

Spatiotemporal Tile Indexing Scheme

Oscar Perez Cruz (Polytechnic University of Puerto Rico)

Dr. Ranga Raju Vatsavai (Geographic Information Science & Technology)

Table of Contents

1. Abstract.....	03
2. Introduction.....	03
3. Database, objects, and indexes.....	05
4. System Architecture.....	08
5. Meta data harvesting system.....	09
6. Spatiotemporal database management application.....	13
7. Web page.....	18
8. Querying database.....	21
9. Conclusion and future direction.....	23
10. Reference.....	25

1. Abstract

Large amount of satellite products have been acquired over large geographic regions for the last few decades. This increasing number of satellite products with varying spatial, spectral, and temporal resolutions makes the data management a difficult task. These images are used in many applications ranging from the creation of high-resolution population databases to monitoring biomass over large geographic regions. The complexity of managing such an amount of images for an efficient search and data discovery is increasingly becoming a problem. To solve this problem, a spatiotemporal tile indexing scheme is created. This scheme is automatically created by harvesting the meta data directly from geographic satellite images and/or ancillary data. This meta data is stored in a database where other users can access the data through a web page anywhere in the world. This web page is developed in a way where they can search for data by spatial, temporal, and spatiotemporal extents using PHP. The system that harvests the image to store its meta data in the database server was developed using C++. The stored data can be indexed to make its search much faster and more accurate. Another feature of the database is the ability to create views for the search engine to search within the results. The end result for this system is that it will be easy to use and it can drastically reduce the time to search and retrieve the satellite data. In the future, this system can be further extended with additional query constraints to specify the search for other attributes that the data may have, for example, percent of cloud coverage in an area and analyze other kinds of images that may contain any spatial and/or temporal data.

2. Introduction

During the past decade a new millennium has come bringing a revolution in the discovery of new technology. As new technology rises, new ways of acquiring data is discover. As the years pass and new technologies are being developed, data just keep pilling up since new and more efficient methods of data management haven't being created. This makes managing data a very difficult and slow process. This problem has led us to the research of developing a data management system that will help in the management of the meta data within geographic satellite images. At the same time we would need a database to sort the meta data of these images by time and space and a way to access and deliver the meta data all around the world. The objective of this system is to generate an improvement in data management, generating an efficient search and data discovery by returning faster results, and granting the user an easier access to the meta data. Currently the system works with geographic satellite images from the Moderate-Resolution Imaging Spectroradiometer (MODIS) and the Advanced Wide Field Sensor (AWIFS).

(MODIS is a payload scientific instrument launched into Earth orbit by NASA in 1999 on board the Terra Satellite and in 2002 on board the Aqua satellite. The instruments can capture data in 36 spectral bands ranging in wavelength from 0.4 μm to 14.4 μm and at varying spatial resolutions (2 bands at 250 m, 5 bands at 500 m and 29 bands at 1 km). Together the instruments image the entire Earth every 1 to 2 days. They are designed to provide measurements in large-scale global dynamics including changes in Earth's cloud cover, radiation budget, and processes occurring in the oceans, on land, and in the lower atmosphere. Three on-board calibrators (a solar diffuser combined with a solar diffuser stability monitor, a spectral radiometric calibration assembly, and a blackbody) provide in-flight calibration.) [7]

(Advanced Wide Field Sensor (AWiFS) camera on board IRS P6 RESOURCESAT satellite is an improved version compared to the WiFS camera flown in IRS-1C/1D. AWiFS operates in four spectral bands identical to LISS-III, providing a spatial resolution of 56 m and covering a swath of 740 Km. To cover this wide swath, the AWiFS camera is split into two separate electro-optic modules, AWiFS-A and AWiFS-B. The IRS-P6 spacecraft mainframe is configured with several new features and enhanced capabilities to support the Payload operations. The payloads can be operated either in Real Time mode by direct transmission to ground station or in Record and Playback mode using an on-board 120 GB capacity Solid State Recorder. The various modes of Payload operations can be programmed a priori through a Telecommand Processor (TCP).) [8]

3. Database, objects, and indexes

A database is a large collection of interrelated data. Each database has an amount of records, which represent an entity or object in the database, and an amount of attributes that describe it. Depending on these attributes, the database maybe object oriented or not. A database that is object oriented has an exact amount of attributes with the necessary types that describe the objects within the database. They system that we are developing is a combination of spatial and temporal databases which represent objects in time and space. This database is called a spatiotemporal database. Its objects have an identifier, a spatial attribute, and a timestamp. A timestamp can mark an event that the object witness. A spatial attribute is a basic type of attribute for representing geometric objects since its basic data types are: point, line, and region.

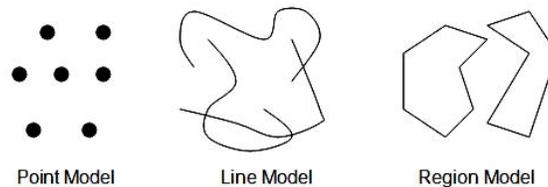


Figure 1: Spatial Attribute Basic Data Types

A point represents a location of an entity, while a line represents a network entity, and a region represent an entity with a large area. For example, a point could be the auditorium of a building, while a line could be the floor where the auditorium is, and a region could be the entire building. For the system that we are developing, combining these attributes to come up with a new type of database is the best idea because if both attributes where separate then the system wouldn't produce an efficient search because it would return the meta data of to many images to analyze. At the same time, having all the data in a database doesn't solve the time issue for searching and getting faster results very well because if the database has millions of records it would have to do a sequential scan through each of them. This is where indexes come in.

An index can ensure a fast access to records in a database based on a search key. With the use of this search key, a sequential scan through the database is avoided. To develop the database, the following index structures are used:

- Tile devised to index a boundary spatial attribute
- Geometric devised to index the spatial attribute
- Temporal devised to index time attributes
- Spatiotemporal index the spatial and time attributes of an objects

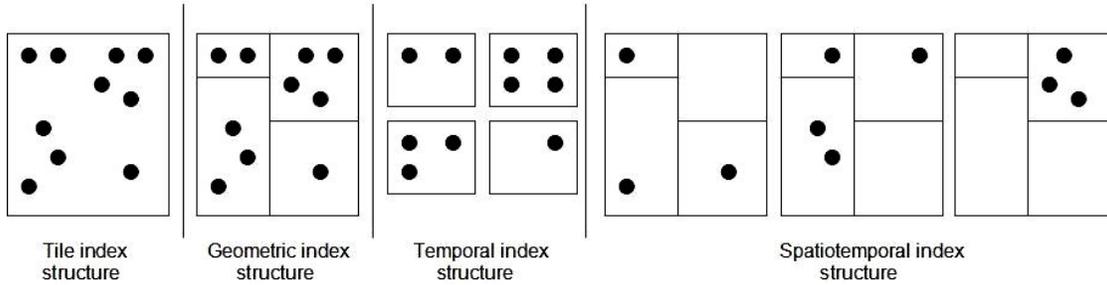


Figure 2: Different Index Structures

An example of how these indexes behave in the database is shown in the following figures. Think yourself as the user and you have this dataset in the database that you want to index for a faster search. You have 15 entities enclosed in a big box known as R. This is what a tile index is. It encloses an amount of data in a box, no matter what kind of data it is. Alongside this box, there is the design of a tree that shows the layers of how the index helps narrow the search for a specific element. This box in the index tree is called a root tree. It is the first node of data that can identify if there is a value that you are searching for inside that box. Now inside the box there are 4 more boxes enclosing up to 4 entities. These 4 boxes in the tree are called intermediate nodes and the entities within those boxes are known as leaf nodes. They enclose until four entities because that is the minimum amount of objects that are permitted in that node array. Now think of the ID that these boxes have as their location for a spatial attribute. As an example, in a spatial query the parameters are specified that it is searching for an entity in the location where region a binds where entity 2 resides. The tree would go from the root node (R) to the intermediate node (a) and search for the leaf node 2. This cuts the search by spatial means by more than half of what it would have taken by scanning the database one object at a time. But for the type of database that is being constructed, this type of search is not enough to give us a definite result because in the real database we could have thousands of results that satisfy that same parameter.

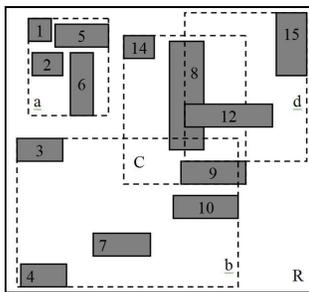


Figure 3: Dataset R

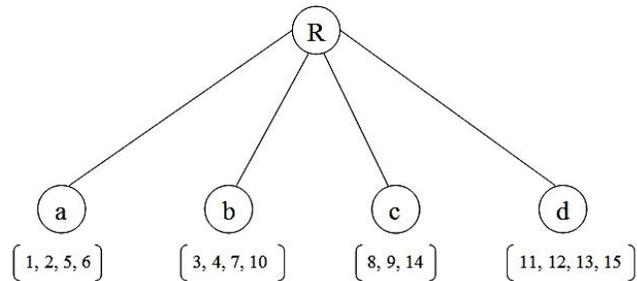


Figure 4: Geometric Index of dataset R

Because of this we bring the temporal index to add another parameter to our search. In our example, each object has a distinctive time in their attributes. These times range from the year 2000 till the year 2010. As the index is constructed, it has an amount of 2 elements per array. The tree is constructed in the theory of what comes before and what comes after by splitting the range of years in two; what came before and after 2005. Now for the parameters of the temporal query it is searching for an object that has the year 2001. Since 2001 came before 2005, the search goes down to the intermediate node containing the year 2002. The same thing happens here and the search goes to the leaf node of the year 2000 and 2001. In this node is where we find the time that we are searching for which contains the entity of object 2. This is a good search for this kind

of dataset since it gave us only one result but for the database that is being constructed, time alone is not enough to give us the desired result for the same reason that location alone is not enough.

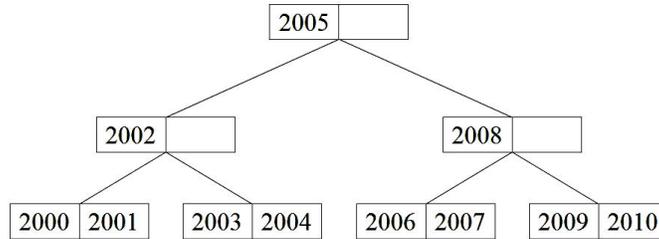


Figure 5: Temporal Index of dataset R

This results in the creation of the spatiotemporal tile indexing scheme and the spatiotemporal query as we combine both search parameters of space and time to get a much more efficient result. As can be seen in the sample of a table from the database, the search takes both parameters into consideration to get the result.

ID	Year
1	2000
2	2001
3	2002
4	2003
5	2004
6	2005
7	DATA 2006
8	2007
9	2008
10	2009
11	2010
12	2004
13	2007
14	2006
15	2009

Figure 6: Example of database containing Dataset R

4. System architecture

This system is devised by five components:

- Meta data harvesting system
- Spatiotemporal database
 - Database management application (SDBMA)
- PostgreSQL database server
 - Extension: PostGIS
- Application server (PHP)
- Web server (Apache)
- Web page

Each of these components has a different task to do in which affects the other component. Shown in the image below is the connection between each part of the system.

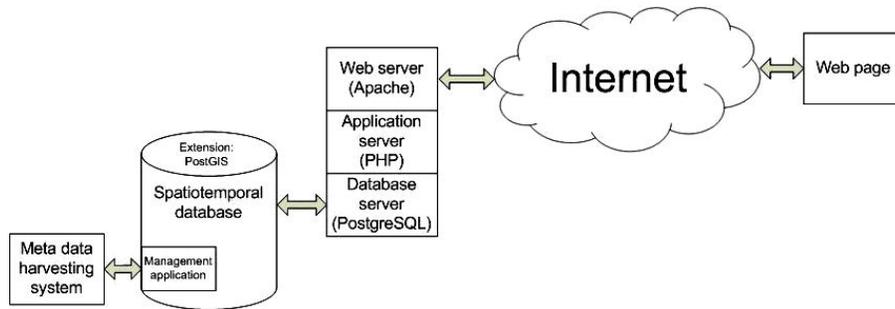


Figure 7: System's Architecture Connections

The meta data harvesting system is in charge of harvesting the image and giving the necessary meta data to the spatiotemporal database management application. The SDBMA is in charge of controlling and managing the database. The spatiotemporal database is where the data will be stored which is connected to the database server. The application server contains the PHP code that connects to the database server that runs the query depending of what it receives from the web server. The web server is what contains the web page that is accessed using a browser, like Internet Explorer and Mozilla Firefox, from anywhere in the world.

5. Meta Data Harvesting System

Meta data is data that provides information about other data and harvesting is the process of extracting meta data from the ancillary data sources. In this case, the meta data in the satellite images is the information that we harvest to get the last modified date and the geographic coordinates of each corner of the image. This system is access by the SDBMA by entering the name of the table of where the meta data will be stored and the path of where the image is located. This system uses the functions of the Geospatial Data Abstraction Library (GDAL), which is an open source library that supports almost 100 types of image formats. Also this library is not dependent of any particular satellite or sensor. Below is shown the meta data of an image that has been harvested:

```

PROJECTION INFORMATION
-Driver: GTiff/GeoTIFF
-Image Size in Pixels:
  -Width: 512
  -Height: 512
  -Raster Bands in dataset: 1
-WKT Image Projection:
PROJCS["unnamed",GEOGCS["NAD27",DATUM["North_American_Datum_1927",SPHEROID["Clarke 1866",6378206.4,294.9786982139006,AUTHORITY["EPSG","7808"]],AUTHORITY["EPSG","6267"]],PRIMEM["Greenwich",0],UNIT["degree",0.0174532925199433],AUTHORITY["EPSG","4267"]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-117],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],UNIT["unknown",1],AUTHORITY["EPSG","26711"]]]
-Pretty WKT Projection Format:
PROJCS["unnamed",GEOGCS["NAD27",DATUM["North_American_Datum_1927",SPHEROID["Clarke 1866",6378206.4,294.9786982139006,AUTHORITY["EPSG","7808"]],AUTHORITY["EPSG","6267"]],PRIMEM["Greenwich",0],UNIT["degree",0.0174532925199433],AUTHORITY["EPSG","4267"]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-117],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],UNIT["unknown",1],AUTHORITY["EPSG","26711"]]]
-Origin: <440720,3.75132e+006>
-Pixel Size: <60,-60>
  
```

Figure 8: Image Harvested Meta Data

As it can be seen, the meta data gives us four key elements that help us calculate the latitude and longitude of the top left and bottom right corner of the image. These elements are:

- Pixel size
- Image size
- Latitude of origin
- Central meridian

During the calculation of the geographic coordinates in the top left and bottom right corner of the image, we had to consider the offset of the distance that exist between the latitude and longitude degrees depending on the latitude. This is because the closer to the poles the more distances there is between latitude degrees and the less distance there is between longitude degrees. These distances are shown in the following tables:

Latitude Range	Distance per Degree
0° to 15°	110,574
16° to 30°	110,649
31° to 45°	110,852
46° to 60°	111,132
61° to 75°	111,412
76° to 90°	111,618

Table 1: Latitude Distance Offset

Latitude Range	Distance per Degree
0° to 15°	111,320
16° to 30°	107,551
31° to 45°	96,486
46° to 60°	78,847
61° to 75°	55,800
76° to 90°	28,902

Table 2: Longitude Distance Offset

After calculation of the offset, the latitude and longitude of the top left corner and bottom right corner were calculated using this offset and the key elements of the meta data. Another element that was harvested from the image was the last date the image was modified to get a temporal type data. The information that is given back to the SDBMA is the:

- Top Corner Latitude
- Top Corner Longitude
- Bottom Corner Latitude
- Bottom Corner Longitude
- Last Modified Date

```

IMAGE SIZE:
-Height: 512 pixels
-Width: 512 pixels
-Pixel Resolution On the X-Axis: 60
-Pixel Resolution On the Y-Axis: -60
COORDINATES:
-Central Meridian: -117
-Latitude of Origin: 0
-Standard Parallel 1: 0
-Standard Parallel 2: 0
COORDINATES IN METERS:
-Top X: 440720
-Top Y: 3.75132e+006
-Bottom X: 471440
-Bottom Y: 3.7206e+006
COORDINATES IN DECIMAL DEGREES:
-Top X: -118
-Top Y: 34
-Bottom X: -112
-Bottom Y: -1
IMAGE FILE PROPERTIES:
-Image Location: utm.tif
-Date Image was Taken: 11/06/2002

```

Figure 9: Key Elements of the Meta Data Organized

Below is the schematic for the meta data harvesting system and how it interacts with the SDBMA.

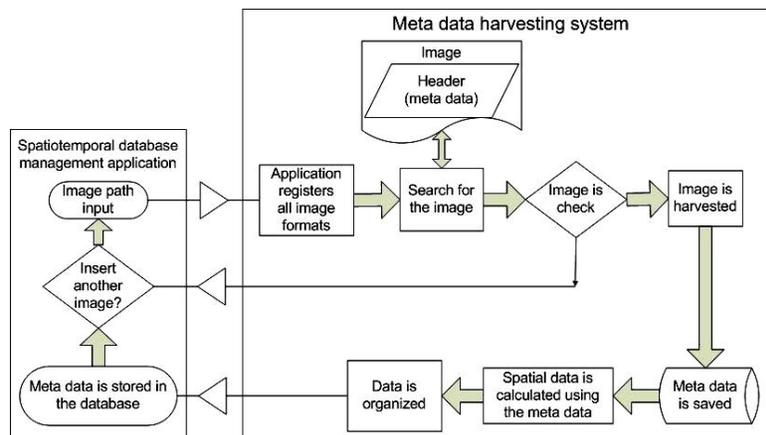


Figure 10: Meta Data Harvesting System

6. Spatiotemporal Database Management Application

A database management system is a set of functions that controls the creation, maintenance, and the use of a database. This system allows organizations to place control of database development in the hands of database administrators and other specialists. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way. This system is called spatiotemporal database management system because it manages a database with spatial and temporal extents. This system is design to be able to connect to any database in a PostgreSQL database server. Once trying to connect to a database, the user should be concerned that the database that he's connecting to is a spatial database with spatial functions. Once a successful connection is made, the user can create, delete, view, and index the tables in the database, has is the ability to insert the meta data of an image by harvesting it and can search for this meta data by spatial and temporal extents.

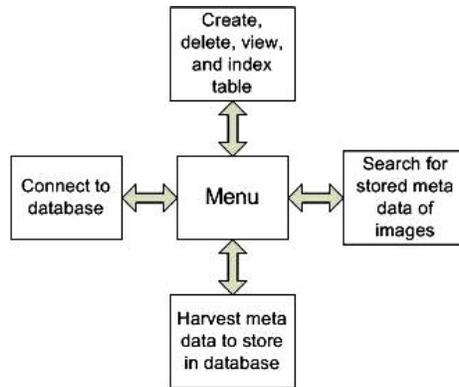


Figure 11: Structure of the Menu for the SDBMS

To connect to a database in the PostgreSQL server, the user needs to input the login information which consists of:

- Hostname
- Port number
- Database name
- Username
- Password

After placing the login information, the server checks for validation and let's the user access the other functions in the SDBMA.

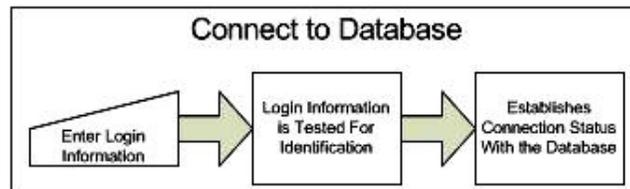


Figure 12: Function Design - Connect to Database

If this is the first time accessing the database, the first thing that the user should do is create a table. The table is created by entering its name and the system will do the rest. Using this system, the tables will be created with the following columns:

- Disc Name
- Image Name
- Image Date
- Top Corner Latitude
- Top Corner Longitude
- Bottom Corner Latitude
- Bottom Corner Longitude
- Numeric Date
- Location

The disc name column is used for the CD or DVD of where the image is taken because that is where the first data for the system is being gathered from. The image name column contains the path of where the image is stored. The image date and the corner latitudes and longitudes columns are for showing purposes when the table is being viewed or an image is being searched for. The numeric date column is used to contain the same date as the image date column, except that it is in integer mode so an easier and more efficient search method can be done in the database. The location column is a geometry type attribute that will combine the latitude and longitude attributes to create region type geometry. The numeric date and location columns are used to index the database and search for an image.

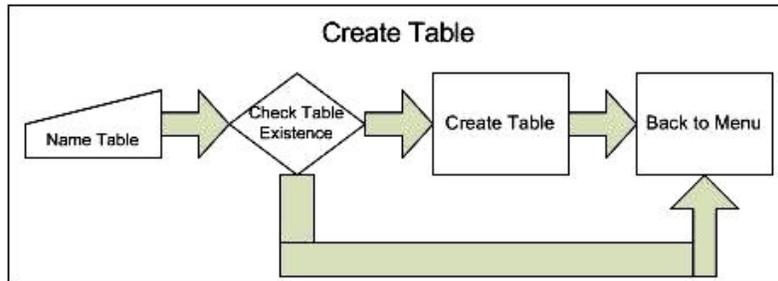


Figure 13: Function Design – Create Table

When the user wants to view, delete, index, insert the meta data of an image or search for an image in the database, all that he has to do is enter the name of the table and the system will check for its existence. When viewing a table the system will print each record with all the columns except for location and numeric date.

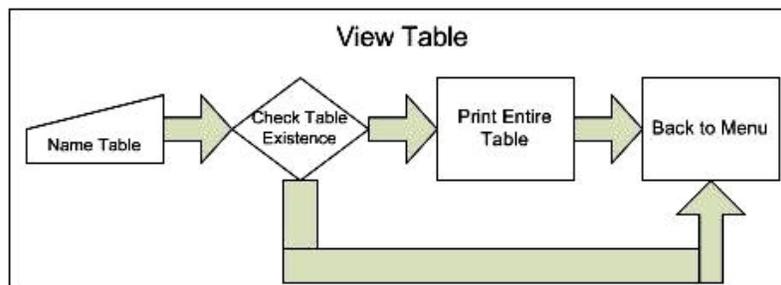


Figure 14: Function Design – View Table

When deleting a table, the user needs to give a final confirmation before deletion.

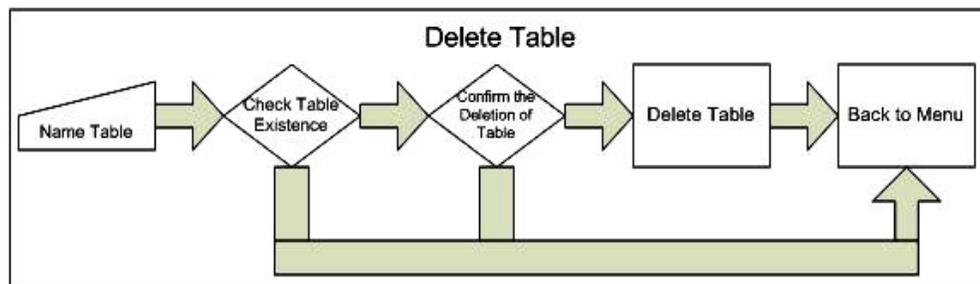


Figure 15: Function Design – Delete Table

When indexing a table, the system will check if the index already exists on it. If not, then the database will index the table by spatial extents using the Location column and by temporal extents using the numeric date column.

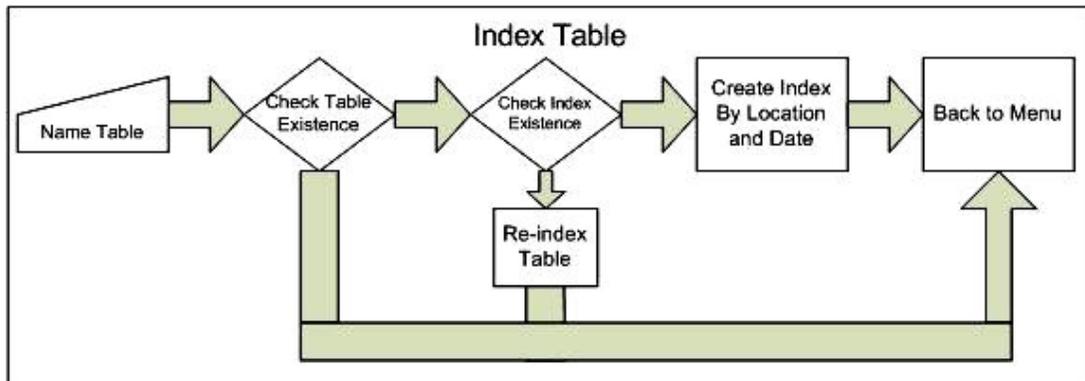


Figure 16: Function Design – Index Table

When inserting the meta data of an image in a table, the user needs to input the path of the image so the system will go to the meta data harvesting system to harvest the image and get the key elements of the meta data to insert in the table. After inserting the meta data, the user is given the choice to insert the meta data of another image in the same table.

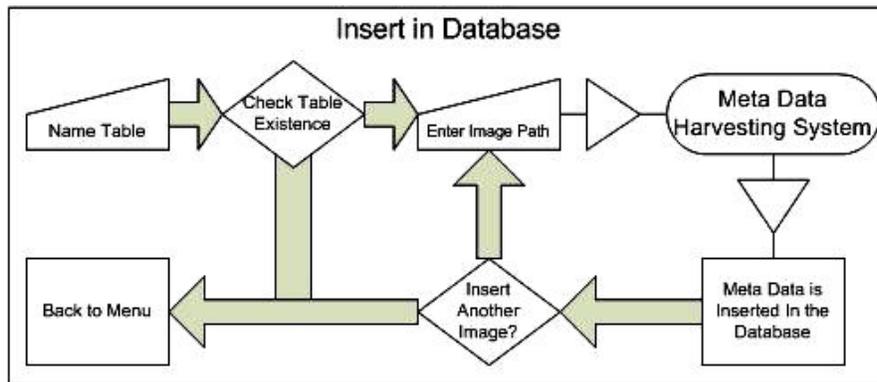


Figure 17: Function Design – Insert in Database

When searching for an image, the user needs to decide which search method he will choose:

- Search by date
- Search by location
- Search by date and location

If searching by date, the user needs to place the year and the month in which to search. If searching by location, the user needs to decide to search by the DMS degree or the decimal format. The decimal format allows placing latitude from -90° to 90° and longitude from -180° to 180° . The DMS format allows placing latitude degrees from 0° to 90° , longitude degrees from 0 to 180° , minutes from 0° to 59° , and seconds from 0° to less than 60° . If searching by date and

location, the user will have the same options of searching by date and location combined. After the results are printed, the user can search within the results for a more precise result.

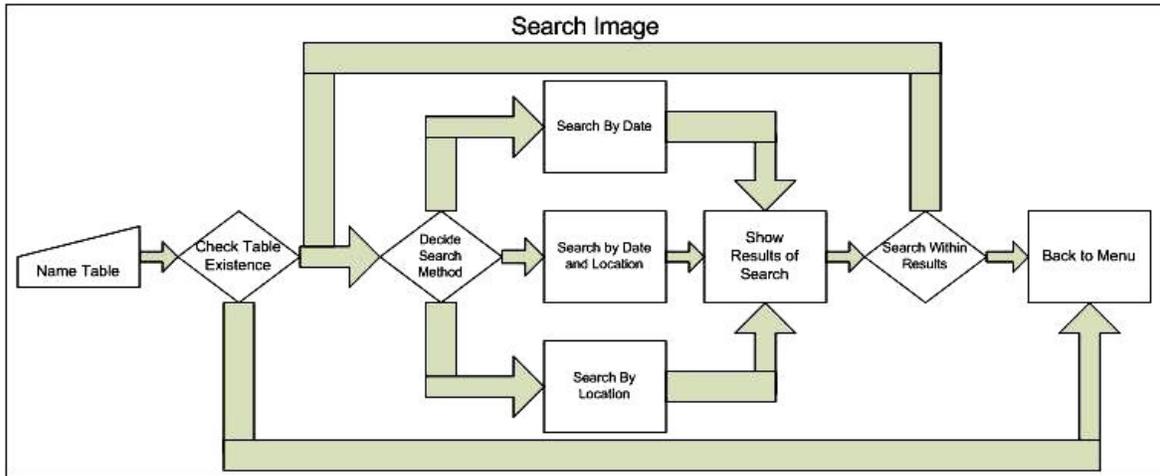


Figure 18: Function Design – Search Image

7. Web Page

A web page is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser and displayed on a computer screen. This information is usually in HTML or XHTML format, and may provide navigation to other web pages via hypertext links. Web pages may be retrieved from a local computer or from a remote web server. The web server may restrict access only to a private network or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using Hypertext Transfer Protocol (HTTP). Web pages may consist of files of static text stored within the web server's file system (static web pages), or the web server may construct the (X)HTML for each web page when it is requested by a browser (dynamic web pages). In this system, the web page is stored in Apache web server that at a later time can be access from anywhere in the world by a web browser, like Internet Explorer and Mozilla Firefox. This web server is connected to an application server that holds the PHP code to execute queries and connect to the database server. The reason why it was chosen to use a web application, instead of a window application, is the difficulty of accessing the system. If a windows application was chosen, then every single person that wished to access the system would need to get the application in their computer systems; while using a web application anyone can access the system from anywhere in the world. Using an application server with PHP code gives an easier connection to the PostgreSQL database server, when trying to connect to the database and execute search queries. This provides an advantage in the design and the development of the web service. Below are the designs on how the web application works.

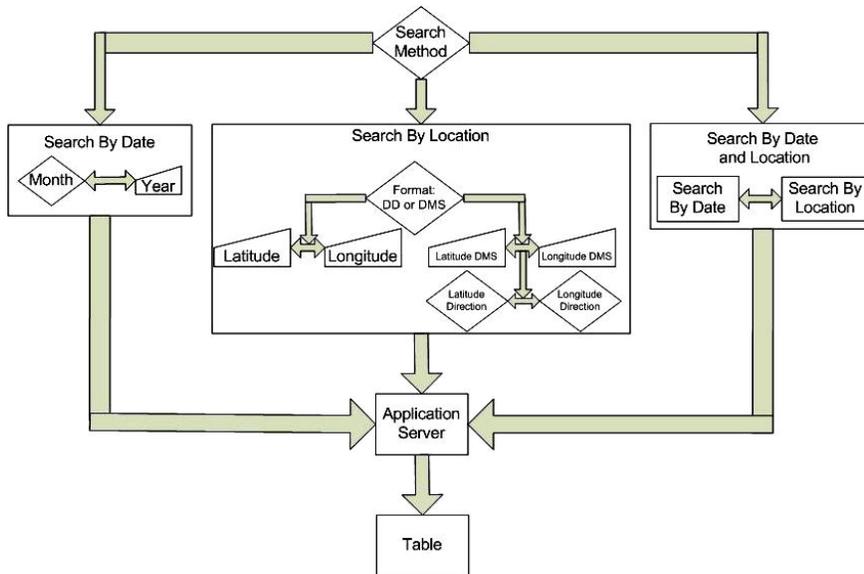


Figure 19: Web Page Design

As shown in the design, there are several ways of searching in the database. The user can search by date, location or both options. When searching by date, the user can search by the exact date or a range of date by choosing the month and the year. When searching by location, the user can input the latitude and longitude coordinates in decimal degrees or in degrees, minutes, and seconds (DMS) format. If the user chooses to input the location in DMS format, the following function is use to change the coordinate to decimal degree format:

$$\text{coordinate} = \text{degree} + \text{minute}/60 + \text{seconds}/3600$$

As an example, here is the prototype of how the web page looks like:

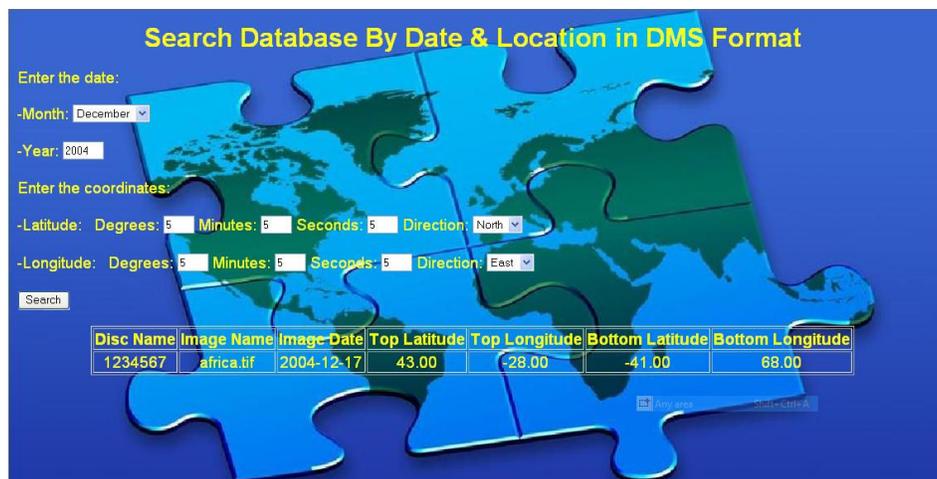


Figure 20: Web Page Demo – Searching the database by date and location in DMS format

In this example, the user chose the month as December and the year 2004 while placing the latitude and longitude as 5°5’5”. When the user places the button “Search” the following steps occur:

1. Information is sent to the application server
2. The server checks for validation of the coordinates
3. The server converts the coordinates from DMS to decimal degrees format
4. A connection is made to the database with a default login
5. The query is constructed
6. The query is executed
7. The results are gathered to create the table
8. The web page is resent with the output of the query

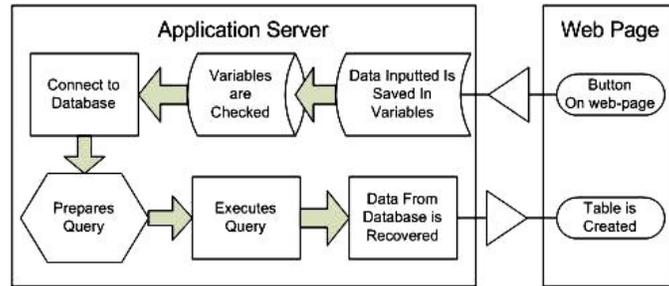


Figure 21: Application Server and Web Page Connection

As shown in the figure of the web page demo, a table was resulted showing the disc name of where the image came from, the name of the image, the last date the image was modified, and the top left and bottom right coordinates of the image.

8. Querying Database

As mention before explaining the SDBMA, the only two columns used for managing the database by indexing or searching for the meta data of an image are location and numeric date. When using the SDBMA or the web page, the queries for searching the meta data of an image in the database are the same. There are three types of query for searching the database:

- Spatial query
- Temporal query
- Spatiotemporal query

A spatial query retrieves all objects whose geometry contain a given point or region. It uses the latitude and longitude coordinates that are given to create a point attribute and searches through the database for all the objects that have a region attribute that contains that point.

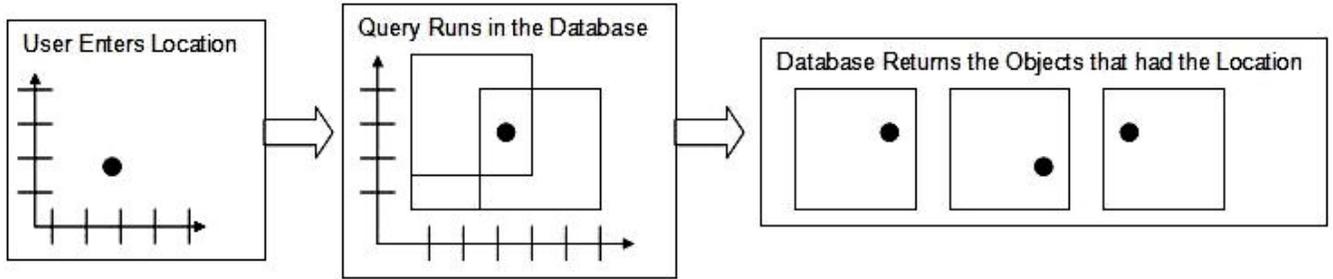


Figure 22: Spatial query - `SELECT * FROM <table_name> WHERE ST_WITHIN('POINT(3 2)', LOCATION);`

Temporal query retrieves all objects whose time has to do with an event. The system uses an exact date or a range of dates and searches through the database for the object that contain the dates that are being searched for.

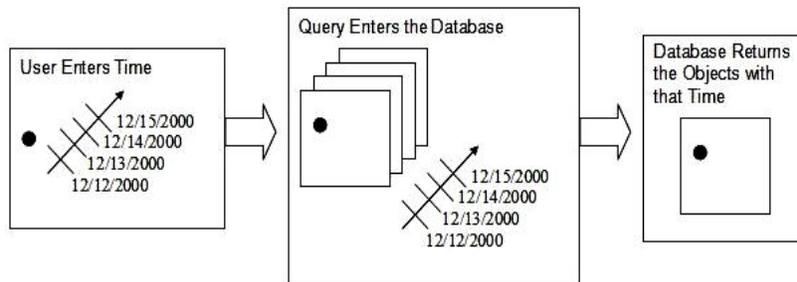


Figure 23: Temporal query - `SELECT * FROM <table_name> WHERE DATE = "12/12/2000";`

Spatiotemporal query retrieves all objects whose time has to do with an event and whose geometry contains a given point.

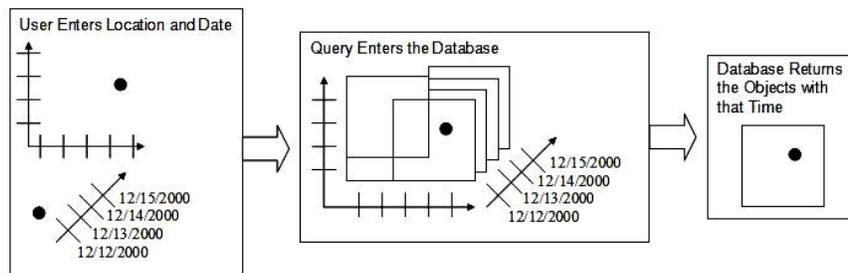
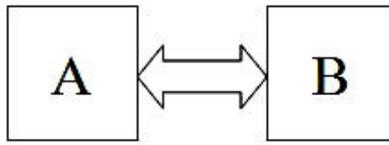


Figure 24: Spatiotemporal query - `SELECT * FROM <table_name> WHERE ST_WITHIN('POINT(4 2)', LOCATION) AND DATE = "12/12/2000";`

The result of these types of search gives the user a table with the meta data of the images that satisfy the parameter of the query. For the moment only the SDBMA can search within the results that the query printed for a more precise search. This feature is still being developed for the web page. Another feature that is development for the entire system is to add new types of way for searching the meta data of images by spatial extents. For example:

- Spatial Query that returns regions objects whose distance of separation is within the parameter <distance>.



distance

Figure 25: Distance Spatial Query - `SELECT * FROM <table_name> WHERE ST_DWITHIN('REGION(geometric_information)', LOCATION, distance);`

- Spatial Query that returns regions objects who overlap each other.

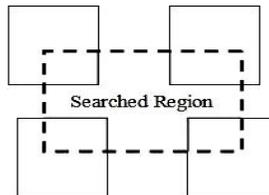


Figure 26: Overlapping Spatial Query - `SELECT * FROM <table_name> WHERE ST_OVERLAPS('REGION (geometric_information)', LOCATION);`

- Spatial Query that returns regions objects that are within other region objects.

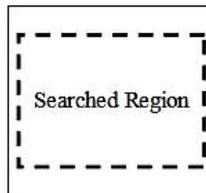


Figure 27: Region Within Object Spatial Query - `SELECT * FROM <table_name> WHERE ST_WITHIN('REGION (geometric_information)', LOCATION);`

9. Conclusion and Future Direction

After doing my research and developing the prototype for this system, I learn a lot about other kinds of databases and indexes and the behavior of those indexes in some manners while indexing the database. I also learn new ways of coding for a web application and a database application. Below is the prototype of the spatiotemporal database management application.

```

Main Menu:
<0>Connect to the Database
<1>Create a New Table
<2>View Table
<3>Delete Table
<4>Insert an Image on the Table
<5>Make an Index for the Table
<6>Search For an Image
<7>Disconnect from Database
<8>Exit Program

What do you wish to do? _

Menu
Enter Your Login Information:
Host: localhost
Port: 5432
Database: postgres
Username: postgres
Password: *****

(0) Connect to the Database
The spatial index has been created.
The temporal index has been created.

5) Make an Index for the Table
Name of the table you wish to create: demo
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "demo_pkey" for t
able "demo"
The table named demo was created.

(1) Create a new table
Name of the table you wish to delete: demo
Are you sure you wish to delete this table and all its attribute? <Y/N> Y
The table known as demo was deleted.

(3) Delete table
14 Managed by UT-Battelle

Name of the table you wish to insert the information of an image: demo
Name of the Disc the Image Resides: 1234567
Image exist.
PROJECTION INFORMATION
-Driver: GTiff/GeoTIFF
-Image Size in Pixels:
-Width: 512
-Height: 512
-Raster Bands in dataset: 1
-UKT Image Projection:
PROJCS["unnamed",GEOGCS["NAD27",DATUM["North American Datum 1927",SPHEROID["Clar
ke 1866",6378206.4,294.9786982139006,AUTHORITY["EPSG","7008"]],AUTHORITY["EPSG",
"6267"]],PRIMEM["Greenwich",0],UNIT["degree",0.0174532925199433],AUTHORITY["EPSG",
"4267"]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],P
ARAMETER["central_meridian",-117],PARAMETER["scale_factor",0.9996],PARAMETER["fa
lse_easting",500000],PARAMETER["false_northing",0],UNIT["unknown",1],AUTHORITY["
EPSG","26711"]]]
-Pretty UKT Projection Format:
PROJCS["unnamed",
GEOGCS["NAD27",
DATUM["North American Datum 1927",
SPHEROID["Clarke 1866",6378206.4,294.9786982139006,
AUTHORITY["EPSG","7008"]],
AUTHORITY["EPSG","6267"]],
PRIMEM["Greenwich",0],
UNIT["degree",0.0174532925199433],
AUTHORITY["EPSG","4267"]],
PROJECTION["Transverse_Mercator"],
PARAMETER["latitude_of_origin",0],
PARAMETER["central_meridian",-117],
PARAMETER["scale_factor",0.9996],
PARAMETER["false_easting",500000],
PARAMETER["false_northing",0],
UNIT["unknown",1],
AUTHORITY["EPSG","26711"]]]
-Origin: (440720,3.75132e+006)
-Pixel Size: (60,60)
IMAGE SIZE:
-Height: 512 pixels
-Width: 512 pixels
-Pixel Resolution On the X-Axis: 60
-Pixel Resolution On the Y-Axis: -60
COORDINATES:
-Central Meridian: -117
-Latitude of Origin: 0
-Standard Parallel 1: 0
-Standard Parallel 2: 0
COORDINATES IN METERS:
-Top X: 440720
-Top Y: 3.75132e+006
-Bottom X: 471440
-Bottom Y: 3.7206e+006
COORDINATES IN DECIMAL DEGREES:
-Top X: -118
-Top Y: 34
-Bottom X: -112
-Bottom Y: -1
IMAGE FILE PROPERTIES:
-Image Location: utm.tif
-Date Image was Taken: 11/06/2002
The image utm.tif was inserted in table demo.
Do you wish to insert the information of another image? <Y/N> N

(4) Insert meta data of an image on a table

```

Figure 28: Spatiotemporal database management application prototype

As can be seen, this prototype has a menu with the different options to manage the database by connecting to one, creating, deleting, indexing, viewing, and inserting the meta data on tables of the database. This system enables an easier data management because thanks to the use of the indexes and the search queries, the user can get much faster and more efficient results. At this time, the time that it takes to get the result from the search queries is not possible to determine because the database is in the process of creation. We, as developers, would like that the results of the search through the database would last only a few seconds. In the future, once the database has been created and the system is working, would like to extend the system to the point where it will be able to use other kind of search criteria like cloud coverage percentage, forestation percentage, population growth, type of resolution, etc. Another extension would also be to interface with other data format and a way to search for meta data through the network so not only one point has all the data together.

10. Reference

- [1] Hartmut Güting, R., and Schneider, M. (2005). *Moving Objects Databases*. San Francisco: Morgan Kaufmann Publishers.
- [2] Rigaux, P., Scholl, M., and Voisard, A. (2002). *Spatial Databases With Application To GIS*. San Francisco: Morgan Kaufmann Publishers.
- [3] PostgreSQL 8.3.3 Documentation, The PostgreSQL Global Development Group.
- [4] Neufeld, K. (2009). PostGIS 1.4.0rc2 Manual.

- [5] Geospatial Data Abstraction Layer (GDAL), GDAL/OGR Project Management Committee, <http://www.gdal.org/>.
- [6] http://en.wikipedia.org/wiki/Web_page
- [7] http://en.wikipedia.org/wiki/Moderate-Resolution_Imaging_Spectroradiometer
- [8] <http://applications.nrsc.gov.in:15001/FAQ.asp?selvar=3>
- [9] http://en.wikipedia.org/wiki/Database_management_system
- [10] <http://en.wikipedia.org/wiki/Metadata>