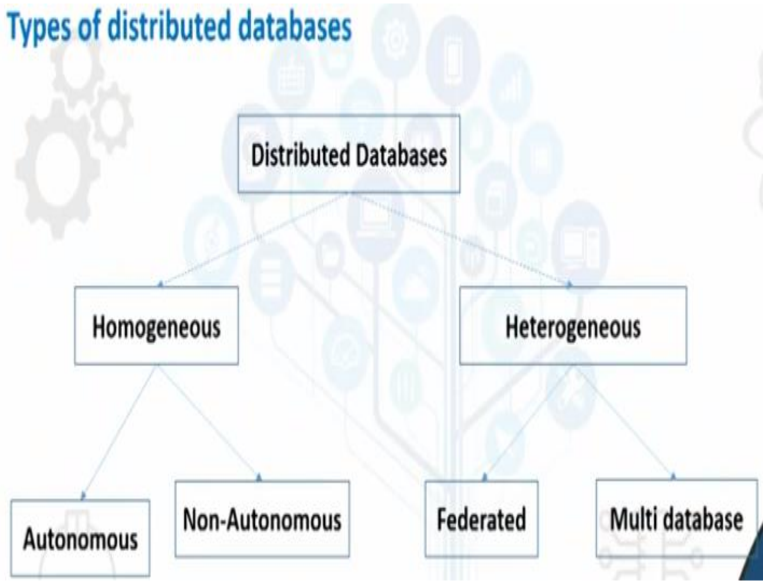
(Refer Slide Time: 05:39)



- Distributed databases have special requirements with respect to common data models , routines for maintaining and checking data quality.
- This results in system to be expensive to run.
- Advantages are increased reliability and access to data ,increased efficiency ,capacity and advantage of sharing resources.

So distributed databases have special requirements with respect to common databases have spatial requirements with respect to common data models routines for maintaining checking data quality so the way they handle it the way they maintain it the way that checked quality as very different the results in the system to be expensive to be expensive to run in terms of distributed databases you have. Advantages are increased reliability and access to data increased efficiency capacity advantage of sharing resources. So all of these are very important in terms of handling a database and maintaining a database.

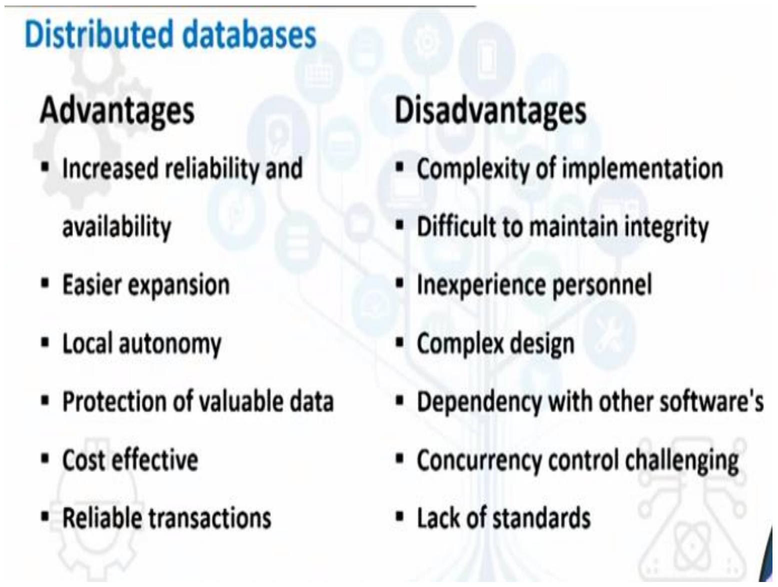
(Refer Slide Time: 06:14)



So when you look different distributed databases as we two types one is homogenous type another one is the heterogeneous type. Homogenous type is divided into autonomous and non-autonomous so each of these have their own way of looking at database that they have been developed for certain issues I would not get into why? How? And how it can store the database but it is different from each other that is why there are different groups.

Then you have heterogeneous which is federated and multi database is a very common this you can find in most of the places. Other than multi database it is federated in terms of he erogenous distributed databases okay.

(Refer Slide Time: 06:57)



Distributed databases

Advantages

- Increased reliability and availability
- Easier expansion
- Local autonomy
- Protection of valuable data
- Cost effective
- Reliable transactions

Disadvantages

- Complexity of implementation
- Difficult to maintain integrity
- Inexperience personnel
- Complex design
- Dependency with other software's
- Concurrency control challenging
- Lack of standards

So these are about distributed databases there are certain advantages certain disadvantage when you look at advantage there are increased reliability and availability at most of the time okay. Most of you look at most of the sites you look at what is downtime? Okay most of the uptime if the site uptime is 99.99 then it is considered to be reliable which means the data that is providing from different databases is reliable.

Easier expansion now reliability not just with the time but you can rely on the particular website for getting the data that is what I meant. Then easier expansion and local autonomy whatever the data autonomy you want to have so you can look at it locally in a distributed database and production of valuable data it is easy because each of the distributed database has its own security rules, policies. So it is easier to segregate data and have different policies and different securities.

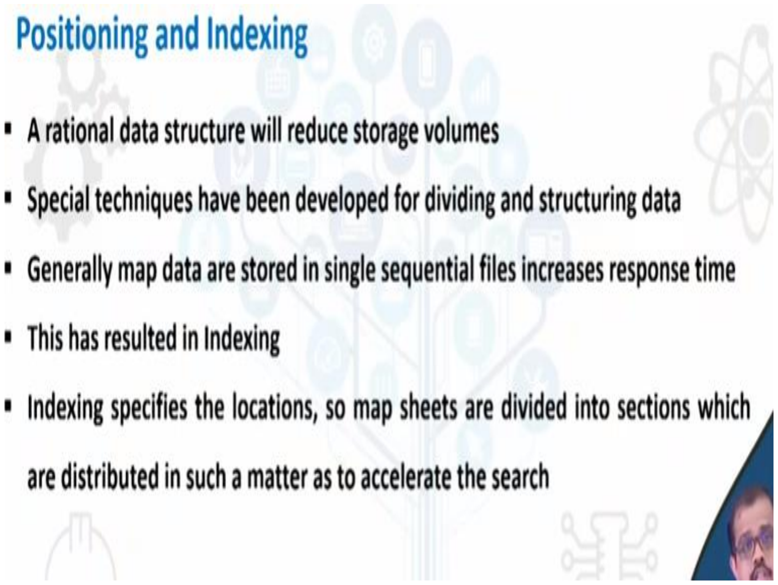
Cost effective in terms of just maintaining it but creating it and in a longer run maintaining it is fine but creating it and then a shorter term it is quite expensive okay. But reliable transaction when you are looking at transaction they are more reliable when you are looking disadvantage the complexity of implementation. So because you should know everything about the different databases distributed databases that they are located and how you have capture that particular data.

So you need the complexity of implementation is very high when you are look at the other disadvantages is the maintenance of integrity of the data which we mention that the first thing that we mention in GIS is data integrity. So that is what it is extremely difficult in terms of distributed I mean distributed database and if you have inexperience personnel you are actually handling the distributed database then you are really going down in terms of maintaining or handling the entire database.

It needs a vast experience in terms of how the database is handled complex designed makes it most difficult in terms of handling. And dependency with other software so let us say that each distributed databases has its means developed in their own software systems and command consoles. So if you have to look at their software and its interoperability with your software in order to give out the data so that is important.

Concurrency control challenging so concurrency yes in today's scenario my not be of much but yes it is challenging task and most importantly in the distributed database it is lack of standard of representation or standards of storing standard of representation and integrity.

(Refer Slide Time: 10:00)



Positioning and Indexing

- A rational data structure will reduce storage volumes
- Special techniques have been developed for dividing and structuring data
- Generally map data are stored in single sequential files increases response time
- This has resulted in Indexing
- Indexing specifies the locations, so map sheets are divided into sections which are distributed in such a matter as to accelerate the search

The next thing that we have to look at in this one using a sequel a positioning and indexing. So why I did mention in a distributed database in reduction to sequence is that because distribute database sequel can handle any kind of databases it can be normal database it can be distributed databases it just can be a local database. So sequel can handle any kind of it and distributed databases are order of day today and sequel has proficiency in handling these data.

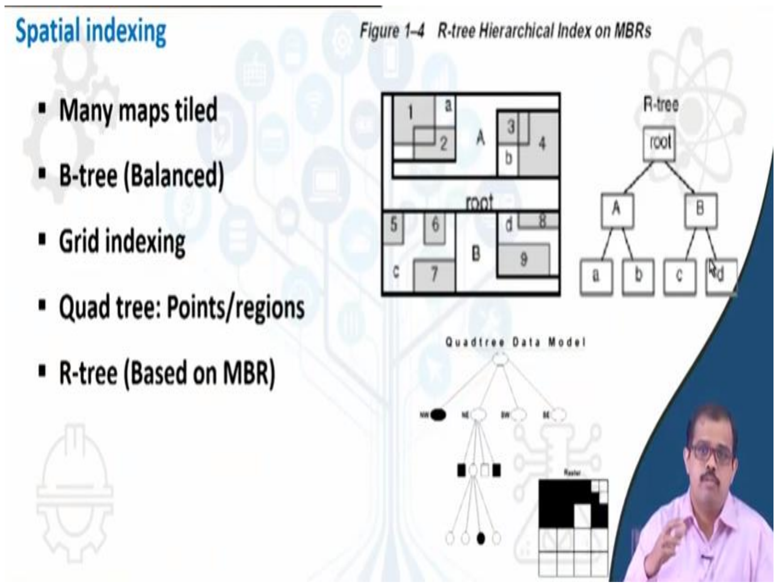
Then when you are looking at positioning and indexing when you have a rational data structure will reduce if you look at positioning and indexing will actually reduce your storage volumes okay. For example if you have positioned or indexed particular data okay it means to say that it is occupied the huge amount of space and it may also again result in certain duplication of data. Then special techniques have been developed for dividing a structuring data.

Generally map data stored in a single sequential file increases response time so what they do is they give a multi sequential file which means that it is not very sequential. So because of the multiple access or the random access it will be much easier and faster to access the map data. So the most important part here is indexing so map data normally indexed or in a multiple parts. So that indexing files locations so keep map sheets are divided into sections which are distributed in such a matter that it can accelerate the search.

So when you are having map data so most of you will face problem only when you are accessing the huge raster data. Vector data may not be much of a (()) (11:51) but when you looking at raster data they go into volumes. So when you are accessing a last data if it is not index properly okay you can look at the (()) (12:02). Click on the vector data it will open the second but if you want to open a raster data it may take huge amount of time because it may not have been indexed.

If you have indexed in multi part indexing then probably it will open up at least in 1 tenth of the time that it use to open up without in indexing okay that is why you need to look at the positioning and indexing okay.

(Refer Slide Time: 12:25)



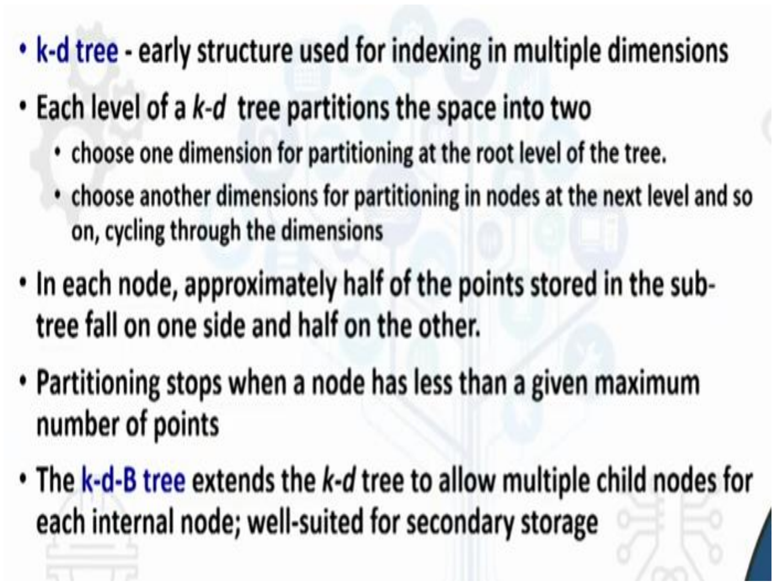
Spatial indexing

- Many maps tiled
- B-tree (Balanced)
- Grid indexing
- Quad tree: Points/regions
- R-tree (Based on MBR)

When you looking at spatial indexing that are many ways of indexing for example to find out a particular data there is B tree there is grid indexing Quad tree R tree. R tree basically based on MBR so it is sometimes very useful whereas when you looking at Quad tree it is based on points and regions whereas B tree is more of a balanced kind for example you have an R tree here first tree starts with the route you have the most top most information that is there in the R tree then you have leaf notes for example these are 2D leaf notes then you have the child note takes in and finally finds out where the particular thing is stored it is based on basically based on MBR's the multi boot records.

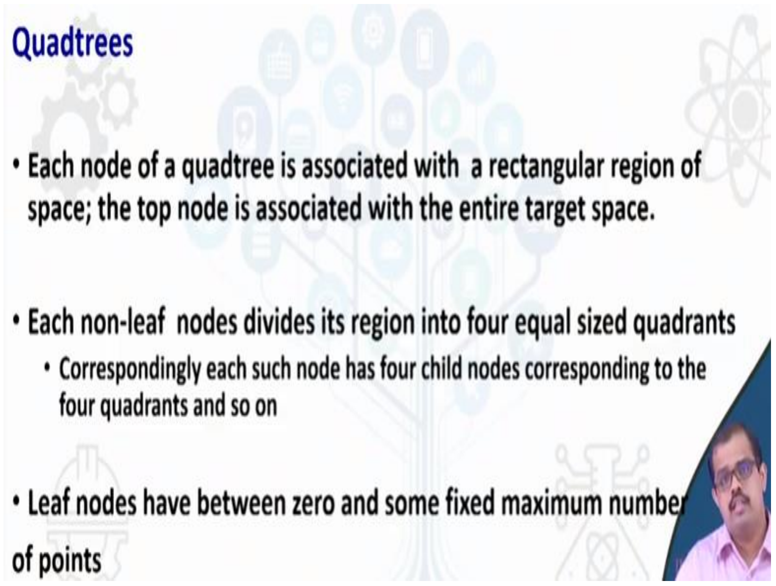
And whereas when you look at a Quad tree for example if you have this particular image okay it divides into 4 different I mean the boxes so once it is in the 4 different quadrants the Quad's now it is starts looking at that particular Quad then bricking the Quad's into 4 different Quads and going into those until it find a particular result. So this is mostly a extremely useful in terms when you have most of the raster data's. Whereas R tree has now become a industry standards for most of a applications in terms of spatial indexing.

(Refer Slide Time: 13:54)


- k-d tree - early structure used for indexing in multiple dimensions
- Each level of a *k-d* tree partitions the space into two
  - choose one dimension for partitioning at the root level of the tree.
  - choose another dimensions for partitioning in nodes at the next level and so on, cycling through the dimensions
- In each node, approximately half of the points stored in the sub-tree fall on one side and half on the other.
- Partitioning stops when a node has less than a given maximum number of points
- The k-d-B tree extends the *k-d* tree to allow multiple child nodes for each internal node; well-suited for secondary storage

So there is a first thing that early structure started was the k-d tree so each level of a k-d tree partition into 2 spaces I did show you with the R. So one dimension for partitioning root level and then next is at the notes each notes. So there is something called as k-d B tree extends the k-d tree that is nothing but a B tree allows the multiple child notes for each external note well suited for secondary storages. So that is why B tree has been extremely useful in terms of applications.

(Refer Slide Time: 14:31)

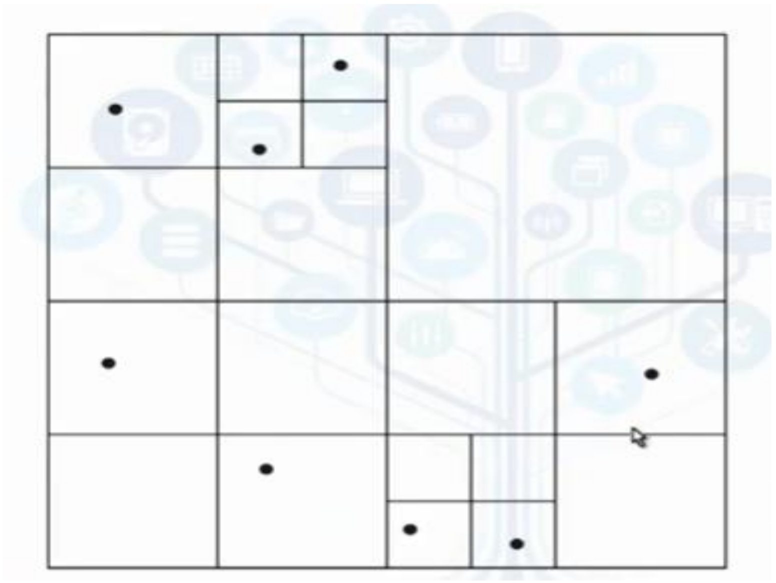

Quadtrees

- Each node of a quadtree is associated with a rectangular region of space; the top node is associated with the entire target space.
- Each non-leaf nodes divides its region into four equal sized quadrants
  - Correspondingly each such node has four child nodes corresponding to the four quadrants and so on
- Leaf nodes have between zero and some fixed maximum number of points

You have Quad trees each node of a Quad tree is associated with the rectangular region that is what I spoke about the top node is associated with the entire target space. So that gives you the entire target space are the entire database then it goes into different nodes different database different worksheets on how the data has to be connected. Each non-leaf nodes divides into region into 4 equal sized quadrant correspondingly each such nodes have 4 child notes corresponding to 4 quadrants and so on.

So once you have 4 child nodes it look at if that is quadrant if that is the quadrant then again divides into 4 child nodes then until it reaches a quadrant where the search operation halts okay. Leaf nodes have between 0 and fixed number of points so when you are non-leaf node is a 0 number of I mean expanding a child node then it will become a search will come down to that particular value.

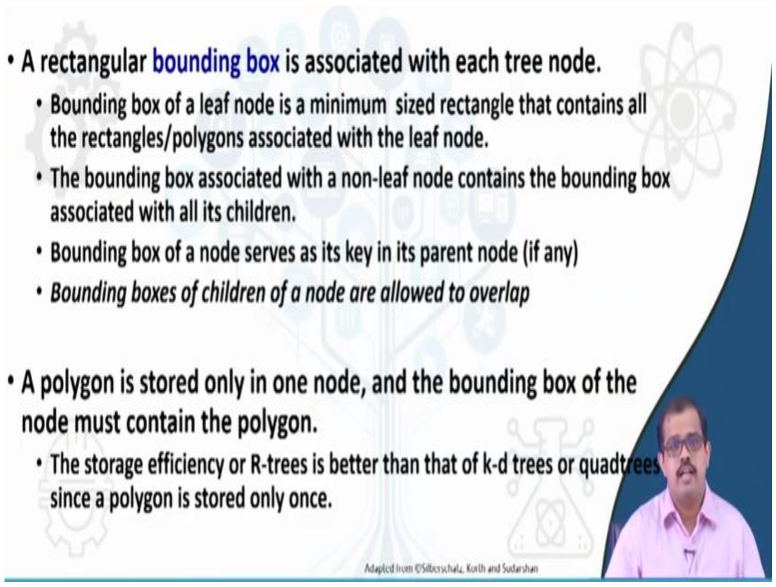
(Refer Slide Time: 15:35)



Just like this first you have your entire database so it divides into different quadrants once this is the first quadrant if you see these are the 4 , 1, 2, 3, 4 quadrants. So now it has to be find out that particular data's so you have based on relationship data you have a lot of data that is present here. So first it looks at this particular quadrant so divides it into 4 then it says that the it finds out 1 particular set of data that is already present then there is again the same ID that is presented in another quadrant also.

So divided into 4 so you get 2 more data sets so you access this particular thing similar it happens out throughout the entire database that is how the Quad tree or a Quad database works.

(Refer Slide Time: 16:26)

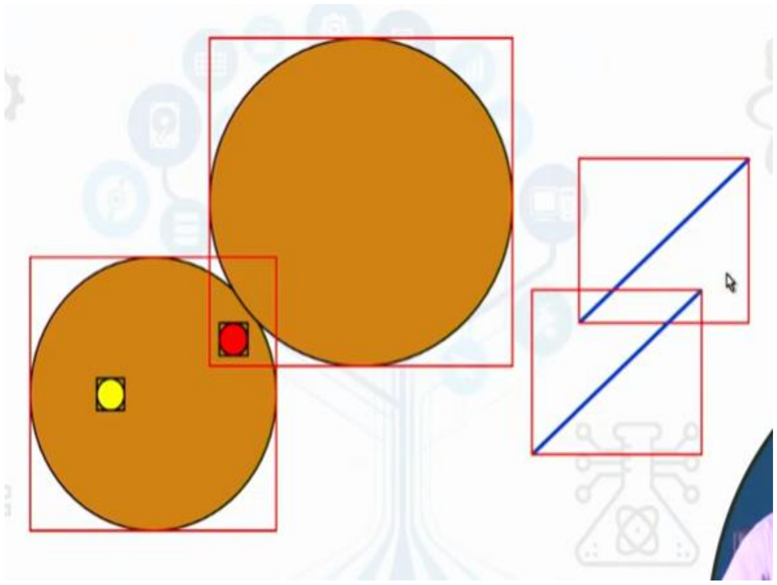


- A rectangular **bounding box** is associated with each tree node.
  - Bounding box of a leaf node is a minimum sized rectangle that contains all the rectangles/polygons associated with the leaf node.
  - The bounding box associated with a non-leaf node contains the bounding box associated with all its children.
  - Bounding box of a node serves as its key in its parent node (if any)
  - *Bounding boxes of children of a node are allowed to overlap*
- A polygon is stored only in one node, and the bounding box of the node must contain the polygon.
  - The storage efficiency or R-trees is better than that of k-d trees or quadtrees since a polygon is stored only once.

Adapted from ©Silberschatz, Korth and Sudarshan

Then next thing is R tree this I have already explained I would not get into these details but R trees have now become an extremely useful in have become as standard protocol in terms of developing it. How does an R tree is formed first a rectangular bounding associated with each tree node.

(Refer Slide Time: 16:47)

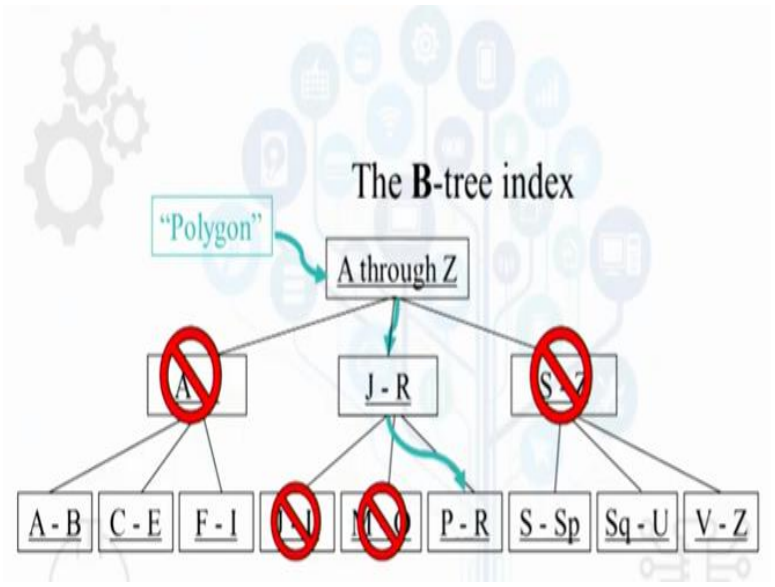


For example like this if these are the different points these are the polygons so you have a bounding box this is the line. So there is a bounding box rectangular bounding box that is created. Then bounding box of leaf node is the minimum size rectangle that contains all the rectangles are polygon associated with the leaf node then the bounding box associated with the non-leaf node contains the bounding box associated with all the children.

If there are many more children then each of them will have bounding boxes so bounding box of node serves as a key to its parent node you do not need to understand all of these in case you are just looking at GIS in just a holistic way. If you want to become an advanced user then only you have to understand how each of this R tree, Quad tree etc., works okay. The polygon is stored only in one node and bounding box of the node must contain that polygon okay so storage efficiency of R tree is better than that of any k-d tree or B tree or even Quad tree.

Since a polygon is stored only once whereas in Quad tree the polygon is repeated that is why there is the data consistency may not be there okay.

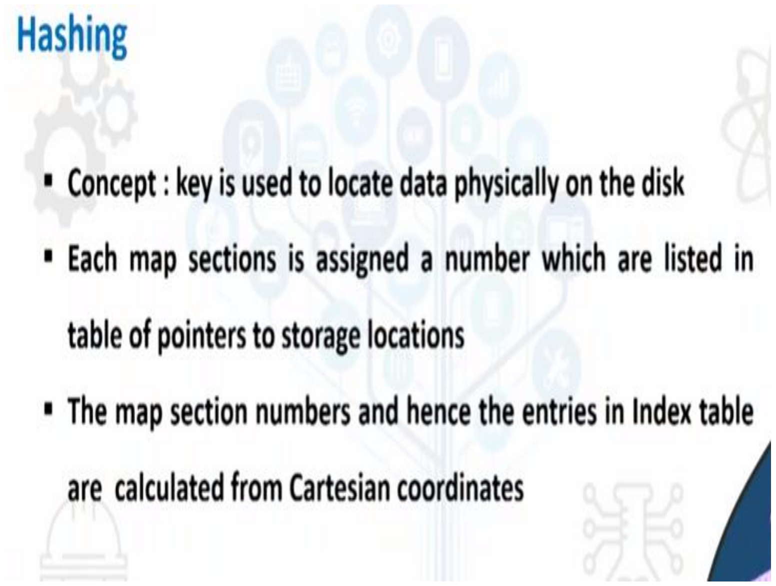
(Refer Slide Time: 18:02)



So this is example of B tree if you want to find out where between B and R polygon okay. So you have entirely through A through Z this is an one of those examples that have adopted from IBM. So if you look at you have an A through Z so first is such as A will have its nodes from A to I so you have these are this is a leaf node and these are non-leaf nodes and when you look at leaf node does not say that this value exist in this.

So it looks at this value exist so it goes into the lead node and digs deep into non-leaf nodes where you find out R okay. So this is how B tree works this is a general way of how it works but if you want to look at more and more detail please look at the textbooks which have already referred as extensive explanations of how a B tree works.

(Refer Slide Time: 18:58)

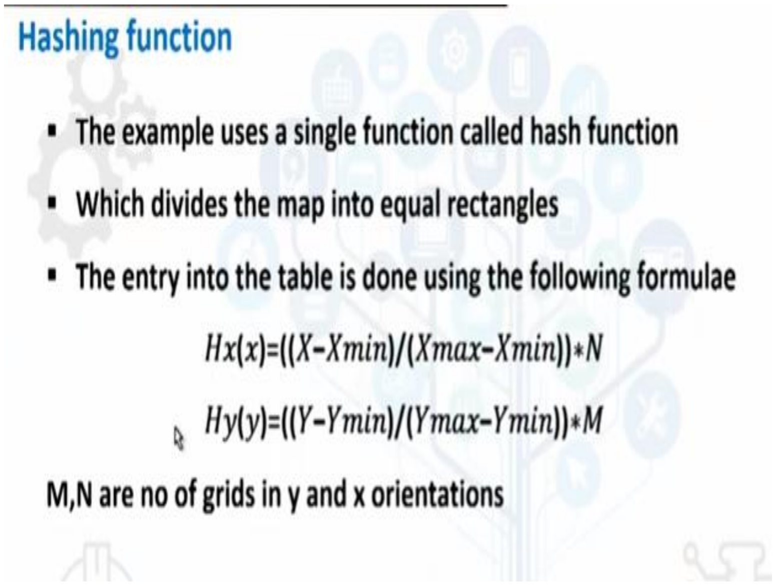


Hashing

- Concept : key is used to locate data physically on the disk
- Each map sections is assigned a number which are listed in table of pointers to storage locations
- The map section numbers and hence the entries in Index table are calculated from Cartesian coordinates

The next concept is hashing it has a key that is used to locate data physically on the disk. Each map sections have assigned a number which are listed in the table on the pointers to storage locations. The map section numbers and hence the entries in table are calculate based on the Cartesian coordinates that is how you hash a map and look at it.

(Refer Slide Time: 19:20)



Hashing function

- The example uses a single function called hash function
- Which divides the map into equal rectangles
- The entry into the table is done using the following formulae

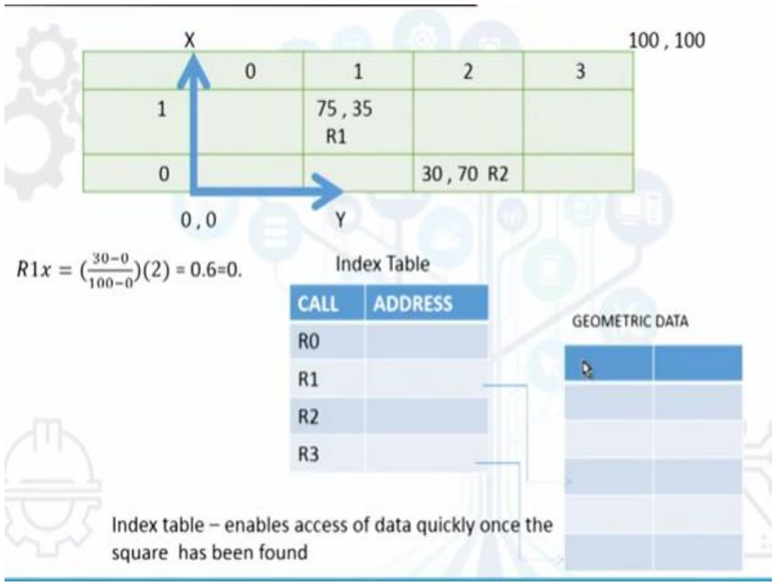
$$Hx(x)=((X-Xmin)/(Xmax-Xmin))*N$$

$$Hy(y)=((Y-Ymin)/(Ymax-Ymin))*M$$

M,N are no of grids in y and x orientations

So how can you do it for example if you want to hash this is one of the formulas that is used in terms of hashing you need Hx of x Hy of y which is giving you point information in a Cartesian coordinate system. So it divides a map into equal rectangles and each entry into a table done based on this okay.

(Refer Slide Time: 19:43)



$R1x = (\frac{30-0}{100-0})(2) = 0.6 = 0.$

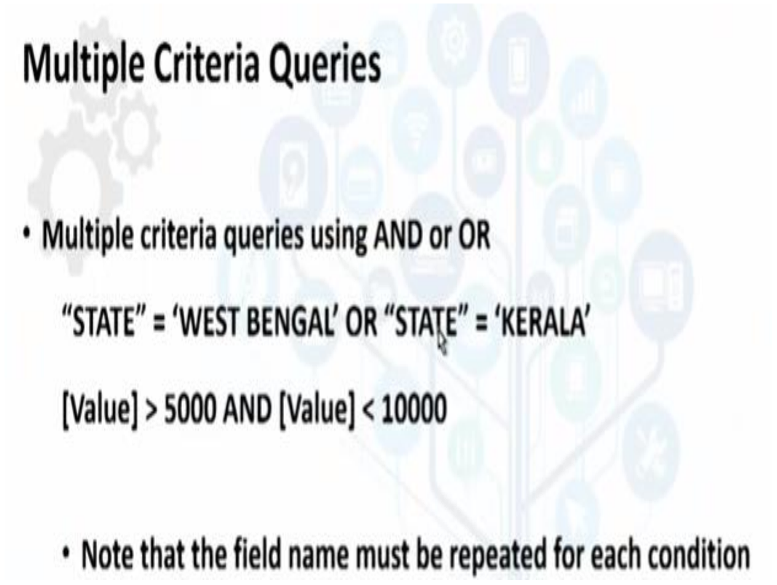
Index Table

CALL	ADDRESS
R0	
R1	
R2	
R3	

Index table – enables access of data quickly once the square has been found

So something like this okay this is how the table is so you have these values so based on this these are the entries that will be made and this is a index table and related geometric data would be stored in the other table okay.

(Refer Slide Time: 20:01)



Multiple Criteria Queries

- Multiple criteria queries using AND or OR

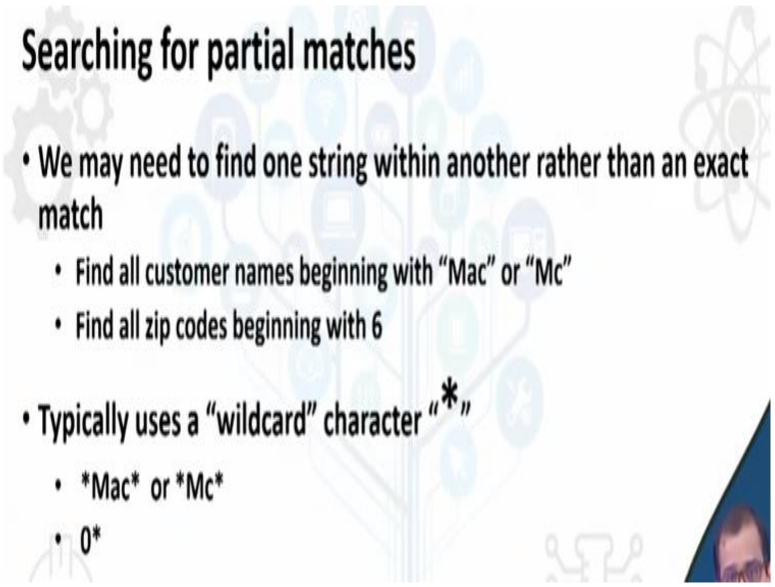
  "STATE" = 'WEST BENGAL' OR "STATE" = 'KERALA'

  [Value] > 5000 AND [Value] < 10000

- Note that the field name must be repeated for each condition

This will help in retrieval of data much fast okay sequel can have a single query sequel can have a multiple query. So if you want to have a multiple query it can handle something like this for example state equal to the west Bengal or state equal to Kerala and value greater than 5000 and value less than 10000 which means i have looking at in the state of west Bengal and Kerala where the population value is less than 5000 and population value is less than 10000. So give me all the queries so this is one way of looking at multiple queries in terms of representation or searching.

(Refer Slide Time: 20:40)



Searching for partial matches

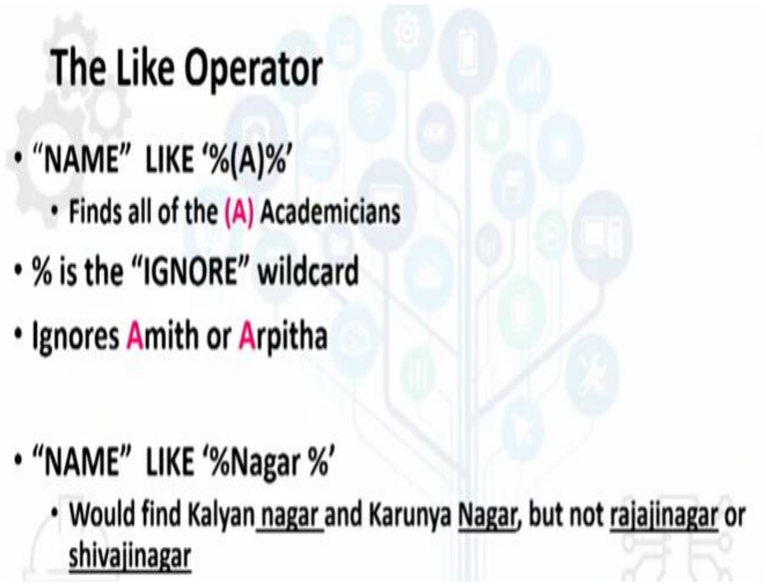
- We may need to find one string within another rather than an exact match
  - Find all customer names beginning with "Mac" or "Mc"
  - Find all zip codes beginning with 6
- Typically uses a "wildcard" character "*"
  - *Mac* or *Mc*
  - 0*

Then you have for example you do not know particular let us say a function or a field okay and you know only the partial name of it. So you can do even the partial matches in terms of querying so for example if you want to find out all customer names begin with Mac or Mc okay you can even write it as Mc Donald or Mac Donald. So how do you write it depends on I mean the way it is represented so it is just the I mean if you want to query so name = Mac and Mc so it will give you all the names that are there or Mac or Mc so that will give only those names which are there.

So find all the zip codes beginning with 6 okay all the beginning with the 6 will be listed okay so you normally when these are there partial matches are there you do not know the other part. For example Mc Donald I do not know the Donald part let us say I know only Mac. So I will write Mc star it means to say that star Mc star which means anything that is previous to Mc anything that is after Mc list all of them wherever Mc exist even in the entire list actually locates okay.

If something like Gunasekaran Mc okay continuous name let us say Gunasekaran Mc so even less even if it is Mca degree because Mc is actually occurring there. So it is also list Mca okay so that is how you can look at the partial match then select the whichever data you need okay you may not be the same data.

(Refer Slide Time: 22:24)



The Like Operator

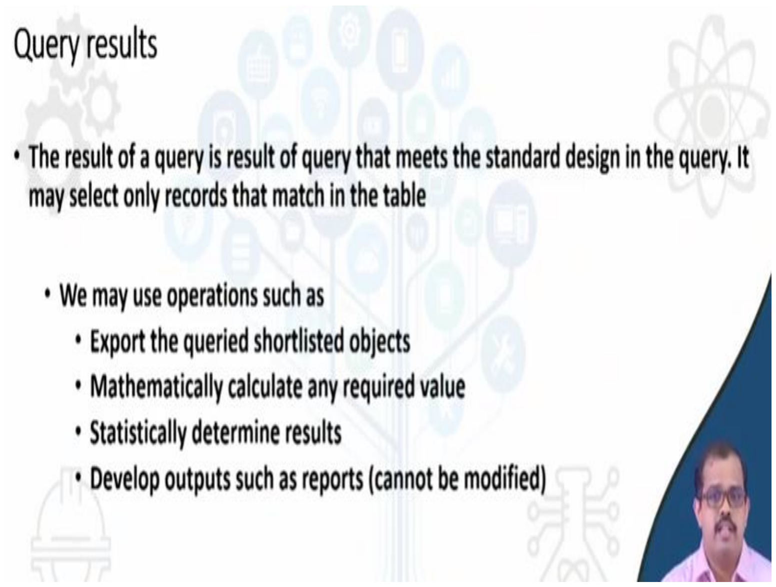
- "NAME" LIKE '%(A)%'
  - Finds all of the (A) Academicians
- % is the "IGNORE" wildcard
- Ignores Amith or Arpitha
- "NAME" LIKE '%Nagar %'
  - Would find Kalyan nagar and Karunya Nagar, but not rajajinagar or shivajinagar

Other one the like operator for example if you want to find out there is certain this for let us say in this example you have NAME, LIKE % A% it will find all the names starting with A. For example I have a field name that is Academicians so academicians and I would give the names. So any Academicians with starting with A will be mentioned in that particular query output. So if you have percentage then is the IGNORE wild card okay.

In case you want to IGNORE it just add a percentage so it ignores this particular wild card. So now if I put it as A there is a Academicians like Amit and Apritha so both of them will be ignored in this particular set of queries. So you can also query something like this you the name you do not know exactly that name so you have to say NAME LIKE % Nagar% it would find Kalyan space Nagar so you need only Nagar you do not something like Kalyan okay.

So it is only searching for Nagar so Nagar exist in Kalyan Nagar and Karunya Nagar so it gives you the output. But if there is something like Rajaji Nagar or Shivaji Nagar it would not give it so if you want to search Rajaji Nagar or Shivaji Nagar then you may have star Nagar star so it may it will give you everything that is in between Rajaji Nagar and Shivaji Nagar including Kalyan Nagar and Karunya Nagar so that is how the like operator works.

(Refer Slide Time: 24:04)

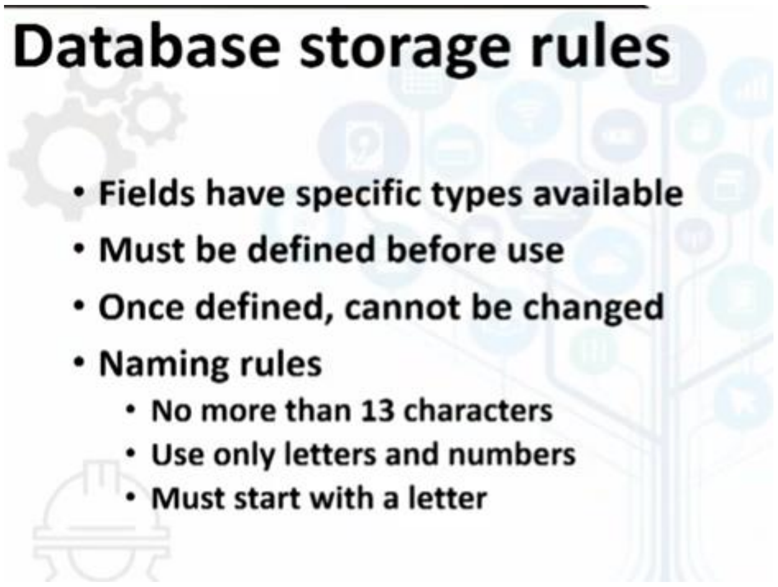


Query results

- The result of a query is result of query that meets the standard design in the query. It may select only records that match in the table
  - We may use operations such as
    - Export the queried shortlisted objects
    - Mathematically calculate any required value
    - Statistically determine results
    - Develop outputs such as reports (cannot be modified)

Then you have Query result the result of a query meets standard design of a query it is how the query has been designed that is where the database creator will actually design how the queries are made so it we may use operations for example export the queries short list objects mathematically calculate any required value statically determine results. So develop outputs such as reports only thing is that reports cannot be modified. So based on this whatever the query you do this if there is database administer has defined the output of a query in a certain way only in that way the query results will be provided okay.

(Refer Slide Time: 24:49)



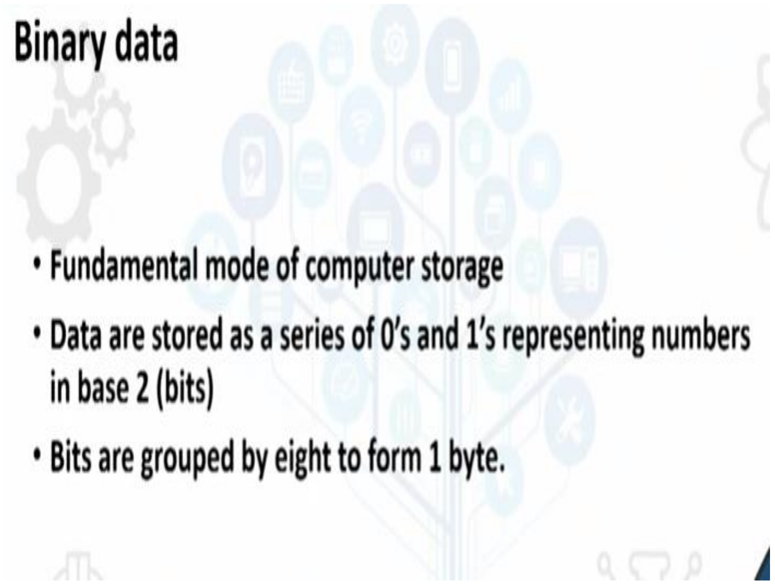
Database storage rules

- Fields have specific types available
- Must be defined before use
- Once defined, cannot be changed
- Naming rules
  - No more than 13 characters
  - Use only letters and numbers
  - Must start with a letter

Then you have certain rules for storage of the database now I am struggling with queries now again go back to database has a storage rules and fields have specific types available for example when you are naming rule not I did mention it before you cannot have more than 13 characters in your database as a name you can use only letter and numbers must start with the letter. Once defined it cannot be there first due to certain constraints many of the databases where not allowing the special characters.

But in today scenario we can use special characters as name of the database so it cannot have hidden spaces so that you should remember. So must be defined before use when you are creating a database you have to defined the name of the particular database. So these are certain storage rules you should know when you are creating a database.

(Refer Slide Time: 25:49)

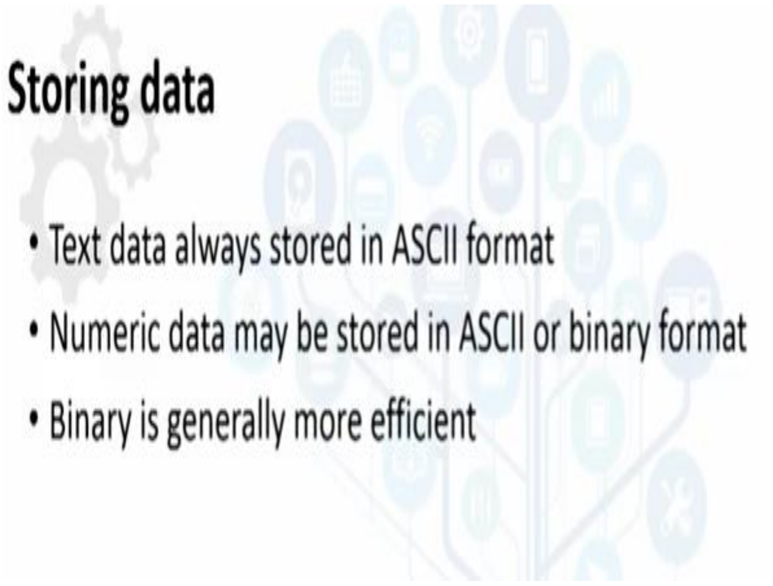


Binary data

- Fundamental mode of computer storage
- Data are stored as a series of 0's and 1's representing numbers in base 2 (bits)
- Bits are grouped by eight to form 1 byte.

So there are different data's that database can handle or a queries can be made of so you have fundamental mode of computer it uses the fundamental mode of computer storage something like a binary data okay. Data's are stored as a series of 0's and 1's representing the numbers into it if you know it is a binary data. So bits are grouped by 8 to form 1 byte 1024 bytes is equal to 1 MB and 1024 MB's is equal to 1 gig and so on okay so that is how the binary data can be used.

(Refer Slide Time: 26:25)

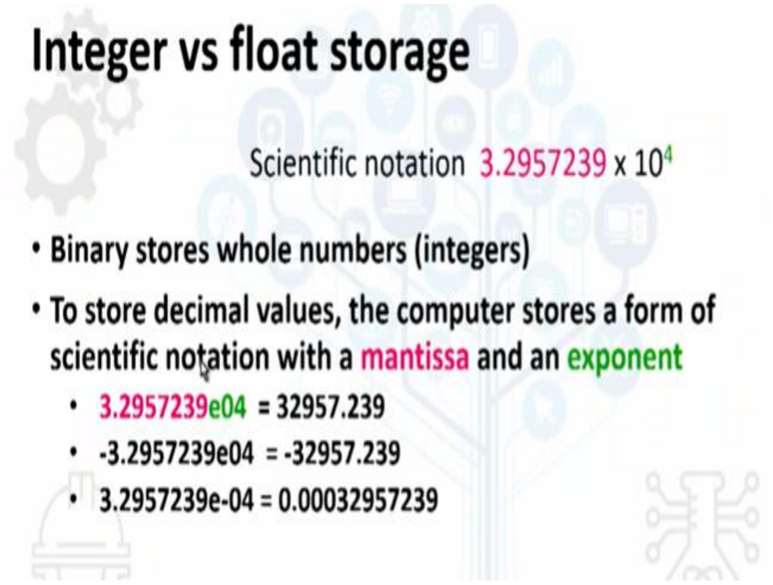


Storing data

- Text data always stored in ASCII format
- Numeric data may be stored in ASCII or binary format
- Binary is generally more efficient

Then you can use it in the form of ASCII okay so normally ASCII is used in alpha numeric storage or those which are non-numeric which are numeric so when you are looking at storing data text data always are normally stored in ASCII format numeric data may be stored in ASCII or binary normal it is binary format. Binary is generally more efficient in terms of storage capabilities.

(Refer Slide Time: 26:54)



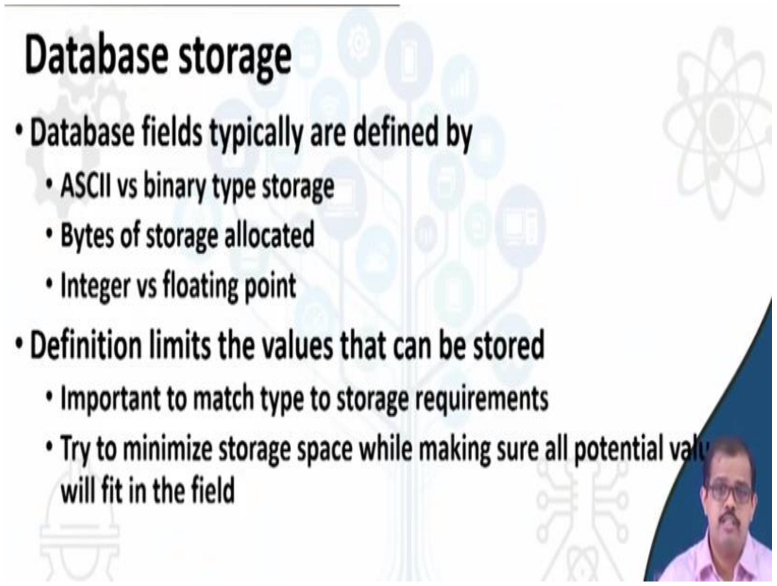
Integer vs float storage

Scientific notation 3.2957239 x 10^4

- Binary stores whole numbers (integers)
- To store decimal values, the computer stores a form of scientific notation with a mantissa and an exponent
  - 3.2957239e04 = 32957.239
  - -3.2957239e04 = -32957.239
  - 3.2957239e-04 = 0.00032957239

And when you are storing the typically this is one of those examples which many students ask in terms of storage. For example binary stores it in form of a whole numbers okay. If you want to store a decimal value okay the computer stores in the form of scientific notation probably you understand what is a mantissa and exponent okay. It stores always in form of a exponent for example this may be scientific notation okay this is your scientific notation the storage in a computer in the form of mantissa and exponent. So always your data is stored in form of a mantissa and exponent okay.

(Refer Slide Time: 27:41)



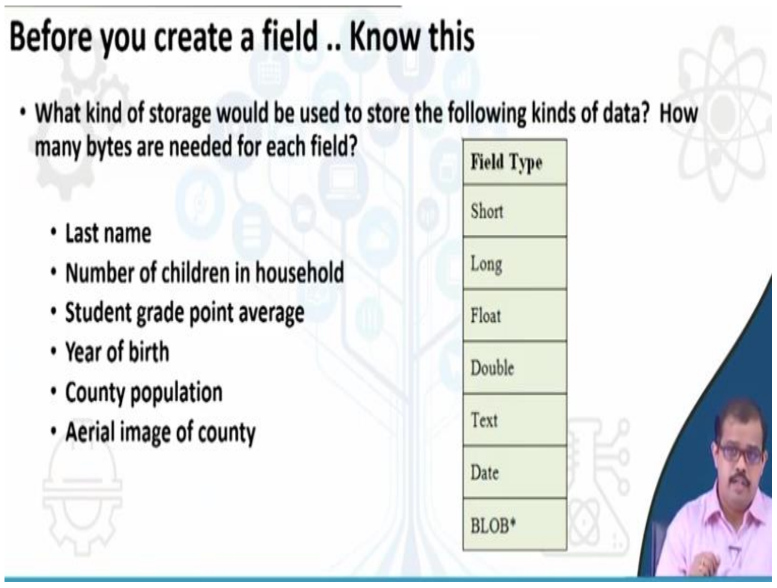
Database storage

- Database fields typically are defined by
  - ASCII vs binary type storage
  - Bytes of storage allocated
  - Integer vs floating point
- Definition limits the values that can be stored
  - Important to match type to storage requirements
  - Try to minimize storage space while making sure all potential valu will fit in the field

So when you are looking at the database storage so you have to typically storage the size of the storage typically depends on 2 things one is the ASCII versus the binary types storage what type of storage is being used? What is those that is behind is being used then bytes of storage allocated so it also depends on user how he or she wants to store the data. Then integer versus floating point when you are looking at floating points there are certain operations where your floating points is must so use floating points okay.

Definition limits to values that can be stored there are certain definitions that can limited then important to match types to storage requirement and try to minimize storage space that is very important in terms of a database storage that why you use compacting.

(Refer Slide Time: 28:32)



Before you create a field .. Know this

- What kind of storage would be used to store the following kinds of data? How many bytes are needed for each field?
  - Last name
  - Number of children in household
  - Student grade point average
  - Year of birth
  - County population
  - Aerial image of county

Field Type
Short
Long
Float
Double
Text
Date
BLOB*

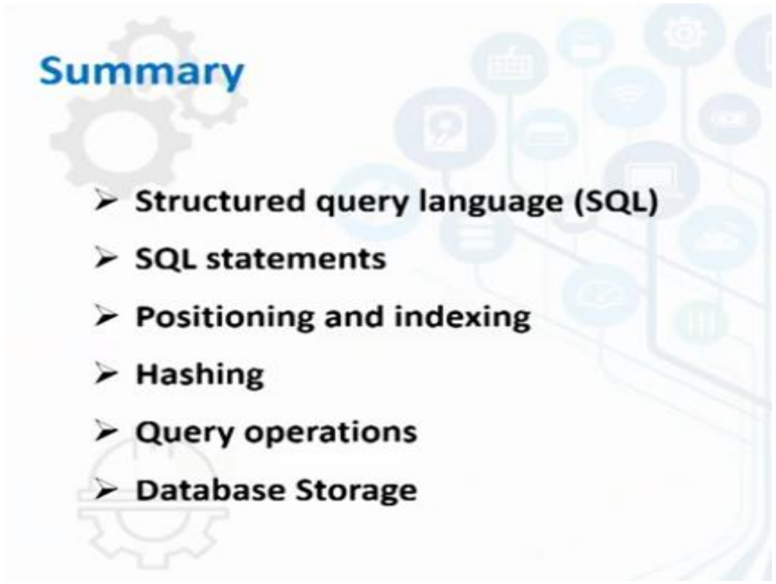
But before you create any field okay or storing attribute information you should know certain things that is first thing is what kind of field you are trying to use if you are using a name students name okay you should are you using a long or short or for example you are storing an age of an person it is long integer short integer that is the way you should know what kind of data you are storing what is the data type you are storing whether it is a integer it is float or it is string.

So how are you actually storing for example if it is double it is double characters so look at the way it has to be stored. First create that kind so that is why you need to plan it first what data you are storing what is the type of the data available the only you can look at it. For example last name may be very different from number of children in household so data that is needed type that is needed in the last name is extremely different.

So when you look at the student grade point averages is more of an floating point whereas number of children in household is an integer okay. So we have to look at how different this data is. Then year of birth the way it is stored is very different okay the country population it cannot be it is in integers it cannot be your floating point and aerial image of country is in certain format it is not you cannot store it in the format as in the above cases okay.

So before you create any field look at what particular thing that you are trying to use and what is the field count how you are looking at the field what is the data that you are capturing. So then only create that particular field okay without that it may not be much useful.

(Refer Slide Time: 30:26)



Summary

- ➢ Structured query language (SQL)
- ➢ SQL statements
- ➢ Positioning and indexing
- ➢ Hashing
- ➢ Query operations
- ➢ Database Storage

(refer time: 30:26)So we looked at certain what do you mean by structure query language we looked at some SQL statements then we looked at positioning and indexing why it is important we looked at hashing then query operations okay how do you actually look at query and how a different operations for example Like etc., We looked at and finally the database storage how do you store a database and very importantly what are the field formats that you have to store and how it has to be stored.

So this is all about the SQL and the database so let us meet in the next class with the next set of different issues of what I have divided this particular module is more dedicated to developing a database and trying to understand with different queries. So once I have understood this I will again go back to the module 5 where would be I will be explaining you certain things about how actually when you start digitizing what are the errors you get and how you correct all those errors that will be in the last lecture of same module which I will go back to the module 5 so until the next module have a nice time thank you very much.