Universidad Nacional Autónoma de México

Facultad de Ciencias

Análisis y Visualización de Información Mediante estadística espacial: GEOSTADS

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

CARLOS SÁNCHEZ PERALES

DIRECTOR DE TESIS:
DR. MIGUEL MURGUÍA ROMERO

13 de enero de 2012

Hoja de datos del jurado

1. Datos del alumno

Apellido paterno	Sánchez
Apellido materno	Perales
Nombre(s)	Carlos
Teléfono	58 51 05 69
Universidad Nacional Autónoma de México	Universidad Nacional Autónoma de México
Facultad de Ciencias	Facultad de Ciencias
Carrera	Ciencias de la Computación
Número de cuenta	303251874

2. Datos del tutor

Grado	Dr.
Nombre(s)	Miguel
Apellido paterno	Murguía
Apellido Materno	Romero

3. Datos del sinodal 1

Grado	Dra.
Nombre(s)	Amparo
Apellido paterno	López
Apellido Materno	Gaona

4. Datos del sinodal 2

Grado	Dr.
Nombre(s)	José de Jesús
Apellido paterno	Galaviz
Apellido Materno	Casas

5. Datos del sinodal 3

Grado	Dra.
Nombre(s)	Hanna Jadwiga
Apellido paterno	Oktaba
Apellido Materno	

6. Datos del sinodal 4

Grado	Act.
Nombre(s)	Carlos Ernesto
Apellido paterno	López
Apellido Materno	Natarén

7. Datos del trabajo escrito

Titulo	Análisis y Visualización de Información Mediante estadística espacial: GEOSTADS.
Número de páginas	153 p
Año	2012

Índice general

Parte I

Introducción

0.1. Antecedentes

El análisis espacial

El análisis espacial es la unión de diversas técnicas análiticas y modelos en los que existe una relación entre los datos cuantitativos y el espacio de donde son tomados [78].

La relación que existe entre el análisis espacial y el uso de un SIG es que el primero proporciona las funciones necesarioas para llevar a cabo el análisis de la información y los SIG los medios necesarios para intrepetar los resultados del análisis, las funciones que en conjunto proporcionan son [79]:

- Análisis y manipulación de los atributos no espaciales almacenados en la base de datos.
- Análisis y manipulación de los atributos espaciales almacenados en la base de datos. - Integración de datos espaciales y no espaciales.
- Algunas funciones de manipulación y análisis espacial que proporcionan los SIG requieren de mayores recursos computacionales y tecnologicos que otros sistemas, esto debido a que realizan operaciones espaciales complejas y representan los resultados en la mayoria de las veces de forma gráfica.

El análisis espacial es el conjunto de habilidades para manipular datos con atributos espaciales o geográficos y a partir de estos, extraer un significado diferente [80] que depende del arreglo espacial de los datos [81]. La característica relevante del análisis espacial es que este requiere no solo de los valores de los atributos, sino además de la localización geográfica de los objetos [53].

En los últimos años se ha ido incorporando a los SIG una herramienta estadística capaz de llevar acabo el análisis de eventos espaciales, la estadística espacial [53]. El por que utilizar una herramienta como la estadística espacial es que está considera la dependecia explicita entre los fenómenos y su ubicación en el espacio .

El presente trabajo toma como punto de partida el trabajo de tesis (CITA TESIS) el cual mediante el uso de técnicas de estadística espacial, logra realizar el análisis de información espacial y obtener conocimiento y conclusiones que no se encontraba en los datos en si de forma explicita.

Requerimientos de análisis en algunas áreas

Demografía

En demografía, las unidades mínimas censales en que se reportan de manera pública los datos, determinan el tipo o alcance del análisis espacial [73]. Los *Census Track*, en inglés, denominadas como "Áreas Geostadísticas Básicas, o AGEB, por el INEGI, son las unidades mínimas en que se tiene acceso a los datos. Sin embargo, el análisis de la información de censos demográficos se ha contextualizado dentro del MAUP, demostrándose que es indispensable un entendimiento profundo de la subdivisión del territorio en las unidades geográficas de análisis, pues de ello depende que los resultados de las herramientas estadísticas muestren un comportamiento positivo o negativo, por ejemplo, en el caso de la autocorrelación espacial. Es por esa razón, que la estadística espacial auxilia en el entendimiento de los procesos espaciales, haciéndose relevante el contar con herramientas de software ágiles, que no requieran de largas curvas de aprendizaje, y que estén disponibles en internet de forma inmediata y permitan el uso empírico y dinámico de técnicas de análisis espacial.

Biodiversidad

El análisis de la distribución espacial de biodiversidad presenta varios retos. Actualmente existen varias técnicas ó herramientas que apoyan a la biología de la conservación, por ejemplo las técnicas y métodos de la biogeografía cuantitativa. Un ejemplo de esto es el software *Biodiverse* [75], que integra el análisis de registros de recolecta de especies mediante técnicas de análisis multivariado. Sin embargo, dicho tipo de herramientas y software aún no contemplan el uso de técnicas de la estadística espacial, y solo están disponible en software poco accesible de manera directa e inmediata a usuarios finales, cono son el lenguaje R, o los sistemas de información geográfica como ArcView, Quatum GIS, DVIA GIS, por nombrar algunos. Dichas herramientas de análisis espacial, requieren de una preparación especial de los datos y una larga curva de aprendizaje.

0.2. Objetivos

El objetivo de este trabajo es desarrollar una herramienta de software centrada en el análisis de datos espaciales mediante técnicas de la estadística espacial. Como se mecionó anteriormente existen muchas herramientas para llevar acabo el análisis de datos, sin embargo muy pocas o ninguna se ha centrado en la utilización de esta rama de al estadística que brinda las

herramientas para estudiar los objetos espaciales utilizando sus propiedaes topológicas, geométricas o geográficas. La herramienta debe ser lo suficientemente simple para evitar que el usuario requiera una preparación especial y extensa.

Parte II

Marco teórico

Capítulo 1

Cartografía

La cartografía es la ciencia que estudia la representación gráfica de la Tierra o parte de ella en una superficie plana [1]. El término cartografía se ha extendido a la realización de cualquier modelo bi o tridimensional de la Tierra.

Para poder representar la forma de geoide[1] de la Tierra en un plano es necesario valerse de un sistema de proyecciones, esto debido a que la forma de la Tierra es más achatada en los polos que en la zona ecuatorial.

La cartografía además de representar los contornos, las superficies y los ángulos de la Tierra, se ocupa también de representar la información en mapas.

1.1. Sistemas de proyecciones

Con el propósito de transformar una superficie esférica en una plana es necesario establecer una serie de puntos sobre la superficie de la esfera y transferir éstos al plano de acuerdo con el sistema de transformacion que se use [2]. Estos puntos se localizan auxiliándose en una red de meridianos y paralelos en forma de malla. Hay tres tipos básicos de proyección: cilíndrica, cónica y polar.

[1] Se denomina geoide al cuerpo de forma casi esférica. La palabra geoide tiene origen del griego : $\gamma\varepsilon\iota\alpha$ — tierra — $\varepsilon\iota\delta\sigma\varsigma$ — forma, apariencia — "forma que tiene la Tierra"

1.1.1. Proyección cilíndrica

Esta se obtiene proyectando el globo terrestre sobre una superficie cilíndrica. Es una de las más utilizadas, aunque por lo general en forma modificada, debido a las grandes distorsiones en las zonas de latitud elevada, lo que impide apreciar a las regiones polares en su verdadera proporción. Es utilizada en la creación de mapamundis[6]. A continuación se explican las proyecciones más utilizadas de esta categoría.

1.1.1.1. Proyección de Mercator

Fue ideada por Gerardus Mercator en 1569 para elaborar planos terrestres y es de tipo cilíndrica. El propósito de esta proyección es representar la superficie de la Tierra sobre una superficie cilíndrica tangente al ecuador, el problema de esta técnica radica en que para zonas alejadas al ecuador (los polos, por ejemplo) se presentan grandes deformaciones[5] (Figura 1.1), por ejemplo:

- Groenlandia aparece aproximadamente del tamaño de África, cuando en realidad el área de África es aproximadamente 14 veces el de Groenlandia.
- Alaska aparece similar en tamaño a Brasil, cuando el área de Brasil es casi cinco veces el de Alaska.

Figura 1.1: Mapa en proyección de Mercator. Tomada de http://www.progonos.com/furuti/MapProj/Normal/TOC/cartTOC.html

1.1.1.2. Proyección transversa de Mercator

Difiere de la proyección de Mercator en que se proyecta la superficie de la Tierra sobre una superficie cilíndrica tangente a cualquier meridiano logrando así, una menor deformación en las regiones de la Tierra (Figura 1.2). En 1947 Estados Unidos de Norte América adopotó esta proyección recibiendo del nombre de Universal Transverse Mercator (UTM)[7].

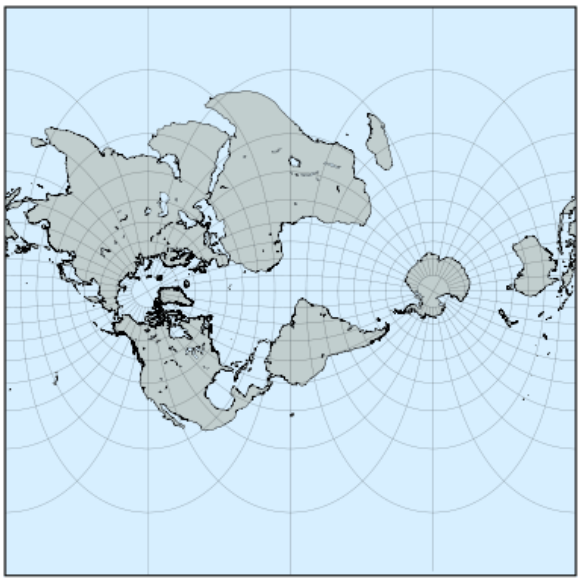


Figura 1.2: Mapa en proyección transversa de Mercator. Tomada de http://www.progonos.com/furuti/MapProj/Normal/TOC/cartTOC.html

1.1.2. Proyección cónica

La proyección cónica se obtiene proyectando los elementos de la superficie esférica terrestre sobre una superficie cónica tangente, situando el vértice en el eje que une los dos polos. Este tipo de proyección es comunmente utilizada en escalas para representar los países y continentes.

1.1.2.1. Proyección de Alber

Es un tipo de proyección cónica que posee la característica de perservar el área que se está proyectando, para realizar esta técnica se utilizan dos paralelos y con ellos se forma un cono (Figura 1.3). Aunque la escala y forma no se conservan, la distorsión es mínima entre los paralelos. La proyección Albers es la proyección estándar de la Columbia Británica[8]. También es utilizado por la Encuesta Geológica y la Oficina del Censo de los Estados Unidos de Norte América [9].

Figura 1.3: Mapa en proyección de Alber. Tomada de http://www.progonos.com/furuti/MapProj/Normal/TOC/cartTOC.html

1.1.3. Proyección azimutal, cenital o polar

En este caso se proyecta una porción de la Tierra sobre un plano tangente al globo en un punto seleccionado, obteniéndose una imagen similar a la visión de la Tierra desde un punto interior o exterior. Las proyecciones azimutales tienen la propiedad de que las direcciones desde un punto central se conservan (y por lo tanto, los grandes círculos a través del punto central están representados por líneas rectas en el mapa). Por lo general, estas proyecciones también tienen simetría radial en las escalas y por lo tanto, en las distorsiones (Figura 1.4).

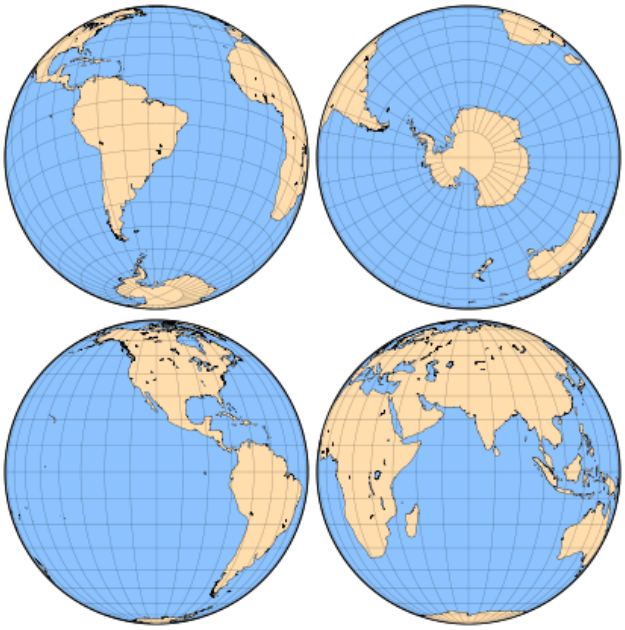


Figura 1.4: Mapa en proyeccion azimutal. Tomada de http://www.progonos.com/furuti/MapProj/Normal/TOC/cartTOC.html

1.2. Sistemas de coordenas

Es un conjunto de valores que permiten definir unívocamente la posición de cualquier punto de un espacio geométrico con respecto de otro punto denominado origen, es un sistema de referencia. Se cuenta con un gran número de sistemas de coordenadas, sin embargo, los más utilizados para cartografía y sistemas de información geográfica son las coordenadas cartesianas, geográficas y Sistema de Coordenadas Universales de Mercator[2].

1.3. Coordenadas cartesianas

Un sistema de coordenadas cartesianas especifica cada punto como único en un plano por un par de coordenadas numéricas en el espacio de dos o tres dimensiones. Cada línea de referencia se llama eje coordenado o simplemente

[2] *Universal Transverse Mercator* (UTM, por sus iniciales en inglés)

eje del sistema, y el punto donde se interesecta es el origen. De esta manera se puede identificar de manera única un punto en el espacio, pues este está dado por dos o tres coordenadas. Este sistema a pesar de ser el más sencillo de los tres mencionados cuenta con un inconveniente; no se ajusta a la forma geoide de la Tierra, si no para superficies planas [10] (Figura 1.5).

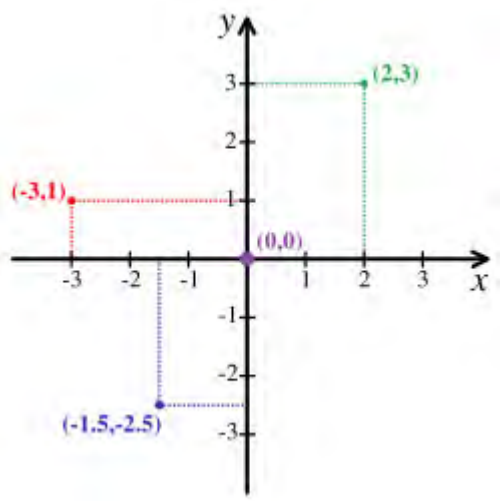


Figura 1.5: Sistema de coordenadas cartesianas en dos dimensiones.

1.4. Coordenadas geográficas

Es un sistema de referencia utilizado para definir puntos sobre una superficie esférica. La ubicación de un punto se hace mediante el uso de latitudes y longitudes las cuales permiten ubicar un punto mediante ángulos en sistema sexagecimal, es decir, la localización de cada punto se mide en grados, minutos y segundos. Por ejemplo, el Distrito Federal tiene coordenadas extremas al norte 19°36', al sur 19°03' de latitud norte; al este 98°57', al oeste 99°22' de longitud oeste.

Las líneas de longitud y latitud se cruzan unas con otras formando ángulos rectos, igual que en las coordenas cartesianas, la diferencia radica en que la longitud y latitud existen sobre una superficie curva. Se toma un número infinito de estas líneas sobre el modelo de la Tierra, por tanto, esto implica que a cada punto sobre la Tierra le corresponde una única linea de latitud y longitud que pasa sobre él [3] (Figura 1.6).

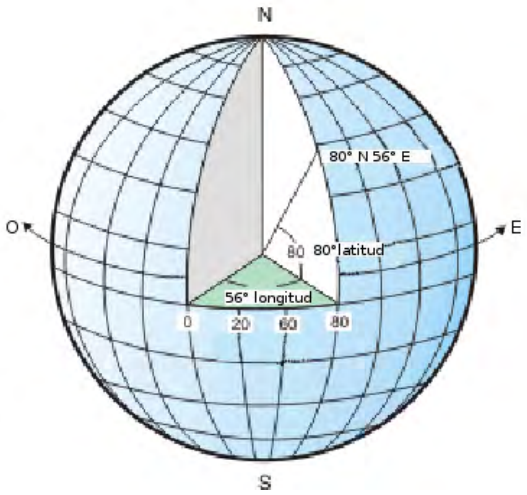


Figura 1.6: Sistema de coordenas geográficas, tomada de [11].

1.4.1. Longitud

Expresa una distancia en ángulos entre un punto dado en la superfice terrestre y el meridiano 0 (el meridiano base es el de Greeenwich). La longitud se mide en grados (°), minutos (') y segundos (") en sistema sexagesimal. La forma común de medirla es tomando de entre 0 y 180 grados indicando a qué hemisferio pertenece (Occidental o W y Oriental o E) o agregando signo negativo para los puntos que se localizan en el hemisferio Oriental. Los meridianos convergen conforme se acercan a los polos, pues la Tierra es achatada en esas regiones.

1.4.2. Latitud

Son un conjunto de líneas que se trazan de occidente a oriente (horizontalmente), llamadas paralelos, ya que son todas paralelas entre sí y no convergen como lo hacen los meridianos. La línea principal que se toma como punto de partida es el paralelo 0 o la línea que pasa sobre el Ecuador. La latitud se mide en grados sexagesimales de 0° a 90° indicando el hemisferio al que pertenece la coordenada o bien agregando un signo menos a aquellos puntos que estén por debajo del paralelo 0.

1.5. Sistema de coordenas UTM

El Sistema de Coordenadas Universal Transversal de Mercator es un sistema basado en la proyección cartográfica transversa de Mercator, las magnitudes en este sistema están expresadas en metros. Es un sistema coordenado plano conveniente para un Sistemas de Información Geográfica (SIG)[3] que trabaja con grandes áreas [4]. UTM divide la Tierra en 60 zonas verticales de 6° de ancho que comienzan en la longitud 180°, cada franja se extiende de norte a sur desde los 84° de latitud hasta 8° latitud (84° N a 80° S) y 20 franjas horizontales de 8° de latitud denominadas con las letras C a X excluyendo las letras I, O y Ñ formando así zonas rectangulares [5] (Figura 1.7).

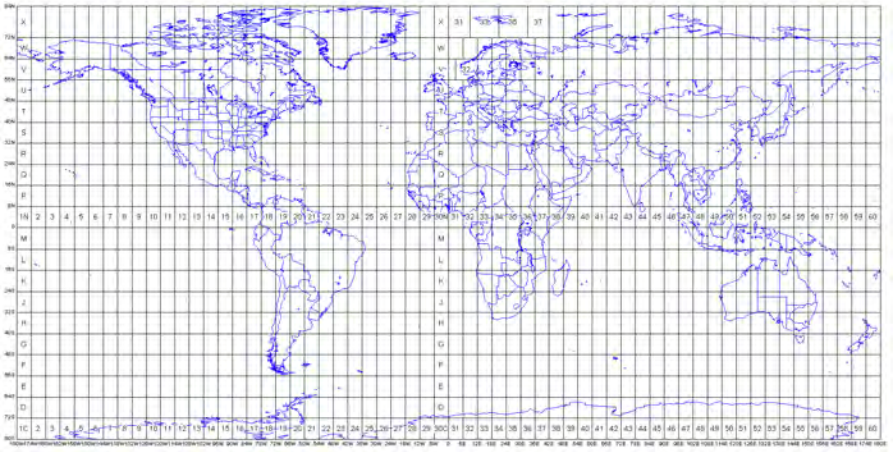


Figura 1.7: Sistema de coordenas UTM, tomada de http://www.dmap.co.uk/utmworld.htm

1.6. Mapa

Un mapa es una representación gráfica y métrica de una porción de territorio, generalmente los mapas se trazan sobre superficies planas aunque los avances tecnológicos computacionales actuales permiten realizar mapas en tres dimensiones. Un mapa tiene propiedades métricas, es decir, es posible tomar medidas de distancias, ángulos o superficies sobre él y obtener un resultado casi exacto.

[3] *Geographic information systems* (GIS, por sus iniciales en inglés)

La utilidad de un mapa va más allá de la simple idea de represntar el territorio o mostrar datos geográficos, en la actualidad con ayuda de los SIG los mapas son una herramienta indispensable para la realización de análisis espacial.

1.6.1. Mapa georreferenciado

La georreferenciación se refiere al posicionamiento de un objeto en el espacio físico mediante un sistema de coordenadas determinado. Así, mediante este proceso, es posible ubicar objetos en la superficie de la Tierra asignándoles un punto en un mapa, al resultado de éste procesos se le llama mapa georeferrenciado. Este proceso nos permite analizar información directamente en su ubicación en el espacio, esta información puede estar contenida en los datos o imágenes que se han producido en un punto en diferente momento de tiempo. Este método permite combinar o comparar estos datos con los actualmente disponibles [12].

Capítulo 2

Bases de Datos

A lo largo de la historia, las civilizaciones han acumulado enormes cantidades de conocimiento y de la misma manera, se han preocupado por hacer que este conocimiento perdure al paso del tiempo, es así que se cuenta con bibliotecas que almacenan colecciones enteras de libros, manuscritos que reunen conocimientos antiguos, tablillas, vasijas, información de todo tipo sobre casi cualquier material imaginable.

Sin embargo, este material continuá en crecimiento, día a día se genera nueva información, miles de datos son costruidos para uno u otro fin y la necesidad de almacenarlos para analizarlos no ha dejado de tener prioridad.

Con la llegada de los sistemas de cómputo se dió un gran paso al llevar toda esta información a un ambiente digital donde puede ser almacenada de forma más eficiente.

Así dependiendo del punto de vista del autor, una base de datos puede definirse como:

- "Depósito o contenedor de una colección de archivos computarizados" [13].
- "Un conjunto autodescriptivo de registros integrados" [14].
- "Una gran colección de datos interrelacionados almacenados en un entorno informático" [27].

Para el presente trabajo, se tomará como definición de una base de datos a la tercera de las anteriores debido a la naturaleza de los datos con los que

se trabajará; y en el presente capítulo se abordarán los fundamentos de una base de datos en un entorno informático, así como los principales modelos que en la actualidad son más utilizados.

2.1. Propiedades

Las propiedades que deben cumplir las operaciones que se realizan en una base de datos son [13]:

- Atomicidad: cada operación que se realice debe ser atómica, éstas deben ser tratadas como una unidad, en caso de error no se realiza la operación y se debe abortar ésta.
- Aislamiento: los datos que son utilizados por alguna operación, no pueden ser utilizados por otra en cuanto la primera no termine.
- Independencia lógica y física: significa que se puede alterar la estructura lógica de la base de datos, sin alterar los sistemas que acceden a la base de datos, además la distribución física de los datos es independiente de los sistemas que los manipulan.
- Consultas complejas optimizadas: la base de datos se puede consultar mediante un lenguaje de consulta estructurado.
- Integridad de los datos: no se pueden ingresar datos de tipo distinto a los establecidos en el dominio de los mismos, por ejemplo, no se puede introducir un valor **boolean** en un campo de tipo numérico.
- Acceso concurrente por parte de múltiples usuarios: permite acceder a diferentes usuarios al mismo tiempo desde diferentes lugares sin errores de consistencia.
- Acceso a través de lenguajes de programación estándar: en la actualidad se cuenta con bibliotecas en varios lenguajes de programación que permiten que los sistemas se comuniquen con las bases de datos con mayor facilidad.

2.2. Niveles de abstracción de los datos

Un sistema de información en general para que pueda ser considerado útil, debe poder recuperar los datos con los que trabaje de manera eficiente, y esto

depende en gran medida en la manera en que se almacena la información en una base de datos. Para ello, dentro de una base de datos se definen niveles de abstracción los cuales son capas donde se esconde la complejidad del almancenamiento para los usuarios [15]:

1. Nivel físico: El nivel más bajo de abstracción describe cómo se almacenan realmente los datos. En el nivel físico se describen en detalle las estructuras de datos complejas de bajo nivel.

2. Nivel lógico: Es un nivel intermedio, entre el nivel interno y el nivel externo, en éste se describe qué tipo de datos se almancenan en la base de datos y las relaciones que existen entre estos.

3. Nivel de vistas: El nivel más alto de abstracción, en este nivel se muestran los datos a cada usuario, dependiendo de los privilegios que éstos tengan y de la información que estos necesiten pues muchas veces los usuarios no requieren ver todos los datos almacenados si no que solamente una parte de estos.

2.3. Sistema Manejador de Bases de Datos (SMBD)[1]

Un SMBD el software que sirve como interfaz entre el usuario, la base de datos y los sistemas que realizan alguna operación en la base. El SMBD permite agregar datos, eliminar tablas o registros, recuperar o almacenar datos y solicitar datos, entre otras operaciones.

La siguiente es una clasificación del tipo de operaciones que se puede realizar en un sistema manejador de bases de datos [14]:

- Descripción de los datos (lenguaje de definición de los datos[2]): es un lenguaje proporcionado por el SMBD que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.

- Manipulación de los datos (lenguaje de manipulación de datos[3]): es un lenguaje igualmente proporcionado por el SMBD, permite a los

[1] *Data Base Management System* (DBMS, por sus iniciales en inglés)
[2] *Data Definition Language* (DDL, por sus iniciales en inglés)
[3] *Data Manipulation Language* (DML, por sus iniciales en inglés)

usuarios manipular la información de la base de datos, este lenguaje está asociado a las operaciones de selección, inserción, actualización y eliminación de datos.

- Control de datos (lenguaje de control de datos[4]): este lenguaje permite realizar la gestión del acceso a los datos almacenados en la base de datos; está asociado a las operaciones de gestión de permisos de los usuarios.

2.4. Arquitectura de los sistemas de bases datos

En esta sección se explican las arquitecturas de un sistema de base de datos, es decir, como están ubicadas y los elementos externos que la conforman (hardware, por ejemplo) y la manera en que se tiene acceso a ella.

La arquitectura de un sistema de bases de datos depende principalmente de cómo se encuentren conectados los elementos que la conforman, el tipo de procesamiento y la distribución de los datos.

A continuación se mencionan las distintas arquitecturas que se han propuesto para un sistema de base de datos [15].

2.4.1. Arquitectura centralizada

Los sistemas de bases de datos centralizados son aquellos que se ejecutan en un único sistema informático sin interactuar con ninguna otra computadora. En esta arquitectura se depende totalmente del poder de procesamiento y de la memoria del equipo [16]. En la Figura 2.1 se muestra un ejemplo de ésta arquitectura.

[4] *Data Control Language* (DCL, por sus iniciales en inglés)

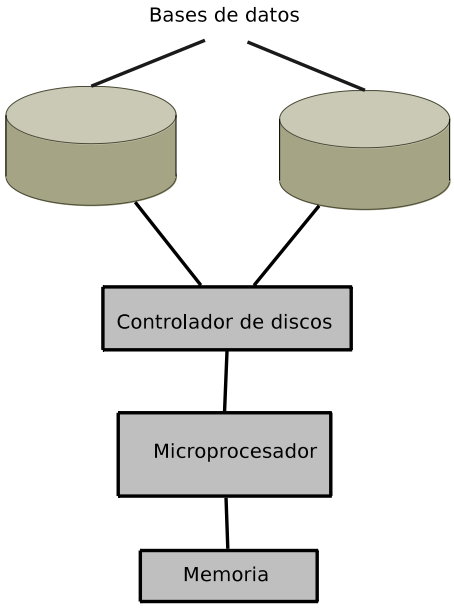


Figura 2.1: Arquitectura centralizada de base de datos. Con base en [15] página 446.

2.4.2. Arquitectura cliente-servidor

En este tipo de arquitectura se tiene una máquina de tipo servidor, la cual brinda los recursos a un conjunto de más computadoras trabajando entre sí. En este enfoque el sistema centralizado solo se encarga de recibir peticiones sobre la base de datos, las procesa, las ejecuta y regresa el resultado, es decir, ya no se lleva a cabo el trabajo por parte de los usuarios en este equipo, éste se realiza en los demás equipos que tienen la función de clientes que solicitan algún servicio al equipo centralizado [17]. En la Figura 2.2 se muestra un ejemplo de esta arquitectura.

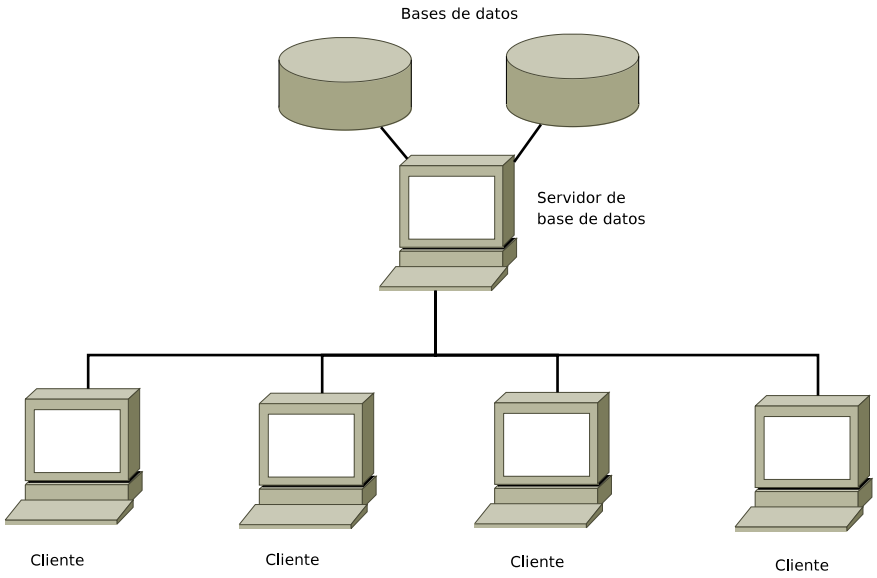


Figura 2.2: Arquitectura cliente-servidor de base de datos. Con base en [15] página 447.

2.4.3. Arquitectura parelela

Existen diferentes arquitecturas de bases de datos paralelas, estas arquitecturas difieren en el número de procesadores y en el número de memorias con las que se cuente [18]:

- Disco compartido: cuenta con un grupo de discos que son accesibles para todos los procesadores y cada procesador tiene su propia memoria a la cual no se permite que acceda otro procesador, la comunicación entre los procesadores es lenta. Tiene tolerancia a los fallos, ya que si algún procesador tiene un error, no se pierde la información, pues la base de datos se encuentra en los discos.

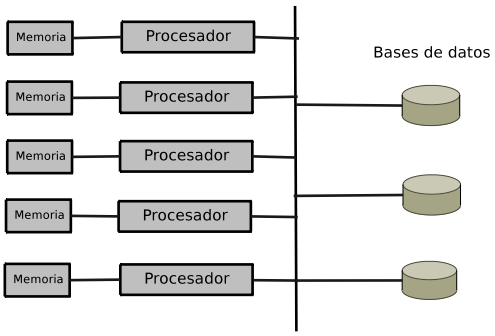


Figura 2.3: Arquitectura paralela mediante disco compartido. Con base en [15] página 453

- Memoria compartida: solo existe una memoria y todos los procesadores acceden a ésta, tiene gran eficiencia en cuanto a la comunicación entre procesadores, sin embargo a partir de cierto número de procesadores la comunicación entre éstos y la memoria se convierte en un cuello de botella.

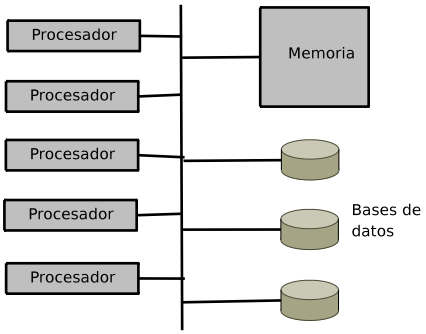


Figura 2.4: Arquitectura paralela mediante memoria compartida. Con base en [15] página 453.

- Sin compartir disco o memoria: cada procesador tiene su memoria y sus discos correspondientes, y cada procesador es visto como un nodo. Los procesadores tienen comunicación entre sí, cada nodo o procesador responde a las peticiones de datos y se transmiten por la red las peticiones a los accesos de discos remotos. En esta arquitectura el software que brinda la comunicación entre los nodos es de gran importancia.

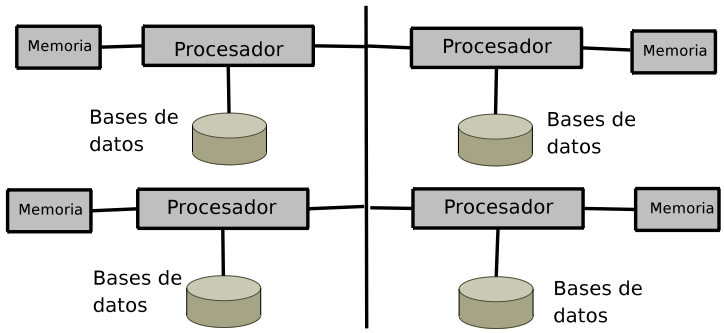


Figura 2.5: Arquitectura paralela sin compartir. Con base en [15] página 453

- Jerárquico: es una combinación de las arquitecturas anteriores, el sistema está formado por nodos o procesadores que pueden o no compartir memoria o discos, sin embargo cada nodo puede ser visto como un sistema de arquitectura de memoria compartida, de disco compartido o sin compartimiento.

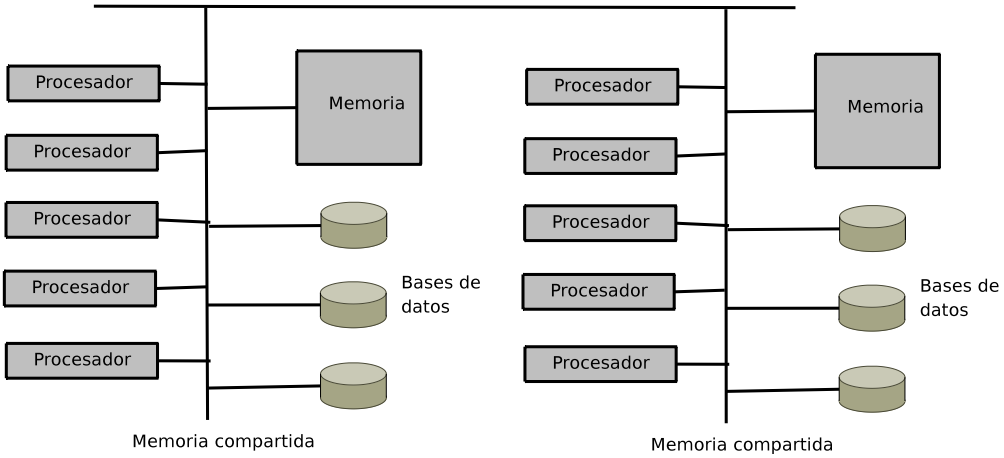


Figura 2.6: Arquitectura paralela jerárquica. Ccon base en [15] página 453

2.4.4. Arquitectura distribuida

En esta arquitectura la base de datos puede ser almacenada en varias computadoras o en una sola. El acceso a la base de datos se realiza mediante el uso

de redes, éstas no comparten memoria ni discos. Las computadoras pueden estar localizadas en diferentes lugares geográficos y administrarse de manera distinta, pero la comunicación entre ellas es más lenta que en las otras arquitecturas. La principal ventaja de la arquitectura distribuida es que los usuarios acceden a los datos de una computadora remota desde la computadora más cercana, sin tener que trasladarse a la ubicación de la base de datos. Si una computadora falla las demás computadoras pueden seguir funcionando sin problemas y en el caso de que los datos de la computadora que tuvo el error estén replicados en las demás computadoras, las operaciones que requieran esos datos pueden ser realizadas [19]. En la Figura 2.7 se muestra un ejemplo de esta arquitectura.

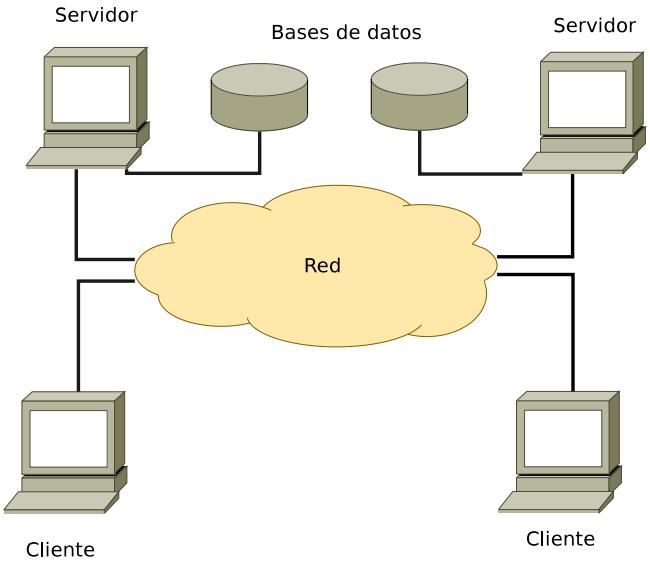


Figura 2.7: Arquitectura distribuida. Con base en [15] página 456

2.5. Modelos de bases de datos

Un modelo de datos es una definición lógica, abstracta e independiente de los objetos, en otras palabras, es una descripción de la base de datos. Las bases de datos se pueden clasificar de acuerdo a su modelo de administración de datos [15]:

- Modelo transaccional: En este modelo solo se permite realizar operaciones como envío y recepción de datos, son muy rápidas pero son utilizadas en su mayoría por grandes empresas, en muchas ocasiones tiene una comunicación con una base de datos relacional.

- Modelo multidimensional: Se utilizan principalmente para crear aplicaciones de tipo Procesamiento Analítico en Línea[5] y pueden verse como bases de datos de una sola tabla. Una base datos de este estilo tiene una alta dimensionalidad lo cual perjudica el tiempo de respuesta a la hora de realizar operaciones sobre ella, para resolver este problema se utilizan estructuras de datos más complejas multidimensionales (o cubos OLAP) que contienen datos resumidos.

- Modelo orientado a objetos (OO): La información se representa mediante objetos como los presentes en la programación orientada a objetos, están diseñadas para trabajar de una mejor manera en conjunto lenguajes de este paradigma. Este modelo tiene la ventaja de que almacena directo en discos objetos y estos se integran de manera transparente a un programa desarrollado con un lenguaje de programación OO, de otro modo se utiliza un puente para realizar esta conversión.

- Modelo deductivo: este modelo permite hacer deducciones mediante inferencias, se basa en reglas y hechos que se encuentran almacenados en la base de datos. Este modelo es también llamado modelo lógico.

- Modelo entidad-relación: Propuesto a mediados de los años setenta como medio de representación conceptual de los problemas y para representar la visión de un sistema de forma global. Físicamente adopta la forma de un grafo escrito en papel al que se denomina diagrama Entidad-Relación. Sus elementos fundamentales son las entidades y las relaciones. Una entidad caracteriza a un tipo de objeto, real o abstracto, del problema a modelar; una relación es una asociación o relación matemática entre varias entidades.

- Modelo relacional: Es un modelo base basado en lógica predicado de primer orden, formulado y propuesto en 1969 por E. F. Codd. El propósito del modelo relacional es proporcionar un método declarativo para especificar los datos y consultas. La representación de los datos y las

[5] *On-Line Analytical Processing* (OLAP, por sus iniciales en inglés). Es una solución utilizada en el campo de la llamada Inteligencia empresarial (o **Business Intelligence**) cuyo objetivo es agilizar la consulta de grandes cantidades de datos.

relaciones que existe entre éstos se establece en tablas. Un registro o tupla es una fila de una tabla, cada columna de la tabla es un atributo. Para obtener un buen diseño con este modelo se debe someter a la base de datos a un proceso de normalización.

A continuación se describe el modelo relacional el cual es el modelo más utilizado por los sistemas de bases de datos y por lo tanto es el utilizado en éste trabajo.

2.5.1. Modelo Relacional

El modelo relacional para la gestión de base de datos está fundamentado en la lógica de predicados de primer orden, formulado y propuesto en 1969 por E. F. Codd [20, 21]. La idea fundamental de este modelo es que todos los datos se representan como una relación matemática n-aria, es decir, un sunconjunto del producto cartesiano con n dominios. Los datos son operados por medio de cálculo relacional y álgebra relacional.

El modelo relacional de datos permite diseñar bases de datos con una representación consistente y lógica de la información, esta consistencia se logra mediante las declaraciones de restricciones en el diseño de la base de datos.

Una base de datos relacional consiste en un conjunto de relaciones que están representadas por tablas, a cada una de las cuales se les asigna un nombre exclusivo, estas relaciones estan formadas por dos secciones bien definidas, una cabecera integrada por un conjunto de atributos y un cuerpo formado por un conjunto de tuplas. Una tupla es un conjunto ordenado de valores de los atributos descritos en la cabecera de la relación[22].

2.6. Lenguaje de consulta estructurado (SQL)[6]

El lenguaje SQL permite realizar consultas a la base de datos para obtener los datos de una manera más sencilla para el usuario, fue creado a principios de 1970 por E. F. Codd que trabajaba en ese entonces para IBM su primer nombre fue Sequel y aunque actualmente se siguen desarrollando nuevas versiones que incrementan las opciones de operaciones para una consulta, es el lenguaje estándar de las bases de datos relacionales[22, 21, 20].

[6] *Structured Query Language* (SQL, por sus siglas en inglés)

Dependiendo del propósito de un comando utilizado en una consulta en SQL, se tienen las siguientes categorías[15]:

- Lenguaje de definición de datos: se puede realizar la definición de la estructura de la base de datos, es decir, permite crear y modificar el esquema.
- Lenguaje interactivo de manipulación de datos: brinda los recursos necesarios para realizar las consultas a la base de datos.
- Control de transacciones: contiene las instrucciones necesarias para definir el comienzo y el final de una transacción.
- Autorización: el lenguaje de definición de datos incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.
- SQL incorporado: son las estructuras que se utilizan en algún lenguaje anfitrión, de tal manera que el SQL queda embebido, los lenguajes anfitriones la mayoría de las veces son lenguajes de propósito general como son Java, C, Ruby, entre otros. El SQL incorporado se utiliza para obtener, actualizar o eliminar datos de una base de datos desde un sistema de información.
- SQL dinámico: este componente permite construir y realizar consultas en tiempo de ejecución. En un sistema de información comunmente el usuario interactúa con el sistema, en este caso, el lenguaje de propósito general en el cual está desarrollado el sistema recibe datos con los que realiza consultas a la base de datos.

Dependiendo de la acción que se desee realizar, la estructura general básica de una expresión en SQL consta de las siguientes cláusulas:

- Proyección
  - SELECT: permite proyectar los atributos que se desea conocer de una relación.
  - FROM: se proyectan las tablas que serán utilizadas en una consulta, no importa el número de relaciones, puede ser sólo una o varias relaciones.
  - WHERE: en ésta se incluye el filtro o condición que deben cumplir los registros que se desean obtener.

- Inserción
  - INSERT INTO: especifica la relación en la que se insertaran los datos.
  - (atributo1,atributo2,...): especificación de los atributos que se insertarán, si no se especifican se asume que se insertaran todos los atributos de la definición de la tabla.
  - VALUES (valor1,valor2,....): se listan los valores que se insertarán para la nueva tupla, estos deben coincidir en orden segun la lista del punto anterior.
- Actualización
  - UPDATE: indica el nombre de la relación que se actualizará.
  - SET: lista de los atributos con su valor que serán actualizados, en el siguiente formato atributo = valor.
  - WHERE: en ésta se incluye el filtro o condición que deben cumplir los registros que se desean actualizar.

2.7. Álgebra relacional

La teoría que está detrás del SQL es el álgebra relacional. Ésta es un conjunto de operaciones que describen paso a paso cómo obtener una respuesta sobre las relación de una base de datos. Estas operaciones toman como entrada una o dos relaciones y dan como resultado una nueva relación. Las operaciones fundamentales del álgebra relacional son *selección, proyección, unión, diferencia de conjuntos y producto cartesiano.*

2.7.1. Selección

Esta operación selecciona las tuplas de una relación que satisfacen un predicado dado, se simboliza con la letra griega sigma minuscula (σ), el predicado aparece como subíndice de σ y la relación es el argumento que se coloca entre paréntesis. En el predicado se permite utilizar los operadores $=, \neq, <, \leq, \geq, >$para indicar comparaciones, así como los símbolos de conexión $y(\wedge)$ y o $(\vee)$ para unir dos o más predicados.

Supongamos que tenemos la relación Alumno(nombre, edad, sexo); para obtener todos las tuplas de Alumno se indica de la siguiente manera:

$$\sigma(Alumno)$$

Para obtener los alumnos mayores a 18 años la expresión sería la siguiente:

$$\sigma_{edad>18}(Alumno)$$

Para obtener los alumnos mayores a 18 años y que además sean del sexo femenino, se obtienen de la siguient forma:

$$\sigma_{edad>18 \wedge sexo=femenino}(Alumno)$$

2.7.2. Proyección

La operación de proyección permite obtener solo los atributos deseados de las tuplas que se buscan. Esta operación es unaria y se denota por la letra griega pi (Π). La lista de atributos que se desea como resultado aparece como subíndice de Π y la relación es el argumento que se coloca entre paréntesis. Por ejemplo, en el caso de la relación anterior, para obtener solo los nombres de los alumnos se escribe de la siguiente manera:

$$\Pi_{nombre}(Alumno)$$

Dado que las operaciones del álgebra relacional generan como resultado una relación, es posible componer estas operaciones para formar una expresión del álgebra relacional, por ejemplo para obtener los nombres de los alumnos que cumplan con la condición de ser mayores a 18 años y del sexo femenino se escribe lo siguiente:

$$\Pi_{nombre}(\sigma_{edad>18 \wedge sexo=femenino}(Alumno))$$

2.7.3. Unión

La operación de unión toma como atributos dos relaciones R y S, devuelve una relación que contiene las tuplas que están en R, o en S, o en ambas, eliminando las tuplas repetidas. Para utilizar esta operación se deben cumplir dos condiciones.

1. Las relaciones R y S deben ser de la misma aridad, es decir, deben tener el mismo número de atributos.

2. Los dominios de los atributos i-esimos de R y de S deben ser iguales para todo i, es decir, deben ser del mismo tipo.

La unión se representa con el simbolo de U . Supongamos que además de la relación Alumno descrita anteriormente contamos con la relación Persona(nombre,domicilio). Queremos obtener la unión de Alumno con Persona.

$$Alumno U\, Persona$$

Si deseamos obtener la unión de Alumno y Persona y obtener solo el campo de nombre:

$$\Pi_{nombre}(Alumno)\, U\, \Pi_{nombrep}(Persona)$$

2.7.4. Diferencia

La operación de diferencia se denota por el simbolo $\sim$, recibe dos relaciones R y S. La expresión $R \sim S$ devuelve las tuplas que están en R pero no en S. Por ejemplo, sean las dos relaciones descritas anteriormente, para buscar los nombres de los alumnos que están en Alumno pero no en Persona se escribe lo siguiente:

$$\Pi_{nombre}(Alumno) \sim \Pi_{nombrep}(Persona)$$

2.7.5. Producto cartesiano

La operación de producto cartesiano se denota con el simbolo $\times$, permite combinar información de dos relaciones.

Por ejemplo, para obtener todos los atributos del producto cartesiano de Alumno y Persona se da la siguiente expresión:

$$Alumno \times Persona$$

El resultado de esta expresión es una relación que contiene todos los atributos de las combinaciones de Alumno y Persona, el hecho importante de este resultado es que tanto Alumno como Prestamo coinciden en un mismo atributo, nombre, para distinguirlos en la relación resultante se les añade como prefijo al nombre de la relación, así los atributos de la relación resultante son

(alumno.nombre, edad, sexo, persona.nombre, domicilio) y esto para todos los atributos que coincidan en ambas relaciones.

El resultado de está operación es de gran tamaño, es decir, si R tiene n tuplas y S m, el resultado tendrá n*m tuplas.

Capítulo 3

Bases de datos espaciales

Actualmente se manejan grandes cantidades de datos: registros bancarios, datos personales, datos estadísticos, etc; sin embargo, en la actualidad mucha de la información con la que se cuenta contiene datos geoespaciales, y existe una gran variedad de aplicaciones que tienen que trabajar con datos geoespaciales y no espaciales, tales como la gestión de recursos, la planificación urbana, meteorología; entre estas aplicaciones se encuentran principalmente un SIG. El objetivo de este capítulo es presentar un panorama de lo que son las bases de datos geoespaciales.

3.1. Definición de base de datos geoespaciales

Como se mencionó en el capítulo anterior, una base de datos es una gran colección de datos interrelacionados almacenados en un entorno informático, sin embargo, hasta este punto se tiene entendido que una base de datos almacena datos alfanuméricos e incluso datos binarios (imágenes, BLOB[1], etc). No obstante, estos datos pueden contar con datos geoespaciales, y bases de datos sin soporte geoespaciales pueden ser insuficientes, pues éstas solo se enfocan en propiedades particulares de los datos léxicos [23, 24, 25].

Una base de datos espacial, es una base de datos que define datos espaciales para objetos geométricos y permite almacenar a estos en tablas regulares.

[1] *Binary Large Objects* (BLOB, por sus iniciales en inglés). Son objetos binarios grandes. Son elementos utilizados en las bases de datos para almacenar datos de gran tamaño que cambian de forma dinámica, generalmente, estos datos son imágenes, archivos de sonido y otros objetos multimedia por lo que se encuentran en representación binaria.

Para dar una mejor idea del concepto, a continuación se muestra un ejemplo [63]. Se tiene una base de datos de una cadena de restaurantes de comida rápida, donde se puede almacenar una gran variedad de atributos, por ejemplo, el ingreso diario, ingreso mensual, el número de clientes que asisten a cada sucursal, historial de ventas, etc. Para realizar análisis sobre estos datos se pueden realizar consultas tales como:

"Dar una lista de todos los clientes con ingresos mayores a una cantidad X "

Para una base de datos estándar sería sencillo, pues se está trabajando con datos alfanuméricos, supóngase que se realiza la siguiente consulta:

"Obtener todos los clientes que vivan a cierta distancia de una sucursal Y"

Para resolver este problema bastaría añadir los atributos de latitud y longitud para tener la ubicación de los clientes y de las sucursales y realizar una operación de distancia, sin embargo, realizar consultas más complicadas a una base de datos sin soporte espacial tal como:

"Obtener todos los clientes que se encuentren contenidos en el area de mercado de una sucursal Y"

Para resolver este problema, se tendría que tener almacenadas en la base de datos todas las coordenadas del área de mercado de una sucursal, la cual es finita pero no constante para cada sucursal. En una base de datos con soporte espacial bastaría con tener almancenado un polígono con las coordenadas del área de mercado de una sucursal, al realizar una operación de contención se podrían obtener todos los clientes que están dentro de ese polígono.

En resumen, una base de datos espacial almacena datos no espaciales y datos geométricos, esto es, datos que son puntos, líneas y polígonos.

3.2. Sistema Manejador de Bases de Datos con soporte espacial

Un Sistema Manejador de Bases de Datos (SMBD) con soporte espacial debe permitir realizar las siguientes operaciones sobre datos geoespaciales y alfanuméricos [26]:

1. Operaciones de entrada de datos y almacenaje de los mismos.
2. Recuperación y análisis de los datos.
3. Despliegue y selección de los datos.

Una de las necesidades de un SIG es almancenar datos, geoespaciales y alfanuméricos, este almancenamiento lo puede realizar el mismo SIG mediante el uso de archivos o con enfoque más actual y descentralizado mediante un sistema manejador de base de datos. Los primeros SIG fueron construidos para el uso de sistemas de archivos propietarios tales como el formato de archivos "shapefile" [28] el cual es un formato multiarchivo, es decir, consta de tres archivos como mínimo.

La lógica de almacenamiento y accesso a la información era controlado por los mismos SIG[29] y esto, no respeta el principio de la independencia de los datos, y conduce a muchos problemas en relación como, por ejemplo, seguridad de datos y el control de concurrencia [30].

Las principales ventajas de este enfoque son:

- Se pueden definir relaciones (tablas) en la base de datos en relación a objetos espaciales.
- Un objeto geográfico es una tupla de una relación.
- Se puede contar con atributos tanto geoespaciales como alfanúmericos.
- Se cuenta con un lenguaje basado en SQL.

En el Cuadro 3.1 muestra un ejemplo de una relación geoespacial usando un SMBD sin soporte espacial, en ella se muestra el esquema de la relación "País":

nombre	capital	población	id_limite

Cuadro 3.1: Esquema de la relación País.

La representación geoespacial se realiza de la siguiente manera:

El atributo geométrico id_limite correponde al límite de un país, y sabiendo que un país consta de distintos partes en su frontera, consideramos una

relación más, llamada "limites" compuesta de contornos. Un "*Contorno*" está caraterizado por un identificador y un punto, uno por vértice del polígono, así definimos una relación más, la relación "Punto" (Cuadro 3.2).

nombre	*capital*	*población*	*id_limite*
México	*México*	*100*	*B1*
...	...	...	...

id_limite	*id_contorno*
B1	*C1*
B2	*C2*
...	...

id_contorno	*num_punto*	*id_punto*
C1	*2*	*P1*
C1	*1*	*P2*
C1	*3*	*P3*
C1	...	...
...	...	...

id_punto	*x*	*y*
P1	*10*	*10*
P1	*11*	*10*
P3	*12*	*14*
P4	*15*	*15*
...	...	...

Cuadro 3.2: Represantación relacional de los límites de países, con base en [27], página 23.

Teniendo de esta manera las relaciones, considérese la siguiente consulta:

"Obtener todos los limites que forman a México"

Resolver esta consulta implica obtener el conjunto de coordendas que conforman el polígono de los límites de México, esta consulta está dada de la siguiente manera en lenguaje SQL:

"SELECT limite.id_contorno,x,y FROM pais,contorno, limite,punto WHERE nombre='México' AND pais.id_limite=limite.id_limite AND limite.id_contorno = contorno.id_contorno AND contorno.id_punto = punto.id_punto ORDER BY limite.id_limite, num_punto"

La mayor ventaja de este enfoque es que se logró integrar datos geoespaciales de manera alfanumérica en un ambiente estandar de SQL, por otro lado sufre de inconvenientes que lo hacen inadecuado para aplicaciones geoespaciales:

- Implica un mal funcionamiento en el método, pues requiere, en particular, una gran cantidad de tuplas para representar una sola relación geoespacial.
- Falta de facilidad de uso.
- Dificultad para definir nuevos objetos espaciales, por ejemplo líneas o multipolígonos.
- Imposibilidad de realizar cálculos geoespaciales como la contención de un punto dentro de un polígono.

Muchos sistemas de información utilizan otro enfoque, dos sistemas que coexisten [31]

- Un SMBD relacional que contiene datos alfanuméricos.
- Un módulo específico para almanecar datos geoespaciales.

Este concepto se muestra en la Figura 3.1.

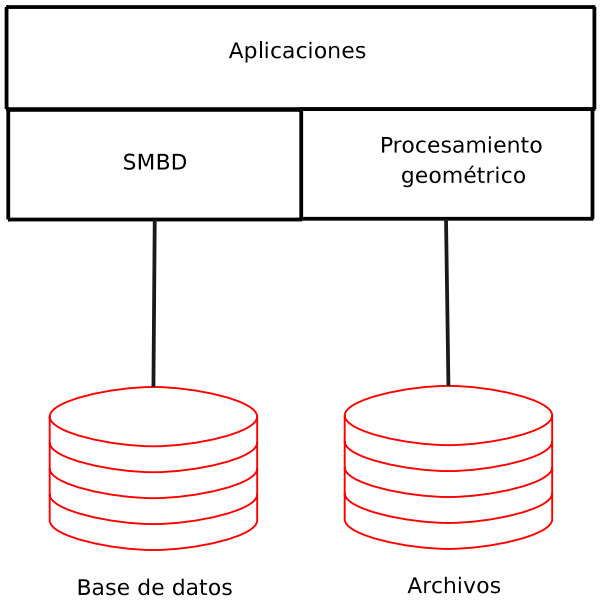


Figura 3.1: Base de datos y sistema de archivos para el almacenamiento de entidades geométricas, con base en [27] página 24.

Sin embargo, este enfoque también sufre de inconvenientes como son [32]:

- La coexistencia de modelos de datos heterogéneos, lo que implica dificultades en el modelado, el uso y la integración.
- La pérdida parcial de la funcionalidad básica de SMBD, tales como la recuperación, consultas y optimización.

Existe un tercer enfoque que evita muchos de los aspectos problemáticos de los anteriores; la exensión de un SMBD para tener soporte espacial. El concepto básico es la posibilidad de añadir nuevos tipos y operaciones de un sistema relacional a datos geoespaciales, así como la extensión del lunguaje SQL para manipular datos espaciales, y los nuevos tipos de datos espaciales son contruidos sobre una base de datos alfanuméricos. El presente trabajo hace uso de este enfoque utilizando un SMBD con soporte espacial [33].

3.3. Requerimientos de un Sistema Manejador de Bases de Datos espacial

Un Sistema Manejador de Bases de Datos con soporte espacial es un programa que sirve como interfaz entre el usuario y una base de datos, la cual tiene datos geoespaciales.

Estos SMBD con soporte espacial deben satisfacer dos necesidades básicas [34] :

1.Integrar la representación y la manipulación de información geometríca con datos no espaciales a nivel lógico.

2.Proveer un soporte eficiente a nivel físico para almecenar y procesar esta información.

A continuación se da una lista de requerimientos más extensa para que un SMBD con soporte espacial pueda satisfacer estos objetivos:

- La representación lógica de los datos debe hacerse extensiva a los datos geométricos, al tiempo que satisface el principio de independencia de datos[34].
- El lenguaje de consulta debe integrar nuevas funciones con el fin de capturar el conjunto de posibles operaciones aplicables a los objetos geométricos[35].
- Debe existir una representación física eficiente de los datos espaciales[36].
- Acceso eficiente a datos espaciales, así como para datos no espaciales[37].
- Nuevos algoritmos para la realización de consultas[38].

3.4. Manipulacion de datos espaciales

A continuación se describen las principales operaciones que lleva a cabo un SMBD con soporte espacial, para esto se introduce el concepto de tema y objeto grográfico, cabe mencionar que estas operaciones son ejemplificadas con un mapa, sin embargo al usar un lenguaje espacial basado en el estándar SQL lo que regresan como resultado es una tabla.

3.4.1. Tema

En un SIG, la información geoespacial que corresponde a un tópico en particular se le conoce como "tema", cuando un usuario visualiza un tema, por lo general es desplegado en pantalla un mapa, en un SMBD un tema es similar a una relación tal como se define en el modelo relacional, donde almacena información tanto alfanumérica como espacial, en este capítulo tomaremos como equivalente los términos tema y relación (Figura 3.2).

Figura 3.2: Ejemplo de un tema geográfico de México y Centro América.

3.4.2. Objeto geográfico

Un objeto geográfico es un elemento, una tupla de un tema, por lo tanto un tema es una colección de objetos geográficos. Un objeto geográfico corresponde a una entidad en el mundo real y consta de dos componentes:

- Una descripción, el objeto es descrito por un conjunto de atributos. Por ejemplo, el nombre y la población de una ciudad constituyen su descripción. Estos también se conocen como atributos alfanuméricos.
- Un componente espacial, incorpora la geometría (ubicación en el espacio, la forma) y la topología (relaciones espaciales existentes entre los objetos, tales como adyacencia). Por ejemplo, una ciudad puede tener como valor geométrico de un polígono en el espacio 2D y como topología un polígono adyacente a él.

3.4.3. Operaciones con temas

Las siguientes son algunas manipulaciones comunes. Se basan en las operaciones del álgebra relacional [39], por lo que se utilizan los mismos símbolos.

3.4.3.1. Proyección

La operación de proyección tiene la siguiente definicion $\Pi_{atributo1,atributo2,..}(tema)$ donde atributo1,atributo2,... es un subconjunto de atributos y devuelve otro tema cuya descripción está basada en los atributos especificados y sus propiedades espaciales no han cambiado. Considerando esta operación y la relación de País, cada país es un objeto geográfico y su nombre y población representa su descripción. Aplicando la proyección sobre el atributo de población, se eliminan el nombre del estado y su capital (Figura 3.3) y el tema resultante consta de los atributos (población y la geometría del país).

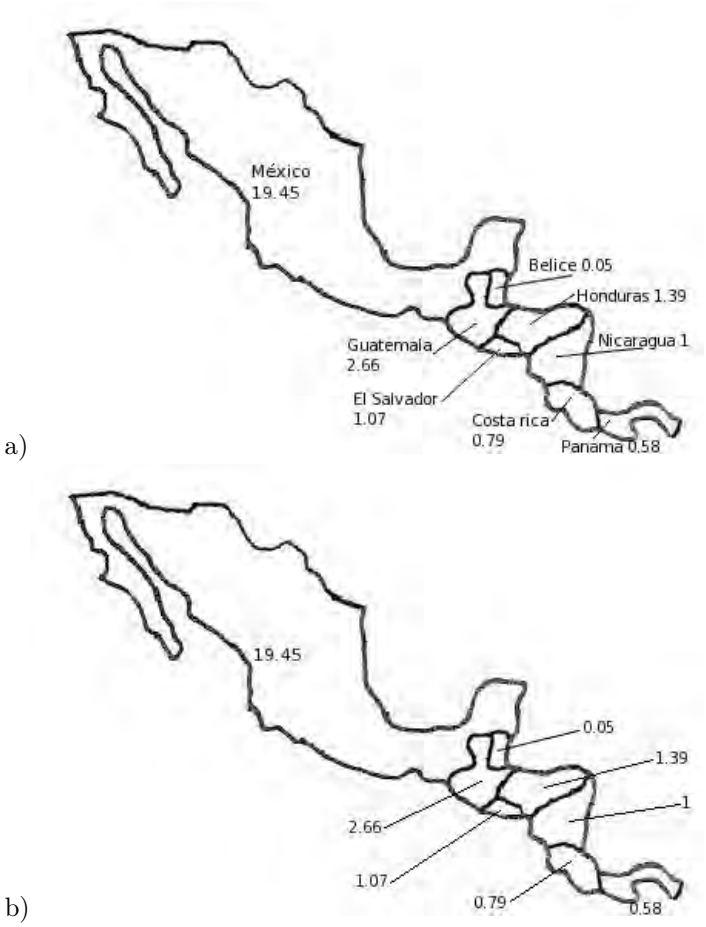


Figura 3.3: Proyección sobre el atributo población del tema Pais (a), en (b) se puede observar el resultado de realizar $\Pi_{poblacion}(Pais)$.

3.4.3.2. Selección

En la selección $\sigma_{predicado}(tema)$, el predicado es similar al de la clausula WHERE del álgebra relacional. Considérese la siguiente consulta:

"Obtener el nombre y población de los países tales que su población sea mayor al 1 por ciento de América"

En la Figura 3.4 se muestra el resultado.

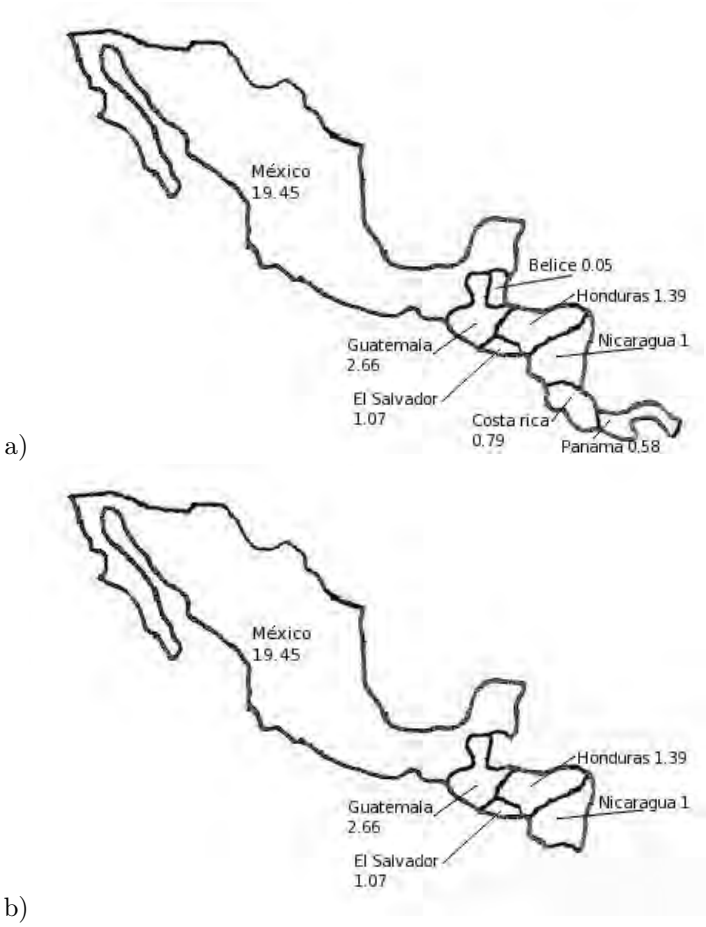


Figura 3.4: Selección de los atributos nombre y población del tema Pais (a), en (b) se puede observar un tema con atributos correspondiente al aplicar $\sigma_{poblacion>1}(Pais)$.

3.4.3.3. Unión

La unión (relacional) de dos temas $tema1 \cup tema2$ consiste en la unión de los conjuntos de objetos geográficos con el mismo esquema. En la Figura 3.5 se puede observar un ejemplo, al unir los países que tiene menos del uno por ciento de población total de América con aquellos que tiene más del uno por ciento.

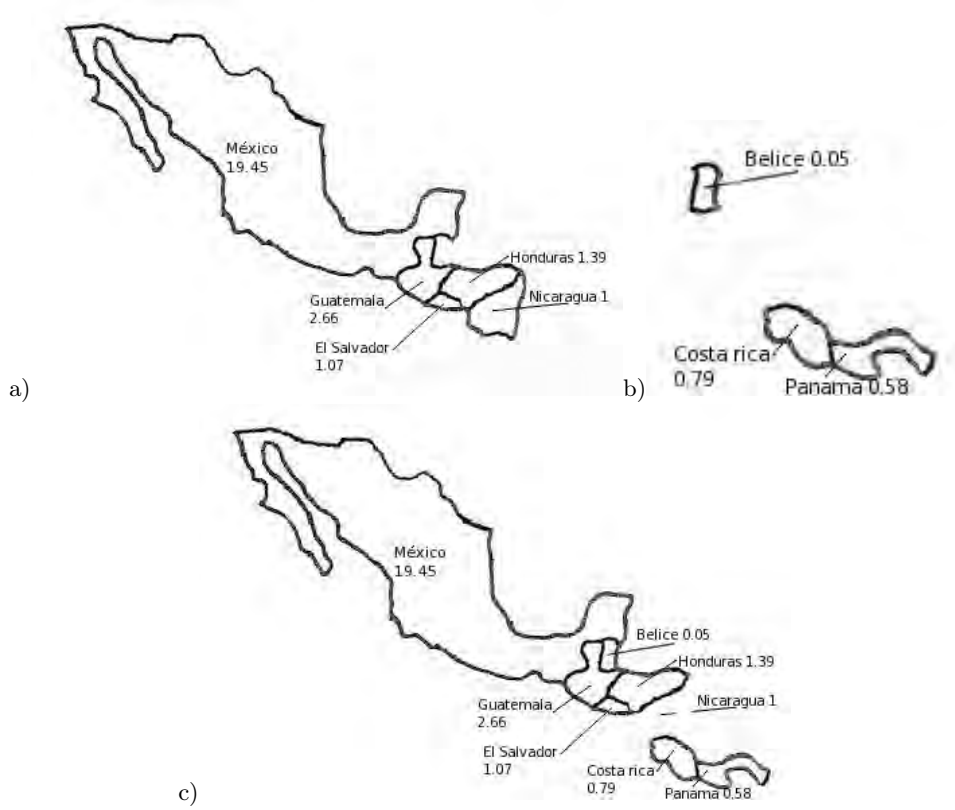


Figura 3.5: Unión de dos temas, (a) muestra los países que tienen más del 1 % de la población de América y (b) los que tienen menos del 1 %. En (c) vemos la unión, es interesante observar que ya no aparece Nicaragua, pues tiene exactamente el 1 %.

3.4.3.4. Superposición

La superposición de dos temas *tema* × *tema* es común en las aplicaciones de SIG. Esta operación genera un nuevo tema de temas superpuestos. Los nuevos objetos geográficos se crean en el tema resultante. Su geometría se calcula mediante la aplicación de la operación de intersección de la geometría de los objetos geográficos involucrados. Existe una gran variedad de operaciones de superposición. La descrita anteriormente se puede expresar como un "join" espacial, donde un objeto de un tema se ha unido con un objeto de otro tema si sus geometrías forman una intersección. Obsérvese la Figura 3.6 donde se muestra un ejemplo de superposición.

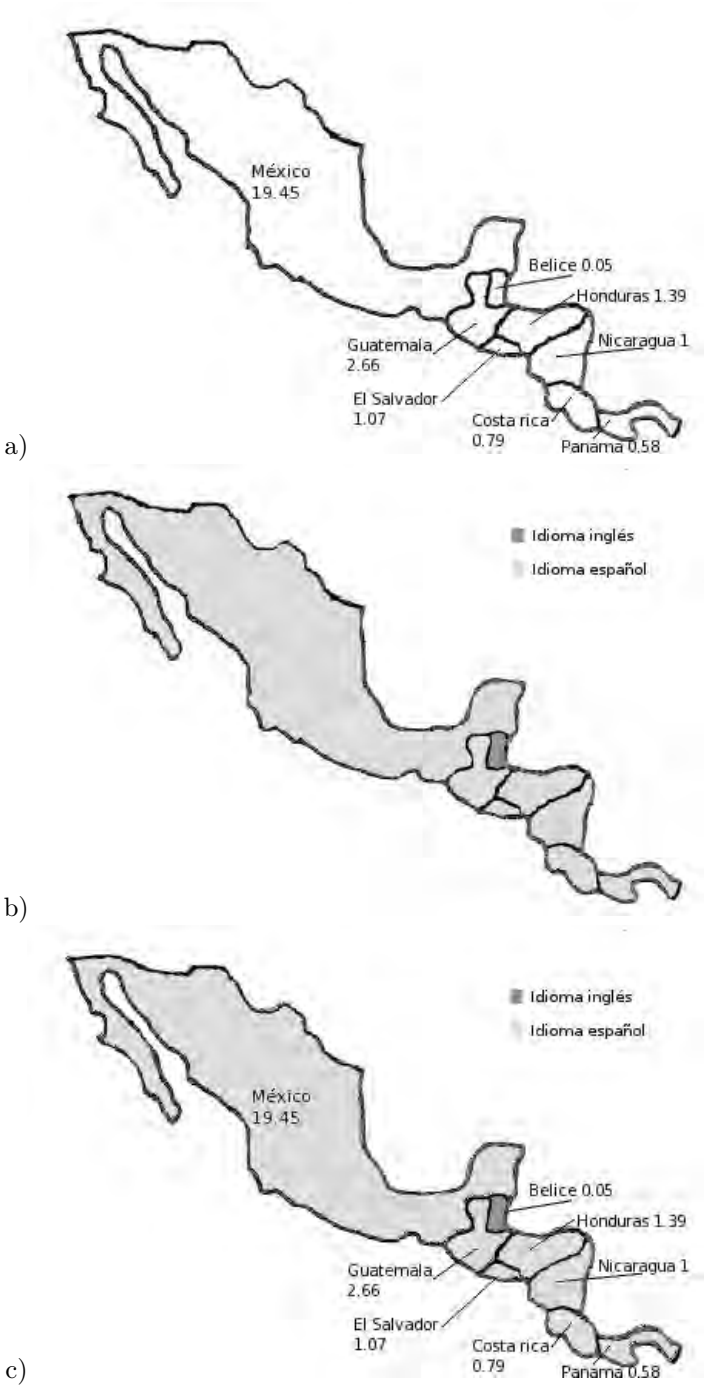


Figura 3.6: Superposición de dos temas. El tema (a) contiene el porcentaje de población de cada país de México y Centro América y el tema (b) contiene el idioma principal en esos mismos países, en (c) se muestra como se superponen los dos temas para formar uno solo.

3.4.3.5. Selección geométrica: Ventanas

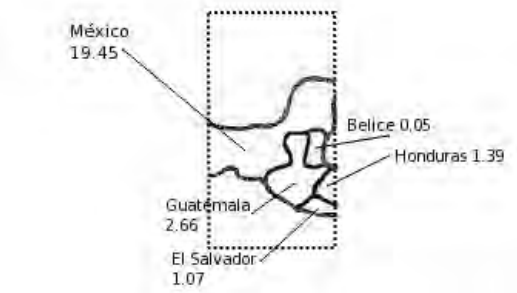
Mediante esta operación se obtiene otro tema, que incluye sólo los objetos del tema de entrada que se superponen a un área determinada: una ventana, que suele ser rectangular. En la Figura 3.7 se puede observar un ejemplo de esta operación.

Figura 3.7: Ventana de una porción de tema (a), el resultado (b) es un tema formado por todos aquellos objetos que alcanzan ser tocados por la ventana.

3.4.3.6. Selección geométrica: Recorte

Recorte extrae la parte de un tema ubicado en un área determinada; En la Figura 3.8 se observa un ejemplo. En comparación con la operación de ventana, la geometría obtenida corresponde a la intersección de los objetos geométricos y un rectángulo dado.

a)



b)

Figura 3.8: Recorte de una porción de un tema (a), el resultado (b) es un tema formado por los trozos de todos aquellos objetos que alcanzan ser tocados por el área de recorte.

3.4.3.7. Fusión[2]

La operación de fusión realiza la unión geométrica de la parte espacial de n objetos geográficos que pertenecen a un mismo tema, sobre una condición dada por el usuario; En la Figura 3.9 se observa un ejemplo de la operación al combinar a todos los objetos del tema Pais.

[2]En inglés *Merge*

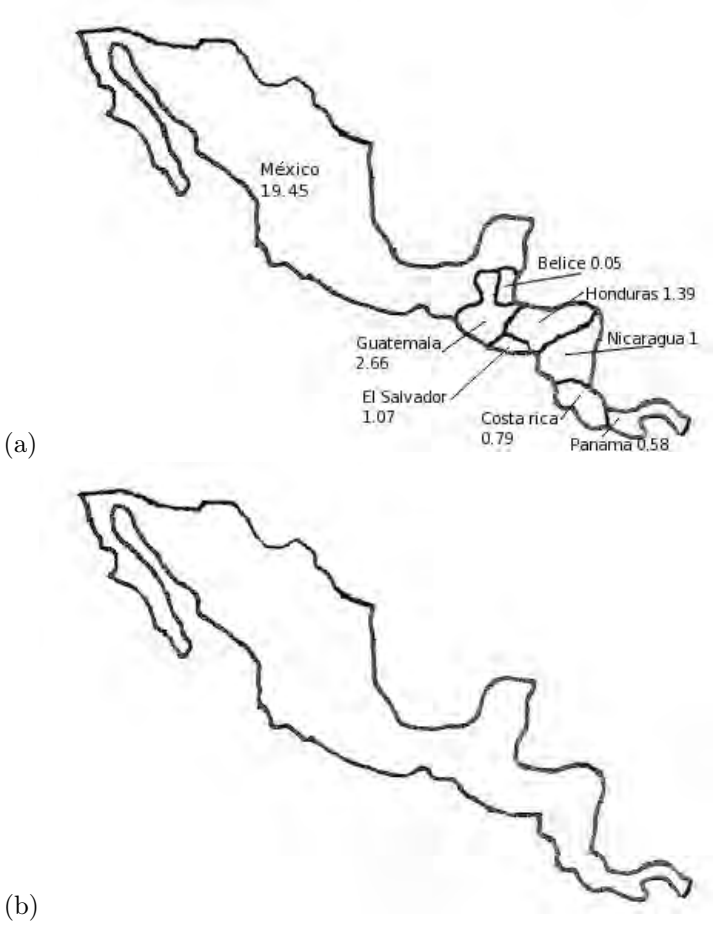


Figura 3.9: Fusión de dos temas, en (a) los países de México y Centro América, en (b) se eliminan los límites y nombres; queda un solo elemento de un nuevo tema.

3.4.4. Tipos geométricos

Un SMBD con soporte espacial debe poder almacenar elementos geométricos para dar soporte a datos geoespaciales. Como se mencionó anteriormente, los tipos básicos no son suficiente para lograr este objetivo. Es por ello que se definen ciertos tipos de datos abstractos[3] especificados por el OGC[4].

[3] *Abstract Data Type* (ADT, por sus iniciales en inglés)

[4] *Open Geospatial Consortium* (OGC, por sus iniciales en inglés) es un conjunto de organizaciones públivas y privadas que tienen como fin la definición de estándares abiertos

3.4.4.1. Punto

Un atributo de tipo punto es un punto en un sistema de coordenadas ya sea definido por el SMBD o por el usuario en un espacio de dos o tres dimensiones, que consta de una coordenada X y Y para dos dimensiones y adiocionalmente, para un punto en tres dimensiones, Z.

Ambos poseen un atributo extra, M, así existe una variante para el punto ya sea 2DM o 3DM, este atributo es conocido como medida y sirve para almacenar información extra para un punto.

3.4.4.2. Línea

Una línea esta definida por almenos dos puntos o un número finito de puntos, al igual que el punto se pueden especificar el espacio de dos o tres dimensiones y el atributo M para cada punto.

3.4.4.3. Polígono

Un polígono está formado por lineas, la línea exterior que forma el polígono se le conoce como anillo exterior. Muchos polígonos consisten de un solo anillo, la frontera de éste, sin embargo un polígono puede tener más de un anillo, es decir, un polígono tiene un anillo exterior y cero o más anillos interiores a estos se les conoce como hoyos.

Al trabajar con polígonos, aparece el concepto de validez. En un polígono válido los anillos no se pueden superponer y los anillos no pueden compartir una frontera en común.

3.4.4.4. Multipunto

Un objeto del tipo Multipunto es un conjunto de puntos. Al igual que el punto, existen cuatro implementaciones, dos para puntos en el espacio de dos dimensiones y tres dimensiones, y otras dos implementaciones para los mismos espacios con el atributo de medida M.

---

dentro de los Sistemas de Informacion Geográfica así como las tecnologías que tienen que ver con información geoespacial.

3.4.4.5. Multilínea

Es un conjunto de objetos del tipo línea con cuatro implmentaciones, dos para puntos en el espacio de dos dimensiones y tres dimensiones y, otras dos implementaciones para los mismos espacios con el atributo de medida M.

3.4.4.6. Multipolígono

Un multipolígono es un conjunto de polígonos, generalmente es utilizado para representar polígonos que estén formados por más de una región o para evitar crear polígonos inválidos.

3.4.4.7. Colección geométrica

Una colección geométrica es un conjunto hetereogéneo de objetos geométricos, es decir, puede contener puntos, líneas, polígonos, multipuntos, multilineas, multipolígonos e incluso colecciones geométricas.

3.5. Lenguaje de consulta estructurado espacial (SSQL)[5]

Recientemente, la atención se ha centrado en el uso de bases de datos espaciales, el número de aplicaciones que utilizan dichos datos aumenta cada vez más [6], es por ello que la necesidad de un lenguaje para tratar dichas bases de datos ha sido identificado como fundamental para le manejo de bases de datos espaciales y aplicaciones que interactúen con ellas tales como los Sistemas de información geográfica [7].

En 1986 apareció el primer estándar para el SQL desarrollado y aprobado por el Instituto Nacional Americano de Estándares[6] en el cual se le consideraba como el lenguaje de consultas para bases de datos [8]. Fue por esta razón, y tomando en cuenta que una base de datos espacial contiene tipos de datos espaciales y no espaciales, que el SSQL fue desarrollado como una extensión de SQL.

[5] *Spatial Structured Query Language* (SSQL, por sus iniciales en inglés)

[6] *American National Standards Institute* (ANSI, por sus siglas en inglés) es una organización sin fines de lucro que supervisa el desarrollo de normas para productos, servicios, procesos, sistemas y personal en los EE.UU, ANSI coordina también las normas de EE.UU. con los organismos de normalización internacionales, por lo que los productos de EE.UU. se puede utilizar en todo el mundo.

Dentro de SSQL existen tres categorías de consultas [42]:

- Consultas exclusivas de datos espaciales, e.g., ***"Obtener todos las ciudades dentro de una nación"***.
- Consultas exclusivas de datos no espaciales, e.g., ***"Obtener la cantidad de habitantes de una ciudad"***.
- Consultas que combinan datos espaciales y no espaciales, e.g., ***"Obtener la cantidad de habitantes de las ciudades dentro una nación"*** .

Una implementación de SSQL debe preservar la estructura de las consultas de SQL así como todas sus funcionalidades alfanuméricas, es decir, debe complir con las siguientes tres condiciones [45]:

- Todo resultado de una consulta debe ser una relación.
- Las claúsulas SELECT-FROM-WHERE de SQL deben ser el marco de trabajo de su par SSQL.
- Los predicados de la claúsula WHERE deben ser formulados sobre los atributos de la relación.

Al ser SSQL un superconjunto de SQL, éste hereda sus limitaciones, las cuales son criticadas por muchos autores entre los que destaca E.F. Codd [43]; la dificultad de integrar al estandar de SQL conceptos espaciales, hacen de éste difícil de extender, por ejemplo, la presentación tabular de los datos, implica que para cada resultado de una consulta en SQL tiene que ser mostrado en pantalla en un tabla; con consultas espaciales esto resulta ser casi imposible, pues muchos de los datos interesantes que puede devolver SSQL son polígonos, los cuales frecuentemente requieren de despliegue de gráficos.

Sin embargo, el uso de "SQL puro" para la realización de consultas espaciales ha demostrado ser sintácticamente complejo es por eso que los datos espaciales son tratados como una abstracción superior de números enteros o cadenas para trabajar más comodamente en conjunto con el álgebra de datos espaciales [44].

Capítulo 4

Estadística espacial

Las técnicas tradicionales de un SIG incluyen consultas espaciales, superposición de mapas, generación y análisis de buffer, interpolación y cálculos de proximidad, entre otros [40]. Junto con la cartografía y herramientas de administración de datos, estas técnicas analíticas han sido por largo tiempo la base de los SIG. Sin embargo, en los últimos años se han incluido nuevas herramientas para extender el análisis de datos en un SIG, entre ellas se encuentran métodos geoestadísticos [41].

La estadística espacial comprende un conjunto de técnicas para describir y modelar datos espaciales, evaluar patrones espaciales, distribuciones, tendencias, procesos y relaciones entre los objetos georeferenciados. A diferencia de la estadística no espacial, la estadística espacial utiliza conceptos como el área, longitud, adyacencia, proximidad, orientación o relaciones espaciales.

4.1. Media espacial

La media espacial es una herrameinta simple de calcular y que a nivel descriptivo muestra datos interesantes, entre ellos, nos muestra el centro de masa de los valores de los eventos espaciales [53].

La media espacial es un promedio ponderado de las coordenadas en las que se ubican los atributos. El resultado de efectuar la operacion es un punto en el espacio, un par coordenado el cual puede ser visto como el centro de masa de los datos. El cálculo de la media espacial puede ser llevado a cabo con puntos o polígonos, sin embargo, cuando se desea calcular sobre polígonos

lo que se procede es obtener el centroide de cada polígono y posteriormente aplicar la formula a los puntos obtenidos. Como se mencionó, el resultado de la media son dos puntos, los cuales se calculan de manera independiente, es decir, un cálculo para el correspondiente valor en el eje de las absisas y otro para el eje de las coordenadas [53].

La media espacial tiene la propiedad de ser independiente de la elección del sistema de coordenadas, el mismo punto relativo a la ubicación de todos los puntos va a surgir como la media espacial, independientemente de la elección de los ejes [54].

La fórmula para calcular la media espacial está dada por la ecuacion de la Figura 4.1:

$$LON = \frac{\sum_{i=1}^{n} x_i * lon_i}{\sum_{i=1}^{n} x_i} \; LAT = \frac{\sum_{i=1}^{n} x_i * lat_i}{\sum_{i=1}^{n} x_i}$$

Figura 4.1: Fórmula para calcular la media espacial. Donde x_i es el valor del atributo del i-esimo dato, n el número de datos, lon_i el valor de la coordenanda en el eje X i-esimo dato y lat_i valor de la coordenada en el eje Y del i-esimo dato.

A continuación se se muestra un ejemplo de un área dividida en cuatro regiones, en cada uno de los casos mostrados, las regiones tienen un valor en sus atributos de 1, 3, 4 y 4, que pueden representar por ejemplo población. En la Figura 4.2 se muestran los centroides de cada uno de los rectángulos y los ejes coordenados.

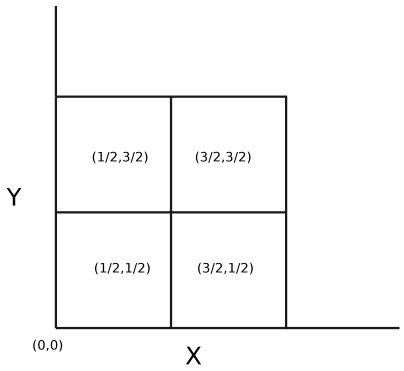


Figura 4.2: Centroides de cada una de las regiones para ejemplficar el la diferencia entre la media clásica y la media espacial

En la figura 4.3 se observa el resultado de calcular la media clásica y en la Figura 4.4 la media espacial. Para calcular la media espacial de cada region, se toma cada rectángulo y se localiza su centro.

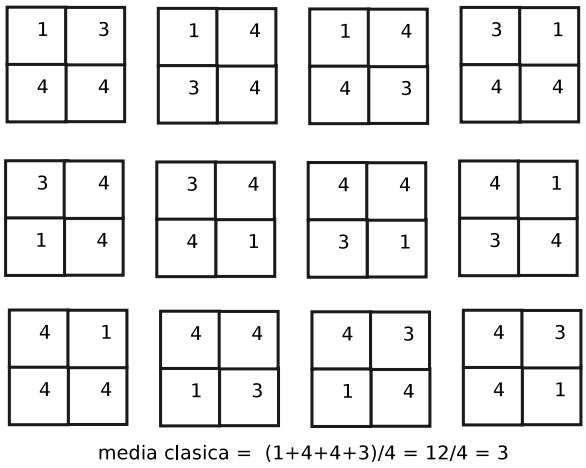


Figura 4.3: Ejemplo de diferentes distribuciones espaciales del mismo conjunto de atributos. En cada caso el valor de la media es 3. El cálculo de la media no distingue entre los diferentes patrones espaciales. Con base en [54], página 19.

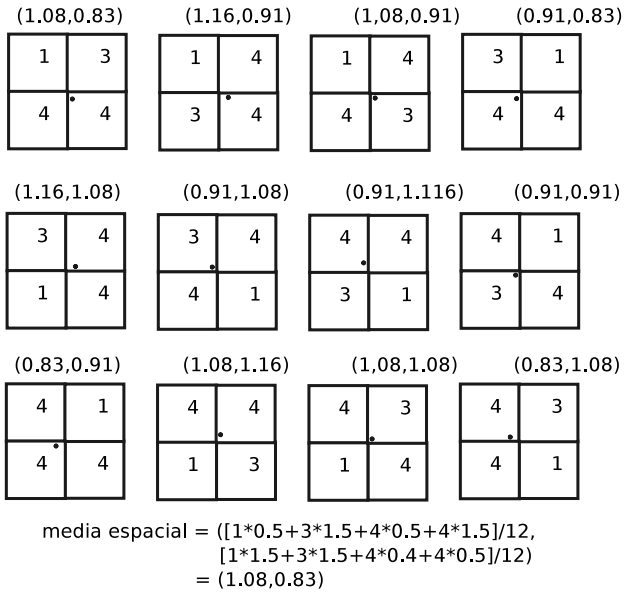


Figura 4.4: Se muestra un ejemplo de media espacial, en cada región de cada cuadro los valores son de 1,3,4 y 4 pero las medias espaciales (par de valores en la parte superior de cada cuadro) varian pues los valores de cada región varian espacialmente. En la parte inferior de la imágen se muestra el cálculo de la media espacial para la primera región

4.2. Vecino más cercano

La búsqueda del vecino más cercano[1] es el problema en el que se busca encontrar para cada punto en un espacio métrico el punto más cercano a él. El problema se define como: dado un conjunto S de puntos en el espacio, y un punto $q \in M$, encontrar el punto más cercano en S a q. Para este trabajo se tomará $M = S$, sin embargo, para el caso en que $M \; ! = S$, es simple la extensión, pues en ambos conjuntos se debe cumplir con que sus elementos contengan un atributo geográfico para poder operar sobre ellos.

Es importante mencionar que esta técnica es utilizada en el análisis de puntos, por lo tanto para realizar el cálculo sobre poligonos se debe obtener el centroide de los objetos geográficos y después operar sobre estos de manera normal [48].

[1] *Nearest neighbor search* (NNS, por sus iniciales en inglés)

Junto al cálculo del vecino más cercano para cada punto, se puede obtener el promedio al vecino más cercano, el cual indica a qué distancia de cada punto en promedio se encuentra el vecino más cercano, esta operacion se calcula con la ecuación de la Figura 4.5.

$$d = (\sum_{i=1}^{N} d_i)/N$$

Figura 4.5: Fórmula para calcular el promedio al vecino mas cercano Donde N es el número de puntos, y d_i es la distancia al vecino mas cercano para el punto i.

Esta operación es usada en Sistemas de Información Geográficos para determinar el objeto más cercano a otro en el espacio, por ejemplo, obtener el estacionamiento más cercano a cada estación de tren subterraneo; es una técnica muy utilizada en la planeación urbana para determinar la ubicación de los recursos [49].

4.3. Autocorrelación espacial: coeficiente de Moran

Dado un conjunto de datos y un atributo, la autocorrelación espacial determina si los datos están distribuidos de manera uniforme, dispersos o aleatoriamente [50]. En otras palabras la autocorrelación espacial es un concepto sobre la cercanía o lejanía de los eventos en el espacio según el valor de los atributos [53].

Cuando el coeficiente de Moran se aproxima a $+1$ se dice que hay autocorrelación positiva. En este caso los valores involucrados tienden a agregarse o ser vecinos.

Cuando el coeficiente de Moran se aproxima a -1 hay autocorrelación negativa. Los valores tienden a separarse y los puntos o eventos vecinos son de diferente tipo.

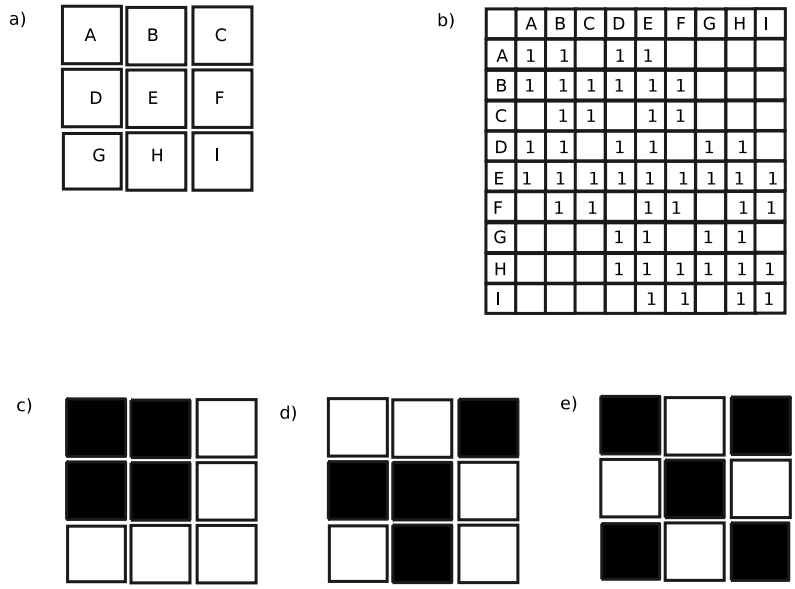
Cuando el coeficiente de Moran se aproxima $-1/(n-1)$ se dice que hay una distribución al azar de los valores (siendo n el número total de observaciones).

La fórmula para calcular el Coeficiente de Moran está dada por la ecuacion de la Figura 4.6:

$$MC = \frac{N}{\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}} \cdot \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(X_i - X)(X_j - X)}{\sum_{i=1}^{N}(X_i - X)}$$

Figura 4.6: Fórmula para calcular el Coeficiente de Moran. Donde N es el número de datos, w_{ij} es el valor en la matriz de conectividad en la posicion (i, j), X_i es el valor del atributo del i-esimo dato, X es el promedio de los X_i.

En la figura 4.7 se se muestra un ejemplo de un área hipotetica dividida en nueve regiones (A,B,C,D,E,F,G,H e I), se muestra la matriz de conectividad asociada a las nueve regiones y se muestran los tres casos de autocorrelación. En c) los valores similares tienden a estar adyacentes. En d) se muestra un ejemplo sin autocorrelacón donde los valores tienden a estar distribuidos al azar. En e) los valores disimilares tienden a estar adyacentes y por tanto ejemplifican una situación con autocorrelación negativa.



	A	B	C	D	E	F	G	H	I
A	1	1		1	1				
B	1	1	1	1	1	1			
C		1	1		1	1			
D	1	1		1	1		1	1	
E	1	1	1	1	1	1	1	1	1
F		1	1		1	1		1	1
G				1	1		1	1	
H				1	1	1	1	1	1
I					1	1		1	1

Figura 4.7: Ejemplo de autocorrelación espacial. a) Área de estudio dividida en nueve regiones. b) Matriz de conectividad. c) Autocorrelacion positiva. d) Distribución sin autocorrelación. e) Autocorrelación negatica. Con base en [54]

4.4. Semivariograma

El semivariograma es una herramienta que permite medir la autocorrelación espacial entre los puntos de muestreo.

La técnica consiste en tomar un valor δ la cual es una distancia que se va incrementando en intervalos constantes y una función $\gamma(\delta)$ que mide las diferencias entre todos los pares de valores que se encuentren a distancia δ, el resultado es una gráfica que tiene como ejes $\gamma(\delta)$ y δ [53].

El cálculo del semivariograma está dado por la ecuación de la Figura 4.8.

$$\gamma(\delta) = \frac{1}{2N(\delta)} \sum_{(i,j)|h_{ij}=h} (x_i - x_j)^2$$

Figura 4.8: Fórmula para calcular el Semivariograma. Donde $N(\delta)$ es el número de pares de puntos que se encuentran a distancia h, h es un escalar tomado para medir la distancia entre los pares de puntos, h_{ij}es la distancia que existe entre los pares de puntos (i, j), x_i es el valor del i-esimo dato.

Si los datos tienen una autocorrelación espacial positiva, las diferencias entre los pares de valores cada vez más distantes crecerán de manera continua, y la gráfica tendrá un aspecto de curva asintótica creciente. En otro caso, si los datos no tienen una autocorrelación espacial positiva o no existe autocorrelación, las diferencias entre los valores estarán alternadas entre grandes y pequeñas al ir aumentando δ. en la Figura 4.9 se muestra un ejemplo de lo anterior.

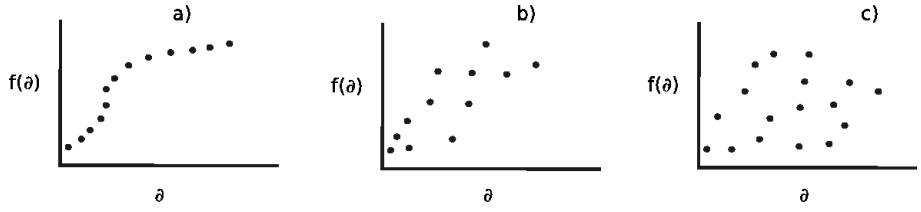


Figura 4.9: Ejemplo de Semivariograma de acuerdo a los diferentes tipos de autocorrelación espacial. a) Semivariograma típico de autocorrelación espacial positiva. b) Semivariograma típico de autocorrelación espacial negativa. c) Semivariograma típico de no autocorrelación espacial. Con base en [53].

Un factor importante a tener en cuenta al aplicar el semivariograma es no elgir valores de δ muy grandes, pues podria ocurrir que la cantidad de pa-

res de puntos que se encuentran a dicha distancia sean muy pequeña y eso sesgaría el análisis.

Capítulo 5

Sistemas de información geográficos

Los Sistemas de Información Geográficos (SIG) surgen para cubrir la necesidad de manipular datos geográficos con mayor facilidad.

La tecnología de los SIG puede ser utilizada para investigaciones científicas, la gestión de los recursos, gestión de activos fijos, la arqueología, la evaluación del impacto ambiental, la planificación urbana, la cartografía, la sociología, la geografía histórica, la mercadotecnia y la logística, por nombrar algunos usos. Por ejemplo, un SIG podría permitir a los grupos de emergencia calcular fácilmente los tiempos de respuesta en caso de un desastre natural. Un SIG puede ser usado para encontrar los humedales que necesitan protección contra la contaminación, o pueden ser utilizados por una empresa para ubicar un nuevo negocio y aprovechar las ventajas de una zona de mercado con escasa competencia.

5.1. Definición

"Como sistema de información se entiénde la unión de la información y herramientas informáticas para su análisis con unos objetivos concretos. Por otra parte, al incluir el término Geográfica se asume que la información es espacialmente explícita, es decir, incluye la posición en el espacio" [62].

Un SIG funciona como un sistema de información, pero con una base de datos que contiene datos geográficos.

La única diferencia que tienen los sistemas de información geográfica de los sistemas de información comunes es que éstos pueden analizar, almacenar, integrar, editar, compartir y mostrar datos geográficos o espaciales.

5.2. Funciones de un SIG

Las principales funciones por las que se desarrolla un SIG son las siguientes:

1. El agrupamiento de datos en conjuntos, que generalmente están almacenados en una base de datos, en las cuales se pueden encontrar datos como puntos, líneas, polígonos, redes topológicas y relaciones.
2. Traducir el lenguaje alfanumérico de los datos almacenados en la base de datos a un lenguaje gráfico que permita visualizar los datos con mayor facilidad, principalmente esta representacion se hace sobre un mapa. Estos sistemas pueden crear mapas, herramientas cartográficas, herramientas dinámicas, entre otras.
3. Realizar operaciones sobre los datos. Los SIG permiten además de visualizar los datos y realizar operaciones sobre éstos, visualizar los cambios producidos.
4. Gestión de los datos almacenados en la base de datos. Soporta la creación, el acceso a la base de datos, provee metodos para la entrada, actualización, borrado y recuperación de datos, el SIG determina los datos accesibles para cada usuario así como sus privilegios.

El uso de los SIG ha ido en aumento en los últimos años, esto se debe a diferentes aspectos como son:

- Conciencia de que los procesos que conllevan la toma de desiciones para alguna mejora tienen una dimensión espacial.
- La interacción del usuario con los SIG facilita la manipulación de los datos geográficos.
- Los SIG mejoran en términos de visualización, gestión y análisis de datos geográficos.
- Ha aumentado la disponibilidad de los datos espaciales, las experiencias y resultados en diversos campos de investigación (puede ser social, ecológica, espacial, entre otros) y gestión de recursos.

5.3. Componentes de un SIG

Como todo sistema de informacion, los SIG también tienen componentes, los cuales son:

- Un sistema manejador de bases de datos, que tenga soporte para datos espaciales o un sistema manejador de archivos.
- Herramientas que permiten la entrada y manipulación de los datos.
- Herramientas que permiten realizar consultas de datos geográficos, así como visualizarlos.
- Una interfaz gráfica fácil de manejar por el usuario.
- Datos: la base de datos de un SIG debe contener como mínimo un atributo espacial, pues de lo contrario no se explotaría al sistema, éstos deben ser fiables y con la menor redundancia posible, ya que sobre éstos se realizará el análisis.

  Las fuentes de información para recabar estos datos son:

  - Imágenes de satelites.
  - Fotografías aereas.
  - Digitalización de mapas en papel.
  - Medidas de campo (un ejemplo de está es el GPS).

5.4. Modelos de datos de un SIG

Para representar la realidad en un SIG se necesita transformar ésta a un formato digital, para lograr esto se tienen dos modelos de representación de los datos en un SIG, los cuales son:

Modelo vectorial: es una estructura de datos utilizada para almacenar datos geográficos, los datos vectorial es constan de tres tipos [76]:

- Puntos: representan las entidades más pequeñas, como pueden ser cabinas de teléfono, pozos, árboles, entre otros.

- Líneas: para representar entidades de una longitud mayor que los puntos o demasiado estrechos se utilizan las líneas, ejemplos de esto pueden ser los ríos, las carreteras, curvas de nivel, entre otros.

- Polígonos: estos objetos se utilizan para representar elementos o entidades con superficie, o con límites, como son áreas de países o estados, parcelas de uso de suelo, áreas inundadas, entre otros.

El almacenamiento de los vectores implica el almacenamiento explícito de la topología, sin embargo solo almacena aquellos puntos que definen las entidades y todo el espacio fuera de éstas no está considerado. En la Figura 5.1 se muestra un ejemplo de la representación de un modelo vectorial.

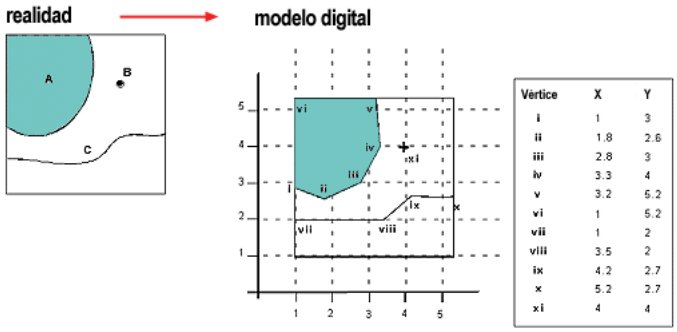


Figura 5.1: Modelo vectorial para un GIS.

Modelo raster: es un método para el almacenamiento, el procesado y la visualización de datos geográficos . Cada superficie a representar se divide en filas y columnas, formando una malla o rejilla regular. Los datos raster son una abstración de la realidad, representan ésta como una rejilla de celdas o píxeles, en la que la posición de cada elemento es implícita según el orden que ocupa en dicha rejilla. En el modelo raster el espacio no es continuo sino que se divide en unidades discretas. Esto le hace especialmente indicado para ciertas operaciones espaciales como por ejemplo las superposiciones de mapas o el cálculo de superficies [77]. En la Figura 5.2 se muestra un ejemplo de este modelo.

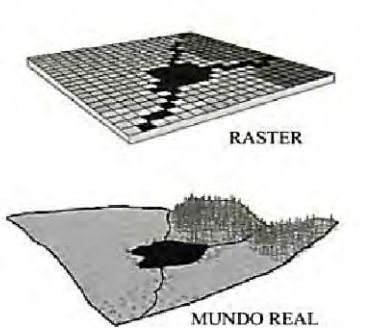


Figura 5.2: Modelos raster para un GIS.

Los elementos que representan la realidad tienen propiedades espaciales como son: la longitud, la forma, la pendiente, la orientación, la superficie y el perímetro.

Los SIG solian ser clasificados de acuerdo al modelo de representación de datos que soporte, aunque actualmente la mayoria son capaces de soportar ambos modelos.

Las representaciones vectoriales alcanza una gran presición en los datos y reducen el espacio en disco, pero su complejidad en cuanto a las estructuras de datos y los algoritmos para su manipulación son más complejos en comparación con las representaciones raster, en los que el almacenamiento de los datos requiere un mayor espacio.

5.5. Aplicaciones de los SIG

Los SIG tienen un amplio ámbito de desarrollo, ya que se utilizan en distintas disciplinas de la ciencia como lo son la estadística, biología, geografía, entre otras. Debido a esto los SIG se pueden utilizar en las administraciones públicas, planificación, detección de daños en catástrofes, empresas de servicio público, gestión de transporte, medio ambiente y agricultura.

A continuación se mencionan algunos ejemplos de los ámbitos antes mencionados:

- Mantenimiento y conservación del transporte.
    - Generación de rutas óptimas en función de condiciones.

- Evaluación del impacto territorial por la construcción de infraestructuras de transporte.
- Análisis de demanda y oferta de transporte en zonas específicas.
- Simulaciones de densidad del tráfico.

- Gestión del territorio:
  - Búsqueda de zonas aptas para proyectos de creación de infraestructura o recuperación del ambiente.
  - Estudios de impacto ambiental.
  - Gestión de las instalaciones públicas.

- Agricultura:
  - Planeación de cultivos.
  - Planeación de aplicaciones eficientes de fertilizantes.
  - Estudios sobre el estado de las cosechas.

- Administraciones públicas:
  - Asesoría de cobranza de impuestos.
  - Soluciones catastrales.
  - Planeación de proyectos de seguridad pública.
  - Planeación de proyectos de desarrollo sustentable.
  - Seguimiento de la delincuencia.

5.6. Tipos de SIG

De acuerdo al ambiente de ejecución los SIG se pueden clasificar en las siguientes categorías [66]:

1. SIG de escritorio: se utilizan para crear, editar y analizar y visualizar datos geográficos, estos a su vez se clasifican de acuerdo a su funcionalidad en:

   a) Visor SIG, estos sistemas solo permiten visualizar la información geográfica.

   b) Editor SIG, estos sistemas permiten la edición de la información geográfica, para que pueda ser analizada posteriormente.
   c) SIG de análisis: Permiten el análisis de los datos así como su visualización.

2. Servidores cartográficos: Se utilizan para distribuir mapas a través de Internet.

3. Servidores SIG: Se puede acceder desde diferentes programas a través de redes a los recursos porporcionados por el sistema, tales como mapas, etc.

4. Clientes web SIG: estos sistemas permiten solo la visualización de los datos, así como su análisis y consulta de servidores SIG.

5. Bibliotecas y extensiones espaciales: estas extensiones añaden funcionalidades que no están implementadas en el sistema principal, pueden ser herramientas para el análisis, la visualización.

6. SIG móviles: se utilizan para la recolección y visualización de datos en campo de trabajo a través de dispositivos móviles, la mayoría de esos aparatos contienen GPS integrados, los cuales son utilizados para la captura de coordenadas espaciales.

Capítulo 6

Ingeniería de software

Algunas definiciones de la ingeniería de software son:

- "Es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software, desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste" [58].
- "Es la disciplina que ofrece métodos y técnicas para desarrollar y mantener el software"[1].

6.1. Principios de la Ingeniería de Software

Los principios por los que se rige la ingenieria de software son:

- Generalidad: se trata de buscar el problema más general, para así diseñar una solución general.
- Abstracción: identificar las propiedades relevantes dejando fuera los detalles, proporcionando una descripción simplificada de un sistema, enfatizando las propiedades del sistema más importantes.
- Modularidad: significa que el problema se divide en piezas o módulos más pequeñas, por lo que también se puede comprender el sistema por piezas.

[1] http://es.wikipedia.org/wiki/Ingenieria_de_software

- Incrementabilidad: el software se construye aumentando sus funcionalidad en periodos, de tal forma que en cada periodo se le agregen funcionalidades al sistema.
- Anticipación al cambio: los cambios pueden surgir cuando se eliminan errores del sistema o por la evolución del sistema, el sistema debe estar diseñado de tal manera que estos cambios sean fáciles de realizar.

6.2. Características del software

La ingeniería de software garantiza, mediante metodologías de desarrollo, que el software tenga las siguientes características:

- *Usabilidad: "*es el grado en el que un producto puede ser utilizado por usuarios específicos para conseguir objetivos on efectividad, eficiencia y satisfacción en un determinado contexto de uso"[59].

  La usabilidad de un sistema es el atributo de calidad del mismo que evalúa cuan fácil es utilizar el sistema para los usuarios, también se refiere a la característica que tiene un sistema para mejorar la facilidad de uso durante el proceso de diseño.

- Facilidad de aprendizaje: el sistema debe contar con una interfaz de usuario adecuada y documentación, como para ser capaz de realizar correctamente la tarea que se desea llevar a cabo. Se pretende requerir el mínimo esfuerzo necesario para aprender, operar y predecir las salidas del programa de acuerdo a las entradas que reciba. Se mide normalmente por el tiempo empleado con el sistema hasta ser capaz de realizar ciertas tareas en menos de un tiempo dado.

  - Eficiencia: El número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema. El software no debe desperdiciar los recursos del sistema lo cual se incrementa al tener un buen diseño.
  - Recuerdo en el tiempo: para usuarios que no utilizan el sistema regularmente es deseable que sean capaces de usar el sistema sin tener que aprender cómo funciona partiendo de cero cada vez.
  - Tasa de errores: se refiere al número de errores cometidos por el usuario mientras realiza una determinada tarea, por lo que se desea obtener la menor tasa de errores posible.

- Mantenibilidad: es la facilidad de comprender, corregir, adaptar y mejorar el software, aún después de haber entregado el software al cliente. El mantenimiento es:
  - correctivo si se trata de corregir los errores.
  - adaptivo si se requiere modificar el software con respecto a su entorno.
  - perfectivo si se le añaden nuevas funcionalidades al software.
- Fiabilidad: es el grado en el que un software se espera que realice su función con una precisión requerida, además debe tener tolerancia a fallos y recuperación de éstos.
- Flexibilidad: se refiere a que el esfuerzo requerido para modificar un programa en funcionamiento debe ser mínimo.
- Reusabilidad: es el grado en el que el software puede ser utilizado en otros sistemas.
- Portabilidad: es la facilidad de transportar el software de un ambiente de ejecución a otro, con las mismas características o completamente diferente y que el software siga cumpliendo con sus funciones correctamente.

  *"**El código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.**"[58]*
- Interfaz de usuario: es el medio con el que el usuario puede comunicarse o interactuar con el software, éste puede emplear elementos del hardware u otros elementos de distinto software, esta interfaz debe ser fácil de usuar y atractiva a la vista del usuario. De acuerdo a sus elementos comprendidos una interfaz de usuario puede ser:
  - Interfaz de hardware: solo utiliza elementos del hardware como son el ratón, el teclado o la pantalla.
  - Interfaz de software: son las interfaces que entregan la información introducida por el usuario al sistema, usualmente son observadas en la pantalla de la computadora.
  - Interfaz de software-hardware: estas interfaces están integradas por ambos elementos.

Según la interacción con el usuario se puede dividir en las siguientes categorías:

- Interfaz alfanumérica: estas interfaces solo presentan al usuario texto.
- Interfaz gráfica de usuario: son las interfaces que le permiten a los usuarios comunicarse con la computadora, representa todos los elementos gráficamente.
- Interfaz táctil: permiten que la interacción entre el usuario y el software se realice en dispositivos táctiles, simulando el flujo de la interacción de una interfaz gráfica de usuario.

Capítulo 7

Metodologías de desarrollo de software

Las metodologías de desarrollo de software son utilizadas para estructurar, planificar y controlar el proceso de desarrollo de software de una manera más eficiente de lo que sería desarrollarlo sin ellas.

A lo largo del tiempo, una gran cantidad de métodos han sido desarrollados diferenciándose por su fortaleza y debilidad y cada una de ellas se adecua mejor dependiendo del tipo de trabajo que se desea realizar, las técnicas requeridas, el tipo de organización que desarrollara el trabajo e incluso según la consideración del equipo de trabajo.

Las metodologías de desarrollo de software tradicionales se crearon en los años 60, la principal idea de estas metodologías es desarrollar los sistemas con un plan y documentación minuciosos.

Algunos años después de que aparecieran las metodologías tradicionales, surgieron las metodologías ágiles de desarrollo de software, donde la documentación no es extensa y el número de miembros del equipo de desarrollo disminuye.

7.1. Metodologías tradicionales de desarrollo de software

- **Modelado en cascada:**

También llamado *Ciclo de vida Clásico*, es un proceso secuencial entre las etapas, las cuales tienen superposición, se hace un énfasis en la planificación de todos los aspectos del sistema, se tiene una retroalimentación hacia las capas anteriores (por esta razón las fallas pueden prevenirse en gran medida), tiene entradas (son los requisitos de cada etapa) y salidas (son el resultado del trabajo realizado en la etapa anterior). La documentación es muy importante, ya que de esta manera se lleva un control estricto del sistema y se debe realizar durante todo el proceso, el sistema entero es descrito y registrado en los documentos generados. Este método asume que una vez especificados los requerimientos, éstos no cambiarán más, pero se ha visto a lo largo de estos años, que el cliente la mayoría de veces cambia o define mejor sus necesidades en cuanto al software. Las etapas de esta metodología se ilustran en la Fgura 7.1:

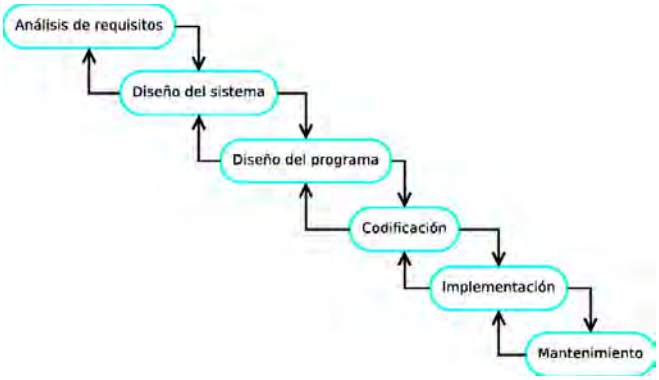


Figura 7.1: Etapas del modelado en cascada

- **Prototipo evolutivo:**

  Es un modelo iterativo, en la cual se desarrollan versiones del software, se busca reemplazar el viejo sistema con uno nuevo, bajo la perspectiva de satisfacer los nuevos requerimientos lo más pronto posible. Se da por hecho que los requerimientos están en un cambio contínuo durante el proceso de desarrollo. El objetivo de crear varias versiones es que sólo se vayan agregando en cada iteración los requisitos que realmente hayan sido bien comprendidos, posterior a la interacción que los usuarios tienen con la versión actual se realiza una retroalimentación, que proporciona los nuevos requisitos del sistema. Las etapas de este modelo agregan funcionalidades al producto de software operacional en cada

iteración, como se muestra en la Figura 7.2:

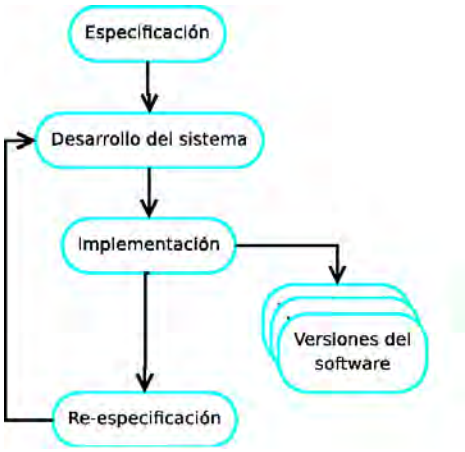


Figura 7.2: Etapas del modelo prototipo evolutivo

- **Prototipos:**

  Este modelo cuenta con iteraciones sucesivas, en las cuales se redefinen los requerimientos del sistema, logrando así adaptar mejor los prototipos a las necesidades de los usuarios. Con este modelo se reduce el riesgo de obtener un sistema que no satisfaga las necesidades de los usuarios. Las principales etapas de este modelo son:

  - Investigación preliminar: se determina el problema y su alcance, se realiza un estudio de factibilidad y se definen los impactos que este sistema tendrá.
  - Definición de requerimientos: se definen los requerimientos y necesidades que el usuario tiene acerca del sistema en desarrollo. Si eventualmente hay más requerimientos entonces el método de prototipos se repite y se definen nuevamente los requerimientos. El prototipo es modificado y evaluado repetidamente hasta que los requerimientos del sistema han sido satisfechos.
  - Diseño y construcción rápido: en esta etapa se construye el prototipo inicial.
  - Pruebas: en esta etapa se comprueba si el sistema cumple con los requerimientos.
  - Modificaciones: en esta etapa se realizan los cambios definidos en la etapa anterior, sobre el prototipo inicial.

- Diseño detallado: esta etapa es también llamada análisis grueso, en esta se rediseña el prototipo y se documenta exhaustivamente para brindar facilidad en la programación y mantenibilidad del sistema.
- Implementación del sistema: se desarrolla el sistema, se realizan las pruebas, la instalación y las modificaciones necesarias.

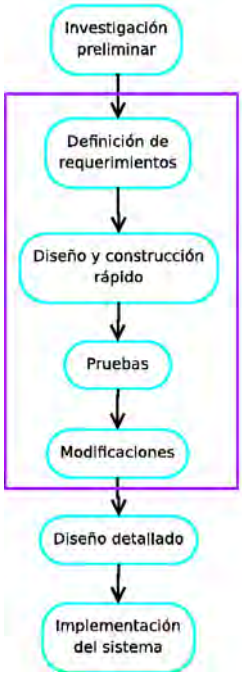


Figura 7.3: Etapas del modelo de prototipos

El cuadro de la Figura 7.3 enmarca las etapas del desarrollo evolutivo que son iteradas un gran número de veces, y es en estas etapas en donde se crean y modifican los prototipos.

- **Incremental:**

  Este modelo sigue la filosofía de la construcción iterativa de prototipos. Se divide en las siguientes etapas principales: análisis, diseño, construcción de código, pruebas y por último integración del sistema. La iteración que se realiza en este modelo tiene la arquitectura de *pi-*

peline[1] como se muestra en la figura 6.4. El cliente se mantiene en contacto con el sistema en cada iteración, por lo que si en alguna iteración no se cumple con los requerimientos o se encuentra una falla solo es necesario desechar la última iteración y retomar la versión anterior del sistema. Generalmete al inicio de este modelo el grupo de desarrollo es reducido y conforme avanzan las iteraciones se añaden miembros al equipo de desarrollo. El análisis de la iteración está basado en la retroalimentación del usuario y el análisis de las funcionalidades del sistema. Una de las principales desventajas de este modelo es que requiere de mucha planeación y por lo tanto de mucha documentación.

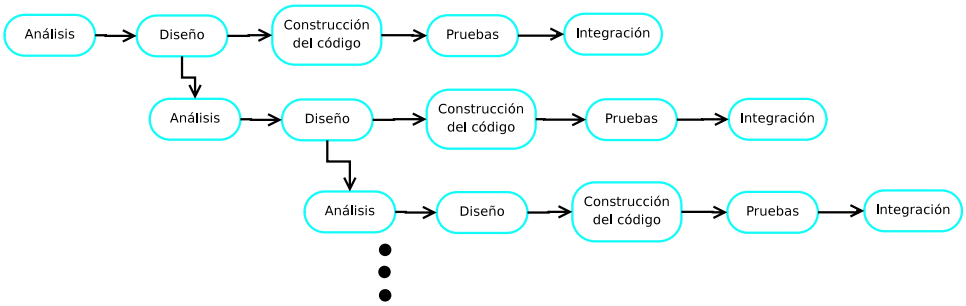


Figura 7.4: Etapas del modelo incremental

- **Espiral:**

  Es uno de los modelos más conocidos, combina las mejores características del modelo en cascada y el modelo de prototipos. Se realizan iteraciones que parecen ciclos o bucles, en cada iteración se identifican los objetivos correspondientes, así como las alternativas y las restricciones del sistema.

  Las etapas de este modelo son:

  - Determinar objetivos y alternativas: en esta etapa se definen los objetivos de la iteración. También se realiza el manual de usuario correspondiente a la iteración.
  - Evaluar alternativas y resolver riesgos: se evalúan los riesgos definidos en la etapa anterior.

[1] Esta arquitectura consiste en transformar un flujo de datos en un modelo que tiene varias faces secuenciales tomando como entrada la salida de la fase anterior.

- Desarrollar y verificar resultados: dependiendo del resultado de la evaluación de la etapa anterior, se elige algún modelo para solucionar y evitar estos riesgos y se desarrolla el sistema.
- Planear la próxima iteración: se revisa todo lo que se lleva desarrollado hasta ese momento y se decide si se continúa con la siguiente iteración y se planifican las siguientes actividades.

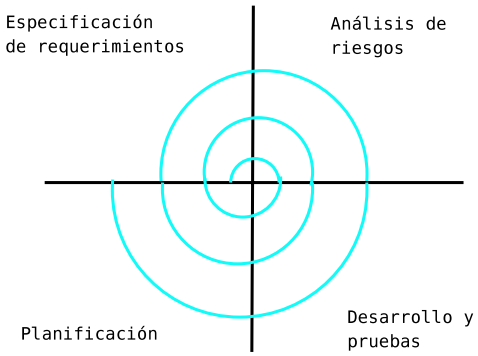


Figura 7.5: Etapas del modelo en espiral

7.2. Metodologías ágiles de desarrollo de software

Como se vio en la sección anterior, existen bastantes metodologías de desarrollo de software, todas estas conllevan una documentación del software exhaustiva, algunas son iterativas, otras dan por hecho que los requerimientos del sistema se pueden denifir desde el principio del desarrollo y que estos no cambiarán, pero desafortunadamente con el avance tecnológico, el cambio de las necesidades de los clientes y los usuarios y otros diversos motivos, el software continuamente necesita ser modificado o reemplazado por uno nuevo, por lo que se crearon las metodologías ágiles de desarrollo de software.

Las metodologías ágiles se rigen por los principios que a continuación se mencionan[2]:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

[2]Principios del manifiesto ágil. http://www.agilemanifesto.org/iso/es/principles.html

2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

3. Entregamos software funcional, frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. El software funcionando es la medida principal de progreso.

8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Algunas de las metodologías ágiles de desarrollo de software son:

- XP (eXtreme Programming):

  La Programación Extrema hace uso de cinco principios básicos:

- Comunicación: El cliente siempre es parte del equipo de desarrollo y mantener una comunicación constante con él permite establecer los requerimientos del sistema; de la misma manera mantener una comunicación con el equipo de trabajo es indispensable, pues todo integrante del equipo es responsable del software que se produce y por lo tanto todos deben tener conocimiento del rumbo que éste lleva.
- Simplicidad: Mantener un código simple y elegante que haga su trabajo no solo habla de una limpieza y educación como programadores, si no que, simplifica la tarea de depuración y mantenimiento del software.
- Retroalimentación: Mantener un contacto con las pruebas de cada módulo del software es una parte fundamental para no caer en los mismos errores.
- Respeto: La idea de este principio se concentra en el hecho de que cada módulo tiene que ser lo suficientemente independiente para que los cambios en éste no afecten el trabajo del resto del equipo.
- Coraje: El software cambia constantemente, así fue concebido, para adaptarse a nuevos requerimientos durante su construcción o hacer uso de nuevas herramientas en su desarrollo, tomando en cuenta el riesgo de tener que reconstruir[3]. Es una tarea que cada miembro del equipo, y en general, cada ingeniero de software tiene que realizar, pues parte de su trabajo es estar al día con las nuevas herramientas que facilitan el desarrollo de software.

- Crystal clear: es una metodología ágil de la familia Crystal, descritas por Alistair Cockburn. Generalmente el equipo de trabajo es de 6 u 8 personas, esta metodología pone mucho énfasis en las personas que serán los usuarios finales. Los elementos de esta metodología son:

  - Roles: los roles existentes en esta metodología son patrocinador, diseñador-programador principal, diseñador - programador y usuarios.

[3]El término 'reconstruir' (*refactoring* en inglés) fue introducido por W. F. Opdyke en su tesis doctoral. "Reconstruir es el proceso de cambiar un sistema de software [orientado a objetos] de tal manera que no se altere el comportamiento exterior del código, pero se mejore su estructura interna". M. Ross Sheldon, **Introducción a la estadística**, Reverte, 2007.

- Productos de trabajo: dentro de estos se encuentran el plan de revisiones, casos de uso, bocetos del diseño, probar el código del sistema, modelos de casos y manual de usuario.
- Políticas estándar: implementar cada dos o tres meses alguna prueba, participación del usuario directa, hay dos revisiones del usuario por iteración

- Adaptive Software Development (ASD):

  Es una metodología ágil creada por Jim Highsmith y Sam Bayer. Esta metodologáa remplaza a la metodologáa tradicional de cascada con ciclos de colaboración, retroalimentación y especulación (supone que algunas partes de la planeación están mal en algunos aspectos durante la definición de los requisitos), lo que permite ciclos de continuo aprendizaje y adaptación a los cambios en el sistema. Durante las iteraciones de esta metodología el aprendizaje que será transmitido a la siguiente iteración es recogido de los pequeños errores que ocurran, basados en los suposiciones falsas.

- Lean Software Development (LSD): fue descrita por Mary Poppendieck y Tom Poppendieck, esta metodología está basada en los principios Lean, los cuales son [64]:

  - Eliminar desperdicios: se trata de eliminar todas las funcionalidades, así como el código que no le aumentan un valor al sistema para el usuario, los cuales solo generan retrasos en el proceso de desarrollo del software.
  - Ampliar el aprendizaje: para esto se realizan las pruebas inmediatamente de que el código ha sido escrito, evitando así acumular defectos, el proceso de ampliar el aprendizaje se amplia realizando pequeñas iteraciones de desarrollo.
  - Decidir lo más tarde posible: se intenta retrasar las decisiones hasta que estas puedan basarse en hechos y no en suposiciones, como sucede con los requerimientos del software, así se dejan las desiciones para después, hasta que los usuarios hayan identificado mejor sus necesidades.
  - Reaccionar tan rápido como sea posible: se trata de entregar tan rápido como sea posible el producto generado en la iteración, para así recibir rapidamente los comentarios acerca de éste, evitando así que pase un largo periodo de tiempo y las necesidades de los usuarios también cambien.

- Posibilitar al equipo: en este principio se refiere a que los desarrolladores le indican a los directivos del proyecto las acciones que podrían tomarse, brindándoles así a los desarrolladores un poco de motivación necesaria para realizar las tareas que se les han asignado y los directivos del proyecto deben encargarse de que no decaiga el espíritu del equipo.
- Crear la integridad: se debe garantizar que los componentes separados del software funcionen bien juntos, logrando cumplir con la mantenibilidad, la eficiencia y un tiempo de respuesta óptimo, esto se logra con el buen entendimiento del problema a resolver.
- Ver el conjunto: para lograr esto se debe pensar en grande y realizar acciones en pequeño, equivocarse y aprender rápido de los errores, se trata de aplicar todos los principios de Lean al proceso de desarrollo.

Parte III

Desarrollo del sistema

Capítulo 8

Método de desarrollo

En los capítulos anteriores se mencionó la motivación de este trabajo, así como la parte teórica que está involucrada en el mismo, siendo ésta una parte fundamental en todo trabajo de investigación, sin embargo, es importante contar con un método bien fundamentado para llevar por buen camino el desarrollo del sistema. En este capítulo se presenta la metodología de desarrollo elegida, la jusitificación de por qué fue elegida y la forma de trabajar con ella.

8.1. Método

Para el desarrollo de GeoStads se decidió utilizar una metodología ágil, pues es importante realizar un buen trabajo pero además que en todo momento se esté conciente de que ocurrirán cambios y que debe desarrollarse en el menor tiempo posible con el mínimo de errores. No se utilizó una metodología en especifica pues todas estos métodos tienen una base homogénea la cual involucra las siguientes fases:

1. Planeación
2. Administración
3. Diseño
4. Codificación
5. Pruebas

8.2. Justificación

El por qué utilizar métodos ágiles en lugar de tradicionales radica en el hecho de que los métodos agiles responde rápidamente a los cambios que pueden surguir durante un proyecto, además estos métodos son incrementales, es decir, están basados en la modularización del código y la construcción de los mismos en pequeños y frecuentes avances guiados por pruebas, son rápidos y no se concentran en realizar una documentación exhaustiva del proyecto.

8.3. Estrategia

La estrategia que se utilizará consiste en analizar los requrimientos del sistema mediante el objetivo que se quiere alcanzar, definir una arquitectura de desarrollo, modularizar el código, definir tareas simples y concisas para la contrucción del sistema.

Una vez realizado esto, el trabajo se divide en etapas, en cada una de ellas se define una tarea a realizar, se escribe el código correspondiente a esta, se realizan las pruebas y de ser necesario se corregen errores.

Capítulo 9

Diseño e Implementación del Sistema

Los procesos de diseño e implementación del sistema se presentaran a lo largo de este capítulo. En primer término, se describen los requrimientos del sistema, en segundo las plataformas, componentes y mecanismos de comunicación que conforman al sistema. Se muestran las herramientas y tecnologías utilizadas, su organización estructural y funcional, y la especificación detallada de los componentes involucrados. De la misma forma, se presentan los prototipos de la interfaz gráfica de usuario donde se establecen los controles y mecanismos interactivos para poner el conjunto de herramientas implementadas en GeoStads. Por último se describen los pasos para el desarrollo del sistema.

9.1. Análisis de requerimientos

9.1.1. Objetivo

Como se mencionó anteriormente, los Sistema de Información Geográficos tienden a ser robustos, caros, centrados en el despligue de información en lugar al análisis de la misma y requieren una gran cantidad de tiempo para entender su funcionamiento. Por tanto, el objetivo del sistema es reunir un conjunto de herramientas para llevar a cabo análisis de información espacial, por lo cual deberá contar con una interfaz simple para lograr obtener un alto grado de usabilidad, que no requiera horas de trabajo para entender su funcionalidad y que sea de fácil acceso.

9.1.2. Requerimientos funcionales

Tomando en cuenta las limitaciones y caracteristicas de un SIG, se definieron los siguientes requerimientos funcionales del sistema:

1. Cálculo de datos estadísticos espaciales: Ya que el objetivo del sistema es realizar análisis de información mediante estadística espacial, es de vital importancia que el sistema cuente con métodos para llevar a cabo operaciones espaciales sobre un conjunto de datos. Las estadísticas a calcular son: media espacial, vecino mas cercano, autocorrelación espacial (Coeficiente de Moran) y semivariograma.

2. Modelo de representación: El sistema debe proveer una representación gráfica del mapa con el que se esté trabajando, internamente consistira de un mapa abstracto para almacenar los datos procesados y de manera visual, se desplegara un mapa.

3. Acceso visual: El sistema deberá permitir la carga de información desde una base de datos y desplegarlos en pantalla. además deberá hacer uso de capas para permitir el filtrado de los datos y visualizar los cálculos realizados.

9.1.3. Requerimientos no funcionales

Para asegurar que el sistema desarrollado en este trabajo sea compatible con el ambiente de operación donde será implantado se deberán satisfacer los siguientes requerimientos no funcionales.

1. Herramientas: La implementación del sistema hará uso en todo momento de herramientas de tipo Software libre.

2. Diseño: El diseño del sistema deberá permitir un facil aprendizaje, así mismo, deberá permitir su extensión en caso de añadir nuevos módulos sin que esto signifique la modificación del sistema en su totalidad.

3. Plataforma de Software: El sistema deberá desarrollarse sobre plataformas que permitan su operación y no compliquen el uso de esté, por ejemplo, el sistema estará instalado en un sistema operativo GNU/Linux para facilitar el uso de herramientas de desarrollo y depuración.

9.2. Diseño y arquitectura del sistema

En esta sección se especifican los detalles del sistema, como son la plataforma de desarrollo, sus componentes y la comunicación que existen entre estos.

9.2.1. Arquitectura

El sistema GeoStads se desarrollará en lenguaje de programación Ruby usando como marco de trabajo Rails. Ruby es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su elegante sintaxis se siente natural al leerla y fácil al escribirla. [69].

Ruby es un lenguaje de programación que comenzó a tener auge a partir de 1995, sin embargo fue hasta el 2006 que empezó a tener un crecimiento masivo[1]. Es un lenguaje orientado a objetos con una sintaxis simple.

También se uitilizará el marco de trabajo[2] Rails. Rails es una herramienta para el desrrollo de aplicaciones web, es de código abierto y al igual que Ruby es relativamtente sencillo de entender y utilizar [70]. La principal razón de utilizar Rails, además de las ya mencionadas, es que hace uso del patron modelo-vista-controlador[3], con el cual podemos separar de manera más adecuda el sistema, permitiendo modularizarlo. Incluye un gran número de clases que permiten al programador enfocarse en la funcionalidad y el objetivo principal del sistema y no en desarrollar el núcleo de cada aplicación.

Como gestor de bases de datos se utilizara MySQL debido a que es un software libre, es un gestor relacional y cuenta con soporte espacial ya incluido, a diferencia de otros sistemas manejadores de bases de datos en que se necesita instalar por separado el soporte espacial, en muchas de estas extensiones hay errores debido a las plataformas sobre las cuales se este instalado el gestor principal.

Para la interfaz, se utilizará HTML, CSS y Javascript usando la herramienta JQuery quer permite desarrollar interfaces de usario en ambiente web de manera sencilla y tener acceso de manera mas sencilla a los elementos DOM[4]

[1] El índice TIOBE lo ubica en el lugar 12. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (07/10/2011 14:00 pm)

[2] En inglés *Framework*

[3] *Model-View-Controller* (MVC, por sus iniciales en inglés) Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

[4] *Document Object Model* (DOM por sus inicales en inglés).

evitando de esta manera escribir todo el código en javascript puro que a pesar de tener la misma funcionalidad solo lograría tener un código mas grande.

9.2.2. Diseño de la base de datos

En la figura 9.1 se muestra el esquema de una base de datos para ejemplificar el uso del sistema. La base de datos consta de dos tablas totalmente ajenas, esto debido a que la naturaleza de los datos es completamente diferente. Las tablas no necesitan de una normalización puesto que cada punto es totalmente ajeno e independiente entre sí es decir, cada tupla de la tabla Inegi no comporte datos con ninguna otra tupla de la tabla, lo mismo ocurre para la tabla de Conabio.

El mantener una estructura simple de la base de datos, utilizando unicamente tablas sin relaciones responde a la necesidad de mantener un sistema ligero y homogéneo, es decir, al no tener relaciones entre las tablas, no hay necesidad de realizar consultas tales como una unión y de está manera se mantienen un acceso mas sencillo y la complejidad de las consultas es mínima.

La característica notable que tienen las tablas es que ambas poseen un atributo espacial, un punto y gracias a este atributo es que se puede llevar acabo el análisis espacial de los datos.

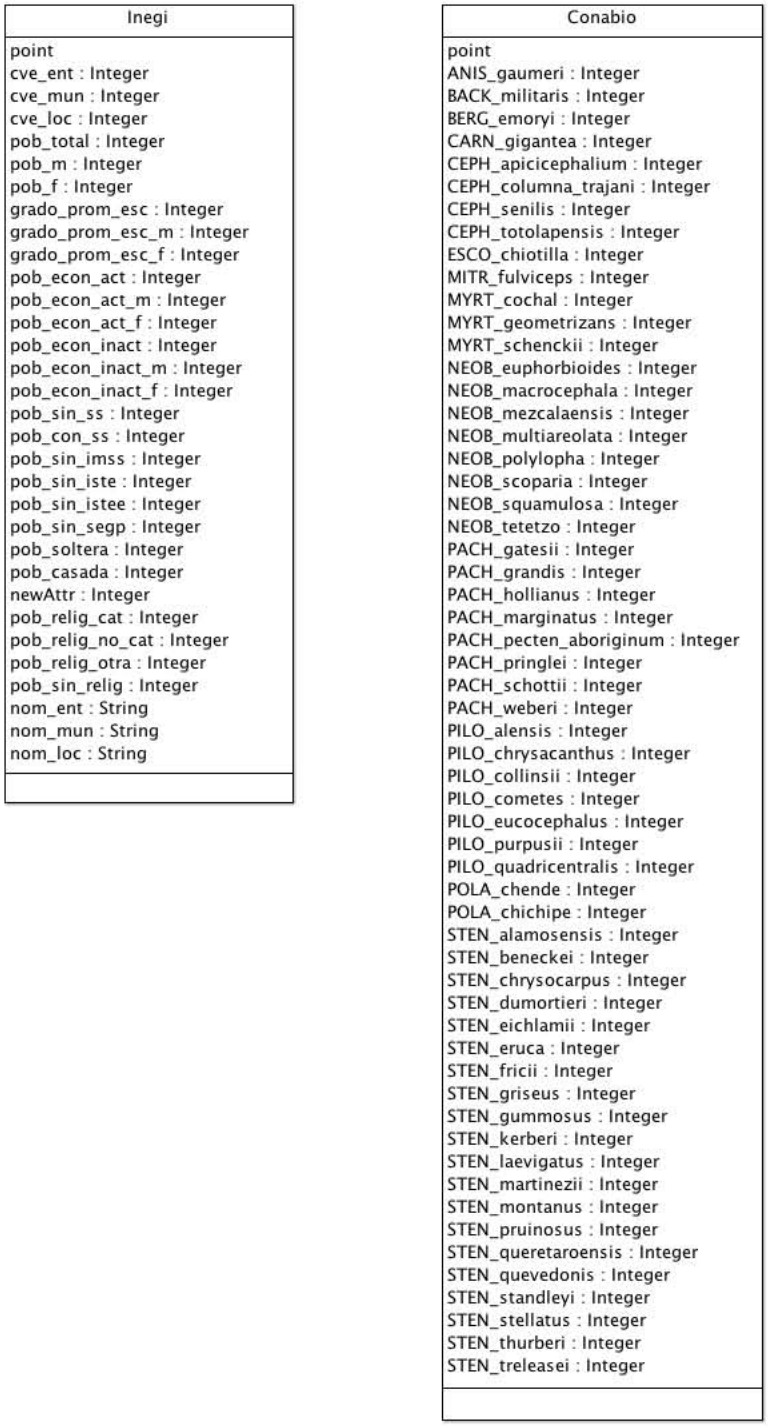


Figura 9.1: Esquema de la base de datos ejemplo utilizada por GepStads

La tabla Inegi fue tomada de los Principales resultados por localidad (ITER) del Instituto Nacional de Estadística y Geografía (INEGI), que consiste de un conjunto de indicadores de población y vivienda a nivel localidad de todo el pais, estos datos provienen del Censo de Población y Vivienda 2010. Este censo consta de 190 indicadores para el censo de todo el país, para el sistema GeoStads se tomó un subconjunto de éste, teniendo en total 24 indicadores y solo las localidades que conforma el Distrito Federal. Estos indicadores representan la cantidad de población en una localidad que cumple con el atributo, por ejemplo, se tiene el indicador poblacion total (pob_total) que para la localidad con nombre Puerto las cruces de la delegación Cuajimalpa de Morelos tiene un valor de 444 habitantes. Ademas se tienen una columna espacial que contiene las coordenadas geogáficas de la ubicación del centroide de la localidad.

En Cuadro 9.1 a 9.3 se muestra la descripción de la tabla Inegi.

Variable	Tipo	Descripción
cve_ent	numérico	Código que identifica a la entidad federativa.
nom_ent	numérico	Nombre oficial de la entidad federativa.
cve_mun	numérico	Código que identifica al municipio al interior de una entidad federativa, conforme al Marco Clave de municipio ó delegación Geoestadístico Nacional.
nom_mun	numérico	Nombre oficial del municipio o delegación ó delegación política en el caso del Distrito Federal.
cve_loc	numérico	Código que identifica a la localidad, al interior de cada municipio (o delegación política) conforme al Marco Geoestadístico Nacional.
nom_loc	numérico	Nombre con el que se reconoce a la localidad dado por la ley o la costumbre.
point	numérico	
pob_total	numérico	Total de personas que residen habitualmente en el país, entidad federativa, municipio y localidad. Incluye la estimación del número de personas en viviendas particulares sin información de ocupantes.
pob_m	numérico	Total de mujeres que residen habitualmente en el país, entidad federativa, municipio y localidad.
pob_f	numérico	Total de hombres que residen habitualmente en el país, entidad federativa, municipio y localidad.
grado_prom_esc	numérico	Resultado de dividir el monto de grados escolares aprobados por las personas de 15 a 130 años de edad entre las personas del mismo grupo de edad.
grado_prom_esc_m	numérico	Resultado de dividir el monto de grados escolares aprobados por los hombres de 15 a 130 años de edad entre los hombres del mismo grupo de edad.
grado_prom_esc_f	numérico	Resultado de dividir el monto de grados escolares aprobados por las mujeres de 15 a 130 años de edad entre los mujeres del mismo grupo de edad.
pob_econ_act	numérico	Personas de 12 años y más que trabajaron; tenían trabajo pero no trabajaron o; buscaron trabajo en la semana de referencia.

Cuadro 9.1: Descripción de la tabla Inegi.

pob_econ_act_m	numérico	Hombres de 12 años y más que trabajaron; tenían trabajo pero no trabajaron o; buscaron trabajo en la semana de referencia.
pob_econ_act_f	numérico	Mujeres de 12 años y más que trabajaron; tenían trabajo pero no trabajaron o; buscaron trabajo en la semana de referencia.
pob_econ_inact	numérico	Personas de 12 años y más pensionadas o jubiladas, estudiantes, dedicadas a los quehaceres del hogar, que tienen alguna limitación física o mental permanente que le impide trabajar.
pob_econ_inact_m	numérico	Hombres de 12 años y más pensionados o jubilados, estudiantes, dedicados a los quehaceres del hogar, que tienen alguna limitación física o mental permanente que le impide trabajar.
pob_econ_inact_f	numérico	Mujeres de 12 años y más pensionados o jubilados, estudiantes, dedicados a los quehaceres del hogar, que tienen alguna limitación física o mental permanente que le impide trabajar.
pob_sin_ss	numérico	Total de personas que no tienen derecho a recibir servicios médicos en ninguna institución pública o privada.
pob_ss	numérico	Total de personas que tienen derecho a recibir Población derechohabiente del servicios médicos en el Instituto Mexicano del Seguro Social (IMSS).
pob_imss	numérico	Total de personas que tienen derecho a recibir servicios médicos en el Instituto Mexicano del Seguro Social (IMSS).
pob_iste	numérico	Total de personas que tienen derecho a recibir servicios médicos en el Instituto de Seguridad y Servicios Sociales de los Trabajadores del Estado.

Cuadro 9.2: Descripción de la tabla Inegi (continuación).

pob_istee	numérico	Total de personas que tienen derecho a recibir servicios médicos en los institutos de seguridad social de los estados (ISSSET, ISSSEMyM, ISSSTEZAC, ISSSPEA o ISSSTESON).
pob_segp	numérico	Total de personas que tienen derecho a recibir servicios médicos en la Secretaría de Salud, mediante el Sistema de Protección en Salud (Seguro Popular).
pob_soltera	numérico	Personas solteras de 12 a 130 años de edad.
pob_casada	numérico	Personas de 12 a 130 años de edad que viven con su pareja en unión libre; casadas solo por el civil; casadas solo religiosamente o; casadas por el civil y religiosamente.
pob_relig_cat	numérico	Personas con religión católica.
pob_relig_no_cat	numérico	Personas con religiones Protestantes Históricas, Pentecostales, Neopentecostales, Iglesia del Dios Vivo, Columna y Apoyo de la Verdad, la Luz del Mundo, Cristianas, Evangélicas y Bíblicas diferentes de las Evangélicas.
pob_relig_otra	numérico	Personas con religiones de Origen oriental, Judaico, Islámico, New Age, Escuelas esotéricas, Raíces étnicas, Espiritualistas, Ortodoxos.
pob_sin_relig	numérico	Personas sin adscripción religiosa. Incluye ateísmo.

Cuadro 9.3: Descripción de la tabla Inegi (continuación).

La tabla Conabio fue tomada de la Comisión Nacional para el Conocimiento y uso de la Biodiversidad (CONABIO) del proyecto G003 (Patrones biogeográficos de las cacatáceas columnares de México), éste proyecto fue desarrollado con el objetivo del análisis de la dsitribución de la riqueza de especies de las cactáceas columnares de México. Consta de 58 columnas y cada una de ellas representa una especia de cacataceas columnares de México. Los valores de cada atributo puede tomar uno de dos posibles; 1 si en el punto de

recolección se encontro la especie X, 0 en otro caso por ejemplo para el punto (-95.5 15.5) el atributo ANIS_gaumeri es 0 pues no se encontró existencia de dicha especia en esa región y para el atributo PACH_pecten_aboriginum el valor es 1.

En el Cuadro 9.4 a 9.5 se muestra la descripción de la tabla Conabio.

Variable	Tipo	Descripción
ANIS_gaumeri	numérico	Anisocereus gaumeri
BACK_militaris	numérico	Backebergia militaris
BERG_emoryi	numérico	Bergerocactus emoryi
CARN_gigantea	numérico	Carnegia gigantea
CEPH_apicicephalium	numérico	Cephalocereus apicicephalium
CEPH_columna-trajani	numérico	Cephalocereus columna-trajani
CEPH_senilis	numérico	Cephalocereus senilis
CEPH_totolapensis	numérico	Cephalocereus totolapensis
ESCO_chiotilla	numérico	Escontria chiotilla
MITR_fulviceps	numérico	Mitrocereus fulviceps
MYRT_cochal	numérico	Myrtillocactus cochal
MYRT_geometrizans	numérico	Myrtillocactus geometrizans
MYRT_schenckii	numérico	Myrtillocactus schenckii
NEOB_euphorbioides	numérico	Neobuxbaumia euphorbioides
NEOB_macrocephala	numérico	Neobuxbaumia macrocephala
NEOB_mezcalaensis	numérico	Neobuxbaumia mezcalaensis
NEOB_multiareolata	numérico	Neobuxbaumia multiareolata
NEOB_polylopha	numérico	Neobuxbaumia polylopha
NEOB_scoparia	numérico	Neobuxbaumia scoparia
NEOB_squamulosa	numérico	Neobuxbaumia squamulosa
NEOB_tetetzo	numérico	Neobuxbaumia tetetzo
PACH_gatesii	numérico	Pachycereus gatesii
PACH_grandis	numérico	Pachycereus grandis
PACH_hollianus	numérico	Pachycereus hollianus
PACH_marginatus	numérico	Pachycereus marginatus
PACH_pecten-aboriginum	numérico	Pachycereus pecten-aboriginum
PACH_pringlei	numérico	Pachycereus pringlei
PACH_schottii	numérico	Pachycereus schottii
PACH_weberi	numérico	Pachycereus weberi
PILO_alensis	numérico	Pilosocereus alensis
PILO_chrysacanthus	numérico	Pilosocereus chrysacanthus
PILO_collinsii	numérico	Pilosocereus collinsii
PILO_cometes	numérico	Pilosocereus cometes
PILO_eucocephalus	numérico	Pilosocereus leucocephalus
PILO_purpusii	numérico	Pilosocereus purpusii
PILO_quadricentralis	numérico	Pilosocereus quadricentralis
POLA_chende	numérico	Polaskia chende
POLA_chichipe	numérico	Polaskia chichipe

Cuadro 9.4: Descripción de la tabla Conabio.

Variable	Tipo	Descripción
STEN_alamosensis	numérico	Stenocereus alamosensis
STEN_beneckei	numérico	Stenocereus beneckei
SSTEN_chrysocarpus	numérico	Stenocereus chrysocarpus
STEN_dumortieri	numérico	Stenocereus dumortieri
STEN_eichlamii	numérico	Stenocereus eichlamii
STEN_eruca	numérico	Stenocereus eruca
STEN_fricii	numérico	Stenocereus fricii
STEN_griseus	numérico	Stenocereus griseus
STEN_gummosus	numérico	Stenocereus gummosus
STEN_kerberi	numérico	Stenocereus kerberi
STEN_laevigatus	numérico	Stenocereus laevigatus
STEN_martinezii	numérico	Stenocereus martinezii
STEN_montanus	numérico	Stenocereus montanus
STEN_pruinosus	numérico	Stenocereus pruinosus
STEN_queretaroensis	numérico	Stenocereus queretaroensis
STEN_quevedonis	numérico	Stenocereus quevedonis
STEN_standleyi	numérico	Stenocereus standleyi
STEN_stellatus	numérico	Stenocereus stellatus
STEN_thurberi	numérico	Stenocereus thurberi
STEN_treleasei	numérico	Stenocereus treleasei

Cuadro 9.5: Descripción de la tabla Conabio (continuación).

9.2.3. Diseño general del sistema

Para obtener un sistema cuyos componentes contaran con una estructura que permita su extensión y delimitación de responsabilidades, se decidió seguir los patrones de diseño Modelo-Vista-Controlador.

El patrón Modelo-Vista-Controlador[5] divide una aplicación en tres capas [72]:

- Las vistas despliegan la información al usuario. Son los mecanismos de interacción y representación de los datos, la interfaz gráfica de GeoStads pertence a éste bloque, cuando los usuarios solicitan alguna operación al sistema que necesita comunicarse con el servidor, la vista envía el evento correspondiente al controlador para su procesamiento. Una

[5]Model-View-Controller (MVS por sus iniciales en inglés)

vez que el controlador devuelve una respuesta, las vistas se encargan de realizar la visualización adecuada.

- Los Controladores manejan las intrucciones del usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz de usuario y el modelo. Este componente se encarga de procesar las solicitudes de las vistas principalmente solicitando los datos a los modelos o regresando una respuesta realizada por él mismo. Por lo general se encargan de devolver el formato adecuado para que la vista se encargue de desplegar el resultado.

- Los Modelos contienen las funcionalidades y los datos centrales. Es el componente responsable de realizar el procesamiento de información, un modelo generalmente representa una entidad en la base de datos, aunque no es obligatorio usar una. Cuando el usuario solicita realizar alguna operación de estadística espacial sobre un conjunto de datos, se le solicita al modelo que representa dicho conjunto que realize las operaciones necesarias, estas son devueltas al controlador el cual le da el formato solicitado (HTML, Json, XML, etc) y lo entrega a las vistas.

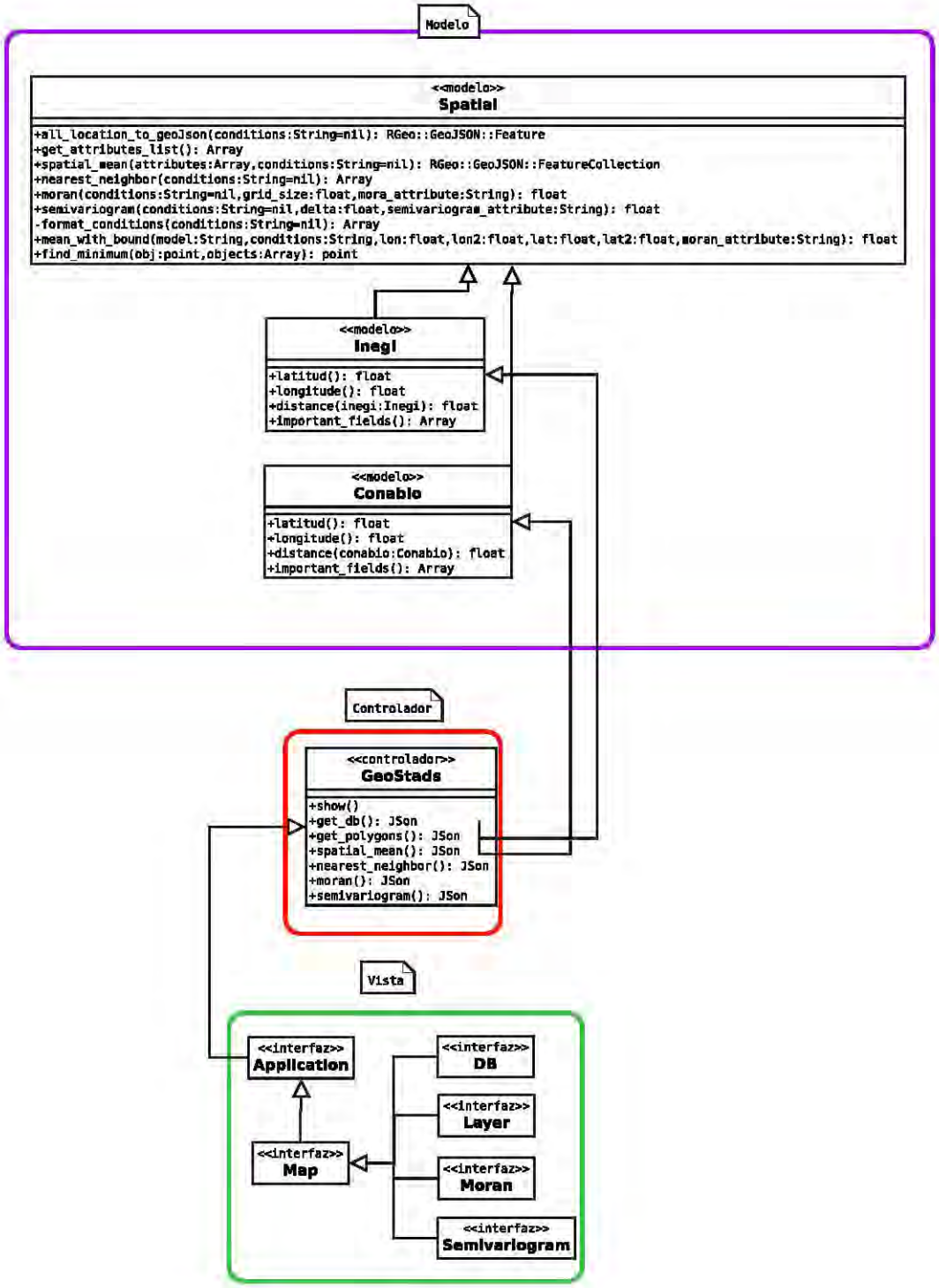


Figura 9.2: Diagrama MVC de GeoStads.

En la Figura 9.2 se muestra el diagrama de clases que conforman GeoStads distribuidos según el patron Modelo-Vista-Controlador. En el cuadro azul están las clases que conforman los tres modelos utilizados en el sistema para llevar a cabo las pruebas: Inegi y Conabio. En el esquema se observa que las tres clases de modelo tienen la misma funcionalidad, esto es por que hacen uso del módulo Spatial, un módulo en Rails es similar a una clase, define métodos y atributos, la diferencia está en que un módulo no puede ser instanciado, es decir, define comportamiento y estado para ser utilizado por alguna clase, es similar a una interfaz cuando se quiere realizar un comportamiento génerico pero no se desea utilizar herencia. Spatial contiene la implementación genérica de los métodos para calcular las estadísticas, los modelos hacen uso de ésta solo enviando los parametros necesarios para el cálculo.

Es importante hacer notar que el hecho de que los tres modelos tengan la misma funcionalidad y se encuentren separados en distintas clases se debe a que Rails está diseñado para mapear un modelo (una clase) a una entidad en la base de datos, de esta manera podemos acceder directamente a los datos de las tablas Inegi, Conabio y Salud con su respectivo modelo. Una de las ventajas de utilizar un marco de trabajo como Rails es no tener que realizar conexiones de manera manual con la base de datos, es decir, solo se necesita definir el esquema, la conexión y definir los modelos que accederan a la base de datos. Para realizar consultas, Rails define una interfaz en la cual los modelos utilizan métodos ya definidos para tener acceso a la tabla correspondiente, de manera que no es necesario ingresar SQL en el código del programa, por ejemplo, para obtener todas las tuplas de la tabla Inegi, tan solo es necesario escribir una línea como la siguiente:

Inegi.all

y Rails genera de manera automática el SQL correspondiente:

SELECT 'inegis'.* FROM 'Inegis'

de la misma manera Rails mapea los datos generados por la consulta a un objeto, permitiendos acceder a los atributos de la tubla como si se trataran de atributos de una clase en nuestro lenguaje de programación orientado a objetos favorito.

En rojo se muestra el único controlador, GeoStads, éste coordina las peticiones que realiza la interfaz y los resultados arrojados por los modelos.

Rails mapea automaticamente todas las rutas de la aplicación, tan solo hay que indicar el nombre y el controlador a donde se mapean las rutas, en el Algoritmo 9.1 se muestran las rutas definidas para el sistema, estas están contenidas en el archivo Routes.rb.

Algoritmo 9.1 Mapeo de rutas definidas para el sistema.

```
1 GeoStads::Application.routes.draw do
2   resources :geo_stads, :only => [:show]
3   match "get_db" => "geo_stads#get_db"
4   match "get_spatial_mean" => "geo_stads#spatial_mean"
5   match "get_nearest_neighbor" => "geo_stads#nearest_neighbor"
6   match "get_moran" => "geo_stads#moran"
7   match "get_semivariogram" => "geo_stads#semivariogram"
8   root :to => "geo_stads#show"
9 end
```

Este es el módulo de Rails que mapea las rutas de la aplicación, en la línea 2 con el método resources se indica que para el controlador geo_stads solo debe añadir la vista show, en las restantes lineas 3 a 7 se utiliza el método match para generar rutas mas personalizadas, todas son mapeadas al controlador geo_stads al método respectivo, todas estas rutas son del tipo GET pues todas devuelven una respuesta al cliente de la aplicación. De la misma manera que en el caso de las consultas, Rails ahorra todo este trabajo generando y mapeando de maneras automatica las rutas, lo único que es necesario es especificar que queremos y adonde debe ir la petición.

Una vez establecidas las rutas del sistema, cuando una petición es recibida por el sistema, Rails se encarga de redirigir al controlador adecuado y éste devuelve una respuesta que se envía a la vista.

En el cuadro verde se encierran las clases que forman parte de las vistas, aunque en realidad hay más archivos que forma parte de las vistas, archivos html y css pero estos no están agrupados en clases pues solo son lenguajes que indican cómo se verán las cosas, en cambio las clases mostradas en la Figura 9.2 si definen comportamiento y son utilizadas para mantener un estado del programa, es decir, van almancenando los resultados que el controlador devuelve para poder visualizarlos. También están los archivos que controlan el comportamiento de la interfaz.

Para el desarrollo de GeoStads se hace uso de múltiples herramientas las cuales se resumen en el Cuadro 9.6 y 9.7, en la primera columna se indica la

herramienta utilizada, en la siguiente columna su descripciòn y en la tercera los beneficios que brinda al utilizarla en GeoStads.

Herramienta	Descripción	Uso
Ruby	Ruby en un lenguaje de programación orientado a objetos, interpratado con una sintaxis sencilla.	La razón principal para hacer uso de Ruby dentro del proyecto es que el marco de trabajo Rails está programado con este lenguaje. Sin embargo, también es importante hacer notar que es un lenguaje muy sencillo en sintaxis y por esta razón no se generan líneas extensas de código.
Rails	Es un marco de trabajo al estilo de Struts para Java, hecho para trabajar con Ruby, Rails es un potente motor para realizar aplicaciones web que antes podían haber tomado semanas en realizarse y con ayuda de Rails mejora enormemente el tiempo de desarrollo.	Dado que GeoStads es un sistema web que corre sobre Ruby, es fundamental utilizar un marco de trabajo que lo soporte como este. Rails es de los mas robustos y populares, además cuenta con una documentación extensa.
Haml	Es una librería desarrollada para Rails que permite combinar código HTML y Ruby en las vistas de manera elegante.	Mediante esta libreria se simplifica de manera sustancial la codificación de las vistas.
Rgeo	Es una librería de Ruby para trabajar con objetos geográficos basada en el estandar de la OGC.	El objetivo de GeoStads es trabajar con objetos geográficos tales como puntos, polígonos, etc, con ayuda de esta librería tan solo hay que indicar las coordenadas de cada una de estos y no en implementar la representación abstracta de los mismos, por otra parte la librería permite conectarnos con una base de datos espacial y hacer la transformación de tuplas en objetos de Ruby, permitiendo manipularlos de manera más simple.
GeojSON	Es un formato de codificación para representar objetos geográficos mediante el ya conocido formato jSON, define un conjunto de etiquetas para objetos tales como puntos, polígonos, etc.	Es importante que GeoStads utilize herramientas que sigan un estándar, de esta manera es más simple realizar la comunicación con otros elementos dentro del sistema y fuera, por ejemplo, la librería Polymaps utiliza GeojSON para colocar puntos en el mapa.

Cuadro 9.6: Librerías y herrameintas utilizadas en el desarrollo de GeoStads, todas ellas son de código abierto

Herramienta	Descripción	Uso
Rgeo-jSON	Es una librería de Ruby que permite realizar conversiones entre objetos geográficos creador por la librería Rgeo ya mencionada en cadenas de texto en formato GeojSON y viceversa.	Con ayuda de esta librería se puede realizar una conversión simple entre los datos procesados por los modelos para ser visualizados por las vistas.
JQuery	Es una librería escrita en JavaScript, permite crear interfaces de usuario en el esplorador web, además de proporcionar una interacción más simple con el DOM de la página web.	Se require que para la interacción del usuario con el sistema, éste cuente con una forma simple y sencilla de utilizarlo, al utilizar esta librería podemos asegurar una interfaz de usuario agradable, además de lograr un código JavaScript más limpio y legible.
Polymaps	Es una librería escrita en JavaScript para la realización de mapas dinámicos e interactivos.	Dado que GeoStads trabaja con datos geográficos es indispensable contar con un método adecuado para la visualización de estos, la forma de realizarlo es mostrarlos en pantalla sobre un mapa, el cual es proporcionado por la librería Polymaps. Otro factor importante es que la librería realiza de manera automática el correcto posicionamientos de los datos sobre la superficie del mapa.
jscharts	Es una libreria escrita en JavaScript que permite la creación y visualización de gráficas.	GeoStads cuenta con dos estadisticas que son visualizadas sobre una gráfica: Autocorrelación espacial y Semivariograma. La principa ventaja de utilizar jscharts es que ajusta de manera automática los rangos de los ejes de la gráfica y la colocación de los valores sobre esta.
MySQL spatial	Es un SMBD relacional que cuenta con soporte de datos geográficos.	Existen en el mercado gran cantidad de software para el manejo de bases de datos, sin embargo, MySQL es el único que cuenta de manera gratuita y ya incluido el soporte espacial, es fácil de instalar, administrar y no requiere de módulos adicionales para dar soporte al uso de datos geográficos.

Cuadro 9.7: Librerías y herramientas utilizadas en el desarrollo de GeoStads, todas ellas son de código abierto (continuación).

El desarollo del sistema consiste en programar en el módilo Spatial los métodos que calcular las estadísticas, los modelos incluirán al módulo Spatial para hacer uso de los métodos. Tanto los métodos de los modelos Inegi, Co-

nabio y Salud se llaman igual que en el módulo Spatial para mantener una coherencia dentro del código. El controlador define los mismos métodos, la tarea del controlador consiste en definir que modelo es utilizado según la petición del usuario, también le dará el formato correspondiente a los resultados arrojados por los modelos.

9.3. Implementación

En esta sección se muestran las etapas de desarrollo del sistema GeoStads, en cada una de ellas se implementó un componente del sistema siguiendo las fases señaladas anteriormente.

9.3.1. Etapa 1: Diseño de la interfaz

Planeación

El obejtivo de esta etapa es tener un diseño previo de la interfaz de usuario sin ningún tipo de interacción con el controlador, la interfaz estará compuesta por un área de trabajo donde se mostrará un mapa y en él se visualizarán los cambios que se realizen. También contará con un menú organizado de la siguiente manera:

1. Archivo
   a) Leer BD: Permite cargar una base de datos en pantalla.
2. Ver
   a) Bases de datos: Muestra las tablas cargadas y sus atributos.
   b) Capas: Desgloza una lista de operaciones efectuadas con los datos para poder hacerlas visibles o no visibles.
   c) Moran: Muestra la gráfica generada por el método de autocorrealción espacial.
   d) Semivariograma: Muestra la gráfica generada por el método del semivariograma.
3. Herramientas
   a) Media espacial: Permite calcular la media espacial.

b) Vecino más cercano: Permite calcular el vecino más cercano.

c) Moran: Permite calcular la autocorrelación espacial con el método del coeficiente de moran.

d) Semivariograma: Permite calcular el semivariograma.

e) Filtros: Permite filtrar una tabla de acuerdo a un atributo y la cantidad deseada.

Administración

En esta etapa se hará uso de HTML, CSS y JavaScript y de las librerías señaladas para el diseño de la interfaz: JQuery, Polymaps, Jscharts y GeoJSON.

Diseño

En la Figura 9.3 se muestra el diseño de la interfaz, este es un prototipo realizado en un editor de imágenes por lo que no cuenta con funcionalidad alguna:

Figura 9.3: Diseño de la interfaz de usuario de GeoStads.

Codificación

GeoStads consta de una única vista que muestra la interfaz de usuario. En la Algoritmo 9.2 se muestran fragmentos del código de esta. El resto de la interfaz está desarrollado con JQuery, esto con el fin de desarrollar menús dinámicos que sean ricos en el aspecto visual y sencillos de codificar.

Algoritmo 9.2 Fragmentos de la interfaz de usuario.

```
1  /El siguiente fragmento de codigo permite mostrar en pantalla el menú
       principal.
2  #menu_tabs
3    %ul
4      %li
5        %a{ :id=> "file_a", :href => "#file"} Archivo
6      %li
7        %a{ :id=> "see_a",:href => "#see"} Ver
8      %li
9        %a{ :id=> "tools_a",:href => "#tools"} Herramientas
10 /El siguiente fragmento de codigo permite crear dos de los submenus
11   #file
12     %ol{:class => "selectable"}
13       %li{:id => "load_db",:class=>"ui-widget-content"} Cargar BD
14   #tools
15     %ol{:class => "selectable"}
16     %li{:class=>"ui-widget-content"} Media espacial
17     %li{:class=>"ui-widget-content"} Vecino mas cercano
18     %li{:class=>"ui-widget-content"} Moran
19     %li{:class=>"ui-widget-content"} Semivariograma
20     %li{:class=>"ui-widget-content"} Filtros
21 /En esta etiqueta  es donde se muestra el mapa.
22 #map
23 /Con el sigueinte fragmento de código indicamos el contenido para la
       ventana que selecciona una base de datos.
24 #load_db_dialog
25   %table{:border=>"0",:id => "select_db_table"}
26     %tr
27       %td
28         %button{:class =>'button_to_load_db', :database=>'inegi'}
               INEGI
29     %tr
30       %td
31         %button{:class =>'button_to_load_db', :database=>'conabio'}
               Conabio
32     %tr
33       %td
34         %button{:class =>'button_to_load_db', :database=>'salud'}
               Salud
```

Para poder mostrar finalmente los menús interactivos se utiliza JQuery para crear los eventos y mostrarlos. En el Algoritmo 9.3 se muestran fragmentos del código javaScript para crear la interfaz de usuario.

Algoritmo 9.3 Código JavaScript para el despliegue de la interfaz

```
1  /**
2  Con esta función se le indica a la interfaz mediante jQuery que el
       elemento que tenga como id file y dentro un ol tendra la propiedad
        de ser seleccionable. En el caso 0 del switch se indica que si
       selecciona la primera opcion del menú principal se debe generar
       una ventana flotante con título correspondiente, se bloquea el
       resto de la interfaz al colocar modal : true e indicamos que el
       tamaño de la ventan no puede ser modificado.
3  ** /
4  $( "#file ol" ).selectable({selected: function(event,ui){
5    var d = $( ".ui-selected", this ).index();
6    switch(d){
7      case 0:
8        $( "#load_db_dialog" ).css("visibility","visible");
9        $( "#load_db_dialog" ).dialog({title: "SELECCIONAR BASE DE DATOS
             ",modal: true,resizable: false});
10       break;
11 }}});
12 /**
13 El siguiente código crea la ventana para el càlculo de la media
       espacial.
14 **/
15 $("#spatial_mean_dialog").dialog({title:"MEDIA ESPACIAL",modal:true,
       resizable:false,buttons:{"Calcular": function(){spatial_mean();$(
       this).dialog("close");}}});
16 /**
17 La siguiente función realiza la llamada ajax al controlador para
       calcular la media espacial.
18 **/
19 function spatial_mean(){
20   var values="";
21   $("#spatial_mean_dialog #content div#spatial_mean_atributes ul input
         .spatial_mean_radio:checked").each(function(index) {
22   if(current_map.layers["spatial_mean_"+$(this).val()] == undefined){
23     values += "spatial_mean["+$(this).attr("name")+"]="+$(this).attr("
           name")+"&";}});
24   $.ajax({
25     url: "get_spatial_mean",
26     dataType: "json",
27     data:values+current_map.get_layer($("#spatial_mean_dialog #content
           div#spatial_mean_layers input[name=
         spatial_mean_layer_checkbox]:checked").attr("id")).query,
28     type: "GET",
29 success: function(resp){respond_to_spatial_mean(resp,values);}});}
```

Algoritmo 9.4 Código JavaScript para el despliegue de la interfaz (continuación)

```
1  /  *
2  La siguiente función permite visualizar en pantalla el resultado de
       calcular la media espacial devuelte por el controlador.
3  *  /
4  function respond_to_spatial_mean(resp){
5    var layer;
6    $.each(resp.features,function(index,feature){
7      layer=current_map.add_marker([feature],"layer_"+feature.properties
           .id,feature.properties.description,feature.properties.
           description,"X");
8
9  $("#layers_dialog #others table").append("<tr><td><div class='
       layer_title'>"+feature.properties.description.toUpperCase()+"</div
       ></td><td><div class='layer layer_spatial_mean'><input type='
       checkbox' CHECKED name='layer' class='layer_checkbox
       layer_spatial_mean' id='layer_"+feature.properties.id+"' /></div
       ></td></tr>");});};}
```

Para el resto de las operaciones se lleva a cabo un proceso similar con respectivas diferencias.

9.3.2. Etapa 2: Media espacial

Planeación

El objetivo de esta etapa es implementar el cálculo de la media espacial mediante un conjunto de datos y un atributo del modelo seleccionado utilizando la formula que se muestra en la Figura 9.4. Recordemos que el resultado es un promedio de los valores de las coordenadas de cada punto ponderado por el valor del atributo, posteriormente, el controlador devuelve el resultado a la vista y está debe mostrar en pantalla el resultado.

$$LON = \frac{\sum_{i=1}^{n} x_i * lon_i}{\sum_{i=1}^{n} x_i} \; LAT = \frac{\sum_{i=1}^{n} x_i * lat_i}{\sum_{i=1}^{n} x_i}$$

Figura 9.4: Fórmula para calcular la media espacial. Donde X_i es el valor del atributo del i-esimo dato, n el número de datos, lon_i el valor de la coordenanda en el eje X i-esimo dato y lat_i valor de la coordenada en el eje Y del i-esimo dato.

Administración

En esta etapa se desarrollan los componentes que forman parte de los modelos y del controlador, estos están programados en Ruby. En base al diagrama de clases se generan los siguientes componentes: las clases Spatial, Inegi, Conabio y GeoStadsController, en cada una de las clases se genera el método correspondiente a la media espacial.

Diseño

En la figura 9.5 se muestra el diagrama de secuencia para obtener la media espacial. Se asume que la base de datos ha sido cargada y seleccionada.

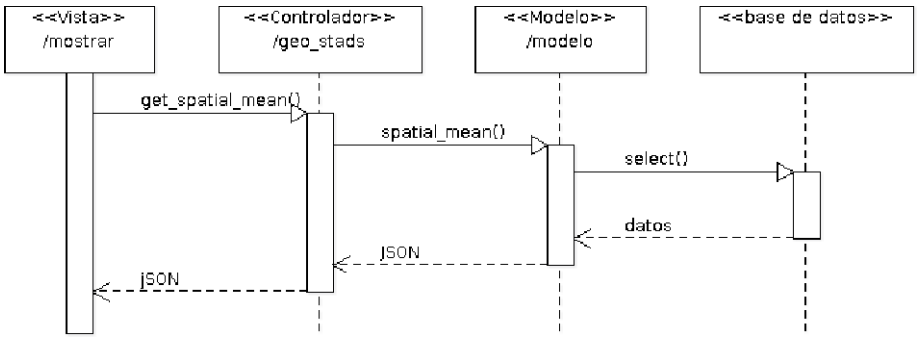


Figura 9.5: Diagrama de secuencia para obtener la media espacial

Codificación

En el Algoritmo 9.2 se muestra el método para calcular la media espacial en la clase Spatial. En la línea 7 evalua el modelo al que hay que llamar, el método eval toma una cadena de texto y ejecuta lo que contenga está, en este caso, la variable model puede ser “Inegi” o “Conabio”, en este caso, el método eval devulve la clase correspondiente y ejecuta el método where.

En las líneas 10 a 21, se cálcula la media espacial de cada atributo, en la línea 20 se crea un objeto geográfico de tipo Punto, esta conversión se realiza por medio de la libreria RGeo, finalmente el algoritmo crea una colección de objetos geográficos en al línea 22 y esta es devuelta como respuesta.

Algoritmo 9.5 Código para el cáculo de la media espacial en el módulo Spatial.

```
1  #Calcula la media espacial de cada atributo en el arreglo attributes
2  #attributes Es un array de atributos de un modelo
3  #conditions Cadena de texto que contiene condiciones para hacer
       filtrado de datos
4  #model Indica el modelo en el cual se efectuará la operación.
5  def self.spatial_mean(attributes,conditions=nil, model)
6    conditions = !conditions.nil? ? [format_conditions(conditions),
         conditions] : nil
7    objects = eval(model).where(conditions)
8    entity_factory = ::RGeo::GeoJSON::EntityFactory.instance
9    features = []
10   attributes.each_key do |attr|
11     spatial_mean_lat=0
12     spatial_mean_lon=0
13     sum=0;
14     objects.map{ |object|
15       spatial_mean_lat += object[attr]*object.latitude
16       spatial_mean_lon += object[attr]*object.longitude
17       sum+=object[attr]}
18     spatial_mean_lat /= sum
19     spatial_mean_lon /= sum
20     features.push(entity_factory.feature(RGeo::Geographic.
           spherical_factory(:srid => 4326).point(spatial_mean_lon,
           spatial_mean_lat),            "spatial_mean_"+attr,{:class => "
           spatial_mean_"+attr,:html_color => "yellow",:id => "
           spatial_mean_"+attr,:description => "Media espacial de la
           variable" + attr}))
21   end
22   entity_factory.feature_collection(features)
23 end
```

En el Algoritmo 9.6 se muestra la implementeción de los modelos, estas dos clases no realizan ninguna operación de estadística espacial, son necesarias para que Rails pueda transformar una tupla de la base de datos a un objeto, en las líneas 2 y 4 se indica al modelo que actúe como un objeto espacial y de esta manera poder acceder a los atributos espaciales, ya que las dos clases solo cuentan con un valor geométrico (punto) solo se indica la forma de acceder a este atributo. A continucaión se definen los métodos latitude y longitude estos solo son para ser mas legibles el código, el método distancia calcula la distancia entre dos objetos de la misma clase. Finalmente el método important_fields, devuelve los atributos para mostrar en pantalla.

Algoritmo 9.6 Código para las clases Inegi y Conabio respectivamente.

```
1  class Inegi   < ActiveRecord::Base
2    self.rgeo_factory_generator = RGeo::Geos.method(:factory)
3
4    set_rgeo_factory_for_column(:point, RGeo::Geographic.
         spherical_factory)
5
6    def latitude
7          self.point.y.to_f
8    end
9
10   def longitude
11     self.point.x.to_f
12   end
13
14   def distance(inegi)
15     point.distance(inegi.point)
16   end
17
18   def important_fields
19     {'cve_ent' => cve_ent, 'nom_ent' => nom_ent, 'cve_mun' =>
           cve_mun, 'nom_mun' => nom_mun, 'cve_loc' => cve_loc, '
           nom_loc' => nom_loc,          'id'=> id.to_s+"_inegi", '
           class'=>"inegi_db_marker", 'html_color' => "red"}
20   end
21 end
```

```
1  class Conabio < ActiveRecord::Base
2
3    self.rgeo_factory_generator = RGeo::Geos.method(:factory)
4
5    set_rgeo_factory_for_column(:point, RGeo::Geographic.
         spherical_factory)
6
7    def latitude
8      self.point.y.to_f
9    end
10
11   def longitude
12     self.point.x.to_f
13   end
14
15   def distance(conabio)
16     point.distance(conabio.point)
17   end
18
19   def important_fields
20     {'id'=> id.to_s+"_conabio", 'class'=> "conabio_db_marker", '
           html_color' => "blue"}
21   end
22 end
```

En el Algoritmo 9.7 se muestra la parte correspondiente al controlador, este método devuelve texto en formato geoJson, esto lo logramos utilizando la biblioteca RGeo::GeoJSON que toma objetos geométricos y los transforma a formato GeoJson. El método choose_database selecciona la base de datos de la cual queremos obtener la media espacial, por último el metodo eval, es definido por el núcleo de Ruby, recibe un texto y lo ejecuta, en éste caso recibe uno texto de dos posibilidades "Inegi" o "Conabio" o, es decir, evalua sobre que clase queremos ejecutar el método spatial_mean.

Algoritmo 9.7 Porción del código para el controlador.

```
1 def spatial_mean
2   respond_with(RGeo::GeoJSON.encode(Spatial.spatial_mean(params
        [:spatial_mean],params[:values],params[:database])))
3 end
```

Pruebas

Para verificar que las operaciones se están llevando de forma adecuada se definen las siguientes pruebas, estas pruebas fueron definidas antes de comenzar con la programación de los módulos, de esta manera se define de alguna manera cómo se tiene que comportar la clase que está siendo probada y en base a la prueba se programa la funcionalidad, en la primera prueba se verifica que el modelo Inegi esté accesando de manera correcta a la base de datos, en las siguientes tres pruebas solo se verifica que al calcular la media espacial de uno o dos atributos, la operación se esté llevando acabo y nos devuelva un objeto distinto a nulo. Finalmente en la cuarta prueba verificamos que en verdad se esté calculando la media espacial de una variable, esto sabemos que es correcto por que se calculó a mano el valor de esa media espacial. Las pruebas para las clases Conabio y Salud son análogas.

Algoritmo 9.8 Pruebas unitarias para la clase Inegi

```
1  require 'test_helper'
2  class InegiTest < ActiveSupport::TestCase
3
4    test "should find inegi" do
5      i = Inegi.first
6      assert !i.nil?, "Inegi element not found"
7    end
8
9    test "should return spatial mean of attribute pob_total" do
10     s_mean = Inegi.spatial_mean({"pob_total" => "pob_total"})
11     assert !s_mean.nil?, "spatial mean of attribute pob_total
           not found"
12   end
13
14   test "should return spatial mean of attribute pob_total with
         db filter" do
15    s_mean =  Inegi.spatial_mean({"pob_total" => "pob_total"}, {:
          pob_total=>10000, :pob_m=>0, :pob_f=>0, :grado_prom_esc
          =>0, :grado_prom_esc_m=>0, :grado_prom_esc_f=>0, :
          pob_econ_act=>0, :pob_econ_act_m=>500, :pob_econ_act_f=>0,
           :pob_econ_inact=>0, :pob_econ_inact_m=>0, :
          pob_econ_inact_f=>0, :pob_sin_ss=>0, :pob_con_ss=>0, :
          pob_sin_imss=>0, :pob_sin_iste=>0, :pob_sin_istee=>0, :
          pob_sin_segp=>0, :pob_soltera=>0, :pob_casada=>0, :
          pob_relig_cat=>0, :pob_relig_no_cat=>0, :pob_relig_otra
          =>0, :pob_sin_relig=>0})          assert !s_mean.nil?, "
          spatial mean of attribute pob_total not found"
16   end
17
18   test "should return spatial mean of attribute pob_total  and
         pob_m with db filter" do
19    s_mean =  Inegi.spatial_mean({"pob_total" => "pob_total", "
          pob_m" => "pob_m"}, {:pob_total=>10000, :pob_m=>0, :pob_f
          =>0, :grado_prom_esc=>0, :grado_prom_esc_m=>0, :
          grado_prom_esc_f=>0, :pob_econ_act=>0, :pob_econ_act_m
          =>500, :pob_econ_act_f=>0, :pob_econ_inact=>0, :
          pob_econ_inact_m=>0, :pob_econ_inact_f=>0, :pob_sin_ss=>0,
           :pob_con_ss=>0, :pob_sin_imss=>0, :pob_sin_iste=>0, :
          pob_sin_istee=>0, :pob_sin_segp=>0, :pob_soltera=>0, :
          pob_casada=>0, :pob_relig_cat=>0, :pob_relig_no_cat=>0, :
          pob_relig_otra=>0, :pob_sin_relig=>0})
20        assert !s_mean.nil?, "spatial mean of attribute pob_total
              and pob_m not found"
21   end
22
23   test "should calculete spatial mean of pob_total with values"
         do
24     s_mean = Inegi.spatial_mean({"pob_total" => "pob_total"})
25     x = s_mean[0].geometry.x
26     y = s_mean[0].geometry.y
27     assert x == -99.14058682673749 && y == 19.38314311671571, "
           Error on spatial mean of attribute pob_total"
28   end
29 end
```

En Figura 9.6 estan los resultados de las pruebas. Nos indica que de cuatro pruebas todas fueron correctas.

```
Started . Finished in 0.245194 seconds.
4 tests, 4 assertions, 0 failures, 0 errors, 0 skips
```

Figura 9.6: Resultados de las pruebas para la clase Inegi

9.3.3. Etapa 3: Vecino más cercano

Planeación

El objetivo de esta etapa es implementar el cálculo del vecino más cercano para cada punto del conjunto de puntos seleccionado mediante la fórmula de la Figura 9.7, la interfaz hace la petición al controlodor para recibir los datos en formato jSON, posteriormente este pedirá al modelo el conjunto de puntos que representan el vecino más cercano a cada punto del conjunto seleccionado, debe mostrarse en pantalla el resultado.

$$d = (\sum_{i=1}^{N} d_i)/N$$

Figura 9.7: Fórmula para calcular el promedio al vecino más cercano Donde N es el número de puntos, y d_i es la distancia al vecino más cercano para el punto i.

Administración

En está etapa se desarrollan los componentes que forman parte de los modelos y del controlador, estos están programados en Ruby. En las siguientes clases Spatial, Inegi, Conabio, Salud y GeoStadsController se añade el método correspondiente para el cálculo del vecino más cercano.

Diseño

En la figura 9.8 se muestra el diagrama de secuencia para obtener los vecinos más cercanos. Se asume que la base de datos ha sido cargada y seleccionada.

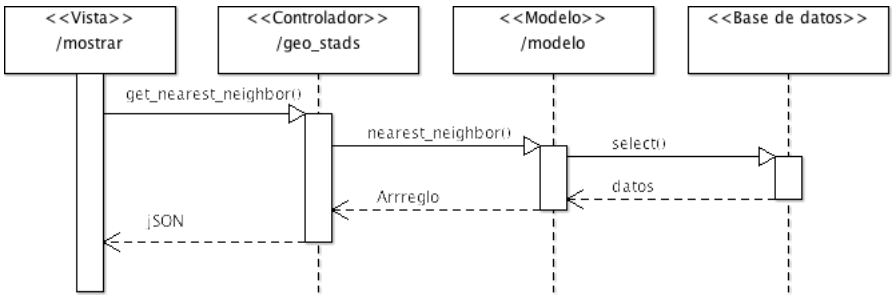


Figura 9.8: Diagrama de secuencia para obtener los vecinos más cercanos.

Codificación

En el Algoritmo 9.9 se muestra el método calcular vecino más cercano en la clase Spatial. En las líneas 10 a 14 el algoritmo calcula el vecino más cercano para cada uno de los puntos. En la línea 13 se crea un objeto geográfico de tipo Line_string, es decir, se genera una linea que consta de dos puntos. Finalmente en la línea 15 el algoritmo cálcula el promedio al vecino más cercano y crea una colección de objetos geográficos en al línea 16 y esta es devuelta como respuesta.

Algoritmo 9.9 Código para el cáculo del vecino más cercano en el módulo Spatial

```
1  #Calcula el vecino más cercano
2  #conditions Cadena de texto que contiene condicones para hacer
       filtrado de datos
3  #model Indica el modelo en el cual se efectuará la operación
4  def self.nearest_neighbor(model,conditions=nil)
5    conditions = !conditions.nil? ?  [format_conditions(conditions
         ),conditions] : nil
6    objects = eval(model).where(conditions)
7    entity_factory = ::RGeo::GeoJSON::EntityFactory.instance
8    features = []
9    prom_min=0;
10   objects.each do |object|
11    min = find_minimum(object,objects)
12    prom_min +=min[:dist]
13   features.push(entity_factory.feature(RGeo::Geographic.
         spherical_factory(:srid => 4326).line_string([min[:p_min].
         point,object.point]),          object.class.name.downcase+"
         _nearest_neighbor",{:distance =>min[:dist],:description =>
         "Vecino mas cercano de la tabla "+object.class.name.upcase
         ,:id_p1=>object.id.to_s+"_"+object.class.name.downcase,:
         id_p2=>min[:p_min].id.to_s+"_"+object.class.name.downcase
         }))
14   end
15   prom_min /= objects.size
16   [entity_factory.feature_collection(features),prom_min]
17 end
```

En el Algoritmo 9.10 se muestra la parte correspondiente al controlador, en la línea 2 se calcula el vecino más cercano según la base de datos seleccionada. En la línea 3 se regresa en formato jSON los marcadores que corresponden a los puntos mas carcano para cada punto de la base de datos y como dato auxiliar el promedio del vecino más cercano a cada punto.

Algoritmo 9.10 Porción del codigo para el controlador.

```
1 def nearest_neighbor
2   resp = Spatial.nearest_neighbor(params[:database],params[:
        values],)
3   respond_with("{\"markers\":"+RGeo::GeoJSON.encode(resp[0]).
        to_json+",\"prom_min\":"+resp[1].to_s+" }")
4 end
```

Pruebas

Para verificar que las operaciones se están llevando de forma adecuada se definen las siguientes pruebas: En la primera prueba se verifica que el método para calcular el vecino más cercano esté regresando una respuesta. En la segunda prueba se verifica que dada la condición de pob_total >= 10 000 regrese la cantidad correcta de puntos esto fue corroborado al realizar la consulta en la base de datos. En la última prueba se verifica el promedio del vecino más cercano tomando en cuenta solo a los 26 puntos anteriores, de nueva cuenta se realizó a mano el cálculo para corroborar los resultados.

Algoritmo 9.11 Pruebas unitarias para la clase Inegi

```
1  test "should return nearest neighbor" do
2    nb = Inegi.nearest_neighbor({:pob_total=>10000, :pob_m=>0, :
         pob_f=>0, :grado_prom_esc=>0, :grado_prom_esc_m=>0, :
         grado_prom_esc_f=>0, :pob_econ_act=>0, :pob_econ_act_m
         =>500, :pob_econ_act_f=>0, :pob_econ_inact=>0, :
         pob_econ_inact_m=>0, :pob_econ_inact_f=>0, :pob_sin_ss=>0,
         :pob_con_ss=>0, :pob_sin_imss=>0, :pob_sin_iste=>0, :
         pob_sin_istee=>0, :pob_sin_segp=>0, :pob_soltera=>0, :
         pob_casada=>0, :pob_relig_cat=>0, :pob_relig_no_cat=>0, :
         pob_relig_otra=>0, :pob_sin_relig=>0})
3    assert !nb.nil?, "nearest neighbor not found"
4  end
5
6  test "should return nearest neighbor" do
7    nb = Inegi.nearest_neighbor({:pob_total=>10000, :pob_m=>0, :
         pob_f=>0, :grado_prom_esc=>0, :grado_prom_esc_m=>0, :
         grado_prom_esc_f=>0, :pob_econ_act=>0, :pob_econ_act_m
         =>500, :pob_econ_act_f=>0, :pob_econ_inact=>0, :
         pob_econ_inact_m=>0, :pob_econ_inact_f=>0, :pob_sin_ss=>0,
         :pob_con_ss=>0, :pob_sin_imss=>0, :pob_sin_iste=>0, :
         pob_sin_istee=>0, :pob_sin_segp=>0, :pob_soltera=>0, :
         pob_casada=>0, :pob_relig_cat=>0, :pob_relig_no_cat=>0, :
         pob_relig_otra=>0, :pob_sin_relig=>0})
8    assert nb[0].size == 26 ?, "nearest neighbor not found"
9  end
10
11 test "should return nearest neighbor average" do
12   nb = Inegi.nearest_neighbor({:pob_total=>10000, :pob_m=>0, :
         pob_f=>0, :grado_prom_esc=>0, :grado_prom_esc_m=>0, :
         grado_prom_esc_f=>0, :pob_econ_act=>0, :pob_econ_act_m
         =>500, :pob_econ_act_f=>0, :pob_econ_inact=>0, :
         pob_econ_inact_m=>0, :pob_econ_inact_f=>0, :pob_sin_ss=>0,
         :pob_con_ss=>0, :pob_sin_imss=>0, :pob_sin_iste=>0, :
         pob_sin_istee=>0, :pob_sin_segp=>0, :pob_soltera=>0, :
         pob_casada=>0, :pob_relig_cat=>0, :pob_relig_no_cat=>0, :
         pob_relig_otra=>0, :pob_sin_relig=>0})
13   assert nb[1] == 2300.7392051088646 ?, "nearest neighbor
         average not found"
14 end
```

En Figura 9.9 estan los resultados de las pruebas. Nos indica que de las cuatro pruebas anteriores mas las nuevas tres todas fueron correctas.

```
Started . Finished in 0.245194 seconds.
7 tests, 7 assertions, 0 failures, 0 errors, 0 skips
```

Figura 9.9: Resultados de las pruebas para la clase Inegi

9.3.4. Etapa 5: Autocorrelación espacial: coeficiente de moran

Planeación

El objetivo del módulo es calcular la autocorrelación espacial de un conjunto de datos espaciales utilizando el coeficiente de moran con base en la fórmula de la Figura 9.10 . la Interfaz hará la petición al controlodar para recibir los datos en formato jSON. El resultado de la operación es un solo dato real, posteriormente el controlador pedirá al modelo el calculo del Coeficiente de Moran, el resultado se muestra en pantalla, el cual consiste en graficar cada vez que se solicite ejecutar la operación con un valor de la rejilla distinto.

$$MC = \frac{N}{\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}} . \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(X_i - X)(X_j - X)}{\sum_{i=1}^{N}(X_i - X)}$$

Figura 9.10: Fórmula para calcular el Coeficiente de Moran. Donde N es el número de datos, w_{ij} es el valor en la matriz de conectividad en la posicion (i, j), X_i es el valor del atributo del i-esimo dato, X es el promedio de los n valores con la variable X.

Administración

En esta etapa se desarrollan los componentes que forman parte de los modelos y del controlador, estos están programados en Ruby. En las siguientes clases Spatial, Inegi, Conabio, Salud y GeoStadsController se añade el método correspondiente para el cálculo de autocorrelación espacial.

Diseño

En la figura 9.11 se muestra el diagrama de secuencia para obtener el Coeficiente de Moran. Se asume que la base de datos ha sido cargada y seleccionada.

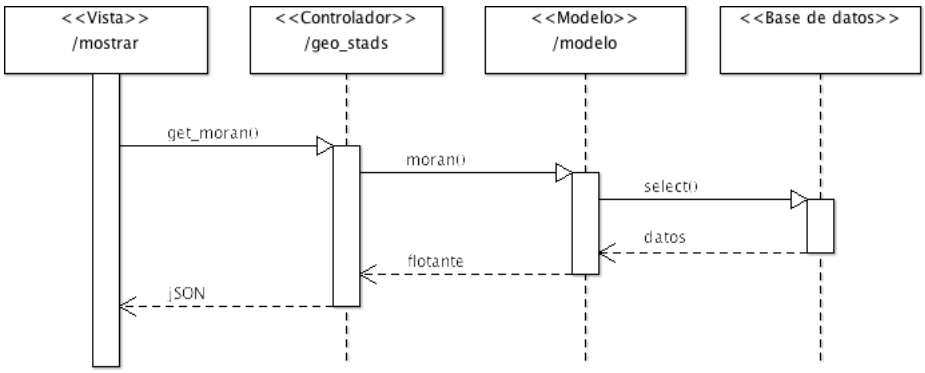


Figura 9.11: Diagrama de secuencia para obtener el Coeficiente de Moran.

Codificación

En el Algoritmo 9.12 y 9.13 se muestra el método para calcular el coeficiente de moran en la clase Spatial. Para calcular el coeficiente se realiza el siguiente procedimiento:

1. Para cada región de la cuadrícula, se toman todos los puntos que estén dentro de la región, se calcula la media clásica sobre los valores del atributo especificado por el parámetro moran_attribute, esto se obtiene mediante el método mean_with_bounds, posteriormente se va almacenando en una suma acumulada estos valores y finalmente se calcula el valor de la media clásica par está suma y el número de regiones.

2. Una vez realizado el paso anterior se precede al cálculo del coeficiente, siguiendo la fórmula anterior, se tendria que realizar una matriz de adyacencias entre las regiones como se explicó en la sección 4.1, sin embargo, realizar tal matriz para tamaños de cuadrícla demasiado pequeños implica realizar demasiados cálculos y el tiempo para realizar el cálculo se vería enormemente afectado, esto se resolvió de la siguiente manera, en lugar de obtener la matriz, para cada región se verifica que otras regiones de la cuadrícula son adyacentes con él, si alguna es adyacente se realizan las operaciones señaladas en la formula del coeficiente de moran.

Algoritmo 9.12 Código para el cáculo del Semivariograma en la clase Spatial

```
1  #Calcula el coeficiente de moran
2  #conditions Cadena de texto que contiene condicones para hacer
       filtrado de datos
3  #model Indica el modelo en el cual se efectuará la operación
4  #delta Especifca a que distancia deben de encontrarse los puntos para
       tomarse en cuenta en el cálculo
5  #semivariogram_attribute Indica el atributo sobre el cual se efectuara
        el cálculo
6  def self.moran(model,conditions=nil,grid_size,moran_attribute)
7    conditions = !conditions.nil? ? [format_conditions(conditions),
         conditions] : nil
8    lon = -99.35 #-117.3
9    lat = 19.49 #32.6
10   num_s_lon = (0.4/grid_size).ceil   #31
11   num_s_lat = (0.4/grid_size).ceil  #18
12   lon2 = lon+grid_size
13   lat2 = lat-grid_size
14   conditions[0] +=  " AND X(point) BETWEEN :x1 AND :x2 AND Y(point)
         BETWEEN :y1 AND :y2"
15   conditions[1].merge!({:x1 => 0, :x2 => 0, :y1 => 0, :y2 =>0 })
16   mean = 0
17   num_s_lat.times do |i|
18     num_s_lon.times do |j|
19       mean += mean_with_bounds(model,conditions,lon,lon2,lat2,lat,
             moran_attribute)
20       lon = lon2
21       if(lon2+grid_size > -98.95)
22         lon2 = -98.95
23       else
24         lon2 = lon2+grid_size
25       end
26     end
27     lon = -99.35
28     lon2 = lon+grid_size
29     lat = lat2
30     if(lat2-grid_size < 19.089)
31       lat2 = 19.089
32     else
33       lat2 = lat2-grid_size
34     end
35   end
36   mean /= num_s_lon*num_s_lat
37   lon = -99.35 #-117.3
38   lat = 19.49 #32.6
39   moran = 0
40   moran_2 = 0
41   sum_wij = 0
```

Algoritmo 9.13 Código para el cáculo del Semivariograma en la clase Spatial (continuación)

```
1   (num_s_lon*num_s_lat).times do |sq|
2     mean_i = mean_with_bounds(model,conditions,lon, lon+grid_size, lat
          -grid_size, lat,moran_attribute)
3     moran += (mean_i - mean)*(mean_i - mean)
4     moran_2 += (mean_i - mean)*(mean_i - mean)
5     sum_wij += 1
6     #para el cuadro i-1 (orilla izq)
7     if(sq %num_s_lon != 0 )
8       moran += (mean_i - mean)*(mean_with_bounds(model,conditions,lon-
            grid_size, lon, lat-grid_size, lat, moran_attribute) - mean)
                        sum_wij += 1
9     end
10    #para el cuadro i+1 (orilla der)
11    if(sq %num_s_lon !=  num_s_lon - 1)
12      if(lon+grid_size*2 > -98.95)
13        moran += (mean_i - mean)*(mean_with_bounds(model,conditions,
              lon+grid_size,-98.95, lat-grid_size, lat,moran_attribute)
              - mean)
14      else
15       moran += (mean_i - mean)*(mean_with_bounds(model,conditions,lon
            +grid_size,lon+grid_size*2, lat-grid_size, lat,
            moran_attribute) - mean)
16      end
17          sum_wij += 1
18    end
19    #para el cuadro i-num_s_lon (orilla sup)
20    if(sq - num_s_lon >= 0 )
21      moran += (mean_i - mean)*(mean_with_bounds(model,conditions,lon,
            lon+grid_size, lat, lat+grid_size,moran_attribute) - mean)
                        sum_wij += 1
22    end
23    #para el cuadro i-num_s_lon-1 (sup izq)
24    if(sq %num_s_lon != 0 && sq - num_s_lon >= 0)
25      moran += (mean_i - mean)*(mean_with_bounds(model,conditions,lon-
            grid_size, lon, lat, lat+grid_size,moran_attribute) - mean)
                        sum_wij += 1
26    end
27    #para el cuadro i-num_s_lon+1 (sup der)
28    if(sq %num_s_lon !=  num_s_lon - 1 && sq - num_s_lon >= 0 )
29      if(lon+grid_size*2 > -98.95)
30        moran += (mean_i - mean)*(mean_with_bounds(model,conditions,
              lon+grid_size,-98.95 , lat, lat+grid_size,moran_attribute)
               - mean)          else
31       moran += (mean_i - mean)*(mean_with_bounds(model,conditions,lon
            +grid_size, lon+grid_size*2, lat, lat+grid_size,
            moran_attribute) - mean)
32      end
33      sum_wij += 1
34    end
```

Algoritmo 9.14 Código para el cáculo del Semivariograma en la clase Spatial (continuación)

```
1      #para el cuadro i+num_s_lon (orilla inf)
2      if(sq + num_s_lon < num_s_lon*num_s_lat)
3        if(lat-2*grid_size < 19.089)
4          moran += (mean_i - mean)*(mean_with_bounds(model,conditions,
               lon, lon+grid_size, 19.089, lat-grid_size,moran_attribute)
                - mean)
5        else
6          moran += (mean_i - mean)*(mean_with_bounds(model,conditions,lon
               , lon+grid_size, lat-2*grid_size, lat-grid_size,
               moran_attribute) - mean)
7        end
8        sum_wij += 1
9      end
10     #para el cuadro i+num_s_lon-1 (inf izq)
11     if(sq %num_s_lon != 0 && sq + num_s_lon < num_s_lon*num_s_lat)
12       if(lat-2*grid_size < 19.089)
13         moran += (mean_i - mean)*(mean_with_bounds(model,conditions,
               lon-grid_size, lon, 19.089, lat-grid_size,moran_attribute)
                - mean)
14       else
15         moran += (mean_i - mean)*(mean_with_bounds(model,conditions,lon
               -grid_size, lon, lat-2*grid_size, lat-grid_size,
               moran_attribute) - mean)
16       end
17       sum_wij += 1
18     end
19     #para el cuadro i+num_s_lon+1 (inf der)
20     if(sq %num_s_lon !=  num_s_lon - 1 && sq + num_s_lon < num_s_lon *
           num_s_lat)
21       lo =  lon+grid_size*2 > -98.95 ? -98.95 : lon+grid_size*2
22       la =  lat-2*grid_size < 19.089 ? 19.089 : lat-grid_size*2
23       moran += (mean_i - mean)*(mean_with_bounds(model,conditions,lon+
               grid_size, lo, la, lat-grid_size,moran_attribute) - mean)
24       sum_wij += 1
25     end
26     lon += grid_size
27     if(lon > -98.95) # if(sq %num_s_lon == num_s_lon - 1 )
28       lon = -99.35
29       lat -= grid_size
30     end
31   end
32   ((num_s_lon*num_s_lat).to_f/sum_wij.to_f)*(moran/moran_2)
33 end
```

En el Algoritmo 9.15 se muestra la parte correspondiente al controlador.

Algoritmo 9.15 Porción del codigo para el controlador.

```
1 def moran
2  respond_with("{\"moran\":"+Spatial.moran(params[:database],
       params[:values],params[:grid_size].to_f,params[:
       moran_attribute]).to_s+"}")
3 end
```

Pruebas

Para verificar que las operaciones se estan llevando de forma adecuado se definen las siguientes pruebas: En la primera prueba se verifica que el método para calcular el coeficiente de moran devuelve un valor no nulo y en la segunda prueba se verifica que devuelve un valor del coeficiente de moran calculado previamente.

Algoritmo 9.16 Pruebas unitarias para la clase Inegi

```
1  test "should return moran" do
2    moran = Spatial.moran("Inegi",{:pob_total=>0 , :pob_m=>0, :
         pob_f=>0, :grado_prom_esc=>0, :grado_prom_esc_m=>0, :
         grado_prom_esc_f=>0, :pob_econ_act_f=>0, :pob_econ_inact
         =>0, :pob_econ_inact_m=>0, :pob_econ_inact_f=>0, :
         pob_sin_ss=>0, :pob_con_ss=>0, :pob_sin_imss=>0, :
         pob_sin_iste=>0, :pob_sin_istee=>0, :pob_sin_segp=>0, :
         pob_soltera=>0, :pob_casada=>0, :pob_relig_cat=>0, :
         pob_relig_no_cat=>0, :pob_relig_otra=>0, :pob_sin_relig
         =>0}, 0.1,"pob_total")
3    assert !moran.nil?, "moran not found"
4  end
5
6  test "should return moran value" do
7    moran = Spatial.moran("Inegi",{:pob_total=>0 , :pob_m=>0, :
         pob_f=>0, :grado_prom_esc=>0, :grado_prom_esc_m=>0, :
         grado_prom_esc_f=>0, :pob_econ_act_f=>0, :pob_econ_inact
         =>0, :pob_econ_inact_m=>0, :pob_econ_inact_f=>0, :
         pob_sin_ss=>0, :pob_con_ss=>0, :pob_sin_imss=>0, :
         pob_sin_iste=>0, :pob_sin_istee=>0, :pob_sin_segp=>0, :
         pob_soltera=>0, :pob_casada=>0, :pob_relig_cat=>0, :
         pob_relig_no_cat=>0, :pob_relig_otra=>0, :pob_sin_relig
         =>0}, 0.1,"pob_total")
8    assert moran == 0.4180533595104267 ?, "moran not found"
9  end
```

En Figura 9.12 están los resultados de las pruebas. Nos indica que de las

nueve pruebas anteriores mas las nuevas dos todas fueron correctas.

```
Started . Finished in 0.245194 seconds.
9 tests, 9 assertions, 0 failures, 0 errors, 0 skips
```

Figura 9.12: Resultados de las pruebas para la clase Inegi

9.3.5. Etapa 6: Semivariograma

Planeación

El objetivo del módulo es calcular el valor del semivariograma dado un conjunto de datos y el valor de una distancia para efectuar la operación en base a la fórmula de la Figura 9.13 . La Interfaz hará la petición al controlodar para recibir los datos en formato jSON, el cual consiste en un valor real, posteriormente este pedirá al modelo el calculo del semivariograma, el resultado se muestra en pantalla, el cual consiste en graficar cada vez que se solicite ejecutar la operación con una distancia distinta.

$$\gamma(\delta) = \frac{1}{2N(\delta)} \sum_{(i,j)|h_{ij}=h} (x_i - x_j)^2$$

Figura 9.13: Fórmula para calcular el Semivariograma. Donde $N(\delta)$ es el número de pare de puntos que se encuentran a distancia h, h es un escalar tomado para medir la distancia entre los pares de puntos, h_{ij}es la distancia que existe entre los pares de puntos (i, j), x_i es el valor del i-esimo dato.

Administración

En esta etapa se desarrollan los componentes que forman parte de los modelos y del controlador, estos están programados en Ruby. En las siguientes clases Spatial, Inegi, Conabio, Salud y GeoStadsController se añade el método correspondiente para el cálculo del Semivariograma.

Diseño

En la figura 9.14 se muestra el diagrama de secuencia para obtener el cálculo del Semivariograma. Se asume que la base de datos ha sido cargada y seleccionada.

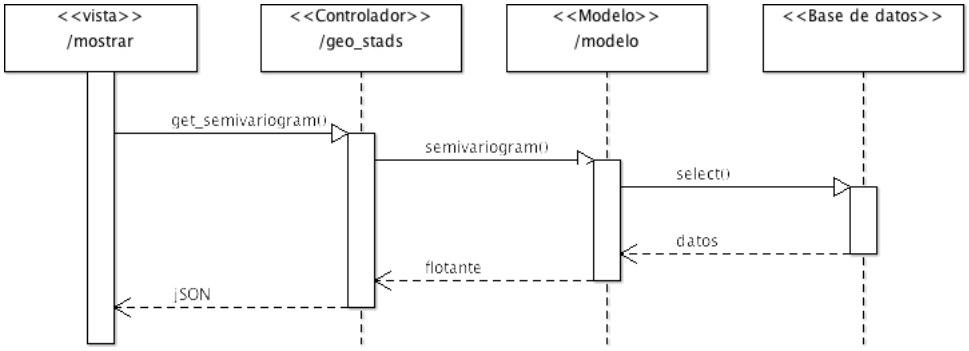


Figura 9.14: Diagrama de secuencia para obtener el Semivariograma

Codificación

En el Algoritmo 9.17 se muestra el método calcular el Semivariograma en el módulo Spatial. Para este cálculo por cada punto del conjunto de datos seleccionado se checan los restantes puntos y se toman en cuenta aquellos que son diferentes al punto en cuestión y que cumplan con la condicion de las distancias, es decir, que su distancia al punto esté entre 10 % más a la distancia establecida o 10 % menos.

Algoritmo 9.17 Código para el cálculo del Semivariograma en el módulo Spatial

```
1  #Calcula el semivariograma
2  #conditions Cadena de texto que contiene condicones para hacer
       filtrado de datos
3  #model Indica el modelo en el cual se efectuará la operación
4  #delta Especidifca a que distancia deben de encontrarse los
       puntos para tomarse en cuenta en el cálculo
5  #semivariogram_attribute Indica el atributo sobre el cual se
       efectuara el cálculo
6  def self.semivariogram(model,delta,conditions=nil,
       semivariogram_attribute)
7    conditions = !conditions.nil? ? [format_conditions(conditions
       ),conditions] : nil
8    objects = eval(model).where(conditions)
9    nh = 0
10   sum = 0
11   objects.each do |object|
12     objects.each do |object2|
13       if(object2 != object && (object.distance(object2) >= delta
           *0.9 &&  object.distance(object2) <= delta*1.1 ))
14         nh = nh+1
15           sum += (object2[semivariogram_attribute.to_sym]-
               object[semivariogram_attribute.to_sym])**2
16       end
17     end
18   end
19   (sum/(2*nh).to_f)
20 end
```

En el Algoritmo 9.18 se muestra la parte correspondiente al controlador.

Algoritmo 9.18 Porción del codigo para el controlador.

```
1  def semivariogram
2    respond_with("{\"semivariogram\":"+Spatial.semivariogram(
       params[:database],params[:delta].to_f,params[:values],
       params[:semivariogram_attribute]).to_s+"}")
3  end
```

Pruebas

Para verificar que las operaciones se están llevando de forma adecuado se definen las siguientes pruebas: En la primera prueba se verifica que el método

para calcular el semivariograma devuelve un valor no nulo y en la segunda prueba se verifica que devuelve un valor del semivariograma calculado previamente.

Algoritmo 9.19 Pruebas unitarias para la clase Inegi

```
1  test "should return semivariogram" do
2    sem = Inegi.semivariogram({:pob_total=>0 , :pob_m=>0, :pob_f
         =>0, :grado_prom_esc=>0, :grado_prom_esc_m=>0, :
         grado_prom_esc_f=0, :pob_econ_act_f=>0, :pob_econ_inact=>0,
          :pob_econ_inact_m=>0, :pob_econ_inact_f=>0, :pob_sin_ss
         =>0, :pob_con_ss=>0, :pob_sin_imss=>0, :pob_sin_iste=>0, :
         pob_sin_istee=>0, :pob_sin_segp=>0, :pob_soltera=>0, :
         pob_casada=>0, :pob_relig_cat=>0, :pob_relig_no_cat=>0, :
         pob_relig_otra=>0, :pob_sin_relig=>0}, 227.0,"pob_total")
3    assert !sem.nil?, "semivariogram not found"
4  end
5
6  test "should return semivariogram value" do
7    sem = Inegi.semivariogram({:pob_total=>0 , :pob_m=>0, :pob_f
         =>0, :grado_prom_esc=>0, :grado_prom_esc_m=>0, :
         grado_prom_esc_f=0, :pob_econ_act_f=>0, :pob_econ_inact=>0,
          :pob_econ_inact_m=>0, :pob_econ_inact_f=>0, :pob_sin_ss
         =>0, :pob_con_ss=>0, :pob_sin_imss=>0, :pob_sin_iste=>0, :
         pob_sin_istee=>0, :pob_sin_segp=>0, :pob_soltera=>0, :
         pob_casada=>0, :pob_relig_cat=>0, :pob_relig_no_cat=>0, :
         pob_relig_otra=>0, :pob_sin_relig=>0}, 227.0,"pob_total")
8    assert sem == 5335.920289855072 ?, "semivariogram not found"
9  end
```

En Figura 9.15 están los resultados de las pruebas. Nos indica que de las nueve pruebas anteriores mas las nuevas dos todas fueron correctas.

```
Started . Finished in 0.245194 seconds.
11 tests, 11 assertions, 0 failures, 0 errors, 0 skips
```

Figura 9.15: Resultados de las pruebas para la clase Inegi

Parte IV

Resultados y conclusiones

Capítulo 10

Resultados

A continuación se presenta una serie de imágenes que ilustran los resultados obtenidos, es decir, la respuesta del sistema para cada una de las operaciones desarolladas para el sistema GeoStads. La primera imágen corresponde al diseño de la interfaz, posteriormente se muestran las imagenes de cada operación aplicada primero a los datos de la tabla Inegi y posteriormente a los datos de la tabla Conabio.

Interfaz

En la Figura 10.1 se muestra la interfaz desarrollada para el sistema.

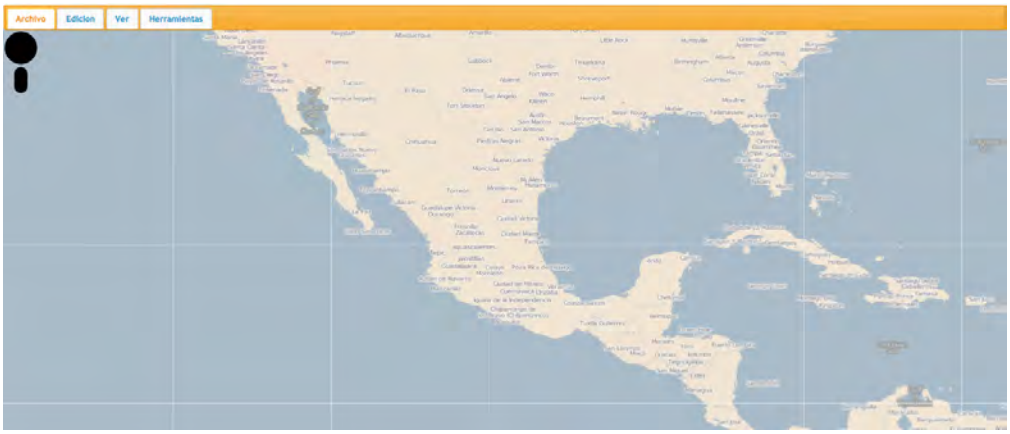
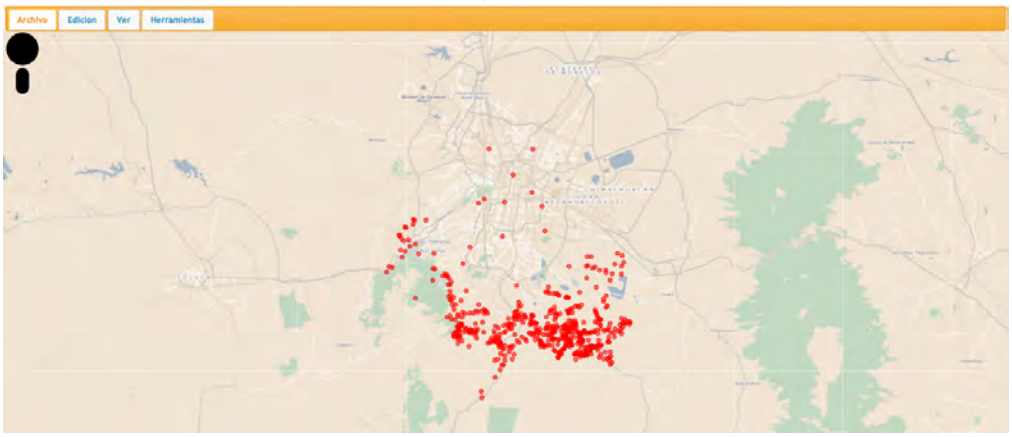


Figura 10.1: Interfaz para el sistema GeoStads

Lectura de datos

A continuación, en la Figura 10.2 se ilustra la lectura de las tablas Inegi y Conabio.

a)



b)

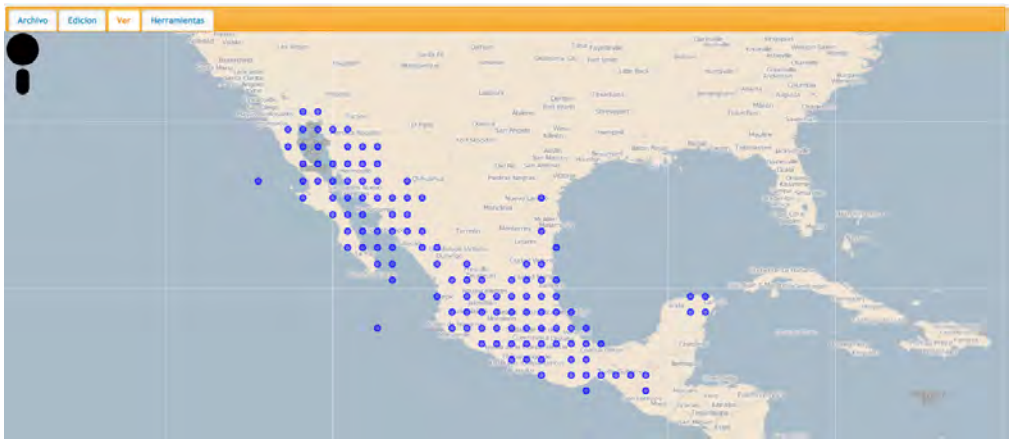
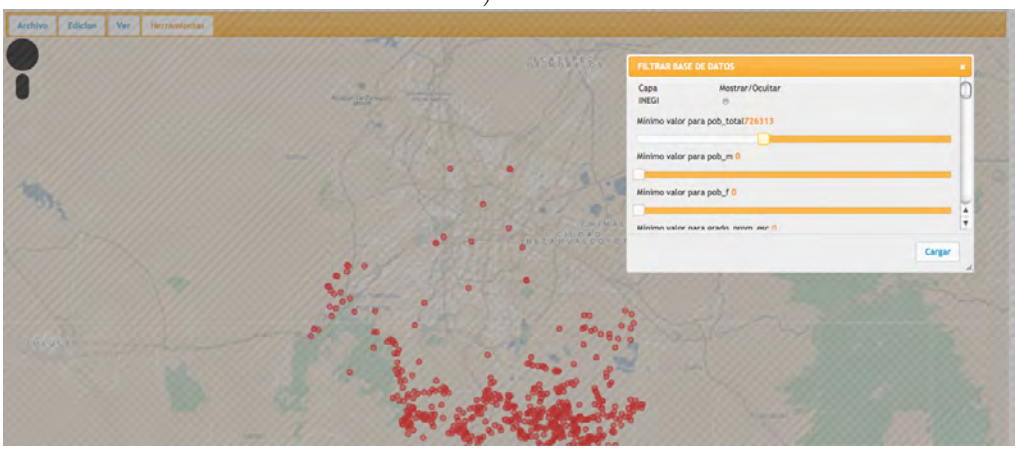


Figura 10.2: Lectura de una tabla en GeoStads

Filtrado de una tabla

En la Figura 10.3 se ilustra la operación de filtrar la tabla Inegi, en a) se muestra el filtro que se le aplicará a los datos de la tabla Inegi, se selecciona el filtro por el atributo pob_total, el sistema muestra en la interfaz todos los puntos que cumplan que su poblacion total sea mayor a 726313. En b) se muestra el resultado de aplicar el filtro.

a)



b)

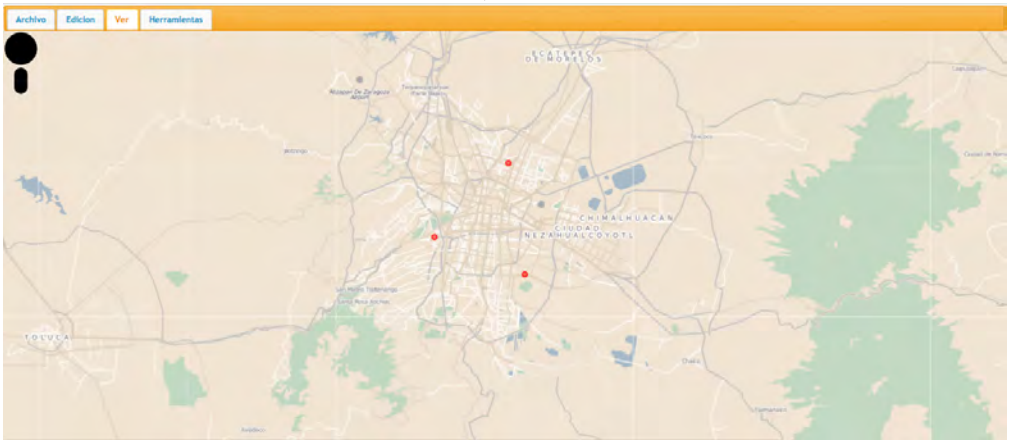
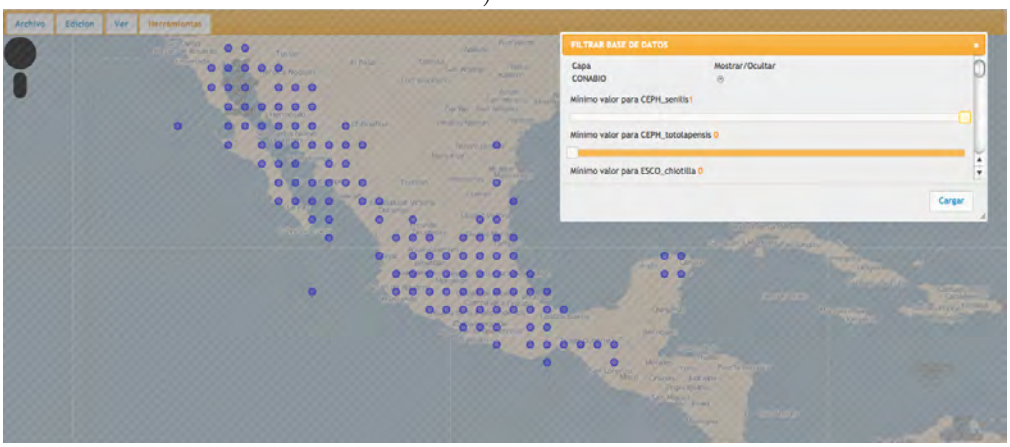


Figura 10.3: Filtrado de la tabla Inegi

En la Figura 10.4 se muestra la operación de filtro sobre la tabla Conabio, en a) se selecciona el filtro por el atributo CEPH_senilis, el sistema muestra en la interfaz todos los puntos que cumplan que CEPH_senilis sea igual a 1, es decir, muestra todos los puntos tales que ahi haya cactáceas de la especie CEPH_senilis. En b) se muestra el resultado de aplicar el filtro.

a)



b)

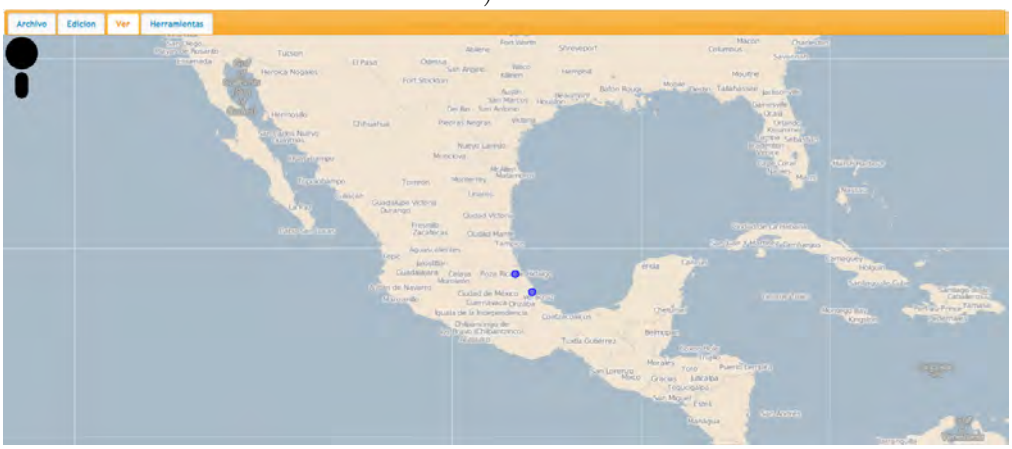
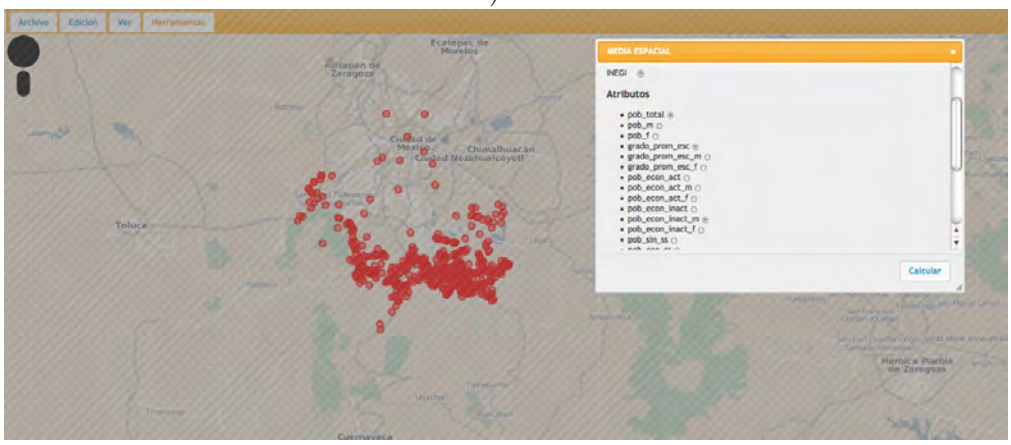


Figura 10.4: Filtrado de la tabla Conabio

Media espacial

En la Figura 10.5 se muestra el resultado de calcular la media espacial sobre los datos de la tabla Inegi. En a) se muestra el menú para seleccionar el calculo de la media espacial, se calculará la media espacial de la tabla Inegi con los atributos pob_total, grado_prom_esc y pob_econ_inact_m. En b) se muestra el resultado de la operación, los puntos en color azul indican la ubicación de la media espacial de cada uno de los atributos.

a)



b)

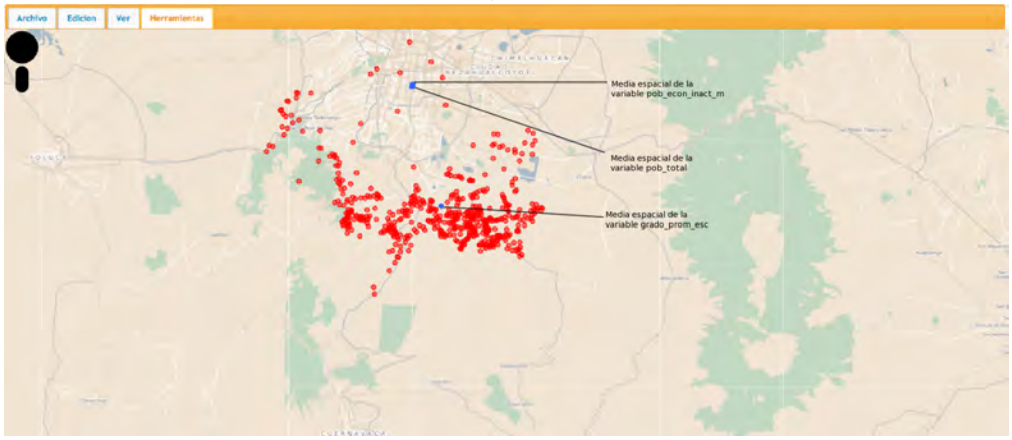
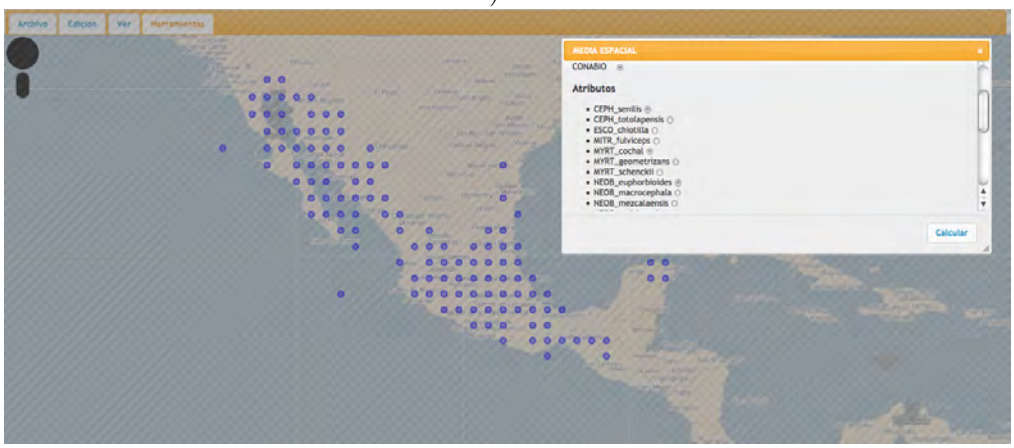


Figura 10.5: Visualizacion del cálculo de la media espacial de la tabla Inegi

En la Figura 10.6 se ilustra el cálculo de la media espacial sobre la tabla Conabio. En a) se muestra el menú para seleccionar el calculo de la media espacial, se calculará la media espacial de la tabla Conabio con los atributos CEPH_senilis, MYRT_cochal y NEOB_euphorbioides. En b) se muestra el resultado de la operación, los puntos en color rojo indican la ubicación de la media espacial de cada uno de los atributos.

a)



b)

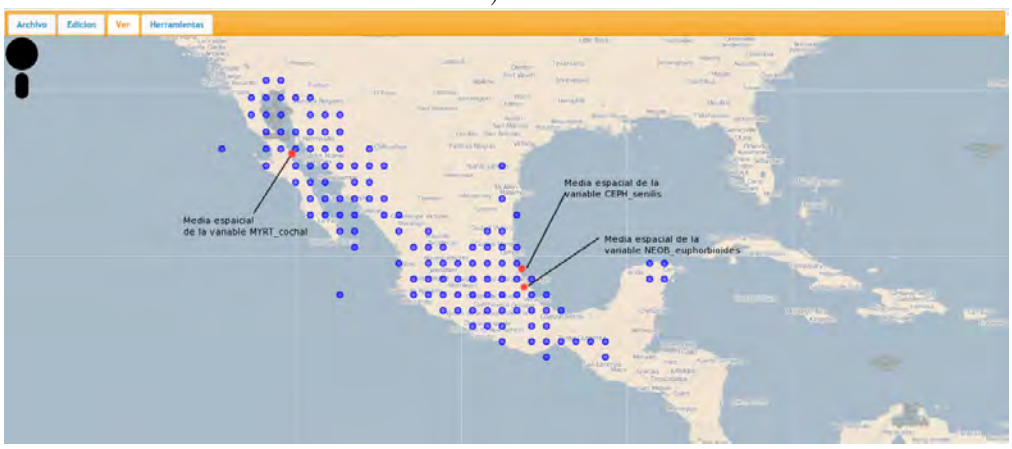
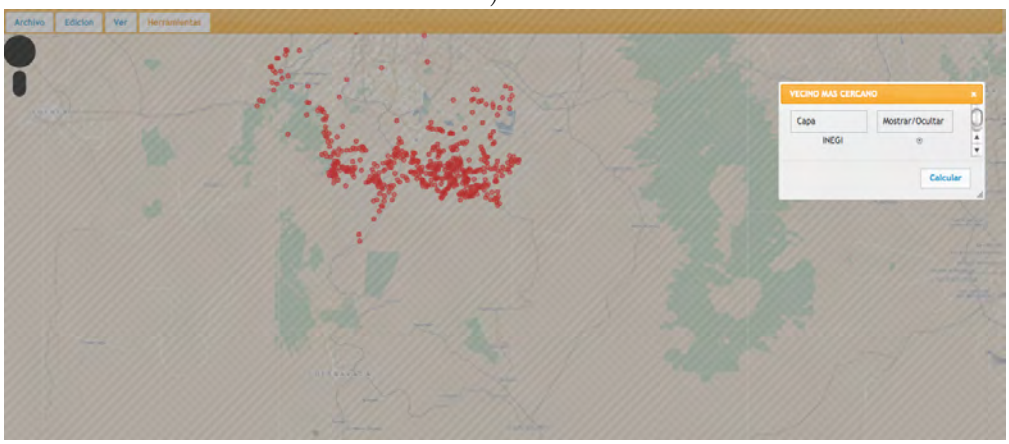


Figura 10.6: Visualizacion del cálculo de la media espacial de la tabla Conabio

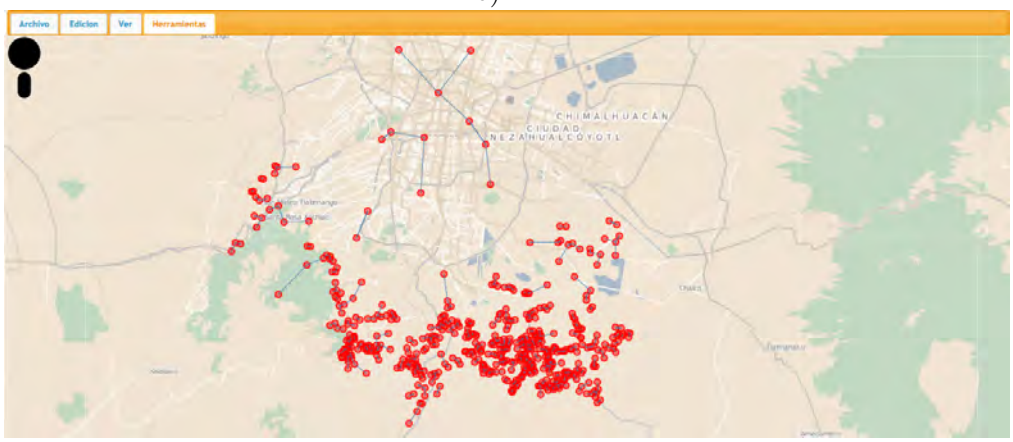
Vecino más cercano.

A continuación en las Figuras 10.7 y 10.8 se muestra el resuldato del cálculo del vecino más cercano para la tabla Inegi y Conabio respectivamente.

a)



b)



b)

Figura 10.7: Vecino más cercano para la tabla Inegi

a)

b)

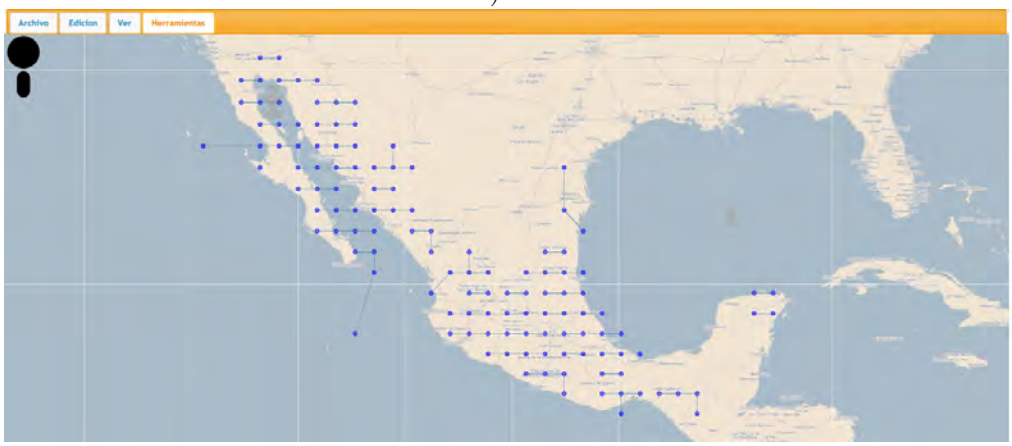
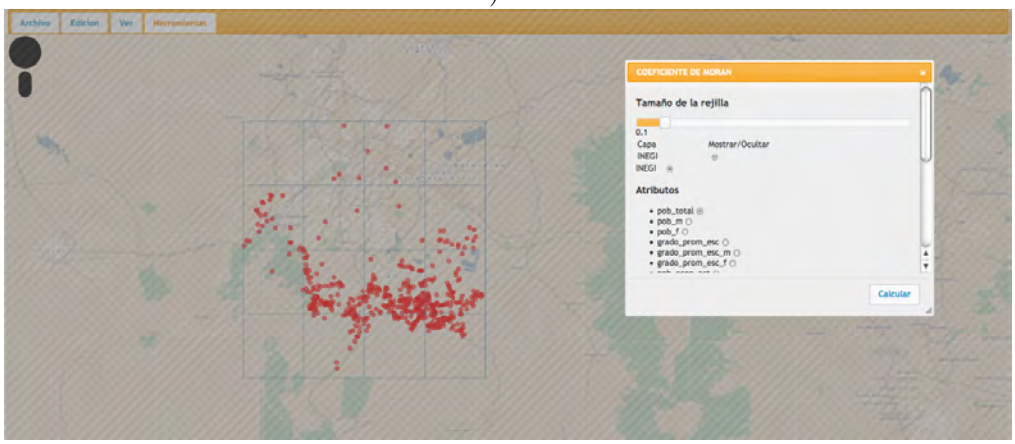


Figura 10.8: Vecino más cercano para la tabla Conabio

Autocorrelación espacial.

En la Figura 10.9 se muestra el resultado de calcular la autocorrelación espacial para la tabla Inegi. En (b) Se muestra el coeficiente de moran calculado con los datos de la tabla Inegi, este se cálculo con tamaños de la rejilla de 0.03^o, 0.04^o, 0.05^o, 0.06^o, 0.07^o, 0.08^o y 0.1^o. Se puede observar en la imágen que los valores comienzan a acercarse a uno lo que indica que con valores de la rejilla pequeños existe una autocorrelación positiva, sin embargo para valores de rejilla cada vez mas grandes la autocorrelación comienza a ser cero y posteriormente es negatva.

a)

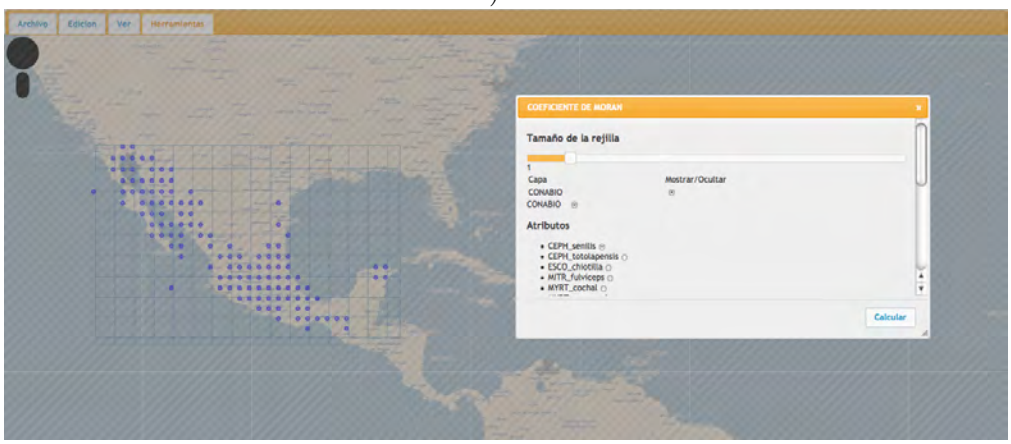


b)

Figura 10.9: Autocorrelación espacial para la tabla Inegi

En la Figura 10.10 se muestra el resultado de calcular la autocorrelación espacial para la tabla Conabio. En (b) se muestra que la autocorrelación espacial de los datos es más uniforme y se mantiene a diferencia de los datos de la tabla inegi aunque tampoco indica si existe o no autocorrelación positiva.

a)



b)

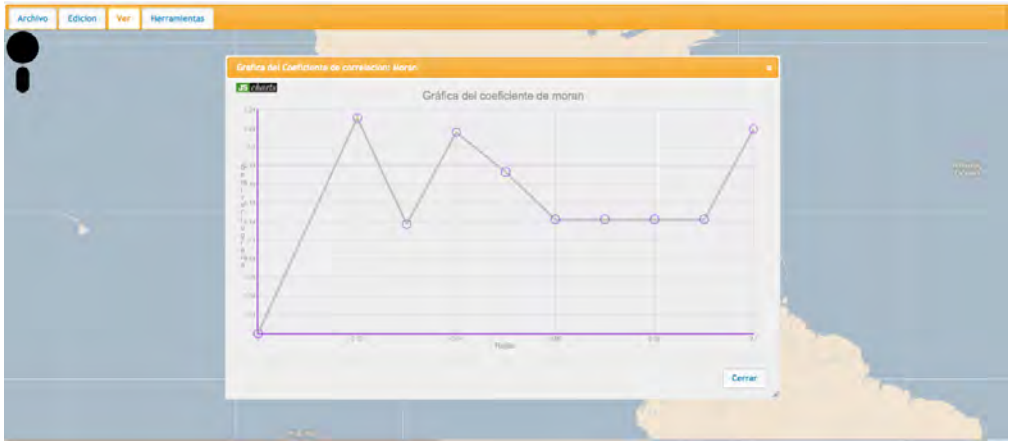
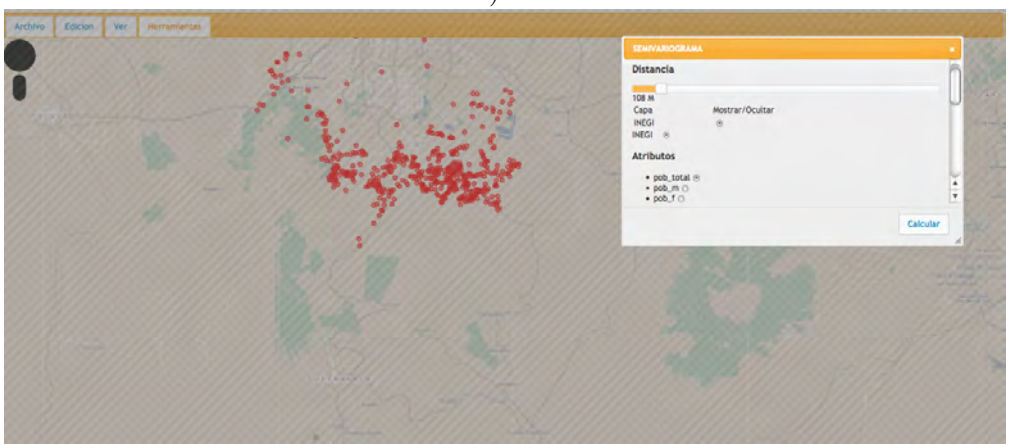


Figura 10.10: Autocorrelación espacial para la tabla Conabio

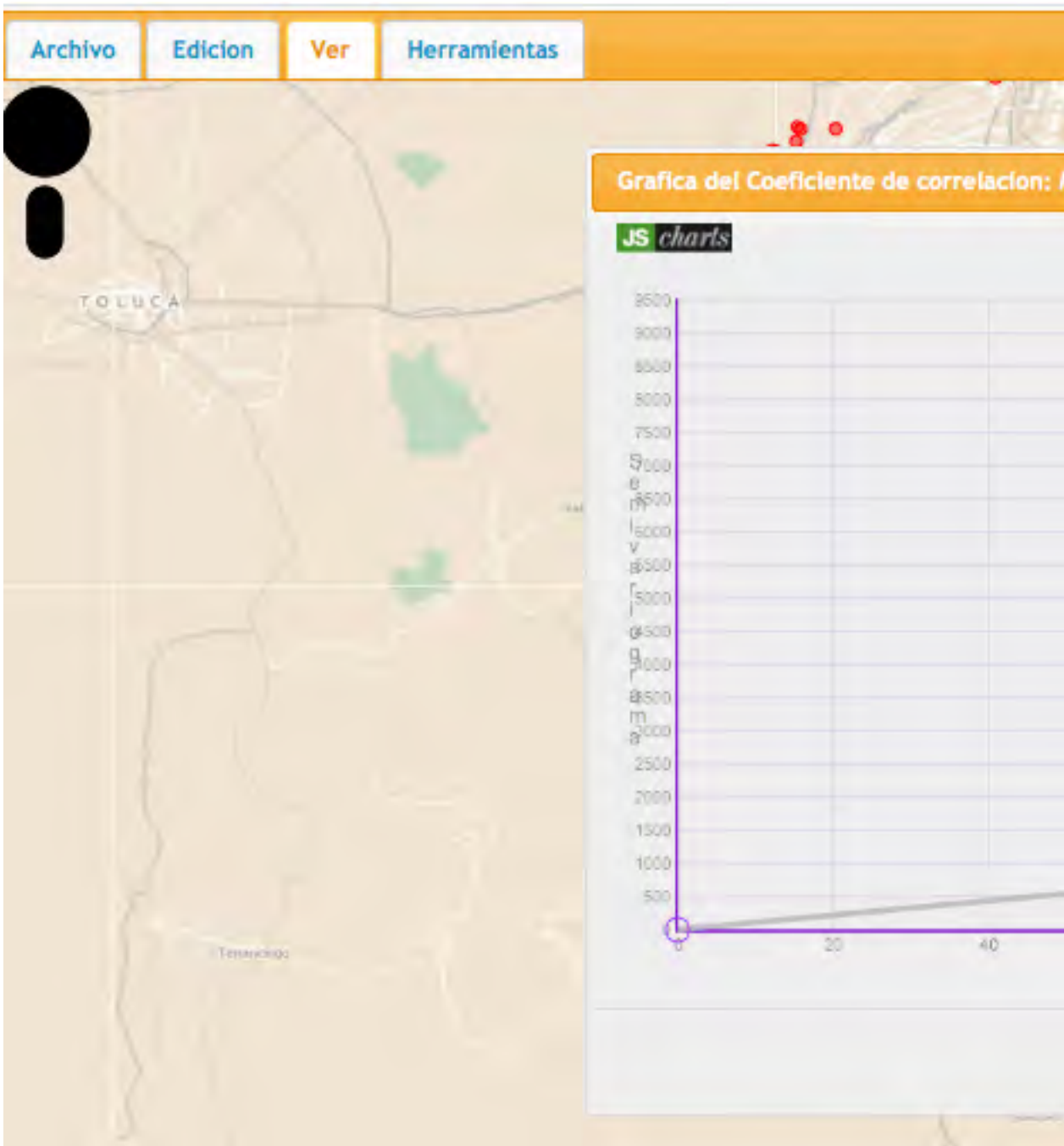
Semivariograma.

En la Figura 10.11 se muestra el resultado de calcular el semivariograma para los datos de la tabla Inegi. En este caso al observar la gráfica de la imágen (b) observamos que es parecida al semivariograma típico de autocorrelación espacial negativa como se mostró en la sección 4.4.

a)

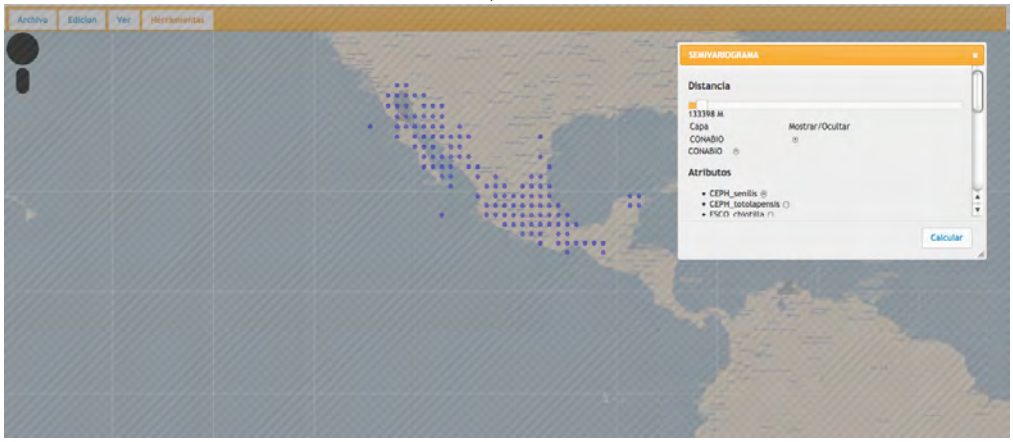

Archivo
Edicion
Ver
Herramientas
SEMIVARIOGRAMA
Distancia
Capa
Mostrar/Ocultar
INEGI
Atributos
pob_total
pob_m
pob_f
Calcular

b)


Archivo
Edicion
Ver
Herramientas
Grafica del Coeficiente de correlacion:
JS charts
TOLUCA
9500
9000
8500
8000
7500
7000
6500
6000
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
20
40

En la Figura 10.12 se muestra el resultado de calcular el semivariograma para los datos de la tabla Conabio. En la gráfica del semivariograma (b) se observa que al tener distancias pequeñas la autocorrelación de los datos es negativa, sin embargo, al crecer la distancia para la muestra del semivariograma es claro que la autocorrelación comienza a ser positiva.

a)



b)

Figura 10.12: Semivariograma para la tabla Conabio.

Capítulo 11

Conclusiones

Este trabajo responde a necesidades planteadas en el área de Sistemas de Información Geográfica y en especial al área de análisis de información geográfica, se puede observar que con ayuda de las herramientas implementadas es posible explotar una base de datos y obtener nuevo conocimiento. Se logró realizar la exploración y el análisis de los datos a fin de encontrar patrones significativos que no se encontraban con otras técnicas, es decir. se logró realizar minería de datos [46].

Sin embargo, estas necesidades ya habían sido planteadas con anterioridad y como resultado existe software que tiene incorporado funcionalidades similares, por esta razón se realiza una comparación entre algunos de los principales paquetes de software para SIG:

- ArcGIS: Desarrollado por la empresa ESRI la cual cuenta con años de experiencia en el campo de SIG, por esta razón ArcGIS es el paquete software más conocido en esta área, sin embargo, a pesar de contar con múltiples herramientas para llevar a cabo análisis espacial tiene las desventajas de ser un software privado y complicado para usuarios que no cuentan con experiencia. Este software cuenta con una extensión, un módulo extra para llevar a cabo análisis geoestadístico, sin embargo, se tiene que pagar para adquirir este producto, Por lo que, si se desea realizar algúnt tipo de análisis espacial con este software es necesario pagar una gran cantidad de dinero[71].
- GRASSGIS: Es un software libre y con funcionalidades complejas para el análisis de datos por lo que tiende a ser dificil de aprender a utilizar. Además no tiene incorporado funciones de estadística espacial,

es necesario instalar en cada máquina cliente donde se desee utilizar una extensión para poder hacer uso de las funciones de lenguaje de programación R.

- GeoStads: Es el software desarrollado a lo largo de este proyecto, cuenta con algunas limitaciones, por ejemplo, no realiza procesamiento de imágenes como la mayoría de los paquetes mencionados anteriormente, sin embargo, la finalidad de GeoStads es brindar herramientas de estadística espacial y una interfaz de usuario amigable y simple. Al ser un sistema web no es necesario instalar ninguna herrameinta del lado del cliente y por esta razón es multiplataforma, otra de las ventajas es que no carga módulos innecesario por lo que es rápido a la hora de realizar cálculos.

GeoStads es una herramienta que permite:

- Visualización de datos con información espacial, como puede ser un punto, un polígono, etc.
- Análisis de información mediante técnicas de estadística espacial.
- Almacenamiento de información en base de datos con soporte espacial en lugar al sistema convecional basado en archivos shapefile.
- Disponibilidad al ser un sistema web.
- Confiabilidad al ser un sistema desarrollado con herramientas de software libre reconocidas ampliamente.

GeoStads es un proyecto que cuenta con posibilidades de crecer de distintas maneras, algunas de ellas son implementar más herramientas de estadística espacial, clustering de los datos, incorporación de procesameinto digital de imágenes para lograr una mayor interacción con los mapas.

Bibliografía

[1] Baselga Moreno S., **Fundamentos de cartografía matemática**, Ed. Universidad Politécnica Valencia, 2006.

[2] Caire Lomelí J., **Cartografía básica**, Facultad de Filosofía y Letras, UNAM, 2002.

[3] Ocampo M. J. A., Céspedes V. O. R., **Cartogtrafía básica aplicada**, Universidad de Caldas, 2005.

[4] Van Sickle J., **Basic GIS coordinates**, CRC Press, 2004.

[5] **Mercator projection**, http://en.wikipedia.org/wiki/Mercator_projection (17/08/2011 15:11 pm)

[6] Osborn Maitland M., **Notes on Cylindrical World Map Projections,** Geographical Review, 1942.

[7] Deetz C. H., **Elements of map projection and its application to the construction of maps and charts**, Washington: Secretaría de Estado de los Estados Unidos de América, 1944.

[8] **British Columbia Map Projection Standard**, BC Integrated Land Management Bureau, http://www.ilmb.gov.bc.ca/risc/pubs/other/mappro/map.htm. (13/09/2011 18:12)

[9] **Projection Reference**. Bill Rankin, http://www.radicalcartography.net/?projectionref. (13/09/2011 18:30)

[10] Shenstone W. F., **Higher geometry: an introduction to advanced methods in analytic geometry**, Ginn and Company, 1922.

[11] **Geographic coordinate systems,**
http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.spat
(19/08/2011 16:10)

[12] Hill L. L., **Georeferencing. The Geographic Associations of Information**, The MIT Press, 2006.

[13] Batini C., **Diseño conceptual de bases de datos: un enfoque de entidades relacionales**, Ediciones Díaz de Santos, 1994.

[14] Date C.J., **Introducción a los sistemas de bases de datos**, Pearson Education, 2003.

[15] Silberschatz A., Korth F. y Sudarshan S., **Fundamentos de Bases de Datos**, MacGraw Hill, 2002.

[16] Patterson D. A. y Hennessy J. L., **Computer Architecture: A Quantitative Approach**, 2a edición, Morgan Kaufmann, 1995.

[17] Gray J. y Reuter A., **Transaction Processing: Concepts and Techniques**, Morgan Kaufman, 1993.

[18] Duncan R., **A Survey of Parallel Computer Architectures**, IEEE Computer, 23(2), 1990.

[19] Ozsu T. y Valduriez P., **Principles of Distributed Database Systems**, 2.a edición, Prentice Hall, 1999.

[20] Codd E.F., **Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks**, IBM Research Report, 1969.

[21] Codd E. F., **The Relational Model for Database Management**, Addison-Wesley Publishing Company, 1990.

[22] Codd E. F., **The relational model for database management: version 2**, Addison-Wesley Publishing Company, 1990.

[23] Chamberlin D., Astrahan M., Eswaran K., Griffiths P., Loriem R., Mehl J., Reisner P. y Wade B., **Sequel 2: a unified approach to data definition, manipulation and control**, IBM J. Res. and Develop. 20, 1976.

[24] Stonebraker M., Wong E., y Kreps P., **The design and implementation of INGRES,** ACM Trans. Database Syst. 1, September 1976.

[25] Zloof M. M., **Query-by-example: a database language**, IBM Syst. J. 16, 1977.

[26] Bennis K., David B., Quilio I. y Vimont Y., **GEOTROPICS: Database Support Alternatives for Geographic Applications**, In Proc. Intl. Symp. On Spatial Data Handling (SDH), 1990.

[27] Rigaux P., Scholl M. y Voisard A., **Spatial databases with application to GIS**, Morgan Kaufmann Publishers, 2002.

[28] ESRI. **Esri Shapefile technical description** 1998, (04/07/2011).

[29] Burrough P. A. y McDonnell R. A., **Principles of Geographical Information Systems**, Oxford University Press, Oxford/New York, 1998.

[30] Lorie R. y Meier A., **Using relational dbms for geographical databases**, GeoProcessing, vol. 2, 1984.

[31] Coppock J. T. y Rhind D. W., **Geographical Information Systems: Principles and Applications. Vol. 1: Principles (chapter: "The History of GIS")** ,Longman Scientific and Technical, London, 1991.

[32] Devogele T., Parent C. y Spaccapietra S., **On Spatial Database Integration**, International Journal of Geographical Information Science, 12(4), 1998.

[33] Erwig M., Guting R. H., Schneider M. y Vazirgiannis M., **Spatiotemporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases**, GeoInformatica, 1999.

[34] Adam N. R. **Gangopadhyay. Database Issues in Geographic Information Systems**, Advances in Database Systems, Kluwer Academic Publishers Vol. 6, New York, 1997.

[35] Becker L. y Guting R. H., **Rule-Based Optimization and Query Processing in an Extensible Geometric Database System**, ACM Transactions on Database Systems, 17(2):247–303, 1992.

[36] Abel D., **SIRO-DBMS: A Database Toolkit for Geographical Information Systems**, Intl. Journal of Geographical Information Systems, 1989.

[37] Agarwal P. K., **Geometric Range Searching**, In CRC Handbook of Discrete and Computational Geometry. CRC Press, Boca Raton, 1997.

[38] Aho A. V., Hopcrof J. E. y Ullman J. D., **The Design and Analysis of Computer Algorithms**, Addison-Wesley, Reading, MA, 1974.

[39] Scholl M. y Voisard A., **Thematic Map Modeling**, Springer-Verlag, 1989.

[40] Mitchell A., **The ESRI guide to GIS analysis, volume 1: geographic patterns and relationships**. ESRI, Redlands, 1999.

[41] Smith M. J., Goodchild M. F. y Longley P. A**., Geospatial analysis**. Troubador, Leicester, 2006.

[42] Barrera R. y Buchmann A., **Sceheme definition and query language for a geographic database system**, IEEE Trans. Comp. Arquitecture: Pattern Analysis and Image Database Management, vol. 11, 1981.

[43] Codd E.F., **Fatal flaws in SQL**, Datamation 34, 1988.

[44] Westrlake A. y Kleinschmidt I., **The implementation of area and membership retrievals in point using SQL**, Conf. Statistics and Scientific Database Management, Charlotte, NC, 1990.

[45] Egenhofer M. J., **Spatial SQL: A Query and Presentation Language**, IEEE Transactions on Knowledge and Data Engineering 6, 1994.

[46] **Data Mining program,** http://dms.stat.ucf.edu/programs/programs2.html (18/11/2011 16:35)

[47] Sarabia A., Pascual J. M. y Sáez J. M., **Curso básico de estadística**, Universidad de Canabria, 2005.

[48] **Average Nearest Neighbor (Spatial Statistics),** http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=average_nearest_neighbor (07/11/2011 19:50)

[49] Zhang J., Papadias D., Mamoulis N. y Tao Y., **All-Nearest-Neighbors Queries in Spatial Databases,** Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on, 2004.

[50] **Spatial Autocorrelation (Morans I) (Spatial Statistics),** http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=Spatial_Autocorrelation_ (7/11/2011 20:30)

[51] **Examine the semivariogram cloud for evidence of autocorrelation and directional trends**, http://webhelp.esri.com/arcgisdesktop/9.3/tutorials/geostat/Geostat_2_4.htm (10/11/2011 17:30)

[52] Ross Sheldon M., **Introducción a la estadística**, Reverte, 2007, 809 páginas.

[53] Murguía M., **La estadística espacial como herramienta de análisis de la biodiversidad**, Sobre diversidad biológica. El significado de las Diversidades Alfa. Beta y Gamma. Monografías tercer milenio, capítulo 5, 2005.

[54] Vasiliev Ren I. **Practical hadnbook of spatial statistics. Visualization of spatial depdendence: and elementary view of spatial autocorrelation**, CRC Press, 1996.

[55] **Interaccion persona ordenador**, http://www.aipo.es(31/07/2011 15:30)

[56] **Medicion de los atributos del software**, http://www.sc.ehu.es/jiwdocoj/mmis/externas.htm (31/08/2011 17:20)

[57] Mooney J.D., **Bringing Portability to the Software Process**, West Virginia University. Dept. of Statistics and Computer Science, 1997.

[58] Sommerville I., **Ingeniería del software**, Pearson Educación, 2005.

[59] **ISO 9241-11. Ergonomic requirements for office work with visual display terminals**. ISO, 1998.

[60] **Principios del manifiesto ágil**, http://www.agilemanifesto.org/iso/es/principles.html (01/08/2011 14:59)

[61] **Modelo incremental**, http://www.mitecnologico.com/Main/ModeloIncremental (05/08/2011 19:45)

[62] Llopis P. J., **Sistemas de información geográfica aplicados a la gestión del territorio**, Club Universitario, 2006.

[63] Obe R. O. y Hsu L. S., **PostGIS in Action**, Editorial Manning, 2011.

[64] Highsmith J. A., **Agile software development ecosystems**, Addison-Wesley Professional, 2002.

[65] Poppendieck M. y Poppendieck T., **Lean software development: an agile toolkit,** Addison-Wesley Professional, 2003.

[66] Botella Plana A., Munoz Bollas A. y Olivella González R., **Introducción a los sistemas de información geográfica y geotelemática**, Editorial UOC, España 2011.

[67] Segrelles S. J. A., **Geografía humana: fundamentos, métodos y conceptos**, Editorial Club Universitario, España, 2002.

[68] Cotos Yáñez J. M., **Sistema de información medioambiental**, Editorial Netbiblio, 2005.

[69] **Sitio oficial del Lenguaje de Programación Ruby.** http://www.ruby-lang.org/es/ (07/10/11 13:13)

[70] **Why Rails?**, http://www.foraker.com/why-rails/ (07/10/11 13:40)

[71] **ArcGIS**, http://www.esri.com/products (18/11/2011 18:00)

[72] Buschmann F., Meunier R., Rohnert H., Sommerlad P. y Stal M., **Pattern Oriented Software Achitecture: A System of Patterns Vol. 1**, Wiley, 1996.

[73] Openshaw, S., **The Modifiable Areal Unit Problem. Norwich: Geo Books**, 1984. Disponible en: http://qmrg.org.uk/files/2008/11/38-maup-openshaw.pdf

[74] Murguía, M. y Villaseñor J. L., **Estimating the quality of the records used in quantitative biogeography using presence-absence matrices**. Ann. Bot. Fennici, 2000.

[75] Laffan, S.W., Lubarsky, E. y Rosauer, D.F., **Biodiverse, a tool for the spatial analysis of biological and related diversit**y, Ecography, 33, 2010.

[76] **SIG vectoriales**, http://www.geogra.uah.es/gisweb/1modulosespanyol/IntroduccionSIG/GISI (03/12/2011 14:09)

[77] **SIG raster**, http://www.geogra.uah.es/gisweb/1modulosespanyol/IntroduccionSIG/GISModu (03/12/2011 14:20)

[78] Asociación de Geógrafos Españoles, **Grupo de Métodos Cuantitativos, Geografía teoríca y cuantitativa**, Universidad de Oviedo, 1986.

[79] Cotos Yáñez J. M., **Sistemas de información medioambiental**, Netbiblo, 2005.

[80] Bailey T. C., **A review of statistical spatial analysis in GIS**, Fotheringham, Y. y P. Rogerson (Eds). Spatial Analysis and GIS, 1994.

[81] Haining R., Designing spatial data analysis for geographic information systems, Fotheringham, Y. y P. Rogerson (Eds). Spatial Analysis and GIS, 1994.