DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACION GEOGRÁFICA AMBIENTAL BASADO EN UML

LUIS ALBERTO MUÑOZ RAMIREZ

NEYBER ARTURO VICTORIA CORRAL

MAESTRÍA EN CIENCIAS COMPUTACIONALES

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA

INSTITUTO DE ESTUDIOS SUPERIORES DE MONTERREY (MÉXICO)

CORPORACION UNIVERSITARIA AUTONOMA DE OCCIDENTE

CALI, 2003

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACION GEOGRÁFICA AMBIENTAL BASADO EN UML

LUIS ALBERTO MUÑOZ RAMIREZ

NEYBER ARTURO VICTORIA CORRAL

Proyecto para optar el titulo de Magíster en Ciencias Computacionales.

Director
FERNANDO ANTONIO MACHUCA B.
Ph. D. en Informática

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA

INSTITUTO DE ESTUDIOS SUPERIORES DE MONTERREY (MÉXICO)

CORPORACION UNIVERSITARIA AUTONOMA DE OCCIDENTE

CALI, 2003

Nota de aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Tabla de Contenidos

LISTA DE FIGURAS

LISTA DE TABLAS

1 INTRODUCCION

La Corporación Universitaria Autónoma de Occidente cuenta con el Centro de Estudios Ambientales para el Desarrollo Sostenible (CEADES), adscrito a la Vicerrectoría de Investigaciones, el cual tiene entre sus objetivos la aplicación de conocimientos y experiencia profesional e institucional para brindar asesoría a entidades territoriales y organismos públicos y privados en diferentes campos de la gestión ambiental como es la planificación, ordenamiento, etc. Es así como, a través de este centro, la Corporación Universitaria Autónoma con el Ministerio del medio Ambiente, la Universidad Nacional, La CVC, La Gobernación del Valle del Cauca y el Municipio de Yumbo, ejecutaron el plan de acción ambiental local, PAAL, para el municipio de Yumbo, el cual es "un conjunto de programas, proyectos y subproyectos que en respuesta a una planificación y una programación, se encaminan a la ejecución de obras y servicios que buscan modificar las condiciones ambientales del Municipio"[1]. De acuerdo a la información recolectada en este proyecto y en otros proyectos el Centro de Estudios Ambientales para el Desarrollo Sostenible ha detectado la necesidad de contar con un Sistema de Información Ambiental y como un subsistema un Sistema de Información Geográfica Ambiental que responda a las necesidades de disponer de información georeferenciada sobre flora, fauna, recursos hídricos, zonas de alto riesgo, zonas de alto impacto ambiental, etc.

Un "Sistema de Información Geográfica" (SIG, o en Inglés Geographic Information System, GIS) permite almacenar y administrar informaciones gráficas espaciales y datos alfanuméricos asociados a esta información gráfica en forma simultánea dentro de un mismo sistema. Un SIG, por lo tanto, integra una función gráfica de dibujo, que permite el despliegue o presentación (en papel o pantalla

de computadora) de las características espaciales de entidades gráficas, con una función de "administración" de base de datos, que maneja los datos alfanuméricos asociados a estas entidades gráficas. Esta integración permite la referencia cruzada, la unión, y lo que es más importante, la consulta de la información almacenada en la base de datos del SIG y la toma de decisiones..

Los Sistemas de Información Geográfica en la actualidad deben manipular simultáneamente dentro del modelamiento o solución de problemas, grandes volúmenes de información. Esta información fundamentalmente está constituida por dos tipos de archivos: georreferenciados y de atributos. Se denominan archivos georreferenciados, aquellos que contienen mapas (formatos vector o raster) y se denominan archivos de atributos a aquellos que tienen información asociada, con datos de tipo texto, numérico o alfanumérico.

Los Sistemas de Información Geográfica en la actualidad están construidos bajo dos modelos:

- El modelo relacional
- El modelo objeto - relacional

El principal problema de los sistemas manejadores de bases de datos de los SIG's ha sido el modelo de datos, que ha presentado fallas al momento de manejar aplicaciones no estándares en que se debe hacer abstracciones y manipular objetos complejos, la mayoría de los SIG's comerciales utiliza Sistemas Manejadores de Bases de Datos, SMBD, convencionales que almacenan y analizan registros, los cuales tienen ciertas deficiencias que dificultan las actividades de los SIG's:

- El tratamiento de objetos complejos no es soportado, lo que no permite manejar por ejemplo una manzana catastral conformada por predios y construcciones como un todo.
- No es posible la representación de aquellos datos que involucran objetos del mundo real que

implican descomposición artificial en partes más pequeñas.

- Al no tener la capacidad para manejar objetos complejos que deben ser trabajados normalmente unidos, los procesos de manipulación se hacen más lentos y complicados.

Estos SIG's orientados a manejar Bases de Datos, en los que simplemente se almacena información de objetos de similares características, están montados sobre modelos relacionales que organizan los datos en tablas o relaciones con columnas y filas, almacenando atributos de tal manera, que todos los valores para el mismo atributo son elementos de un dominio común y las filas son registros que describen una entidad.

Estos modelos no cuentan con el poderoso concepto de recursión[2] que se presenta cuando un polígono debe ser descompuesto en muchos polígonos y estas partes también pueden ser descompuestas, lo que es vital para los modelamientos de situaciones complejas que se manejan normalmente al trabajar objetos espaciales y sus divisiones.

Lo anterior se puede resolver con un sistema implementado con la filosofía de manejar objetos y no tablas, en los cuales un objeto físico, de cualquier complejidad, se representa tal como es en la realidad sin efectuar divisiones artificiales.

Por lo tanto, se plantea**: Hacer uso del UML (Unified Modeling Language) para el diseño e implementación de un Sistema de Información Geográfico Ambiental.**

2 MARCO TEORICO

2.1 Los Sistemas de Información Geográfica (SIG's)

El término "Sistema de Información Geográfica" (SIG) apareció a finales de los años 60. En sus orígenes los SIG's estaban orientados hacia trabajos de dibujo computarizado de mapas, actualmente, el campo es mucho más amplio. Un SIG ofrece un gran número funciones y una mayor flexibilidad que un sistema tradicional (tipo CAD) de dibujo de mapas y a través de diversos análisis permite la toma de decisiones en diversos ámbitos donde es aplicado.

La función clave en un SIG es su capacidad para almacenar y administrar informaciones gráficas espaciales y datos alfanuméricos asociados a esta información gráfica en forma simultánea dentro del mismo sistema. Un SIG, por lo tanto, integra una función gráfica de dibujo, que permite la presentación o despliegue (en papel o pantalla de computador) de las características espaciales de entidades gráficas, así como una función de "administración" de base de datos, que maneja los datos alfanuméricos asociados a estas entidades gráficas. Esta integración permite implementar una serie de operaciones como, la unión, la intersección, y lo que es más importante, la consulta de la información almacenada en la base de datos del SIG y a partir de ellas: la generación de nueva información.

Los SIG's incluyen generalmente un amplio número de funciones adaptables que pueden ser aplicadas a situaciones particulares. Por ejemplo, por medio de imágenes en computador superpuestas a mapas temáticos, un planificador del territorio puede usar un SIG para dibujar las manzanas y la red de calles de un área geográfica determinada, junto con la información asociada catastral, social, demográfica, de servicios públicos (agua, alcantarillado, electricidad, teléfonos, etc.) y otros servicios (Televisión por cable, Usuarios de Internet, telefonía celular, etc.); de igual manera, un especialista del medio ambiente o de los recursos naturales puede usar estas mismas herramientas para analizar áreas de diferente vegetación, recursos de agua, la topografía, el tipo de suelos, el uso actual y potencial de las tierras y la geología de diferentes regiones o finalmente, un

analista de mercados puede utilizar la herramienta para determinar las zonas de influencia, de distribución de un producto, de un distribuidor, una red de tiendas, mercados, hospitales, etc..

La siguiente lista ilustra algunas de las áreas donde se está empleando exitosamente los SIG's:

- Planeamiento Territorial Urbano y Rural
- Sistema de información de tierras (Uso actual, uso potencial)
- Sistemas catastrales urbanos y rurales
- Sistemas de servicios públicos
    - Agua potable (Tuberías, Desagüe, drenaje, caudales, etc.)
    - Electricidad (Torres, postes, cables, transformadores, aisladores, etc.)
    - Teléfonos fijos y móviles (Suscriptores, Repetidoras, zonas de influencia, etc.)
    - Televisión por Cable (Usuarios, Localización, Tarifas, etc.)
    - Distribución de gas (Ductos, usuarios, zonas de abastecimiento, etc.)
- Planeamiento de transporte y tráfico
- Análisis demográficos
- Exploración y registro de recursos naturales:
    - Geología
    - Bosques
    - Tierras
    - Aguas
    - Minas
    - Hidrocarburos
- Evaluación y Administración de Impacto Ambiental
- Aplicaciones a Negocios
    - Mercadeo de productos
    - Posicionamiento de marcas
    - Zonas de distribución
    - Zonas potenciales de expansión de un mercado
    - Hospitales

- Aplicaciones Militares

2.2 Nacimiento y Evolución de la Tecnología de los SIG's[3].

Los Sistemas de Información Geográficos (SIG's) surgieron fundamentalmente, por las necesidades de actualización cartográfica de países desarrollados como Canadá y Estados Unidos a principios de los años sesenta. Igualmente, se generó la necesidad del Análisis Espacial de la Información y la elaboración de mapas temáticos a través de computadoras.

La creación de los sistemas CAD (Computer Aided Desing/Drafting – Diseño asistido por computadora) promovió un avance significativo en el camino de desarrollo de los SIG's. Otro aspecto fundamental para la evolución de los SIG's ha sido el avance de la tecnología computacional en cuanto a capacidad, almacenamiento de la información, procesamiento de imágenes y manejo a altas velocidades.

El desarrollo de los SIG's ha sido muy costoso en los países desarrollados, ya que se alimenta y retroalimenta a través de tecnologías de alta inversión como son la Percepción Remota, la Ingeniería Satelital y los Sistemas de Posicionamiento Global (GPS).

Las universidades también han tenido participación significativa para el desarrollo de los SIG's. El MIT (Instituto Tecnológico de Massachusetts) desarrolló la tecnología CAD a finales de los años cincuenta y comienzo de los sesenta. Hacia 1965 en la Universidad de Iowa se creó la llamada "Revolución Cuantitativa", es decir el Análisis Geográfico Cuantitativo y uso del computador en investigación de Simulación Espacial. Igualmente, la Universidad de Washington se convirtió prácticamente en un centro de investigación y desarrollo de SIG's; gran cantidad de los posteriores centros de SIG, tales como el Laboratorio de Harvard para Gráficas por Computador y Análisis Espacial y el Laboratorio SIG de SUNY/Buffalo, fueron gerenciados por alumnos graduados de la Universidad de Washington.

En 1962 el doctor Horwood desarrolló con un grupo de estudiantes el "Mapping Program" como

parte de un proyecto prototipo para proveer herramientas de Planeación Urbana. De la misma forma, la Universidad de Carolina del Norte adelantó trabajos en Planeación Urbana y Usos de la Tierra, desarrollando un Modelo Celular Numérico para Información Espacial.

La Northwestern University igualmente desarrolló el SYMAP "Synagraphic Mapping System" el cual podía integrar diferentes datos espaciales en un mapa, una gráfica u otro medio de despliegue visual. El SYMAP sirvió de modelo para el desarrollo de posteriores SIG's. Esta tecnología abrió una visión acerca del uso del mapeo automatizado y la integración de estructuras de datos para análisis espacial, lo cual estimuló el trabajo con Relaciones y Modelos Espaciales. También el departamento de Arquitectura de Harvard desarrolló la tecnología de "Grid Cell" que se define como una área contenida dentro de una red o malla espaciada uniformemente, tanto horizontal como verticalmente y que es la base del procesamiento.

En 1967, en la Universidad de Oregon, un investigador en digitalización y superposición de polígonos creó una herramienta analítica llamada Map/Model, que fue un SIG ya orientado al usuario y de propósito general. Por la misma época, la Universidad de Minnesota en su Centro de Asuntos Regionales y Urbanos, desarrolló un Sistema de Información Geográfico llamado LMIC (Centro de Información para el manejo de tierras).

Los Sistemas de Información Geográficos comenzaron a ganar aceptación en múltiples campos en los años setenta. De esta manera se introdujeron varias aplicaciones dentro de las que se encuentran Planeación Urbana, Desarrollo Territorial, Estudios de Impacto Ambiental y Desarrollo Sostenible, Exploración y Explotación de minerales e hidrocarburos, Usos del Suelo, Estrategias de Mercadeo y Ventas, Navegación y Transporte, entre otros.

2.3 Sistemas de Información Geográfica

2.3.1 Definiciones.

- Un Sistema de Información Geográfica es un poderoso conjunto de herramientas para obtener,

almacenar, recuperar a voluntad, transformar y desplegar datos especiales del mundo real para determinados propósitos. Un SIG también se puede considerar como un sistema especializado de base de datos capaz de manipular información espacial. [4]

- Un SIG es un sistema computarizado, conformado por la colección organizada de equipos, programas, datos georeferenciados y personal. Todos estos trabajan en conjunto para el almacenamiento, análisis y despliegue de información espacial asociada a una base de datos de atributos.[5]

- Un sistema de información geográfica según la definición del National Center for Geographic Information and Analysis (NCGIA) es un "sistema de hardware, software y procedimientos elaborados para facilitar la obtención, gestión, manipulación, análisis, modelado, representación y salida de datos espacialmente referenciados, para resolver problemas complejos de planificación y gestión".[6]

- Cualquier sistema de información que permita:
    - Capturar, almacenar y mostrar información basada en su localización espacial.
    - Analizar los datos relacionados espacialmente como ayuda en la toma de decisiones.
    - Permitir la selección e intercambio de información con modelos analíticos capaces de establecer diferentes alternativas.[7]

De estas definiciones se puede considerar, esencialmente, al SIG como una tecnología aplicada a la resolución de problemas territoriales, aunque sus áreas de aplicación son muy variadas. En otras palabras, un SIG es potencialmente aplicable en cualquier área que requiera del manejo de información espacial. Los SIG's como sistemas de información, se crean para dar respuesta a preguntas no predefinidas de antemano, y por lo tanto, incluyen: una base de datos, una base de conocimientos (conjunto de procedimientos de análisis y manipulación de los datos) y un sistema de interacción con el usuario.

La característica fundamental de un SIG es su capacidad de análisis; es decir, no sólo capacidad para generar nueva información a partir de un conjunto previo de datos (mediante su manipulación y

reelaboración) sino, de relacionar elementos gráficos con elementos de una base de datos temáticos.

Se usan siempre datos con coordenadas conocidas, a estos se les denomina "datos espaciales georeferenciados". Se trabaja con mapas en formato digital, con escalas y coordenadas, En estos mapas la información puede ser analizada, modificada, presentada de diversas formas, etc. gracias a las herramientas computacionales. Además de la información cartográfica (los datos georeferenciados), el SIG maneja una base de datos donde se incluyen tantas características como las requieran los objetos digitalizados, ésta se llama "base de datos de atributos". De modo que se tienen dos tipos de datos estrechamente integrados: datos georeferenciados y datos de atributos.

En general la información espacial se muestra en forma de “capas” (layers) (fig. 2.1), en las que se pueden describir características como: la topografía, la disponibilidad de agua, los suelos, los bosques, el clima, la geología, la población, la propiedad de la tierra, los límites administrativos, la infraestructura (carreteras, vías férreas, sistemas de electricidad o de comunicaciones), los hospitales, los distribuidores, etc. Una de las funciones más importantes del SIG es la capacidad de combinar distintos capas en una sola operación, conocida con el nombre de “superposición”.

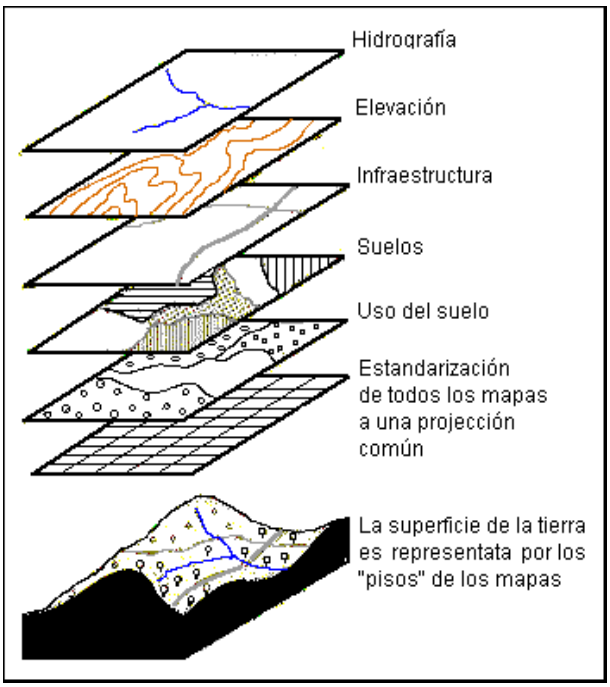


Fig. 2.1 Información almacenada por capas (layers) de una determinada región.

Básicamente, el SIG permite obtener una gran cantidad de información de distinto tipo, tratarla para

convertirla en conjuntos de datos compatibles, combinarlos y exponer los resultados sobre un mapa.

2.3.2 Operaciones con un SIG

Algunas de las operaciones estándar del SIG son:

- Integración de mapas trazados a escalas diferentes, o con proyecciones o leyendas distintas;
- Cambios de escala, proyecciones, leyendas, inscripciones, etc. en los mapas.
- Superposición de distintos tipos de mapas de una determinada zona para construir un nuevo mapa en el que se incluyen los datos de cada de ellos. Por ejemplo, un mapa de vegetación podría superponerse sobre un mapa de suelos, tal como aparece en la figura 2.2. Este a su vez podría colocarse sobre un mapa donde figure la duración del periodo vegetativo a fin de conseguir un mapa de idoneidad de la tierra para un determinado cultivo.[8]

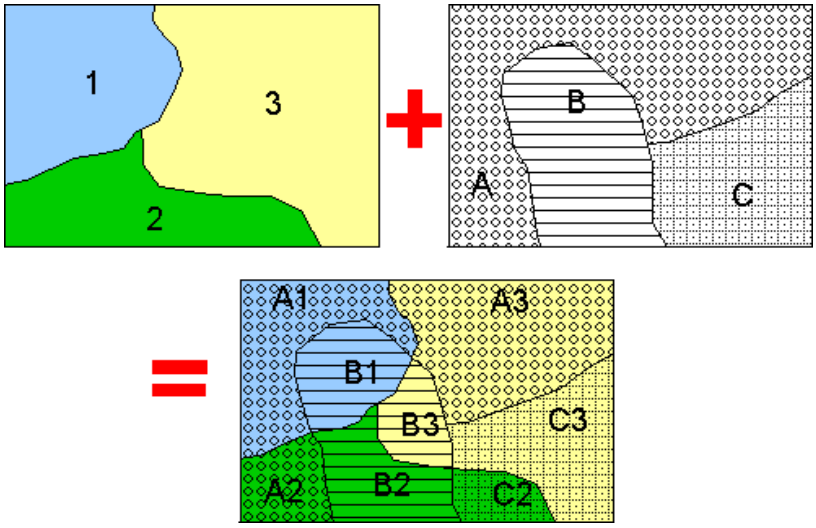


Fig. 2..2. Ilustración de cómo funciona la superposición. Un mapa con tres polígonos (áreas) y 3 clases, por ej. 1, 2 y 3 se sobreponen con otro mapa con otros 3 polígonos y 3 clases A, B, y C. El resultado de la superposición consiste de 8 polígonos con los nombres: A1. A2. A3. B1. B2. B3. C2 y C3

- Creación de zonas intermedias o próximas en torno a las líneas o polígonos de un mapa. Esta técnica se utiliza para buscar zonas a una distancia dada de las carreteras, ríos, etc., o

de ciertas condiciones temáticas. Estas zonas intermedias pueden a su vez utilizarse como otra capa de superposición;

- Preguntas de carácter espacial e informativo a través de bases de datos.

Existen preguntas típicas como:

- ¿Cuál es la capital de un país, departamento, etc.?
- ¿Cuál es el área de determinada región?
- ¿Cuál es la distancia entre dos lugares?
- ¿Dónde se produce tal o cuál producto?
- ¿Qué hay en una determinada región?

que pueden ser respondidas por un SIG, pero se puede igualmente obtener respuesta para preguntas más específicas y complejas:

- Cambios temporales. (Por ej. ¿Ha aumentado el área deforestada de una zona protegida en los últimos 10 años?, ¿Ha habido aumento/disminución de zonas de erosión?

- Patrones espaciales o de distribución. (Por ej. distribución de flora y fauna)

- También se pueden simular modelos y realizar nuevas consultas. ¿Qué pasaría si..... se abre una industria en este sitio, crecen las lluvias vertiginosamente en una región, se otorga un permiso de explotación de una mina en esta región, se descarga material contaminante en este río, cambio la localización de un distribuidor de productos, se construye un nuevo hospital, etc.?

Algunos ejemplos específicos de la utilización del SIG son:

- Facilidad de creación y actualización de la cartografía
- Manejo ágil y fácil de información catastral y demográfica

- Planificación del área de siembra de cultivos y plantaciones forestales basado en datos de usos potenciales y cobertura de la tierra actualizados
- Distribución de recursos en zonas de desastres
- Planificación de mercado de productos
- Manipulación y análisis de imágenes de sensores remotos para diversos usos
- Planificación de recursos naturales
- Evaluación y monitoreo del impacto de actividades humanas en el ambiente[9]

La calidad de la base de datos y su contenido, son determinantes para obtener información a través del SIG pues manejará la cantidad y calidad del resultado obtenido por el mismo.

2.3.3 Componentes principales del SIG

Los sistemas de información geográfica tienen tres componentes principales:

- Equipos,
- Programas informativos,
- Recursos humanos y la organización que hace que el sistema funcione.

2.3.3.1 Equipos.

Los componentes del equipo del SIG poseen una unidad central de proceso (CPU), discos para almacenar datos y programas, cintas utilizada para almacenar datos adicionales, pantalla de visualización y otras unidades periféricas (plotter, quemadores de CD's, impresoras, tabletas o mesas digitalizadoras, escáner, conexión a redes, etc.).

Entre los periféricos es importante destacar los de carácter especializado, entre ellos: la mesa o tableta digitalizadora, que se utiliza para convertir la información geográfica obtenida de los mapas en papel en datos digitales y enviarla a la computadora y el trazador de gráficos (plotter) para imprimir los mapas y otros gráficos del sistema.

2.3.3.2 Programas.

Los principales componentes del programa del SIG están destinados a desempeñar las siguientes funciones:

- Entrada de datos (digitalización e ingreso de los atributos utilizando un teclado);
- Almacenamiento de datos y gestión de la base de datos;
- Análisis y tratamiento de datos;
- Interacción con el usuario (edición de gráficos/mapas); y
- Salida y presentación de datos (representación gráfica).

2.3.3.2.1 Entrada de datos.

Incluye la conversión de datos procedentes de los mapas, la recolección de datos sobre el terreno (obtenidas por ejemplo a través de una GPS), las imágenes procesadas obtenidas mediante satélites y fotografías aéreas en datos digitales compatibles. La mayor parte de los SIG's utilizan un sistema de digitalización manual para introducir en el sistema los datos de los mapas, o sea que alguien debe sentarse ante el mapa colocado sobre una mesa digitalizadora y utilizando un mouse (ratón) especial, seguir las líneas que forman el mapa. Regularmente para introducir estos datos se utilizan capas (layers), en donde se separan los atributos gráficos, como por ejemplo una capa contiene curvas de nivel, otra ríos, otra quebradas, etc. y cuando se despliega el mapa se pueden seleccionar las capas de interés o se pueden superponer si es del caso.

También se pueden introducir datos cartográficos utiliza sistemas automatizados de digitalización como los dispositivos de exploración. Eliminan el trabajo manual de seguir las líneas y aseguran resultados coherentes y repetibles cada vez que se explora un mapa. La exploración es más rápida que la digitalización, pero solo pueden someterse a ese proceso los mapas de buena calidad e incluso así, el resultado del producto no es por lo general tan satisfactorio

La calidad de los datos introducidos influirá en la calidad de los productos del SIG independientemente de lo perfeccionado que sean su equipo y sus programas. En muchos casos,

los inventarios están incompletos o atrasados y hay que revisar la información de los mapas antes de digitalizarla.

2.3.3.2.2 Almacenamiento y Gestión de la base de datos

Consisten en determinar la forma en que los datos han sido estructurados y el sistema de preguntas, análisis e informes de los atributos relacionados con las características sobre los mapas.

2.3.3.2.3 Procesamiento de datos

Abarca dos tipos de operaciones: la primera, consiste en preparar los datos eliminando los errores o actualizándolos, o bien haciéndolos compatibles con otro conjunto de datos; la segunda, en analizar los datos para dar respuestas a las preguntas que pueda formular el usuario al SIG. El procesamiento de datos puede referirse a los aspectos espaciales y no espaciales de la información, o a ambos. Las operaciones típicas de elaboración automatizada de datos incluyen la superposición de diferentes mapas temáticos, la adquisición de información estadísticas sobre los atributos, el cambio de escala, la adaptación de los datos a las nuevas proyecciones, el calculo de las superficies y los perímetros y preparando perspectivas tridimensionales.

2.3.3.2.4 Interacción con el usuario (edición de gráficos/mapas)

Permite al usuario la organización y manipulación de los datos (gráficos, mapas y tablas) para satisfacer las necesidades de presentación de la información de acuerdo con sus requerimientos, generación de nueva información a través de las operaciones propias del SIG, correcciones, nueva información, etc.

2.3.3.2.5 Salida y presentación de los datos

Es la forma en que se presenta la información al usuario. Puede hacerse mediante despliegue en pantalla (visualización transitoria) o en mapa impreso (copia impresa) realizada con un plotter, o copia magnética o información impresa en forma digital.

2.3.4 Recursos humanos y organización

Cuando se describe un SIG se tiende a pensar en términos de equipos y programas como el sistema completo, sin pensar en las personas que hacen funcionar eficazmente todo el sistema. Aunque puede parecer que el SIG es la panacea para el planificador de recursos, es evidente que, en la realidad, no lo es tanto. Como sucede con cualquier sistema de computadoras, la información producida solo tiene el valor de los datos introducidos previamente. Una información incorrecta o insuficiente introducida en el SIG produciría respuestas incorrectas o insuficientes, aunque se tenga la mejor tecnología.

Al igual que en cualquier trabajo cartográfico, la recolección de datos y la introducción de los mismos en el sistema requieren una gran calidad de diseño y trabajo, una capacitación intensiva y una comprobación frecuente para controlar la calidad. En otras palabras, además de contar con equipos y programas adecuados para realizar el trabajo, la utilización eficaz del SIG requiere contar con personal suficientemente capacitado, así como con servicios de planificación, organización y supervisión, que permitan mantener la calidad de los datos y la integridad del producto final.

2.4 Etapas de construcción de un SIG. Tipos de Datos Gráficos[10]

Para trabajar un Sistema de Información Geográfica es posible adquirir datos gráficos de dos tipos:

- Datos vectoriales y
- Datos raster (celdas o cuadriculas).

2.4.1 Datos Vectoriales

Se obtienen mediante el proceso de digitalización, esto permite leer las coordenadas de cada punto del mapa y almacenarlas en archivos codificados, que serán interpretados luego, por el correspondiente software de construcción de gráficos. La información vectorial esta compuesta por tres clases de datos:

- Puntos,
- Líneas y
- Polígonos.

En un mapa digital para la representación cartográfica de los diferentes temas se pueden usar, por ejemplo, puntos para los municipios, veredas, corregimientos; líneas para calles, quebradas, ríos, vías férreas, límites administrativos, etc.; ó polígonos para zonas, departamentos, regiones de uso actual, de uso potencial, comunas, entre otros elementos.

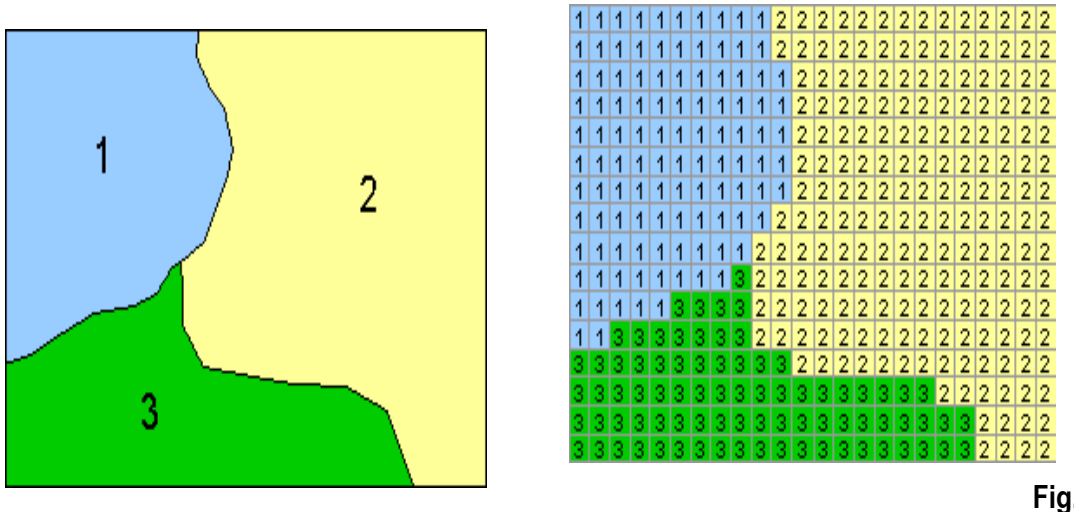


Fig. 2.3 Datos vectoriales y datos raster

En los sistemas vectoriales, el trabajo lineal se representa mediante una serie de segmentos rectos llamados vectores. Las coordenadas X, Y del final de cada vector se digitalizan y se almacenan de forma explícita, y las conexiones se indican mediante la organización de los puntos en la base de datos.

En los sistemas raster, con base en cuadrículas o de celdas, el mapa se representa en formato rectangular o en células rectangulares o cuadradas, a cada una de las cuales se le asigna un valor.

2.4.2 Datos Raster

El segundo tipo de información, (raster, cuadrícula o celda), se adquiere mediante el proceso de rasterización de la imagen, esto es, de codificar cada uno de los pixeles que conforman la imagen y

almacenar su posición dentro de la misma. Cada pixel en la pantalla será coloreado de acuerdo a su código y cada código representará una característica diferente.

Los datos raster, están especialmente disponibles para representar los fenómenos geográficos que varían continuamente en el espacio como: la elevación, inclinación, precipitación; pero pueden ser usados para representar tipos de información menos tradicionales tales como: densidad de población, comportamiento del consumidor y otras características demográficas. Las celdas también son datos ideales de representación para el modelo espacial, el análisis de flujos y tendencias sobre los datos representados como superficies continúas tales como: modelado de vertientes o los cambios dinámicos de población sobre el tiempo.

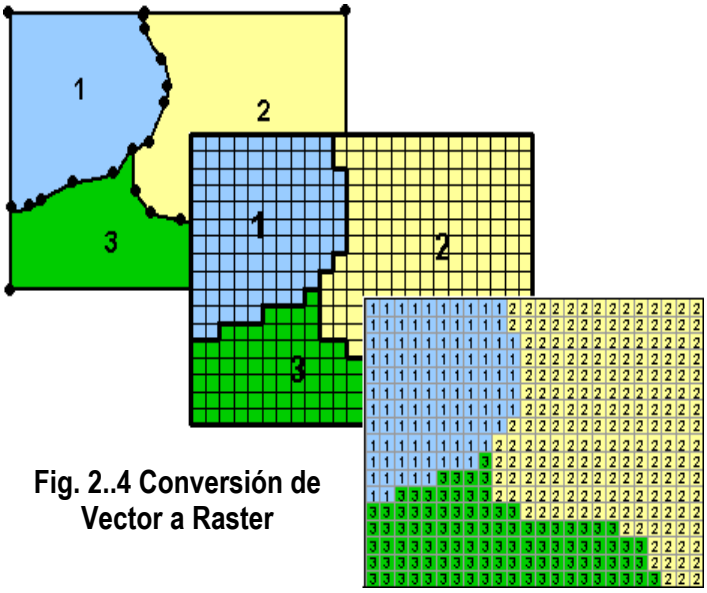


Fig. 2..4 Conversión de Vector a Raster

La mayoría de los SIG's tienen la capacidad de transformar los datos a partir de un formato, al otro.

La figura 2.4 ilustra la conversión de datos en formato vectorial a formato de cuadrícula o celda (raster a vector).

Cada uno de los sistemas presenta sus ventajas y desventajas, la tabla 2.1, muestra un resumen de las potencialidades y limitaciones de cada uno de los formatos de datos gráficos:

	Ventajas	Desventajas
Raster	• El tratamiento de los algoritmos es más sencillo y simple de escribir que en los sistemas por vectores. • Los sistemas de celda o cuadrícula son más adecuados para las entradas en forma de retícula como es el caso de las imágenes digitales de teledetección, y los sistemas del formato de celda, son más compatibles con los dispositivos de salida de forma reticular como las impresoras de líneas y muchas terminales gráficas.	• Los requerimientos para el almacenamiento son mayores que las de los sistemas vectoriales, • La representación de un recurso depende del tamaño de la celda, y resulta especialmente difícil presentar adecuadamente los rasgos lineales, como las curvas de nivel, las carreteras, las líneas férreas, etc., a menos que la cuadrícula sea pequeña y la mayor parte de los datos de entrada estén digitalizados en forma de vector y deben ser trasladados a formato celda para poder almacenarlos en un sistema de ese tipo.
Vector	• Se necesita mucha menos capacidad de almacenamiento que en los sistemas del formato de celda, • El mapa original puede representarse en su resolución original. • Múltiples atributos pueden ser fácilmente representados.	• Los algoritmos para las funciones realizadas son complejos y menos confiables que los sistemas del formato de celda, • Los datos espaciales de variación continua (como las imágenes por satélites) no pueden ser representados en forma de vector, y hay que convertirlos al sistema raster para procesar la información de ese tipo.

Tabla 2.1. Comparación entre los formatos de datos raster y vector

2.4.3 Digitalización

Para realizar el proceso de transformar la información gráfica en mapas digitales y en cartografía temática, se utiliza el software administrador del SIG, usando para la etapa de digitalización, una mesa o una tableta digitalizadora, con buena área activa y alta precisión, lo que posibilita, en la mayoría de los casos, trabajar con planos enteros evitando, la subdivisión para su digitalización.

La técnica de digitalización, consiste en indicar con la cruz del cursor y mediante la presión de una tecla del ratón, las coordenadas de cada punto o vértice de las líneas o arcos del plano. Esta

información de coordenadas en un sistema propio de la mesa o tableta digitalizadora es almacenada en archivos, que luego serán utilizados conforme sean requeridos en un determinado proyecto.

Fig. 2.5. Mesa digitalizadora Summagrid IV, de Summagraphics

2.4.4 Edición

En esta etapa, se edita la información digitalizada, permitiendo la corrección de los errores cometidos durante la etapa de digitalización, tales como:

- Incorrecta asignación de rótulos (labels)
- Arcos duplicados.
- Arcos que deberían cortarse y no lo hacen.
- Polígonos que no cierran.
- Nodos innecesarios.
- Suavizado de curvas.
- Agregado de arcos faltantes.

2.4.5 Topología

A continuación, se pasa a la etapa de generación de la topología, en la cual se logra que todas las entidades del mapa digital (áreas, líneas y puntos) cumplan ciertas condiciones necesarias de

relación entre sí, lo que permite identificar a cada polígono, arco o punto como una entidad individual dentro del conjunto. Requisito que posibilita, posteriormente, asignarle a cada entidad los atributos descriptivos correspondientes.

A cada entidad del mapa digital se le asigna una cantidad atributos diferentes; éstos se almacenan en bases de datos alfanuméricas que se interrelacionan con las entidades gráficas a través de un código que identifica de igual manera a ambas, y que luego permitirá ubicar un elemento en el espacio gráfico mediante la selección de uno o varios de sus atributos en la base de datos alfanumérica; o bien, indicando un elemento del mapa digital, encontrar inmediatamente sus atributos descriptivos en la base de datos alfanumérica.

2.5 Los Sistemas de Información Geográfica y las Empresas de Servicios Públicos.

Para las Empresas de Servicios Públicos Domiciliarios concretamente se desarrollaron los Sistemas AM/FM (Automated Mapping and Facilities Management)[11], que manejan, con elementos específicos, las Redes de Planta Externa, proporcionando capacidad de análisis espacial y manejo de datos alfanuméricos asociados a los puntos georeferenciados (con identificación de su posición).

Dentro de las empresas de telecomunicaciones, la United Telephone Company of Florida fue la primera compañía de teléfonos en Estados Unidos en completar la automatización de su base cartográfica y administración de la red. El proyecto se inició en 1982 y la conversión de datos se terminó en 1986. La meta principal fue proveer un sistema de registro único de tal forma que con sólo una digitación se actualizaría los datos de Ingeniería de Planta Externa, Contabilidad e Inversión. El SIG se usa a través de todo el proceso, comenzando con el diseño de la orden de trabajo y pasando por todas las herramientas para auditoría y estadística. En 1988 alrededor de 100 empresas de servicios públicos en Estados Unidos habían adoptado el sistema.

Igualmente, Southern Bell, Southwestern Bell y la British Columbia Tel., fueron líderes en la implantación de Sistemas AM/FM. En la South Bell, en 1990, se encontraba automatizado el 100% de la base cartográfica (dentro de un sistema AM/FM) y el 75% de la información de red.

Para la Pacific Bell, la justificación primaria para la implantación de la tecnología fue la significativa reducción en los costos de ingeniería, en segundo término la velocidad y la calidad en el procesamiento de la información de red y tercero, la imagen de la compañía como líder en tecnología dentro del ámbito de las telecomunicaciones.

Existe también una gran afinidad entre los SIG's AM/FM y los sistemas SCADA (Supervisory Control and Data Acquisition) que es un conjunto de software y hardware que permite un acompañamiento y recolección de datos de sistemas distribuidos en grandes complejos industriales. A través de estos sistemas, ingenieros y gerentes pueden hacer estadísticas en tiempo real, colocar alarmas contra fallas o dirigir procesos a distancia. Algunas aplicaciones de los mismos son:

- Control de Bombas y Pozos
- Campos petrolíferos y de gas natural
- Monitoreo de oleoductos y gasoductos
- Control de Acueductos y Desechos
- Bodegas Refrigeradas y Sistemas de Aire Acondicionado Sistemas contra incendios y de seguridad
- Criaderos e invernaderos
- Sitios de repetición en sistemas celulares, troncalizados y convencionales
- Monitoreo de Flujo, Nivel, Temperatura, Presión, Vacío
- Monitoreo de tanques
- Monitoreo de Condiciones Ambientales
- Subestaciones eléctricas
- Camaroneras y criaderos de pescado
- Detección de fugas
- Monitoreo de acidez (pH)
- Advertencia Comunitaria de Defensa Civil

2.6 CINTEL y los Sistemas AM/FM en Colombia.[12]

En el mercado actual las compañías están reduciendo y reorganizando su fuerza de trabajo para competir más efectivamente en un ambiente de cambios acelerados, así como evaluando y acondicionando los procedimientos de trabajo a través de reingeniería. Como una respuesta a este fenómeno necesario, CINTEL (Centro de Investigación de las Telecomunicaciones), asesora y

apoya el desarrollo e implantación de las aplicaciones de los Sistemas de Información Geográficos AM/FM en las empresas de Telecomunicaciones.

Los beneficios tradicionalmente cuantificables son el incremento de la productividad, tiempo real de actualización de datos, mayor exactitud en los mapas y registros de la base de datos, eliminación de datos redundantes, automatización en línea de los inventarios, mejor manejo de los recursos, capacidades de análisis y generación de ordenes de trabajo.

El trabajo de CINTEL apunta no solamente a identificar la inversión del proyecto sino también a definir completamente los procesos definitivos para la implantación de la tecnología computarizada AM/FM. Los Sistemas de Información Geográficos permiten simular tanto el desarrollo del proyecto como la financiación del mismo antes de ser implementado, lo que constituye uno de los elementos de balance en la relación costo/beneficio.

Durante 1995, CINTEL firmó convenio con ICONTEC para el establecimiento de la Secretaría Técnica de Normalización del Sector de las Telecomunicaciones, dentro del cual se conformó el Comité 383204 "Sistemas de Información Geográficos", del cual emanarán las normas para la implementación de la Tecnología AM/FM en las Empresas de Telecomunicaciones en Colombia. Se conformaron tres grupos: Normalización en Simbología y Planos, Normalización en Cartografía, y Normalización en Software Genérico.

2.7 El modelo relacional[13]

El modelo relacional presenta dos principales vacíos: primero, la falta de constructores adecuados

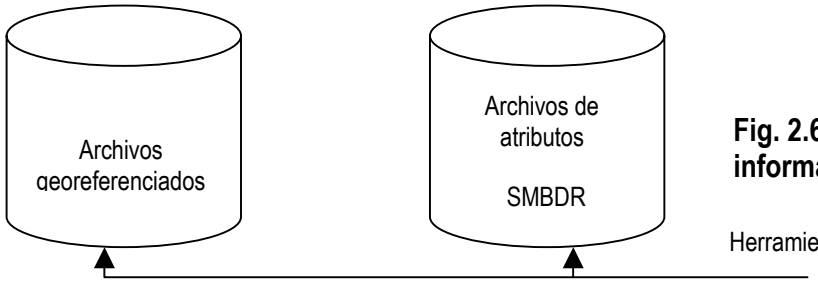


Fig. 2.6. Manejo de la información por un SIG actual.

para captar directamente la estructura del dominio que se quiere modelar; y segundo, el que sus lenguajes de manipulación no permiten desarrollar aplicaciones completas, lo que provoca que sea necesario recurrir a un lenguaje "host" conllevando el denominado "impedance mismatch". Las bases de datos orientadas a objetos intentan mejorar estos dos aspectos mediante la introducción de jerarquías y de la descripción no solo de las características atributivas de las entidades sino también de su comportamiento.

Por otra parte, una base de datos activa no es simplemente un depósito pasivo de datos que responde a consultas externas, sino un depósito de datos que pueda responder en forma autónoma y asíncrónica a eventos detectados en el medio y sobre la misma base de datos. Este tipo de actividad es importante para aplicaciones que requieran seguimiento continuo y reacción oportuna, por ejemplo, control de plantas de proceso, control de vuelos, prevención de riesgos, respuestas inmediatas de evacuación, administración de redes de comunicación y redes de generación de potencia, etc.

2.8 El modelo objeto - relacional[14]

Dadas las características del modelo anterior y el avance investigativo y tecnológico, las grandes empresas diseñadoras de software han buscado soluciones que permitan mejorar el modelo relacional. Para no desmontar completamente el modelo anterior (estrategia de mercado) se ha logrado integrar a éste características de la tecnología de objetos y en el mercado se ha introducido un modelo intermedio (híbrido) objeto - relacional.

Un sistema de gestión de base de datos objeto - relacional (SMBDRO) debe combinar las prestaciones más útiles tanto de la modelación de objetos como del almacenamiento relacional. Un SMBDRO debe ofrecer las siguientes posibilidades:

- Tipos de datos definidos por el usuario
- Las funciones de los tipos de datos definidos por el usuario deben ser flexibles y seguras.
- Mejor soporte de aplicaciones multimedia y grandes objetos de datos.
- Compatibilidad con SQL y los estándares actuales relativos a los SMBDR, así como con

nuevos estándares de objetos tales como CORBA[15] y DCOM.[16]

- Prestaciones avanzadas de servidor de base de datos, tales como optimización de consultas, escalabilidad y seguridad.

Los SMBD relacionales definen un conjunto limitado de tipos de datos: caracteres, números y fechas. Asimismo, las operaciones realizadas con estos tipos de datos están codificadas en el gestor de la base de datos.

Una base de datos objeto - relacional ofrece tipos de datos definidos por el usuario que permitan a las empresas y organizaciones ampliar la funcionalidad del servidor para gestionar las operaciones realizadas con estos tipos. Además, un tipo definido por el usuario debe interactuar con los otros datos del sistema de la misma manera que cualquier otro tipo de datos.

Por ejemplo, un SMBDRO debe permitir a los usuarios definir un tipo como, por ejemplo, "Pedido", que contiene información sobre el pedido de un cliente. Los atributos de este tipo serían similares a las columnas de una tabla relacional. En este tipo, los métodos definen cómo interactúa una aplicación con los datos, de la misma manera que los procedimientos almacenados se pueden definir en una base de datos relacional.

Una aplicación sólo tiene que solicitar el objeto de pedido deseado, y la base de datos devuelve toda la información asociada a ese pedido. La aplicación cliente no necesita relacionar los atributos del objeto cliente con la estructura de almacenamiento subyacente, que puede ser relativamente compleja.

Otro tipo es "Punto", que describe unas coordenadas X y Y. Puede haber funciones geométricas bidimensionales para ese tipo, como por ejemplo Distancia, Superficie o SuperficieEnRadioDado. Este nuevo tipo de datos debería poder utilizarse en consultas con otros tipos de datos de forma que puedan encontrarse respuestas a preguntas tales como "¿A qué distancia esta…? ¿Cuál es el área de…? ¿Qué distribuidores de un producto están situados en un radio de 2.000 metros del nuevo distribuidor propuesto?".

Un SMBDRO también permite a las empresas y organizaciones modelar objetos con cualquier nivel de detalle y a continuación referenciar esos objetos para crear objetos más complejos. Para que un sistema sea verdaderamente objeto - relacional, los tipos de datos definidos por el usuario deben proporcionar un mecanismo flexible y seguro para ampliar las funciones. Las siguientes características son esenciales para las extensiones de tipos objeto - relacional:

- Las extensiones de tipo deben estar enlazadas dinámicamente a los métodos, con el fin de que los tipos puedan ampliarse sin necesidad de recompilar el código del servidor.
- Las extensiones de tipo debe activarlas y ejecutarlas el cliente, el servidor de aplicaciones o el servidor de base de datos.
- Las extensiones de tipo deben proporcionar seguridad con el fin de que las funciones escritas por el usuario no pongan en peligro la seguridad del servidor.
- Las extensiones de tipo deben proporcionar "callback" al servidor de manera que las consultas u otras funciones puedan ejecutarse desde la función ampliada.

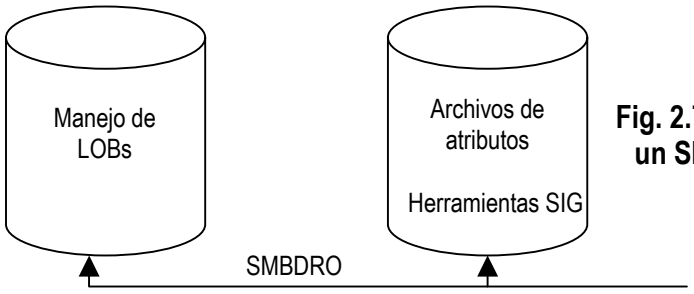


Fig. 2.7. Manejo de la información por un SIG, modelo objeto - relacional.

Las funciones de un tipo pueden ser procedimientos ejecutados en el servidor, o bien llamadas vinculadas dinámicamente y realizadas a aplicaciones codificadas mediante lenguajes de tercera generación. En la parte del servidor, las funciones se pueden escribir en PL/SQL; o en Java. Ambos protegen el código del servidor contra un posible bloqueo si la función suministrada por el usuario experimenta un fallo, y ambos se pueden añadir dinámicamente a un objeto. Cualquier función de un objeto tiene acceso a los datos del servidor para realizar consultas u operaciones de "callback".

Al añadir vídeo, imágenes y sonidos a los datos escalares tradicionales se mejora la utilidad de la aplicación, pero también aumenta la cantidad de almacenamiento que ocupa un objeto. El SMBDRO

debe gestionar objetos grandes (LOBs)[17] de manera eficaz y con un buen rendimiento. La base de datos también debe proporcionar a las aplicaciones clientes un acceso aleatorio y por partes a grandes objetos, con el fin de que sólo sea necesario recuperar a través de la red la parte solicitada de los datos. Los LOBs tales como vídeo, programas ejecutables, o incluso grandes bloques de texto se pueden almacenar en "tablespace" separados o en dispositivos independientes, como unidades de CD-ROM. Las empresas y organizaciones deben ser capaces de especificar los nombres de los sistemas de archivos para los objetos grandes almacenados externamente, o bien nombres de "tablespace" independientes para los objetos grandes almacenados en la misma base de datos que otros datos. Asimismo, uno o más LOBs pueden formar parte de un objeto y aprovechar cualquier función de objeto ampliada, independientemente de las características de su almacenamiento.

2.9 Sistemas manejadores de bases orientados a objetos (SMBDOO)[18]

2.9.1 Definición de un Sistema Manejador de Bases de Datos (SMBD)

Un SMBD es un conjunto de datos relacionados entre sí y un grupo de programas para tener acceso a esos datos. Las principales razones para emplear un SMBD son:

- **Tamaño**. Cuando se manejan grandes volúmenes de información, se requiere algún sistema que facilite el intercambio de información con memoria secundaria, la búsqueda rápida, etc.
- **Concurrencia**. Necesidad de un mecanismo de control sobre la información cuando sobre ella pueden existir varias personas o programas interactuando, de modo que coordine sus accesos para que no sean invalidados los trabajo que se realicen.
- **Persistencia**. Posibilidad de que la información permanezca aún después de desconectar el ordenador.
- **Recuperación e Integridad**. La información almacenada es importante y se hace necesario algún mecanismo que encargado de protegerla de pérdidas accidentales provocadas por fallos de energía, fallos de la propia aplicación, etc.

Además de lo anterior un SMBD suele proporcionar otras capacidades, tales como:

- **Distribución** Posibilidad para que la información pueda estar almacenada en lugares diferentes. Los usuarios no necesitan conocer dónde está la información, el SMBD la encuentra para ellos.
- **Seguridad.** Restricción al acceso a la información a usuarios no autorizados. Existen muchas posibilidades tales como listas de acceso, definición de niveles etc.
- **Administración.** Posibilita que los usuarios o administradores de bases de datos examinen, controlen y ajusten el comportamiento del sistema. Permite por tanto mantener las restricciones de integridad definidas por el usuario.

En cuanto a los lenguajes de la base de datos, un SMBD debe soportar lenguajes para los tres aspectos siguientes:

- **Definición de Datos (Data Definition Language).** Permitir a los usuarios definir nuevas estructuras y tipos de información.
- **Manipulación de Datos (Data Manipulation Language).** Permitir acceder a las instancias de dichas estructuras de información, permitiendo incluso su modificación.
- **Consultas (Query).** Permitir a los usuarios enunciar declarativamente la información que desean, siendo el propio SMBD quién se encargue de obtenerla. El soporte para la consulta suele incluir un motor que examinará la declaración de la información que se ha solicitado, elaborará un plan de ejecución para obtenerla, optimizará ese plan basándose en el conocimiento de la base de datos (índices, que información es local y cual es remota, etc.) y finalmente la ejecutará.

2.9.2 Características de los Sistemas Manejadores de Bases Orientados a Objetos (SMBDOO)

Un SMBDOO es un sistema de objetos y un sistema de bases de datos. Soporta:

- Identidad,
- Encapsulamiento,

- Clases,
- Herencia,
- Persistencia,
- Recuperación,
- Concurrencia, etc.

Se puede entonces decir que un SMBDOO es un SMBD que almacena objetos, permitiendo concurrencia, recuperación, etc. Esto quiere decir que se puede trabajar directamente con objetos, no teniendo que hacer la traducción a tablas o registros, también implica que los objetos se conservan, pueden ser gestionados, aunque su tamaño sea muy grande, compartidos entre múltiples usuarios, y se mantiene tanto su integridad como sus relaciones.

Las bases de datos tradicionales almacenan sólo datos, mientras que las bases de datos orientadas a objetos almacenan objetos, con una estructura arbitraria y un comportamiento dado.

Para aclarar la diferencia entre los dos modelos vale la pena observar el problema de almacenar un automóvil en un garaje al final del día. En un sistema de objetos el automóvil es un objeto, el garaje es un objeto, y hay una operación simple que es almacenar_el_automóvil_en_el_garaje. En un sistema relacional, todos los datos deben traducirse a tablas, de esta forma el automóvil debe ser desarmado, y todos los pistones almacenados en una tabla, todos los instrumentos eléctricos en otra, etc. Por la mañana, antes de irse a trabajar hay que componer de nuevo el automóvil para poder conducir.

El problema del almacenamiento en tablas implica:

- **Tiempo de desarrollo**. El tiempo empleado en generar el código para la traducción de objetos a tablas y viceversa.
- **Errores**. Debidos precisamente a esa traducción.
- **Tiempo de Ejecución**. Empleado en el ensamble/desensamble.
- **Inconsistencia**. Debida a que el ensamble/desensamble puede realizarse de forma diferente en las distintas aplicaciones.

Para solucionar en parte el problema algunas empresas productoras de software de bases de datos relacionales están incorporando capas de objetos sobre sus motores relacionales (basados en tablas), que son bastante útiles para la integración con aplicaciones y bases de datos orientadas a objetos, y que ayudan en la generación de algunos códigos de ensamble/desensamble, pero no mejoran en absoluto el tiempo de ejecución del problema.

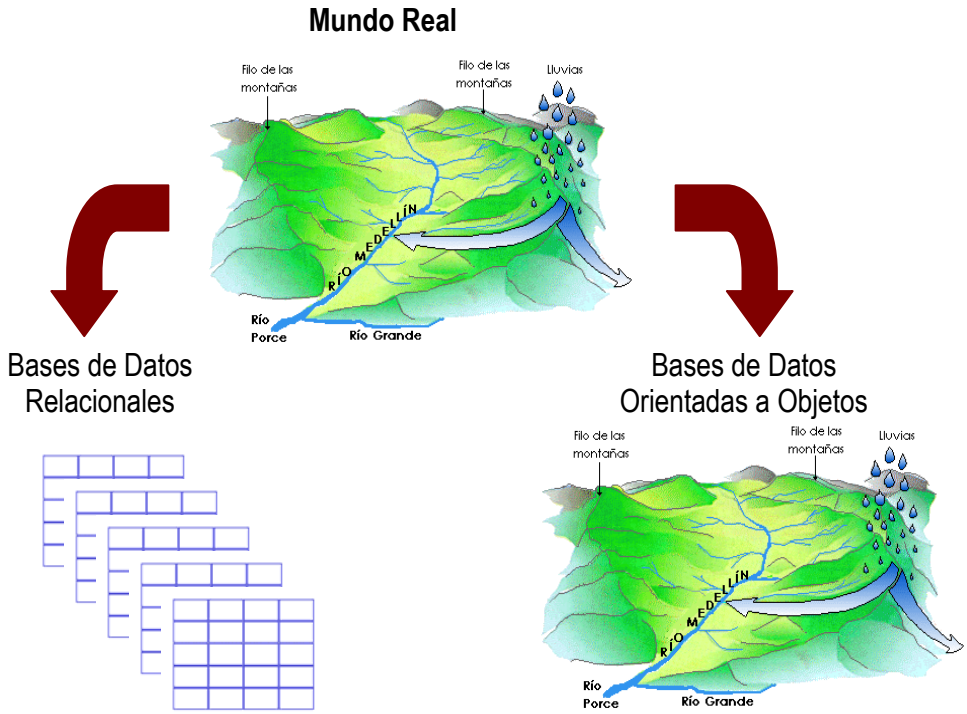


Fig. 2.8. Representación de la información en un SMBDR (Sistema Manejador de Bases de Datos Relacionales) y en un SMBDOO (Sistema Manejador de Bases de Datos Orientado a Objetos).

Un SMBDOO debe satisfacer dos criterios:

- Ser un sistema Orientado a Objetos, y
- Ser un sistema Manejador de Bases de Datos.

El primer criterio se traduce en ocho características generales:

- Abstracción,
- Encapsulamiento,
- Modularidad,
- Jerarquía,

- Control de Tipos,
- Concurrencia,
- Persistencia y
- Genericidad.

El segundo criterio se traduce en cinco características:

- Persistencia,
- Concurrencia,
- Recuperación ante fallos del sistema,
- Gestión del almacenamiento secundario y
- Facilidad de consultas

2.9.3 Sistema Manejador de Bases de Datos[19]

Las características básicas que ofrece un SMBD tradicional son:

2.9.3.1 Persistencia

Se refiere a la capacidad que tiene el programador para que sus datos se conserven al finalizar la ejecución de un proceso, de forma que se puedan reutilizar en otros procesos. Los datos además, tienen que persistir sin necesidad de una traducción explícita, es decir, sin que sea necesario que el usuario explícitamente los copie.

2.9.3.2 Concurrencia

Se relaciona con la existencia de muchos usuarios interactuando concurrentemente en el sistema. El sistema debe controlar la interacción entre las transacciones concurrentes para evitar que se destruya la consistencia de la base de datos. El sistema debe proporcionar el concepto estándar de atomicidad de una secuencia de operaciones, es decir, debe ofrecer la seriación de las operaciones, pero también otras alternativas menos estrictas.

En el caso de que la base de datos del sistema sea distribuida, hay que prestar especial atención al

mecanismo de control de la consistencia de los datos, de forma que además de controlar las transacciones concurrentes sobre un determinado nodo, hay que controlar la forma en que se reactualizan esos datos con relación al resto de los nodos.

2.9.3.3 Recuperación

El sistema debe proveer mínimamente el mismo nivel de recuperación que los sistemas de bases de datos actuales. De forma que, tanto en caso de fallo de hardware o de software, el sistema pueda retroceder hasta un estado coherente de los datos. Los fallos de los que un sistema debe ser capaz de recuperarse pueden producirse por diferentes motivos, por ejemplo: Interrupción en el suministro de energía, de forma que se pierda la información almacenada en la memoria principal y en los registros de uso general y errores de software, debido a que los resultados generados son incorrectos, lo que provoca respuestas erróneas a los usuarios y que la base de datos entre en un estado incongruente.

Una vez que el subsistema de recuperación de la base de datos ha detectado el fallo, procede a su clasificación (ya que cada uno de ellos debe manejarse de manera diferente), para posteriormente restaurar la base de datos al estado anterior al de la ocurrencia del fallo.

2.9.3.4 Gestión del Almacenamiento Secundario

La gestión del almacenamiento secundario es una característica clásica de los SMBD. Esta gestión es soportada por un conjunto de mecanismos que no son visibles al usuario, tales como gestión de índices, agrupación de datos, selección del camino de acceso, optimización de consultas, etc. Estos mecanismos evitan que los programadores tengan que escribir programas para mantener índices, asignar el almacenamiento en disco, o trasladar los datos entre el disco y la memoria principal, creándose de ésta forma una independencia entre los niveles lógicos y físicos del sistema.

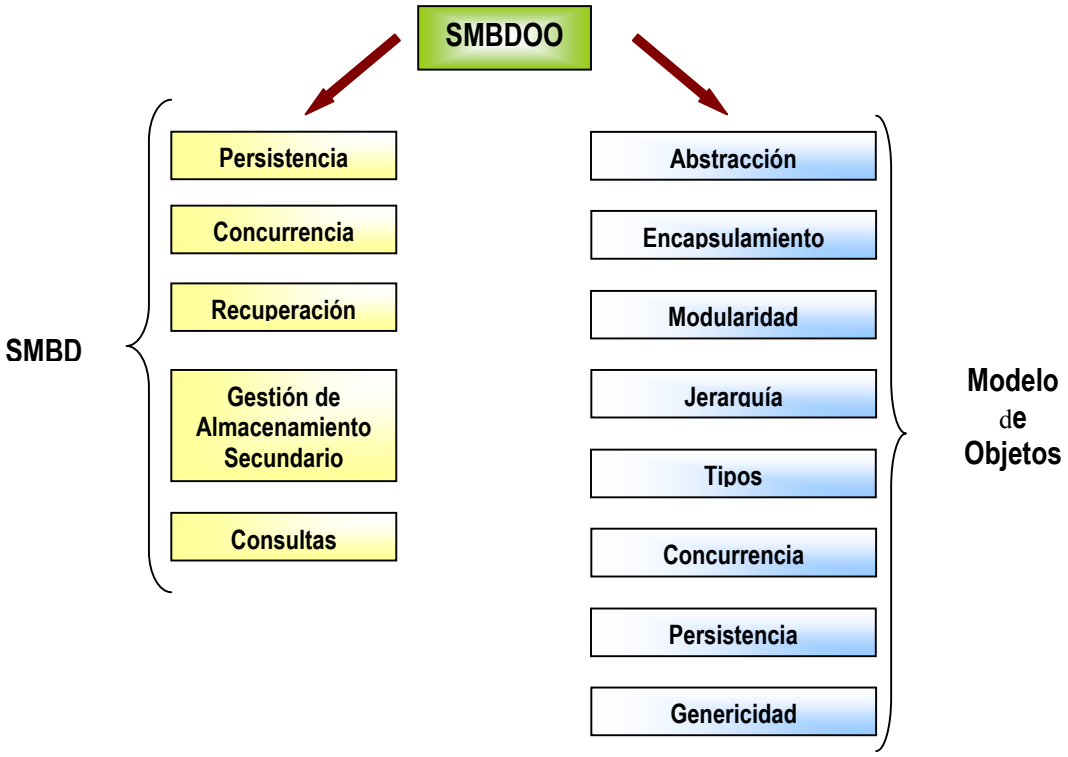


Fig. 2.9. Características básicas de un SMBDOO

2.9.3.5 Facilidad de Consultas

El sistema debería proporcionar la funcionalidad de un lenguaje de consulta *ad hoc*, es decir, permitir al usuario hacer cuestiones sencillas a la base de datos. Este tipo de consultas tienen como misión proporcionar la información solicitada por el usuario de una forma correcta y rápida.

2.9.3.6 Lenguajes de Consulta

En los SMBDOO los lenguajes de consulta constituyen una funcionalidad importante. Estos lenguajes de consulta deben satisfacer cuatro criterios[20]:

- Ser de alto nivel, es decir, deben ser capaces de expresar consultas no triviales en pocas palabras.
- Ser declarativos, es decir, que la atención se dirija hacia el QUÉ se desea, y no hacia CÓMO se obtiene.

- Ser eficientes, esto es, la formulación de la consulta debe permitir alguna forma de optimización.
- Ser independientes de la aplicación es decir, debería trabajar sobre cualquier base de datos.

Pero a diferencia de los SMBD relacionales, en los sistemas orientados a objetos, los lenguajes de consulta no constituyen la única forma de acceder a los datos. En los SMBDOO también se puede acceder a los datos de forma navegacional. Esta forma de acceso se basa en los identificadores de los objetos y en la jerarquía de agregación, de forma que dado un IDO, el sistema accede al objeto directamente y *navega* a través de los objetos a los que se refieren sus atributos.

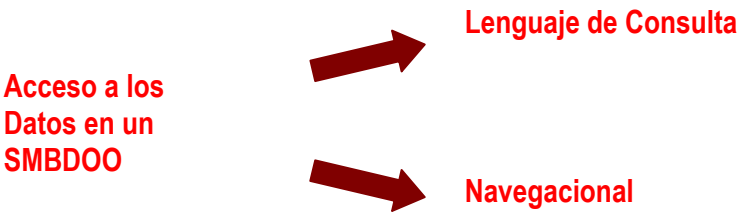


Fig. 2.10. Tipos de accesos

Estos dos tipos de acceso se pueden emplear de forma complementaria. Una consulta convencional puede ser útil para seleccionar un conjunto de objetos (devuelve los identificadores de objetos) que luego serán accedidos y procesados por las aplicaciones usando las capacidades navegacionales.

2.9.4 Falta de correspondencia

Uno de los principales problemas que se destacan al aplicar la orientación a objetos a los SMBD es la falta de correspondencia (“impedance mismatch”) o diferencia existente entre el sistema de tipos del lenguaje de programación y el sistema de tipos de los lenguajes de definición y manejo de datos del SMBD.

Esta falta de correspondencia se centra principalmente en el hecho de que los lenguajes de consulta utilizan conjuntos como unidades computacionales lógicas, mientras que, los lenguajes de

programación generalmente utilizan un razonamiento basado en una única entidad (registro). Estas diferencias exigen a los programadores el conocimiento de los dos lenguajes, de forma que puedan trasladar las estructuras de datos del lenguaje de programación a las estructuras de datos de los lenguajes de definición y manejo de datos. Por eso es muy importante solucionar este problema, no sólo para lograr la uniformidad en el formalismo entre los lenguajes involucrados, si no para aumentar la productividad del programador.

La solución a este problema de correspondencia requiere la integración de la tecnología de los lenguajes de programación con la tecnología de las bases de datos.

2.10 El Modelo Orientado a Objetos

El modelo orientado a objetos posee cuatro características fundamentales:

- **Abstracción**

  La abstracción denota las características esenciales de un objeto que lo distinguen de todos los demás tipos objeto, y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador". Una abstracción se centra en la visión externa de un objeto, y, por tanto sirve para separar el comportamiento esencial de un objeto de su implantación.

- **Encapsulamiento o encapsulación**

  La abstracción y el encapsulamiento son conceptos complementarios: la *abstracción* se centra en el comportamiento observable de un objeto, mientras que el encapsulamiento se centra en la implementación que da lugar a este comportamiento. El encapsulamiento se consigue a menudo ocultando información, es decir, se basa en ocultar todos los secretos de un objeto que no contribuyen a sus características esenciales. El encapsulamiento proporciona, por tanto, barreras explícitas entre abstracciones diferentes.

Existen dos visiones diferentes del encapsulamiento, la primera y original que es la del lenguaje de programación; y la segunda que es la adaptación de esa visión para la base de datos.

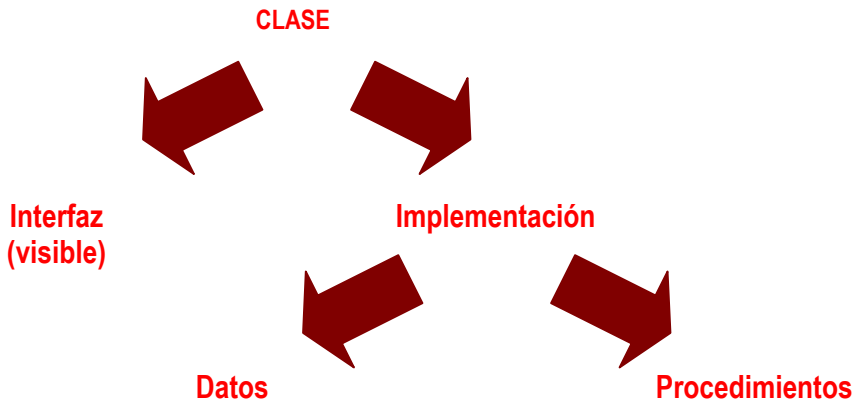


Fig. 2.11 Componentes de una clase

La idea de encapsulamiento en los lenguajes de programación proviene de los tipos abstractos de datos. Desde este punto de vista una clase tiene una parte de interfaz y una parte de implementación (Fig. 2.11). La *interfaz* es una especificación del conjunto de operaciones que pueden actuar sobre las instancias de una clase. Es la única parte visible del objeto. La parte de la *implementación* incluye una parte de datos (que describe la estructura interna de los datos de los objetos) y una parte de procedimientos (describe en algún lenguaje de programación la implementación de cada operación).

Desde el punto de vista de las bases de datos, esto se traduce en el hecho de que un objeto abarca operaciones y datos, pero con una diferencia. En las bases de datos no está claro si la parte estructural es parte de la interfaz, mientras que en los lenguajes de programación la estructura de datos es claramente parte de la implementación y no de la interfaz.

Como se puede observar, el encapsulamiento proporciona una forma lógica de independencia de los datos, ya que se puede cambiar la implementación de un tipo sin cambiar ninguno de los programas

que usan ese tipo.

El encapsulamiento real sólo se obtiene cuando las operaciones son visibles y el resto del objeto está oculto. Sin embargo, el uso del sistema se puede simplificar significativamente si no se obliga a un estricto encapsulamiento. Un ejemplo de una de estas situaciones es el manejo de las consultas. Las consultas se expresan a menudo en términos de predicados sobre los valores de los atributos. Por tanto, la mayoría de los SMBDOO permiten el acceso directo a los atributos, incluyendo "operaciones definidas por el sistema", las cuales leen y modifican estos atributos.

2.11 Métodos

En los SMBDOO, como se ha dicho anteriormente, los objetos sólo se manipulan con los métodos. Un método se define por: el encabezamiento, que especifica el nombre del método, los nombres y clases de sus argumentos y la clase del resultado (si devuelve alguno); y el cuerpo, que representa la implementación del método, y consiste en un conjunto de instrucciones expresadas en lenguaje de programación. Los diversos SMBDOO utilizan diferentes lenguajes de programación:

SMBDOO	Lenguaje de Implementación	SMBDOO	Lenguaje de Implementación
ORION[21]	LISP	Vbase/Ontos[22]	C++
GemStone[23]	Extensión de Smalltalk	Versant[24]	C++
O2[25]	C	Object Store[26]	C++

Tabla 2.2. Lenguajes de implementación

Algunos sistemas, como ORION[27], no necesitan especificar en el encabezamiento las clases de los argumentos, en otros, por el contrario, la especificación es opcional, y en otros obligatoria.

2.12 Acceso y Manipulación de los Atributos.

Como ya se ha dicho anteriormente, algunos SMBDOO violan el principio de encapsulamiento, permitiendo leer y escribir directamente los valores de los atributos de un objeto. El objetivo de esto

es facilitar el desarrollo de algunas aplicaciones que simplemente necesiten acceder a los atributos de un objeto o modificarlos. Este acceso directo a los atributos de los objetos presenta dos ventajas:

- Evitarle al programador tener que desarrollar un gran número de métodos que son, por lo general, bastante convencionales.
- Aumentar la eficiencia de las aplicaciones debido a que el acceso directo a los atributos está implementado por operaciones que ofrece el sistema.

Por otro lado, esta violación del encapsulamiento puede causar problemas si se llega a modificar la definición de los atributos de un objeto. Como se puede apreciar existen dos posturas opuestas a las que los sistemas ofrecen diferentes soluciones. Algunos sistemas, como Vbase/Ontos incluyen métodos implementados a bajo nivel por el sistema para la lectura y escritura de los atributos de los objetos. Otros sistemas, como O2, permiten especificar en la interfaz que atributos y métodos son visibles y pueden invocarse desde afuera. Estos métodos y atributos se denominan públicos. Inversamente, los atributos y métodos no visibles desde afuera son privados. Por último, en otros sistemas, como ORION, se puede acceder directamente a todos los atributos, lo mismo para leer que para escribir, y todos los métodos pueden ser invocados. En ORION lo que sí existen son mecanismos de autorización que pueden utilizarse para proteger el acceso a ciertos atributos y evitar la ejecución de ciertos métodos.

2.13 Modularidad

La modularidad es "la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos (agrupan abstracciones que tienen cierta relación) y débilmente acoplados (minimizan las dependencias entre módulos)". Se basa en el concepto de fragmentación de los programas en componentes individuales para reducir su complejidad en algún grado, y para crear además una serie de fronteras bien definidas y documentadas dentro del programa, dónde estas fronteras o interfaces tienen un incalculable valor para a la comprensión del programa.

La mayoría de los lenguajes que soportan el módulo como un concepto adicional distinguen también entre la interfaz de un módulo y su implementación. Se puede decir entonces que encapsulamiento y

modularidad son dos conceptos que van de la mano. De esto se deduce que los conceptos de abstracción, encapsulamiento y modularidad son sinérgicos. Un objeto proporciona una frontera bien definida alrededor de una sola abstracción, y tanto el encapsulamiento como la modularidad proporcionan barreras que rodean a esta abstracción.

2.14 Jerarquía

Es "una clasificación u ordenación de abstracciones". Frecuentemente un conjunto de abstracciones forma una jerarquía, y la identificación de esas jerarquías simplifica la comprensión de los problemas.

Las dos jerarquías más importantes en un sistema complejo son su estructura de clases ("jerarquía de clases") y su estructura de objetos (la jerarquía "de partes"). Posee además tres características secundarias, lo que quiere decir que son útiles al modelo, pero no esenciales.

2.15 Tipos (control de tipos)

Existen dos categorías principales de sistemas orientados a objetos, los que soportan el concepto de clase y los que soportan el concepto de tipo. Un tipo en un sistema orientado a objetos se corresponde con el concepto de tipo abstracto de datos. Es un conjunto de objetos que tienen un mismo comportamiento (comparten una misma funcionalidad) que se puede observar desde afuera. Esto significa que el tipo al cual un objeto pertenece depende de qué operaciones puedan invocarse sobre el objeto, cuál es el orden y tipo de sus argumentos y cual es el tipo del resultado. En los lenguajes de programación, los tipos son medios para incrementar la productividad de los programadores, garantizando la corrección de los programas al forzar a los usuarios a declarar los tipos de las variables y de las expresiones que emplean, de forma que se pueda chequear la información que se deposita en ellas. Este chequeo de tipos se realiza principalmente en tiempo de compilación.

El concepto de clase es diferente al de tipo. Una clase es un conjunto de objetos, los cuales tienen

exactamente la misma estructura interna y, por tanto, los mismos atributos y métodos. La especificación es la misma, pero es un concepto de tiempo de ejecución. Las clases no se emplean para chequear la corrección de los programas, pero sí para la creación y manipulación de objetos. En la mayoría de los sistemas que emplean el mecanismo de clases, las clases se pueden manipular en tiempo de ejecución, es decir, pueden ser modificadas o pasadas como parámetros. De esta forma las clases proporcionan una mayor uniformidad y flexibilidad al sistema, pero hacen imposible el chequeo de tipos en tiempo de compilación.

A menudo los conceptos de tipo y clase se utilizan indistintamente. Sin embargo, cuando ambos están presentes en el mismo lenguaje, se puede decir que las clases implementan a los tipos. Podemos entonces concluir afirmando que los tipos son la puesta en vigor de la clase de los objetos, de modo que los objetos de tipos distintos no pueden intercambiarse o, como mucho, pueden intercambiarse de formas muy restringidas.

2.16 Concurrencia

La concurrencia es una característica que los SMBDOO heredan tanto de los SMBD como del modelo de objetos. La concurrencia heredada del modelo de objetos hace referencia a la propiedad de distinguir un objeto activo de otro que no lo está, dónde un objeto activo es aquel que puede representar un hilo separado de control (abstracción de un proceso). En un sistema basado en diseño orientado a objetos, se puede conceptualizar el mundo como un conjunto de objetos cooperativos, algunos de los cuales son activos y sirven así como centros de actividad independiente.

Si se introduce concurrencia en nuestro sistema hay que considerar cómo los objetos activos sincronizan sus actividades con otros, así como con objetos puramente secuenciales. Por ejemplo, si dos objetos activos intentan enviar mensajes a un tercer objeto, hay que tener la seguridad de que existe algún mecanismo de exclusión mutua, de forma que el estado del objeto sobre el que se actúa no está corrupto cuando los dos objetos activos intentan actualizarlo simultáneamente. En presencia de la concurrencia no hay que definir simplemente los métodos de un objeto; hay que asegurarse

también de que la semántica de estos métodos se mantiene a pesar de la existencia de múltiples hilos de control.

2.17 Persistencia

La persistencia es otra de las características que los SMBDOO heredan tanto de los SMBD como del modelo de objetos. La diferencia está en que la persistencia proporcionada por el SMBD tradicional, se refiere únicamente a la conservación de los datos, mientras que la persistencia heredada del modelo de objetos hace referencia no sólo a la conservación del estado de un objeto, si no también a la conservación de la clase, que debe trascender a cualquier programa individual, de forma que todos los programas interpreten de la misma manera el estado almacenado.

El OMG (Object Management Group)[28] establece la siguiente afirmación: "Los objetos persistentes se utilizan para representar el estado a largo plazo de una computación en ejecución, como por ejemplo el conjunto de empleados de una corporación y las relaciones entre los mismos. Tales instancias poseen, por ejemplo, la cualidad intrínseca de sobrevivir al proceso que las creó; incluso sobreviven a la extinción temporal de la CPU que las generó."

Según la definición anterior, vemos que se distingue entre una persistencia en el tiempo, y una persistencia en el espacio. La persistencia en el espacio hace referencia al hecho de que los objetos creados en una máquina puedan llevarse a otra, y que incluso puedan tener representaciones diferentes en diferentes máquinas.

La persistencia de un sistema de objetos se implementa, en la práctica, proveyendo a éstos de un mecanismo cuyo objetivo básico consiste tanto en el almacenamiento de objetos existentes en la memoria como en la recuperación posterior a éste de los mismos. La eficiencia es un factor crítico fuertemente ligado a tal mecanismo, que supone la preservación, no sólo de los datos estrictamente contenidos en un objeto sino también de la identidad del mismo, así como de las relaciones, apuntadores y referencias a otros objetos.

Se define entonces la persistencia, como la cualidad de un determinado objeto de almacenar y

recuperar eficiente, automática y selectivamente la estructura compleja de objetos relacionados con el mismo (se incluye aquí la auto-relación). De acuerdo con tal definición, la aplicación de la persistencia a un objeto inmerso en una estructura equivaldría a la impulsión de un primer mensaje que a su vez y en razón de un mecanismo de extensibilidad posiblemente generaría otros mensajes dirigidos a objetos de la estructura, dando lugar al trazado figurado de un complejo grafo derivativo, y transformándose así en un mensaje que recorrería la estructura de objetos procediendo, bien al almacenamiento bien a la recuperación en memoria de los objetos calificados como persistentes.

Un aspecto importante de la persistencia en los SMBDOO son las modalidades bajo las que los objetos se hacen persistentes y bajo las que pueden eliminarse. Existen dos enfoques básicos:

- La persistencia es una característica implícita de todas las instancias de las clases. La creación de una instancia tiene el efecto de insertarla en la base de datos. Por tanto, la creación del objeto automáticamente implica su persistencia. Esta modalidad es la más simple, y es típicamente utilizada en sistemas en los que las clases tienen también una función extensional.

- La persistencia es una característica ortogonal. La creación de la instancia no tiene el efecto de insertarla en la base de datos. Una instancia se crea durante la ejecución de un programa y se elimina al terminar éste, a menos que se haya hecho persistente. Un mecanismo para hacer la instancia persistente puede ser asignarle un nombre dado por el usuario, o bien insertarla en una colección de objetos persistentes.

Se puede adoptar un enfoque intermedio entre ambos, como puede ser dividir las clases en persistentes y temporales, de forma que, las instancias de las clases persistentes se crean automáticamente como persistentes, pero las temporales no.

En cuanto a la eliminación de objetos persistentes existen también dos modalidades:

- La primera, implica tener una operación explícita para borrar. La capacidad de borrar crea problemas con la integridad referencial (si se elimina un objeto al que otros estaban

apuntando, las referencias de éstos se invalidan). Una posible solución puede ser utilizar un contador de referencias, que determine si un objeto está siendo referenciado por otros objetos antes de borrarlo. Otra posibilidad consiste en no mantener ninguna información adicional y permitir las operaciones de borrado libremente. De esta forma, la operación de borrado es eficiente, pero las referencias a objetos borrados causarán excepciones por lo que se requerirá código adicional tanto en las aplicaciones como en los métodos para tratar las excepciones. En este último caso, los IDOs de los objetos borrados no podrán ser reutilizados.

- La segunda, se basa en no tener una operación explícita para borrar, si no en que un objeto persistente se cancele cuando se eliminen todas las referencias y nombres externos asociados a él. Esto asegura la integridad de las referencias.

Y por último, existe una característica discutida según diversos autores:

- **Genericidad**

Es la propiedad que permite construir clases genéricas para otras clases. Una clase genérica o parametrizada es una que sirve de modelo para otras clases (un modelo que puede parametrizarse con otras clases, objetos y/o operaciones). Una clase parametrizada debe ser instanciada (sus parámetros deben ser rellenados) antes de que puedan crearse los objetos.

2.18 Características del Modelo Derivadas de los Conceptos de Objeto y Clase

2.18.1 Características derivadas del concepto de objeto

Un objeto tiene estado, comportamiento e identidad; la estructura y comportamiento de objetos similares están definidos en su clase común; los términos instancia y objeto son intercambiables.

- El estado de un objeto abarca todas las propiedades (normalmente estáticas) del mismo

más los valores actuales (normalmente dinámicos) de cada una de esas propiedades.

- El comportamiento es cómo actúa y reacciona un objeto, en términos de sus cambios de estado y paso de mensajes, es decir, representa su actividad visible y comprobable exteriormente.
- La identidad es aquella propiedad de un objeto que lo distingue de todos los demás objetos.

Un objeto por si mismo es poco interesante. Los objetos contribuyen al comportamiento de un sistema colaborando con otros. Existen dos tipos de relaciones entre objetos de especial interés:

- Enlaces
- Agregación

2.18.2 Identidad

La identidad de los objetos siempre ha existido en los lenguajes de programación. El concepto es más reciente en bases de datos. La idea es que en un modelo con identidad de los objetos, la existencia de un objeto es independiente de su valor. Cada objeto se representa por un IDO (Identificador de objeto) que es único dentro de toda la base de datos. Estos IDOs no tienen que ser gestionados por el programador ya que son implementados a bajo nivel por el sistema, lo que a su vez incrementa el rendimiento del mismo. De esta forma dos objetos serán diferentes si tienen IDOs diferentes, aunque sus atributos tengan los mismos valores. Así, dos objetos pueden ser idénticos (son el mismo objeto, tienen el mismo IDO) o iguales (tienen el mismo valor). Esto implica que dos objetos idénticos son a su vez iguales, mientras que lo inverso no es cierto.

La identidad de los objetos es además una potente primitiva que puede ser la base para el manejo de tuplas, conjuntos, y objetos complejos recursivos. También permite operaciones tales como asignación o copia de objetos.

Hay algunos modelos en los cuales se permiten objetos y valores, y donde no todas las entidades se representan como objetos. Un valor, es autoidentificable y no tiene ningún IDO asociado a él. Todas

las entidades primitivas, tales como enteros o caracteres, se representan por valores, mientras que las entidades no primitivas se representan por objetos.

Los modelos basados en la identidad son los normales en los lenguajes imperativos, ya que cada objeto que se manipula en un programa tiene una identidad y puede ser modificado. Esta identidad la proporciona el nombre de la variable o bien una asignación física en memoria. Pero este concepto es bastante nuevo en los sistemas relacionales puros, donde las relaciones se basan en valores. Se puede simular la identidad de objetos en un sistema basado en valores introduciendo identificadores explícitos de objetos, pero esta posibilidad ocasiona al usuario una carga, ya que él será el que se encargue de evitar la duplicidad de los identificadores de objetos y de conservar la integridad referencial.

2.18.3 Construcción de los Identificadores de Objetos

Los identificadores de objetos (IDOs) pueden ser físicos o lógicos.

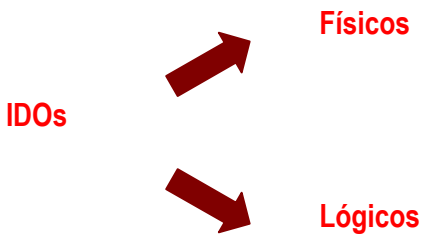


Fig. 2.12 Identificadores

Los IDOs *físicos* contienen la dirección real del objeto, mientras que los *lógicos* son un índice a partir del cual se obtiene la dirección real. En el caso de los identificadores lógicos es necesario mantener una tabla de correspondencia entre los IDOs y su posición física. A continuación, se señalan algunos enfoques para la representación de ambos tipos de IDOs.

2.18.3.1 Dirección Física

El IDO es la dirección física del objeto. Esta representación es muy eficiente pero se utiliza muy poco en los SMBDOO, puesto que, si un objeto se cambia de lugar o se elimina, se tienen que modificar todos los objetos que contienen este IDO.

Este es el tipo de identificador empleado en el sistema O2, dónde cada objeto se almacena en un

registro WiSS (Subsistema Wiscosin de Almacenamiento), y el IDO es el identificador de registro (IDR). El WiSS garantiza que el IDR no cambia, aunque el registro se mueva a otra página. Esto se logra mediante una técnica que va enlazando las posiciones antiguas del registro hasta llegar a la actual. Sin embargo, si un objeto se mueve mucho en el almacenamiento secundario, el acceso al mismo es costoso, puesto que hay que seguir la cadena de enlaces, con un acceso a disco en cada enlace.

La ventaja de este tipo de IDO es que el acceso a los objetos es muy rápido, ya que no se necesita el nivel de indirección introducido por la tabla de correspondencias de IDO lógico a posición física. Una desventaja es que si se crea un objeto en un nodo diferente del nodo donde se va a almacenar de forma definitiva, es necesario asignarle un IDO temporal hasta que se almacene definitivamente, puesto que hasta entonces no tendrá un IDR.

2.18.3.2 Dirección Estructurada

El IDO está formado por un par <segmento+página, desplazamiento>, donde el primer elemento contiene el número de segmento y el número de página, lo que facilita un conocimiento rápido de la dirección del disco, y el segundo elemento contiene un número lógico de desplazamiento que se utiliza para obtener la posición del objeto dentro de la página. Este tipo de representación permite fácilmente la reubicación del objeto dentro de la página, así como una emigración hacia otra página.

Este tipo de IDOs permiten obtener el objeto con un único acceso a la página; en el peor de los casos (el objeto ha sido movido hacia otra página) tiene que hacer dos accesos a disco. Esta es la técnica empleada en ONTOS, Objetivity/DB, etc.

2.18.3.3 Subrogado.

El IDO se genera mediante un algoritmo (por ejemplo, mediante la fecha y la hora) que garantiza su unicidad. Estos IDOs subrogados se transforman en direcciones físicas de objetos usando un índice.

Los IDOs subrogados no son muy eficientes para la obtención de los objetos, ya que a menudo hay que transformarlos en una dirección mediante funciones de dispersión (hash). Sin embargo, en el

mejor caso, una función de dispersión bien balanceada, puede conseguir que la mayoría de los objetos se puedan obtener con un único acceso a disco. Los IDOs subrogados se utilizan en GemStone y POSTGRES.

2.18.3.4 Subrogados con tipo

Es una variante de los anteriores en la que los IDOs incluyen un identificador de tipo y una parte de identificador de objeto. El identificador de tipo en el IDO permite determinar el tipo del objeto sin necesidad de traer el objeto de disco. Para cada tipo se utiliza un contador diferente para generar la porción de identificador del objeto. Los IDOs subrogados con tipo, permiten un nivel de rendimiento similar al proporcionado por los subrogados. Con este tipo de identificadores cuando se invoca una operación sobre un objeto, el sistema extrae el identificador de la clase a partir del IDO y determina el método para ejecutar la operación. Como inconveniente, el identificador de tipo dificulta la migración de objetos entre clases/tipos, ya que hay que modificar los IDOs de todos los objetos que migran, y además todas las referencias a los objetos que han migrado quedan invalidadas.

2.18.3.5 Longitud de los Identificadores

La selección de como representar los IDOs es un factor que afecta al rendimiento del SMBDOO, ya que un objeto se obtiene a través de su IDO. Otro factor de los IDOs que afecta al rendimiento, es la longitud de los mismos. Los IDOs con una longitud en el rango de los 32 a 48 bits pueden tener un efecto significativo sobre el tamaño global de una base de datos, sobre todo cuando la base de datos contiene un gran número de objetos complejos interrelacionados. Pero hay situaciones en las que la longitud exigida al IDO es superior (64 bits):

- Cuando los IDOs deben ser únicos durante la vida del objeto, de modo que se puedan identificar las referencias colgantes.
- Cuando los IDOs se representan por medio de subrogados generados por una función monotónicamente creciente.
- Cuando se está trabajando en un entorno distribuido, dónde los IDOs deben ser únicos. En este caso es necesario añadir un prefijo a los IDOs que permita identificar unívocamente la máquina o la base de datos, además de garantizar la unicidad de éstos.

2.18.3.6 Visibilidad de los identificadores

Algunos sistemas permiten que el usuario acceda directamente a un IDO, y que, por ejemplo, lo imprima, pero la mayoría no lo permiten para asegurarse de que el IDO no se pueda modificar. Ciertos sistemas como GemStone y O2, permiten al usuario asignar nombres de variables a los objetos a los que va a permitir el acceso directo en la base de datos.

2.18.3.7 "Swizzling"

Los IDOs se utilizan para representar referencias entre los objetos. En muchas implementaciones, los IDOs se convierten en direcciones de memoria, cuando los objetos se traen de disco a memoria central. Esta transformación conocida como "swizzling", se lleva a cabo para aumentar la velocidad con la cual se puede "navegar" entre los objetos usando los IDOs.

La ventaja de esta transformación es el aumento de la velocidad con la cual se navega por las referencias entre los objetos, ya que direccionar la memoria central es más rápido que estar recorriendo IDOs. La desventaja que tiene es que es muy costosa, lo que lleva a veces a que la rapidez en la navegación entre objetos no justifique el costo de la transformación. Según estudios realizados se concluye que es mejor utilizar tablas que asocian los IDOs a direcciones de objetos en memoria (como en Objetivity/DB) cuando los objetos tienen una probabilidad alta de sacarse de memoria principal, y cuando las referencias no se usan un número significativo de veces.

2.18.3.8 Enlaces

Representa una conexión física o conceptual entre objetos. Un objeto colabora con otros objetos a través de sus enlaces con estos. Es decir, un enlace denota la asociación especifica por la cual un objeto (el cliente) utiliza los servicios de otro objeto (el servidor), o a través de la cual un objeto puede comunicarse con otro.

2.18.3.9 Agregación

La agregación denota una jerarquía todo/parte, con la capacidad de ir desde el todo (agregado) hasta sus partes (atributos). La agregación puede o no denotar contención física.

2.19 Características derivadas del concepto de clase

Una clase es un conjunto de objetos que comparten una estructura común y un comportamiento común. En una clase tenemos un interfaz y una implementación. La interfaz de una clase proporciona una visión externa y por tanto, enfatiza la abstracción a la vez que oculta su estructura y los secretos de su comportamiento. La implementación, por el contrario, es la visión interna que engloba los secretos de su comportamiento.

Los modelos orientados a objetos usan el concepto de clase como la base para la instanciación (entendiendo por instanciación que la misma definición se puede utilizar para generar objetos con la misma estructura y comportamiento). En este sentido una clase es un objeto que actúa como plantilla, y que especifica:

- una estructura, esto es, el conjunto de atributos de las instancias
- un conjunto de operaciones
- un conjunto de métodos que implementan las operaciones.

La clase incluye dos aspectos: una fábrica y un almacén de objetos. La fábrica se utiliza para crear nuevos objetos. El almacén permite que la clase sea asociada al conjunto de objetos que son instancias de ella. El usuario puede manipular el almacén aplicando operaciones sobre todos los elementos de la clase.

A continuación se van a describir algunas de las relaciones principales entre clases:

2.19.1 Asociación

El concepto de asociación es muy utilizado en algunos modelos semánticos y de diseño conceptual de la base de datos. Una asociación es un enlace entre entidades en las aplicaciones. Se caracterizan por el grado o número de entidades que participan en la asociación, así como por las restricciones de cardinalidad, que indican el número máximo y mínimo de asociaciones en las que

puede participar una entidad.

Sin embargo, en la mayoría de los sistemas de datos orientados a objetos no existe el concepto explícito de asociación. Las asociaciones se representan por medio de referencias entre objetos. Pero, esta forma de representar las asociaciones presenta una serie de desventajas:

- Dificultad para representar las asociaciones de grado 3, y las asociaciones que tienen sus propios atributos.
- Las asociaciones son en los dos sentidos, pero al implementarlas de esta manera el recorrido únicamente es fácil en una dirección.

2.19.2 Herencia Simple y Múltiple

La herencia es el concepto más poderoso de la programación orientada a objetos. Permite la definición de una clase, llamada subclase, sobre la base de la definición de otra clase llamada superclase. La subclase hereda los atributos, métodos y mensajes de su superclase. Adicionalmente la subclase también puede definir sus propios atributos y métodos. Así, la herencia implica una jerarquía de generalización/especialización, en la que una subclase especializa el comportamiento o estructura, más general, de sus superclases.

La herencia tiene dos ventajas: es una potente herramienta de modelado ya que proporciona una descripción precisa del mundo; y ayuda a compartir especificaciones e implementaciones en las aplicaciones. Supongamos que tenemos dos clases: Empleados y Estudiantes. Cada Empleado tiene un nombre, una edad y un número de afiliación a la seguridad social; y cada uno de ellos puede trabajar o divertirse. Cada estudiante tiene una edad, un nombre y un número de matrícula; también puede divertirse o estudiar.

En un sistema relacional, el diseñador de la base de datos definiría una relación para Empleado y otra para Estudiante, y escribiría el código para las operaciones trabajar y divertirse de la relación Empleado, y para las operaciones estudiar y divertirse de la relación Estudiante. De esta forma el programador escribe cuatro programas.

En un sistema orientado a objetos que emplee la propiedad de herencia, se reconocería que Empleados y Estudiantes son Personas (Fig. 2.13) que tienen cosas en común y además poseen algunas características específicas. Se introduce entonces una clase Persona con los atributos de nombre y edad, y se escribe la operación de divertirse para esa clase. A continuación ya se declararían las clases Empleado y Estudiante, que heredarían todos los atributos de Persona e incorporarían los específicos de cada uno. De esta forma se obtiene una descripción del esquema más estructurada y concisa ya que se comparte la especificación y la implementación (sólo se escriben tres programas).

El aspecto esencial de la herencia es la relación que se establece entre las clases, como es el hecho de que las superclases son a su vez subclases de otras clases. Las clases en un esquema de base de datos se pueden agrupar en una *jerarquía de herencia*. El grafo que representa esta jerarquía se convierte en árbol cuando el sistema no presenta herencia múltiple. La jerarquía de herencia permite lograr una descripción más concisa y precisa de la realidad que queremos modelar.

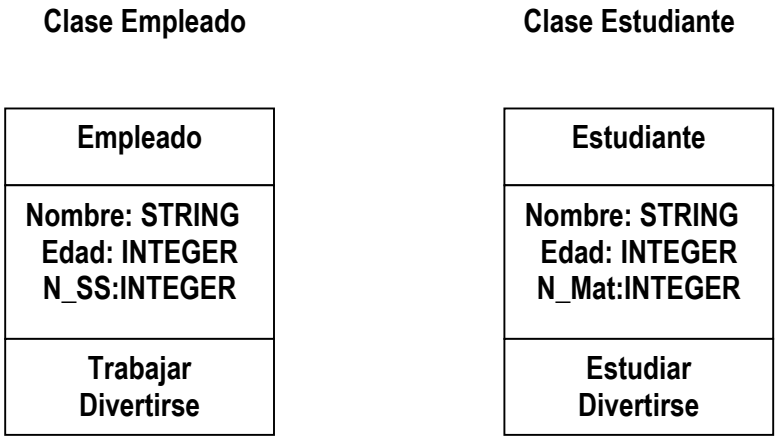


Fig. 2.13. Representación de clases sin herencia

Un problema importante que se plantea es que la estructura de las instancias de una clase también debe estar encapsulada con respecto a las subclases. Los métodos de una clase pueden acceder directamente a todos los atributos de sus instancias.

Sin embargo, cuando se aplica la herencia, el conjunto de los atributos de las instancias de una

clase consiste en la unión de los atributos heredados y de los específicos de la clase. La implementación de un método puede depender de atributos que no están definidos en la clase en la que se define el método, sino en alguna superclase. De esta forma, una modificación de la estructura de las instancias de alguna superclase puede invalidar un método definido en alguna subclase. Esto limita los beneficios del encapsulamiento, en tanto que los efectos de las modificaciones a una clase no se limitan a la clase misma. Se han propuesto algunas soluciones, pero hasta ahora los sistemas actuales no se inclinan por ninguna.

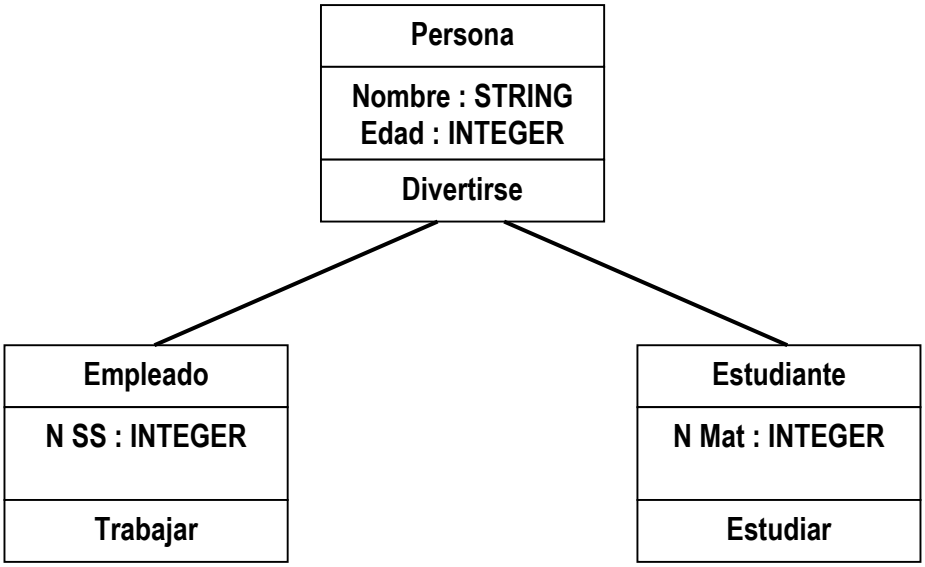


Fig. 2.14. Representación de las clases con herencia

2.19.3 Herencia Múltiple

El hecho de que un sistema presente herencia múltiple se traduce en que una subclase puede tener más de una superclase. Así, si un sistema ofrece herencia múltiple pueden surgir una serie de conflictos, como el hecho de que dos o más superclases tengan un atributo con el mismo nombre, pero con dominios diferentes. En estos casos se deben establecer reglas apropiadas para resolver dichos conflictos: si los dominios están ligados por una relación de inclusión, entonces debe tomarse el dominio más especifico como dominio de la subclase. Si no existiera este tipo de relación, entonces, una posible solución sería escoger el dominio de acuerdo con un orden de precedencia entre las superclases definido a priori.

Otro conflicto que puede surgir con la herencia múltiple, es lo que se conoce como herencia repetida, que ocurre cuando dos o más superclases "hermanas", comparten una superclase común,

ya que entonces hay que decidir si la clase que hereda de ambas debe tener una sola copia o muchas copias de la estructura de la clase compartida.

2.19.4 Polimorfismo

Existen casos en los que se desea tener el mismo nombre para diferentes operaciones. Supongamos la operación display, que toma un objeto como entrada y lo muestra en pantalla. Dependiendo del tipo de objeto (cuadrado, estrella, flecha,…) debemos emplear diferentes mecanismos de visualización. Es decir, necesitamos visualizar un conjunto cuyos miembros no se conocen en tiempo de compilación.

```
case of type(x)
cuadrado: display_cuadrado (x);
estrella: diplay_estrella (x);
flecha: display_flecha (x);
…
end;
```

En una aplicación que emplee el sistema convencional, habrá tantas operaciones como figuras a representar: display cuadrado, display estrella, display flecha, etc. Los programadores tendrán que averiguar el tipo de cada objeto en el conjunto y asociarle la operación de display correspondiente. Esto obliga a los programadores a conocer todos los tipos de objetos en el conjunto, todas las operaciones de display disponibles y a asociar ambos correctamente.

En un sistema orientado a objetos se definirá la operación en una clase más general. Así display tendrá un único nombre y podrá emplearse indiferentemente sobre cualquier figura. Únicamente se redefinirá la implementación de las operaciones para cada una de las subclases; esto es lo que se llama suplantación. El hecho de que el mismo nombre de operación denote varios programas distintos es lo que se conoce como sobrecarga o polimorfismo. De esta manera, para visualizar un conjunto de elementos simplemente aplicaremos la operación display a cada uno de ellos, y el sistema será el que se encargue de seleccionar la implementación adecuada en tiempo de ejecución. Así, los programadores siguen teniendo que escribir el mismo número de programas, pero los diseñadores de las aplicaciones no tienen que preocuparse de los tres programas diferentes. El código es más simple porque no tiene que incluir la instrucción Case, y el mantenimiento es más

sencillo, ya que cuando se añaden nuevos tipos el programa de visualización continúa trabajando sin modificación.

Para proporcionar esta nueva funcionalidad, el sistema no puede asociar los nombres de las operaciones con los métodos correspondientes en tiempo de compilación; se hará en tiempo de ejecución. Esto es lo que se conoce como *ligadura tardía*, y dificulta o imposibilita el chequeo de tipos.

2.19.5 Agregación

Las relaciones de agregación entre clases tienen un paralelismo directo con las relaciones de agregación entre los objetos correspondientes a esas clases. La agregación entre clases relaciona una clase "ensamblada" con una clase "componente". La relación de agregación entre clases, al igual que entre objetos, no precia contención física.

2.19.6 Metaclase

Si en un sistema se desea sustentar el principio mediante el cual cada objeto es instancia de una clase, y las clases son objetos, entonces el sistema debe incluir el concepto de metaclase. Con este concepto se considera cada clase como un objeto en si misma, en el cual los atributos y métodos comunes a las instancias de la clase se agrupan entre sí y se almacenan de forma que no puedan ser considerados como características globales de las instancias. Una metaclase es a su vez un objeto, y debe ser por tanto una instancia de una metaclase de un nivel más alto, y así sucesivamente. En ORION, por ejemplo, hay una clase del sistema, llamada CLASS, que representa a la clase de todas las clases y está en la raíz de la jerarquía; es decir, es la superclase de todas las clases presentes en el sistema.

Las metaclases, generalmente, no pueden ser manipuladas directamente por el usuario. Su propósito es simplificar el manejo de las clases por parte del sistema. Por ejemplo, si la operación new se aplica a una clase, se provoca una búsqueda del método apropiado que debe ejecutar dicha

operación, llamada búsqueda del método, que es la misma que se utiliza cuando se busca un método para una operación invocada sobre una instancia de la clase.

Algunos modelos también permiten la definición de atributos y operaciones que caracterizan a las clases (consideradas como objetos). Por tanto, estas operaciones y atributos no las heredan las instancias de las clases (por ejemplo, un valor de un atributo calculado a partir de otro atributo de todas las instancias de la clase).

La mayoría de los sistemas orientados a objetos no soportan el concepto de metaclase.

2.20 Características Avanzadas de los SMBDOO

Dentro de las características avanzadas del modelo de objetos, se destacan las siguientes:

2.20.1 Versiones

La mayoría de las nuevas aplicaciones, tales como CAD/CAM y CASE, comprenden una actividad de diseño que requiere alguna forma de versionado.

Para manejar las versiones en un SMBD se necesitan una serie de conceptos organizativos y de mecanismos operacionales. Los primeros deben estar incluidos de alguna forma dentro del modelo de datos, mientras que los segundos son el resultado de las operaciones que se pueden aplicar a las entidades del modelo de datos.

En las bases de datos para las aplicaciones de diseño, un objeto de diseño es una agregación de datos de diseño que los diseñadores tratan como una unidad coherente. Un objeto de diseño es generalmente un objeto complejo, que se ha obtenido ensamblando entre sí una serie de objetos que pueden ser a su vez primitivos o complejos. Los objetos de diseño se almacenan en una base de datos de diseño. Un ejemplo de objeto de diseño, puede ser un paquete de software, consistente en los módulos de código fuente, módulos ejecutables y los manuales de documentación.

Una versión es una instantánea semánticamente significativa tomada al objeto de diseño en un momento dado en el tiempo. Una versión de un objeto se crea a partir de las modificaciones hechas en el tiempo a las versiones previas de éste, comenzando con una versión inicial. La traza de estas derivaciones se mantiene a través de una historia de las versiones.

Una versión de un objeto complejo puede constar de las versiones específicas de sus objetos componentes. Una configuración establece los enlaces entre una versión del objeto compuesto y una versión de cada uno de sus objetos componentes. La configuración puede ser estática, cuando el usuario define de forma explícita y permanente el enlace entre la configuración y las versiones específicas de sus objetos componentes; o dinámica, cuando una o más de las versiones específicas que componen la configuración se determinan en tiempo de ejecución.

Una versión específica se debe poder identificar dentro de la historia de una versión. El sistema de manejo de versiones debe proveer al usuario de primitivas que le permitan seleccionar la versión actual (no tiene porque coincidir con la última), y, en el caso de falta de indicación por parte del usuario, el sistema debe mantener una versión actual por omisión.

Una versión de un objeto de diseño es un objeto complejo que contiene referencias a otros objetos (primitivos o complejos) que pueden, a su vez, ser referidos por otros objetos o versiones. Cuando se crea una nueva versión de un objeto, ésta puede invalidar alguno, o incluso todos los objetos que hacen referencia al objeto dado. Por tanto, son necesarios mecanismos para la notificación y propagación de los cambios.

La propagación de los cambios genera dos problemas principales:

- Limitar el alcance de la propagación de los cambios. La sustitución de una versión por otra en el árbol que representa la historia de las versiones de un objeto puede requerir la propagación de dicha sustitución hasta el nodo raíz, con lo cual, puede que otras versiones que fueron generadas a partir del nodo raíz ya no sean válidas.

- Definir el camino a lo largo del cual se deben propagar los cambios. Aquí el problema surge a la hora de seleccionar la configuración por la que se ha de propagar el cambio, cuando dos configuraciones comparten la versión que necesita ser reemplazada. Para solucionar este problema se presentan varias soluciones:

    - Evitar cualquier propagación de cambio cuando hay estructuras de grafos dirigidos acíclicos. Esta opción es muy restrictiva.

    - Propagar el cambio a todas las ramas involucradas y considerar las versiones resultantes como alternativas de la versión raíz. Esto ocasiona una proliferación de versiones, dónde no todas son útiles para el usuario.

    - Propagar el cambio, interrumpiéndolo tan pronto como surja la primera ambigüedad.

    - Definir mecanismos operacionales, por medio de los cuales los usuarios puedan evitar la ambigüedad, mediante la definición de reglas apropiadas y operaciones de verificación de entrada y salida.

2.20.2 Evolución de Esquemas

Las posibilidades de modificación de un esquema en un SMBDOO son mucho mayores que las existentes en un SMBD relacional; esto es debido al incremento en la complejidad del modelo de datos, y al hecho de que las modificaciones realizadas en una clase pueden involucrar varias clases. Por ejemplo, si se elimina un atributo de una clase, se debe eliminar en todas las subclases.

El concepto de jerarquía de herencia requiere que se definan otras operaciones para modificar el esquema, además de aquellas que son semánticamente significativas en el modelo relacional.

Una vez que se han ejecutado las modificaciones a un esquema, éste debe seguir siendo consistente. Con el fin de mantener esa consistencia se definen unas reglas, que deben ser verificadas cada vez que se modifica el esquema. A estas reglas se les denomina invariantes de

esquema. Cada modelo de bases de datos define sus propios invariantes de esquema.

Determinadas modificaciones realizadas en las clases afectan a las instancias de dichas clases, y existen diferentes mecanismos para aplicarlas:

- Estrategia diferida, que consiste en diferir las modificaciones sobre las instancias. Un mecanismo basado en esta estrategia consiste en que las instancias se modifiquen en el momento en el que las aplicaciones acceden a ellas. La modificación propuesta con esta estrategia no es costosa, pero el acceso a las instancias puede ser lento.

- Estrategia inmediata, implica la modificación de todas las instancias tan pronto como la definición de la clase se modifica. Esta estrategia hace la modificación más costosa que la anterior.

Otro problema que puede ocasionar la modificación del esquema, es la inconsistencia de métodos. El hecho de eliminar un atributo de un objeto, hace que un método que emplee dicho atributo no sea consistente en el esquema actual. Una posible solución, puede ser recompilar el esquema completo y sus métodos. Pero esta solución puede ser bastante costosa si los esquemas son modificados frecuentemente, y además, sólo puede emplearse en sistemas que emplean compilación. Para este problema no se ha encontrado todavía una solución que se pueda aplicar con carácter general, ya que los métodos a menudo se implementan en lenguajes de programación imperativos, y no es fácil determinar los efectos de una modificación de esquema en el conjunto de los métodos de las clases involucradas en la modificación.

2.20.3 Migración de instancias entre clases

Representa el hecho de que un objeto pueda convertirse en instancia de una clase diferente de la clase a partir de la cual se generó. Más concretamente, esto significa que un objeto puede modificar sus propias características (atributos y operaciones), manteniendo la misma identidad. Ciertos sistemas, como Iris y Encore, son capaces de hacer esto, mientras que la mayoría de los otros no. Si los objetos pueden migrar entre clases, entonces surgen problemas concernientes a la integridad

semántica. Así, supongamos que tenemos un objeto M con un atributo T, el cual tiene como valor otro objeto M'. Si M' cambia de clase y su nueva clase no es compatible con la clase dominio de T, entonces el objeto M contendrá un valor incorrecto en T.

2.21 Modelo Propuesto por ODMG-93[29]

El éxito de las bases de datos relacionales, ha venido en gran medida ocasionado por la estandarización que ofrecían. La aceptación de un SQL estándar ha permitido un alto grado de portabilidad e interoperabilidad entre sistemas, simplificando a su vez el aprendizaje de nuevos sistemas de gestión relacionales.

En el caso de las bases de datos orientadas a objetos la carencia de un estándar es la mayor limitación para su uso generalizado. ODMG-93 (Object-Oriented Database Management Group) es un punto de partida muy importante para conseguir un lenguaje estándar de bases de datos orientado a objetos. Adopta una arquitectura que consta de un sistema de gestión que soporta un lenguaje de bases de datos orientado a objetos, con una sintaxis similar a un lenguaje de programación también orientado a objetos, como puede ser C++ o Smalltalk. El lenguaje de bases de datos, es especificado mediante un lenguaje de definición de datos (ODL), un lenguaje de manipulación de datos (OML), y un lenguaje de consulta (OQL).

El objetivo principal que persigue ODMG-93 es la portabilidad de la aplicación completa, es decir, pretende conseguir el desarrollo de aplicaciones que puedan ejecutarse sobre más de un SMBDOO, siendo portable tanto el esquema de la base de datos, como el lenguaje de programación asociado, como los lenguajes de manipulación y consulta.

La arquitectura de los SMBDOO es sustancialmente diferente de la de los otros SMBD, ya que más que proporcionar un lenguaje de alto nivel como SQL para la manipulación de datos, un SMBDOO integra transparentemente las capacidades de la base de datos con las del lenguaje de programación de la aplicación. Esta transparencia hace innecesario aprender un DML separado, evitando tener que traducir explícitamente los datos entre las representaciones de la bases de datos y la de los lenguajes de programación.

En definitiva, ODMG-93 intenta definir un SMBDOO que integre las capacidades de las bases de datos con las capacidades de los lenguajes de programación, de forma que los objetos de la base de datos aparezcan como objetos del lenguaje de programación. El SMBDOO extiende el lenguaje con persistencia, concurrencia, recuperación de datos, consultas asociativas, etc.

La figura (Fig. 2.15) ilustra los SMBDOO que ODMG-93 está intentando estandarizar. En esta arquitectura el programador escribe declaraciones para el esquema de la aplicación, más un programa fuente para la implementación de la aplicación. El programa fuente se escribe en un lenguaje de programación (PL) como C++, que ha sido ampliado para proporcionar un lenguaje de manipulación de datos completo, incluyendo transacciones y consulta de objetos. Las declaraciones del esquema pueden escribirse mediante una extensión del lenguaje de programación, PL ODL en la figura, o en un lenguaje de programación independiente ODL.

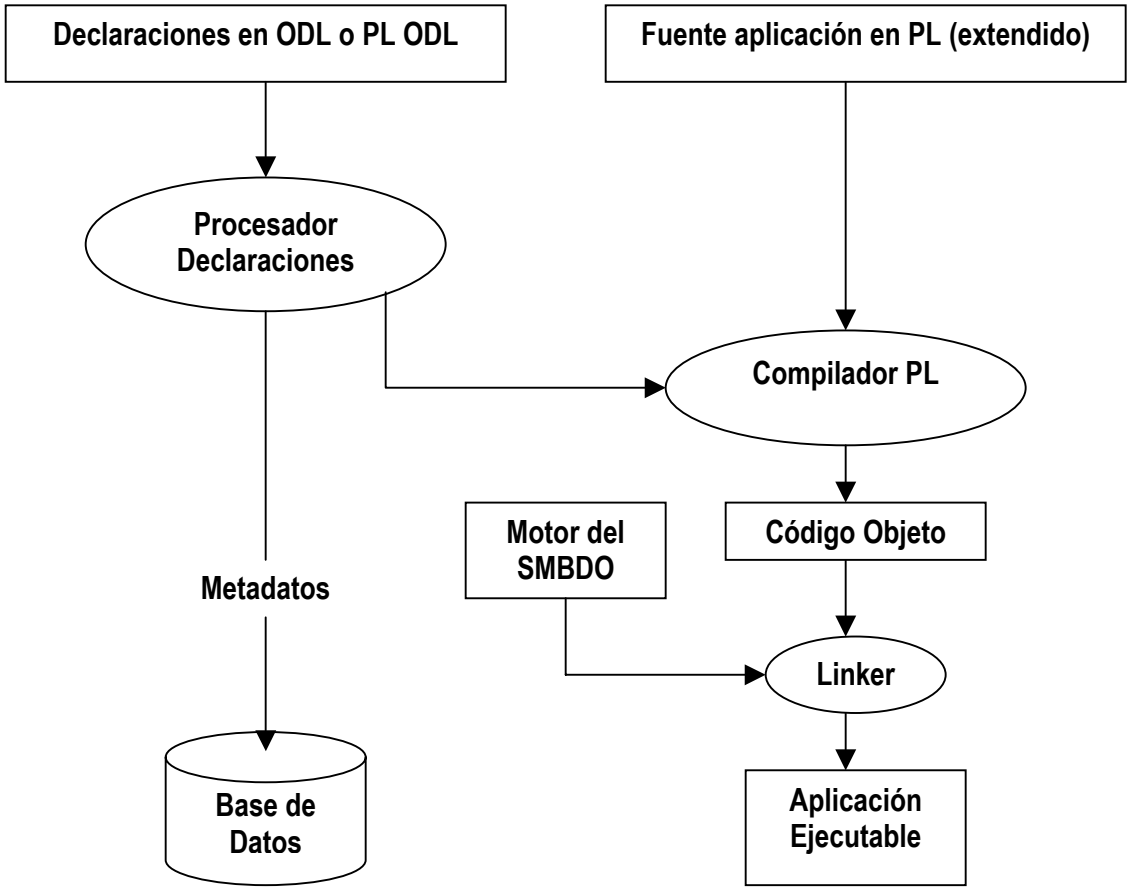


Fig. 2.15 Arquitectura propuesta por ODMG - 93

Las declaraciones y programa fuente son compilados con el SMBDOO para producir la aplicación ejecutable. La aplicación accede a bases de datos nuevas o ya existentes, cuyos tipos deben estar de acuerdo con las declaraciones.

2.22 Modelo de Objetos de ODMG - 93

El modelo de objetos propuesto por ODMG-93[30] se puede resumir en los siguientes puntos:

- La primitiva básica del modelo es el objeto.
- Los objetos se categorizan en tipos. Todos los objetos de un tipo dado muestran un comportamiento y un rango de estados común.
- El comportamiento de un objeto es definido por un conjunto de operaciones que pueden ser ejecutadas sobre un objeto del tipo.
- El estado de los objetos es definido por los valores asignados a un conjunto de propiedades. Estas propiedades pueden ser atributos del propio objeto, o interrelaciones con otro u otros objetos.
- Un tipo tiene una interfaz y una o más implementaciones. El interfaz define las propiedades y operaciones que pueden invocarse sobre las instancias del tipo. Una implementación define las estructuras de datos en función de las que se representan físicamente las instancias del tipo, y los métodos que operan sobre esas estructuras de datos para soportar el estado y comportamiento definidos en el interfaz.

La definición del interfaz de un tipo está constituida por propiedades del tipo, propiedades de las instancias y operaciones de las instancias.

Los tipos son también objetos, y pueden tener por lo tanto propiedades específicas. La definición del interfaz de un tipo puede especificar valores para estas propiedades. Hay tres propiedades de tipo:

- **Supertipos**. Los tipos se pueden representar en grafos de subtipo/supertipo. Todos los atributos, relaciones y operaciones definidas sobre un supertipo son heredadas por los subtipos. Los subtipos pueden añadir propiedades y operaciones adicionales para proporcionar un

comportamiento especializado a sus instancias.

- **Extensión.** Entendiendo por extensión el conjunto de todas las instancias de un tipo dado. El sistema puede mantener automáticamente un índice a los miembros de este conjunto incluyendo una declaración de extensión en la definición de tipos. El mantenimiento de la extensión es opcional y no necesita ser realizada para todos los tipos.
- **Llaves**. Se entiende por llave, la propiedad o conjunto de propiedades que identifican de forma única las instancias de un tipo. Las llaves simples están constituidas por una única propiedad. Las llaves compuestas están constituidas por un conjunto de propiedades. Las llaves pueden estar constituidas no sólo por atributos, sino también por interrelaciones.

Las propiedades de las instancias son las propiedades para las que los objetos del tipo admiten valores. Las operaciones de las instancias son las operaciones que los objetos del tipo soportan.

Tanto las operaciones, como los atributos, como las relaciones se expresan en el interfaz mediante una serie de prototipos:

- En el prototipo de las operaciones se indica el nombre de la operación, el nombre y tipo de los argumentos, el nombre y tipo de los valores de retorno (si los hay), y el nombre de cualquier excepción que la operación pueda elevar.

- En el prototipo de los atributos se indica únicamente el nombre del atributo y el tipo de los valores permitidos.

- En el prototipo de las relaciones, se define para cada objeto el tipo del otro objeto o conjunto de objetos involucrados en la relación, y el nombre de la función empleada para referirse al objeto o conjunto de objetos relacionados.

- El modelo incluye un conjunto de tipos colección predefinidos, como son: sets, bags, lists y arrays.

2.22.1 Herencia

Los tipos en ODMG-93 se pueden organizar en grafos de subtipos y supertipos. Es decir, un subtipo hereda todas las características de su supertipo, y además puede definir características adicionales que se apliquen sólo a sus instancias. Además de la herencia simple, permite la *herencia múltiple*, y en el caso de que dos de las características heredadas coincidan en el subtipo, se redefinirá el nombre de una de ellas.

Este modelo soporta la existencia de *tipos abstractos*, es decir, aquellos que únicamente contienen características para que sean heredadas por sus subtipos, y que no pueden ser instanciados directamente. Estos tipos no definen implementación, por lo que cualquier subtipo instanciable de un tipo abstracto deberá definir una implementación que soporte cada una de las características heredadas de ellos.

2.22.2 Clases

Un tipo puede tener una o más implementaciones. A cada una de estas implementaciones se le da un nombre, que además debe ser único dentro del ámbito definido por un tipo. Las implementaciones asociadas a un tipo son separadas léxicamente en el lenguaje de definición de datos (ODL). Una *clase,* en este modelo, es la combinación del interfaz del tipo y una de las implementaciones definidas.

El hecho de permitir varias implementaciones, presenta varias ventajas; la primera de ellas, es que soporta fácilmente bases de datos que están sobre redes, donde puede haber máquinas con arquitecturas diferentes (soportando mezcla de lenguajes y compiladores); la segunda, es que facilita al programador su tarea al poder comparar fácilmente cual de las implementaciones se comporta mejor. La implementación que emplee un objeto se especifica en tiempo de creación.

En caso de que el lenguaje asociado no permita la existencia de diferentes implementaciones (caso de C++) para un mismo interfaz, es necesario acudir al mecanismo de subtipos, definiendo un

subtipo diferente para cada una de las implementaciones.

2.22.3 Jerarquía de Tipos

El conjunto de tipos predefinidos en la base de la jerarquía de tipos del modelo propuesto por ODMG-93 es el que se representa a continuación. Con el fin de simplificar, en esta representación se han incluido los tipos y los generadores de tipos, pero hay que tener en cuenta que un generador de tipos no puede ser un subtipo de un tipo, y que tampoco un tipo puede ser un subtipo de un generador de tipos.

- Objecto_Principal
  - Objeto
    - Objeto_Atómico
      - Type
      - Excepción
      - Iterador
    - Objeto_Estructurado
      - Colección<T>
        - Set <T>
        - Bag <T>
        - List <T>
          - String
          - Bit_String
        - Array <T>
      - Estructura <e1:T1…en:Tn>
  - Literal
    - Literal_Atómico
      - Integer
      - Float
      - Character
      - Boolean
    - Literal_Estructurado
      - Colección_Inmutable <T>
        - Set_Inmutable <T>
        - Bag_Inmutable <T>
        - List _Inmutable<T>
        - String_Inmutable
          - Bit_String_Inmutable
          - Array_Inmutable <T>

        - Enumeración
    - Estructura_Inmutable<e1:T1…en:Tn>
        - Date
        - Time
        - Timestamp
        - Interval
- Característica
    - Propiedad
        - Atributo
        - Interrelación
    - Operación

Todos los objetos que derivan de Objeto_Principal tienen identidad, así en el caso de Literal la identidad es su propio valor, mientras que la identidad de Objeto viene expresada por medio de un identificador de objeto (IDO). En ambos casos heredan la operación igual? de Objeto_Principal.

2.22.4 Tipo Objeto

Los valores de los atributos de las instancias de Tipo Objeto, así como las relaciones en que participan pueden cambiar, pero la identidad de los objetos permanece invariante a estos cambios.

El dominio de un identificador de objeto es la base de datos en la que existe el objeto. La estructura del patrón de bits que representa el IDO no se define en el modelo de objetos.

ODMG-93 permite la asignación de nombres a los objetos de forma que un objeto tendrá un único IDO, pero puede tener más de un nombre. Un nombre sólo podrá referir a un objeto dentro del ámbito de su definición.

ODMG-93 permite también que los objetos sean identificados por predicados definidos sobre sus características. El predicado es una conjunción o disyunción booleana de operaciones soportadas por los tipos que aparecen dentro del predicado, y que será aplicado a una colección para seleccionar los miembros que lo satisfacen. Para facilitar o acelerar este proceso, el encargado de definir el tipo puede indicar los valores que definen de forma única los objetos dentro de la extensión

del tipo, lo que obliga al SMBDOO a crear y mantener índices sobre esas propiedades que lo identifican unívocamente. Esto es la base para las consultas en el SMBDOO.

Las propiedades definidas sobre el tipo Objeto son:

- tiene_nombre?
- nombres
- tipo:Type

Las operaciones principales definidas sobre este tipo son:

- crear
- borrar
- mismo?

La operación crear, asigna espacio para la representación del objeto, le asocia un IDO y devuelve ese IDO como valor de la operación. También se le pueden indicar a la operación una lista de pares de valores de la forma (nombre_de_la_propiedad, valor_de_la_propiedad) como argumentos.

La operación borrar, borra el objeto de la base de datos, y libera el almacenamiento empleado en su representación. Este borrado implica la eliminación del objeto de cualquier relación en la que haya participado. Si el objeto es una instancia de un tipo para el que el SMBDOO está manteniendo su extensión automáticamente, al borrar el objeto se elimina de la extensión del tipo del que era una instancia. Los IDOs de los objetos borrados no pueden emplearse de nuevo.

La operación mismo?, devuelve el valor true si los objetos comparados son el mismo objeto, y false en caso contrario.

El tiempo de vida de un objeto puede especificarse cuando se crea el objeto, y no puede cambiarse posteriormente. Además, la duración de un objeto es ortogonal a su tipo. El modelo especifica tres duraciones diferentes para los objetos: de procedimiento, de proceso y de base de datos. Los objetos especificados con duración de base de datos son gestionados por el motor de base de

datos.

2.22.5 Tipo Type

El tipo Type es a la vez una instancia y un subtipo del tipo Objeto_Atómico. Entre las propiedades de instancia que tiene definidas, están:

- tiene_operaciones?
- tiene_propiedades?
- tiene_subtipos?
- tiene_supertipos?
- nombre
- extensión
- nombre_de_extensión

Entre las operaciones que tiene definidas están las siguientes:

- crear_instancia
- crear_extensión

2.22.6 Tipo Excepción

Las excepciones son objetos, y pueden ser organizadas por tanto en una jerarquía subtipo/supertipo. El SMBDOO proporciona un tipo básico *Excepción.* Este tipo incluye una operación que imprime un mensaje para notificar que se ha producido una excepción y terminar el programa. La información sobre las causas de una excepción o el contexto en que haya ocurrido se le pasa al manejador de excepciones como propiedades del objeto excepción.

2.22.7 Tipo Iterador

Es un tipo que permite iterar sobre los elementos de una colección. Si la colección está ordenada, el orden puede ser del primero al último o viceversa. Si por el contrario, la colección no está ordenada el orden de iteración es arbitrario. Los iteradores se crean sobre las colecciones, y puede haber más de un iterador definido sobre una colección.

2.22.8 Tipo Colección

Una colección es un objeto que agrupa a otros objetos. Las colecciones contienen un número arbitrario de elementos que son instancias del mismo tipo. La inserción y el borrado de un elemento se hace teniendo en cuenta la posición dentro de la colección. Las colecciones pueden definirse sobre cualquier subtipo instanciable del tipo Objeto_Principal. Se permiten así colecciones de objetos atómicos, colecciones de literales atómicos, colecciones de colecciones, colecciones de estructuras, etc. Además, las colecciones puedan estar ordenadas o no, y permiten la repetición de elementos.

Las colecciones individuales son instancias de los tipos colección. Los tipos colección son instanciaciones de los tipos generadores de colección o tipos parametrizados. El modelo de objetos de ODMG-93 define un conjunto estándar de tipos generadores de colecciones, que derivan del tipo Colección, que es un tipo abstracto, y que por tanto no puede ser directamente instanciado:

Set <T>, colecciones no ordenadas que no permiten duplicados.
Bag <T>,colecciones no ordenadas que permiten duplicados.
List <T>, colecciones ordenadas que permiten duplicados. El orden es el de inserción.
Array <T>, son arrays unidimensionales de longitud variable. Durante su creación se les asigna un tamaño que podrá cambiar posteriormente.

Este tipo Colección permite definir opcionalmente una serie de propiedades y operaciones relativas a los índices, que posibilita entre otras cosas la creación de un índice sobre los elementos de una colección (tanto si son objetos como literales).

2.22.9 Tipo Estructura

Una estructura se compone de un número fijo de slots con un nombre, donde cada uno de estos slots contiene un objeto o un literal. Los tipos de los elementos que llenan los slots son generalmente diferentes, y tanto la inserción como el borrado de objetos de los slots se hace referenciando el nombre del slot.

2.22.10 Tipo Literal

Los literales son objetos cuyas instancias no pueden cambiar. En este modelo se definen dos subtipos de este tipo: Literales_ Atómicos y Literales_Estructurados. Los Literales_ Atómicos no tienen la operación de crear asociada, y tienen una identidad única pero no un IDO asociado. Los subtipos del tipo Literales_Atómicos que soporta el sistema son : Integer, Float, Boolean y Character.

Los Literales_ Estructurados una vez que han sido creados, no pueden ser modificados, ya que no tienen IDOs asociados. Los Literales_Estructurados tienen dos subtipos: Colección_Inmutable y Estructura_Inmutable.

El programador de tipos puede definir nuevos subtipos del tipo Literal, sin embargo no puede redefinir operaciones sobre ninguno de los tipos literales predefinidos. Esta restricción viene impuesta por el hecho de que muchas de las operaciones definidas sobre literales son implementadas directamente en el hardware de la máquina sobre la que se ejecuta el SMBDOO.

2.22.11 Tipo Propiedad

Un tipo objeto define un conjunto de propiedades mediante las que los usuarios de las instancias del tipo pueden interrogar y en algunos casos manipular directamente el estado de esas instancias. En este modelo se definen dos clases de propiedades: atributos e interrelaciones.

2.22.12 Tipo Atributo

Los atributos se definen sobre un tipo objeto y sus valores son literales. No tienen IDOs asociados, y su individualidad se determina por el objeto individual al que se aplican. Se definen dos operaciones sobre los atributos:

- almacenar_valor -> da al atributo un nuevo valor
- recuperar_valor -> devuelve el literal aplicado por última vez con la función almacenar_valor.

2.22.13 Tipo Relación

Las relaciones se definen entre tipos objeto. Este modelo sólo soporta relaciones binarias uno-a-uno, uno-a-muchos y muchos-a-muchos. Las relaciones no tienen nombres, en su lugar se nombran los caminos o direcciones en las que se expresa la relación. Estos nombres son declarados dentro de la definición del interfaz de los tipos objetos que participan en la relación. No tienen IDOs asociados, y se identifican por las instancias de los objetos que participan en ellas.

2.22.14 Tipo Operación

El comportamiento de las instancias de un tipo objeto se especifica como un conjunto de operaciones. Las operaciones se definen siempre sobre un tipo objeto simple. No existe la noción de operación independiente de un tipo objeto o definida sobre dos o más tipos objeto. Este modelo permite la sobrecarga de operaciones. Las operaciones definidas sobre el tipo Operación son: invocar, devolver y devolver_anormalmente

El modelo asume una ejecución secuencial de las operaciones, es decir, no requiere soporte para operaciones concurrentes, aunque no excluye sistemas que soporten esta posibilidad. El modelo no requiere tampoco soporte para operaciones remotas, aunque no excluye que un SMBDOO pueda proporcionar tal soporte. El lugar por defecto para la ejecución de una operación es el lugar dónde se ha invocado.

2.22.15 Transacciones

Los programas que emplean datos persistentes se organizan en transacciones. Así, cualquier acceso, creación, modificación y borrado de objetos persistentes debe ser realizado dentro de las transacciones.

El modelo de objetos soporta transacciones anidadas del tipo:

```
Transaccion::begin()->t:Transaccion
```

```
Transaccion::begin()->x:Transaccion
Transaccion::begin()->y:Transaccion
y.commit ()
x.commit ()
t.commit ()
```

De esta forma los cambios hechos por *t* antes de que comience *x*, son visibles a *x* y a *y*. Los cambios hechos por *x* son visibles para *t* una vez que *x* finaliza. Sin embargo, ninguno de los cambios hechos por *x* o *y* son visibles fuera de *t* hasta que *t* finaliza. De igual forma, si *t* aborta, los cambios hechos por *x* e *y* serán anulados.

2.23 Base de Datos

Una base de datos proporciona almacenamiento para objetos persistentes de un conjunto dado de tipos. Cada base de datos tiene un *esquema*, es decir, un conjunto de definiciones de tipo, y podrá almacenar por tanto, instancias de los tipos definidos en él. Una base de datos lógica puede estar almacenada en una o más bases de datos físicas, pero esto es un detalle de implementación que el modelo deja abierto a los programadores. Cada base de datos es una instancia del tipo *Database*. Este tipo soporta las siguientes operaciones:

- Abrir
- Cerrar
- Contiene_objeto?
- Buscar_objeto?

Además puede soportar operaciones diseñadas para la administración de la base de datos: mover, copiar, reorganizar, verificar, backup, restore, etc. que están fuera del modelo de objetos. Los nombres de los tipos en el esquema de la base de datos, así como sus extensiones asociadas, son globales a la base de datos, y son accesibles a un programa una vez que se abre la misma. Las tres clases de nombres globales: nombre de tipo, nombre de extensión y nombre de objeto, proporcionan puntos de entrada en la base de datos, permitiendo al programador seleccionar un grupo de objetos a partir del cual él puede navegar siguiendo las relaciones, o por medio de una recuperación asociativa.

2.24 Compatibilidad de Tipos

El modelo de objetos propone un fuerte control de tipos. Cada objeto tiene un tipo, y cada operación requiere que los operandos tengan un tipo.

2.24.1 Compatibilidad de tipos entre objetos

Dos objetos tienen el mismo tipo sí y solo sí han sido declarados como instancias del mismo tipo. Los objetos que hayan sido declarados como instancias de dos tipos diferentes no son el mismo tipo, aunque ambos tipos definan el mismo conjunto de propiedades y operaciones.

2.24.2 Compatibilidad de tipos entre literales

Dos literales atómicos tienen el mismo tipo si pertenecen al mismo conjunto de literales (por ejemplo, Integer, Float, etc.). Los literales estructurados tienen el mismo tipo si en cada nivel tienen la misma estructura, y en cada parte atómica tienen el mismo tipo.

2.25 Lenguajes de Definición (ODL), Manipulación(OML) y Consulta(OQL)

2.25.1 Lenguaje de Definición

El ODL o lenguaje de definición de datos, se utiliza para expresar la estructura y condiciones de integridad sobre el esquema de la base de datos. En una base de datos relacional define las tablas, los atributos en la tabla, el dominio de los atributos y las restricciones sobre un atributo o una tabla. En un SMBDOO el ODL debe ser empleado para definir no sólo lo anteriormente mencionado, si no también para definir métodos, datos compuestos, relaciones IS-A, herencia, etc.

El DDL propuesto por ODMG-93 (ODL) pretende principalmente facilitar la portabilidad de los esquemas de las bases de datos. Este ODL no es un lenguaje de programación completo, define las propiedades y los prototipos de las operaciones de los tipos, pero no los métodos que implementan

esas operaciones.

El ODL intenta definir tipos que puedan implementarse en diversos lenguajes de programación; no está por tanto ligado a la sintáxis concreta de un lenguaje de programación particular. De esta forma un esquema especificado en ODL puede ser soportado por cualquier SMBDOO que sea compatible con ODMG-93.

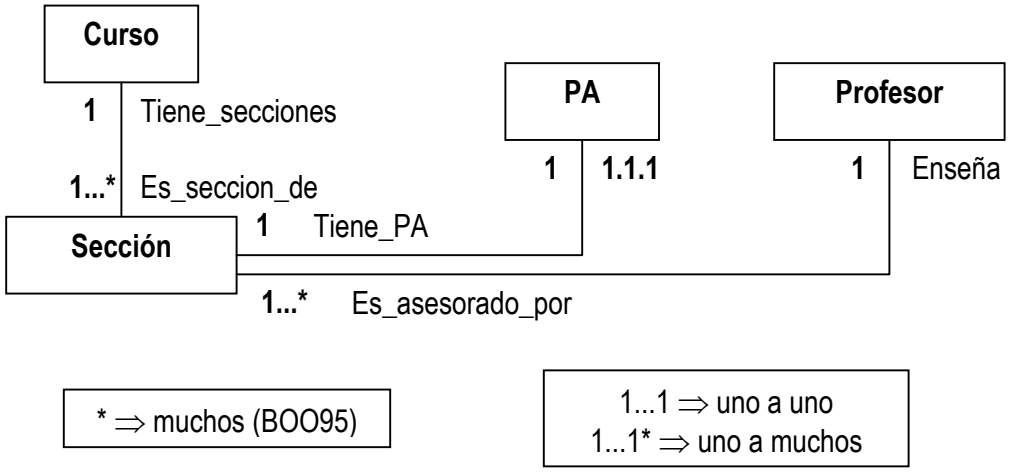


Fig. 2.16. Representación gráfica de un esquema de una base de datos

La sintáxis de ODL es una extensión de la del IDL (Interface Definition Language) desarrollado por OMG como parte de CORBA (Common Object Request Broker Architecture).

A continuación expone un ejemplo de definición de una interfaz mediante el lenguaje de definición (ODL). El interfaz representado corresponde al tipo Sección de la Fig. 2.16.

```
Interface Seccion
( extent secciones
Key (es_seccion_de, numero))
{
attribute String numero;
relationship Profesor Es_asesorada_por inverse Profesor::Enseña;
relationship PA Tiene_PA inverse PA::Asiste;
relationship Curso Es_seccion_de inverse Curso::Tiene_secciones;
};
```

La traducción ODL-C++, por ejemplo, se expresará como una librería de clases y una extensión a la gramática estándar de definición de clases de C++. La librería de clases proporcionará clases y funciones para implementar los conceptos definidos en el modelo de objetos de ODMG, y la extensión consistirá en un conjunto de palabras reservadas y su sintaxis asociada, que proporcionarán soporte declarativo para las relaciones en la declaración de clases de C++.

2.25.2 Lenguaje de Manipulación

El lenguaje de manipulación es empleado para la elaboración de programas que permitan crear, modificar y borrar datos que constituyen la base de datos.

ODMG-93 no propone un OML estándar, simplemente sugiere que este lenguaje sea la extensión de un lenguaje de programación, de forma que se puedan realizar entre otras las siguientes operaciones sobre la base de datos:

- Creación de un objeto
- Borrado de un objeto
- Modificación de un objeto
- Identificación de un objeto

2.25.3 Lenguaje de Consulta

El lenguaje de consulta propuesto por ODMG-93, presenta las siguientes características:

- No es computacionalmente completo. Sin embargo, las consultas pueden invocar métodos, e inversamente los métodos escritos en cualquier lenguaje de programación soportado pueden incluir consultas.
- Tiene una sintaxis abstracta.
- Su semántica formal puede definirse fácilmente.
- Proporciona un acceso declarativo a los objetos.
- Se basa en el modelo de objetos de ODMG.
- Tiene una sintáxis concreta al estilo SQL, pero puede cambiarse con facilidad.

- Puede optimizarse fácilmente.
- No proporciona operadores explícitos para la modificación, se basa en las operaciones definidas sobre los objetos para ese fin.
- Proporciona primitivas de alto nivel para tratar con conjuntos de objetos, pero no se restringe sólo a ellas.

2.26 Aspectos de la Tecnología

Los objetos pueden estar compuestos o consistir de cualquier tipo de información que, eventualmente, puede almacenarse en forma digital; por ejemplo, imágenes barridas ("scanned"), voz y sonido, dibujos y planos arquitectónicos complejos, esquemas electrónicos y diagramas desarrollados por ingenieros, así como los tradicionales tipos de datos alfanuméricos. Comúnmente, las aplicaciones que producen este tipo de objetos complejos, al terminar, guardan los objetos en archivos de datos en distintos formatos. Cuando el programa es reactivado, los objetos son cargados nuevamente. En estos ambientes, los objetos son accesibles sólo a un usuario en cada momento, no existen mecanismos de seguridad y no hay manera de protegerse ante la eliminación accidental de un objeto complejo. Las BDOO superan todas estas dificultades porque permiten que múltiples usuarios compartan objetos complejos y los manipulen en un ambiente seguro y estructurado.

Las bases de datos convencionales fueron diseñadas para manejar tipos de datos alfanuméricos y por esto, difícilmente, pueden manipular objetos y "métodos" (los métodos son los comportamientos definidos de los objetos). Algunos proveedores de bases de datos relacionales han respondido a las tendencias de la tecnología proporcionando "front - ends" orientados a objetos, una capa filtrante que traduce entre objetos y la base de datos interna. Sin embargo, este enfoque es limitado porque los objetos deben ser interceptados y desmenuzados en una forma que se almacene en la base de datos relacional, lo que resulta en un proceso difícil. Los objetos deben ser repetidamente ensamblados (para trabajar con ellos) y desarticulados (para guardarlos).

Una base de datos en red o jerárquica puede almacenar objetos complejos, pero esta arquitectura no es flexible, lo cual motiva, por ejemplo, el uso del modelo relacional. El problema principal con los

modelos en red o jerárquicos es que la estructura es definida, rígidamente, cuando la base de datos es creada. Estos sistemas casi no permiten flexibilidad para modificaciones, y el sistema debe desactivarse cuando se requiere modificar estructuras de objetos o métodos.

Una base de datos relacional tiene una estructura más flexible, pero no puede manejar tipos de datos complejos. Para sobreponerse a estas limitaciones, algunos proveedores han desarrollado las bases de datos orientadas a objetos, las cuales son diseñadas para manipular objetos con los conceptos de la programación orientada a objetos y proporcionando un depósito persistente en un ambiente multiusuario seguro.

Existen niveles en los cuales las bases de datos incorporan los conceptos alrededor de la metodología de objetos. La primera clase, puede denominarse BDOO pasivas o "estructuralmente orientadas a objetos", que permiten manejar objetos compuestos. Una base de datos pasiva puede almacenar objetos complejos pero no puede definir comportamientos. Este tipo de base de datos se utiliza para almacenar objetos de otras aplicaciones. Una BDOO pasiva incorpora conceptos como "jerarquía parte de", pero no incorpora mecanismos para tipos definidos por el usuario o aspectos que definen comportamientos. Una BDOO es activa u "orientada a objetos por comportamiento" si permite definir y ejecutar comportamientos de los objetos dentro de la base de datos, incorpora conceptos como "herencia" y permite el manejo de tipos definidos por el usuario. Si se incorporan todos los aspectos se denomina "plenamente orientada a objetos". En bases de datos activas, es sencillo programar una señal de alerta en un objeto inventario cuando se llega a un nivel mínimo.

2.27 Ejemplos de Ventajas en BDOOs

Entre las ventajas más ilustrativas de las BDOOs está su flexibilidad y soporte para el manejo de tipos de datos complejos. Por ejemplo, en una base de datos convencional, si una empresa adquiere varios clientes por referencia de clientes previos, pero la base de datos existente, que mantiene la información de clientes y sus compras, no tiene un campo para registrar quién proporcionó la referencia, y de qué manera fue dicho contacto, o si debe compensarse con una comisión, sería necesario reestructurar la base de datos para añadir este tipo de modificaciones. Por el contrario, en una BDOO, el usuario puede añadir una "subclase" de la clase de clientes para manejar la

modificación que representan los clientes por referencia. La subclase heredará todos los atributos y características de la definición original y se especializará en especificar los nuevos campos que se requieren y los métodos para manipular solamente estos nuevos campos. Naturalmente se generan los espacios para almacenar la información adicional de los nuevos campos. Esto presenta la ventaja adicional que una BDOO puede ajustarse a usar siempre el espacio de sólo los campos que son necesarios, eliminando espacio desperdiciado en registros con campos que nunca se usan.

Una segunda ventaja es que en una BDOO manipula datos complejos rápida y ágilmente. La estructura de la base de datos está dada por referencias (o apuntadores lógicos) entre objetos. No se requieren búsquedas en tablas o uniones para crear relaciones. Esta capacidad resulta atractiva en aplicaciones de la ingeniería, donde las relaciones entre componentes dependen de factores diversos. Por ejemplo, considérese una aplicación en el diseño de vehículos automotores. El fabricante que quiere determinar una lista de partes necesarias para un auto para un modelo particular requiere de diferentes decisiones subsecuentes para elaborar la lista de partes. Si el modelo es automático o estándar, se requiere de un chasis particular y de la caja de velocidades correspondiente. Escoger un tipo de motor demanda decidir sobre otras partes requeridas, todo esto hasta el nivel de componentes y piezas individuales. Armar esta lista de componentes resulta más ágil en una BDOO que en una base de datos relacional. En un modelo relacional, las tablas deben ser barridas y buscadas cada vez que se indica una condición, resultando, posiblemente, en miles de redireccionamientos.

2.28 Posibles Problemas

Al considerar la adopción de la tecnología orientada a objetos, la inmadurez del mercado de BDOOs constituye una posible fuente de problemas, y debe analizarse con detalle la presencia en el mercado del proveedor para adoptar su producto en una línea de producción sustantiva.

El segundo problema es la falta de estándares en la industria orientada a objetos. Sin embargo, el Object Management Group (OMG), es una organización internacional de proveedores de sistemas de información, usuarios e investigadores dedicada a promover estándares para el desarrollo de aplicaciones y sistemas orientados a objetos en ambientes de cómputo en red. El OMG fue fundado

de 1989, y mucho de su esfuerzo se concentró inicialmente en el diseño de un estándar para una capa de manejo de objetos sobre una red, denominado el "Object Request Broker". La mayoría de los grupos que buscan estándares en la industria informática se crean para formalizar un estándar que está activo "de facto" por su aceptación extendida y ampliamente mayoritaria en el mercado o para codificar una tecnología en maduración que se ha desarrollado de los esfuerzos de múltiples proveedores. Naturalmente, en cuanto a BDOOs, debemos decir que se trata del segundo caso, y en cierto sentido, este grupo va a la vanguardia, ya que los estándares de OMG son definiciones de productos ideales a los que los proveedores esperan llegar.

La implantación de una nueva tecnología requiere que los usuarios iniciales acepten cierto riesgo. Aquellos que esperan resultados al corto plazo y con un costo reducido quedarán desilusionados. Sin embargo, para aquellos usuarios que planean a un futuro intermedio con una visión tecnológica de avanzada, el uso de tecnología orientada a objetos paulatinamente compensará todos los riesgos.

2.29 Proveedores y Productos

Objectivity/DB es el paquete de BDOOs principal de Objectivity, Inc. Este paquete está enfocado a las necesidades de los ingenieros y usuarios científicos y técnicos. Objectivity/DB puede instalarse en una amplia variedad de plataformas y dominios de ingeniería incluyendo redes heterogéneas que combinan estaciones de trabajo de Sun y Digital.

Además de su soporte en sistemas distribuidos, Objectivity/DB proporciona control distribuido de los datos. El manejo de memoria y el control de recursos en la red han sido diseñados pensando en aplicaciones con grandes volúmenes de datos y programas complejos y largos, con requerimientos de desempeño acordes con las necesidades de una gran compañía de ingeniería.

Para satisfacer de manera óptima las necesidades de los ingenieros usuarios, Objectivity/DB maneja los datos al nivel de objetos (no al nivel de archivos), por lo que los datos pueden ser combinados para formar objetos compuestos, y cualquier tipo de datos (inclusive arreglos de dimensión dinámicamente variable) pueden formar parte de objetos compuestos. Ligas y asociaciones entre

objetos se pueden definir dinámicamente. El cliente más renombrado de Objectivity es el fabricante de aeroplanos Boeing, en Seatle.

ObjectStore es el producto desarrollado y comercializado por Object Design. ObjectStore es un manejador de BDOOs para las estaciones de trabajo SUN3 y Sun Sparc. Existen versiones para Microsoft Windows 3.0 y otras estaciones de trabajo en UNIX. ObjectStore, y sus herramientas de desarrollo, está basado en C++. El preprocesador proporciona una interfaz de desarrollo que permite comprimir código de C++ en enunciados optimizados. Además, el manejador de ObjectStore permite tipos parametrizados. Los tipos parametrizados son comúnmente utilizados para definir clases que contienen e incorporan código de C++ y que permiten el desarrollo de código C++ reutilizable.

ObjectStore puede complementarse con un diseñador de esquemas y herramienta CASE, un manejador de diccionario de datos y un depurador.

ONTOS es la BDOO que ofrece la compañía Ontologic, Inc. La característica principal de ONTOS es que incluye una interfaz totalmente transparente con C++ (esto es, transparentemente activa objetos en la memoria como se requiere dinámicamente). Además, permite, transparentemente, activar objetos desde el servidor de la base de datos y proporciona una interfaz muy versátil con otros productos y herramientas de ONTOS como ONTOS SQL, Ontos Studio, Ontos Shorthand y Ontos Ptech. ONTOS está completamente basado en el modelo cliente- servidor, proporciona soporte para datos distribuidos y aplicaciones distribuidas en redes cliente - servidor. El protocolo de transacciones distribuido soporta "two-pahse commits" atómicos. El producto ONTOS está disponible para Sun 3, 4 y SPARC, y las series Apollo 3000 y 4000 y múltiples equipos en OS/2.

GemStone es el producto distribuido por Servio Corporation. Gemstone está desarrollado en C y está basado en un diseño sobre una arquitectura cliente - servidor con múltiples hilos de ejecución. La mayor virtud es su soporte para múltiples plataformas, estaciones de trabajo y servidores que van desde IBM, Digital y Sun. Además puede implantarse en PCs IBM y compatibles, así como Macintosh II. El lenguaje de definición de datos y el lenguaje de manipulación puede integrar código en C o C++, Smalltalk V, Smalltalk-80, ADA y otros lenguajes de programación. Gemstone puede compartir una interfaz con Sybase. Tiene un sistema de respaldo que permite uso continuo las 24

horas del día. Objetos definidos en C++ tienen su identidad propia y coexisten independientemente de la aplicación que los crea, luego pueden almacenarse y posteriormente reinvocados por la misma aplicación u otras aplicaciones. El paquete incluye una herramienta gráfica para la definición y creación de clases llamada Gemstone Visual Schema Designer. Además existe otra serie de aditamentos para el desarrollo disciplinado de aplicaciones. Gemstone tiene más de 120 sistemas instalados internacionalmente.

Versant es la BDOO de tipo distribuido y multiusuario, en un modelo multicliente/multiservidor que desarrolló Versant Object Technology. Está disponible para la Silicon Graphics IRIS 4D y todas las estaciones Sun. Versant es el producto que soporta más estándares, incluyendo X Window, OSF/Motif, ANSI C, AT&T C++2.0, Smalltalk-80 y TCP/IP.

La arquitectura de Versant permite a varios usuarios compartir los mismos datos. Una aplicación puede actualizar datos desde varios servidores en una sola transacción, eliminando la necesidad de grandes bases de datos centralizadas, y permitiendo distribuir los datos donde son más frecuentemente utilizados. Los objetos pueden extraerse "en préstamo" de un grupo a una base de datos local para transacciones locales. Versant soporta "two-phase commits" y recuperación de "rollback", garantizando actualizaciones atómicas. Acompañan al manejador varios productos y herramientas de desarrollo entre ellas Versant View, Versant Schema Designer, Versant C++ y Versant 4GL.

ORION/ITACSA consiste de una serie de manejadores de bases de datos de una generación superior al modelo relacional cuyo desarrollo fue iniciado por el Advanced Computer Technology (ACT) Program, en el Microelectronics and Computer Technology Corporation (MCC) en 1985. ORION fue desarrollado como un vehículo para la integración de lenguajes de programación orientados a objetos y bases de datos avanzadas para aplicaciones en Inteligencia Artificial, CAD/CAM y sistemas de automatización de oficinas. La serie ORION incluye ahora tres sistemas, ORION-1, es un sistema monousuario, multitarea; ORION-1SX está diseñado para redes de área local en una arquitectura cliente - servidor; y ORION-2 es un manejador de bases de datos distribuidas. ORION "per se", está sólo disponible a miembros patrocinadores de ACT. La versión comercialmente disponible en el mercado se llama ITACSA y la distribuye Itacsa Systems Inc.

El sistema manejador de bases de datos O2 es producido por la firma francesa Altair y es un producto comercial bastante conocido. Los expertos consideran a O2 como un manejador de BDOO del mismo desarrollo que ITACSA. O2 es el resultado de un proyecto de investigación a largo plazo y constituye un sistema avanzado, tanto desde el punto de vista de la tecnología de bases de datos, como del ambiente de desarrollo. O2 no sigue la filosofía de diseñar una extensión de un lenguaje OO para incluir persistencia (como puede pensarse de Gemstone a partir de Smalltalk); tiene su propio lenguaje CO2 para implantar métodos, el cual es una extensión de C.

O2 está basado en un modelo de datos orientado a objetos en un sentido convencional donde la identidad de un objeto es independiente del valor y el encapsulamiento de valores y operaciones es central. Además O2 permite "valores complejos", que son valores definidos por constructores que pueden aplicarse a cualquier nivel de recursividad. Los objetos complejos son parejas (identificador, valor) cuya manipulación es sólo por métodos pero pueden ser compartidos.

O2 permite intercambiar código con lenguajes como LISP y BASIC y soporta tipos atómicos, estructurados y clases. O2 tiene herencia múltiple y los objetos se hacen persistentes al insertarlos en un conjunto persistente, o al formar parte de objetos persistentes o al recibir un nombre externo.

En O2 el lenguaje de consulta no puede usarse conjuntamente con el lenguaje de aplicación CO2 (al menos en la versión actual) y se utiliza principalmente en la interfaz gráfica. Sin embargo, es tan poderoso como SQL. O2 permite modificaciones dinámicas de las instancias de una clase y permite borrar clases. Se promete la modificación dinámica de esquemas para versiones futuras.

El manejador persistente de objetos OBST fue desarrollado bajo la coordinación del Forschungszentrum Informatik (FZI) como una contribución al proyecto STONE. Este proyecto fue financiado por el Ministerio Alemán de la Investigación bajo el fondo número ITS8902A7 con miras a desarrollar un ambiente de programación en ingeniería con propósitos educativos y fue realizado por 9 instituciones alemanas de investigación (universidades y centros de investigación).

Una característica esencial de STONE es que el paradigma de orientación a objetos es el principal

concepto a seguir. OBST es el almacén o depósito persistente para todas las herramientas en el ambiente STONE.

El modelo de datos de OBST se caracteriza por las siguientes propiedades:

- Definición de esquemas en un lenguaje sintácticamente muy similar a C++.
- Apoyo a la herencia múltiple.
- Clases Genéricas.
- Clases abstractas y métodos.
- Diferenciación entre métodos públicos, protegidos y privados.
- Métodos redefinibles.
- Métodos sobrecargables (overloading).

2.30 Nuevas Tecnologías

Actualmente los Sistemas de Información Geográficos continúan comercialmente implementados con SMBDR y las empresas que los utilizan invierten gracias al desarrollo tecnológico en máquinas más poderosas "encubriendo" de esta manera el manejo del volumen de datos y logrando ocultar la necesidad de implementaciones más rápidas, las consultas pueden ser más lentas pero en muchos casos, la mayoría, los requerimientos no son en tiempo real y por lo tanto parece ser que el tipo actual de SIG's satisface de cierta manera sus necesidades. Vale la pena destacar que en le caso de volúmenes pequeños de información las consultas parecen ser rápidas.

A pesar de lo anterior algunas empresas comercializadoras de Software están comenzando a generar nuevas herramientas (híbridos) para buscar la forma de acelerar los tiempos de respuesta. Se consigue ahora un motor de acceso a datos geográficos orientado a objetos con un alto rendimiento denominado SDE (Spacial Database Engine), el cual se implementa dentro de un ambiente de bases de datos relacionales (SMBDR) utilizando una arquitectura cliente/servidor. El almacenamiento, control y mantenimiento de las bases de datos se encuentran embebidos totalmente en el SMBDR.

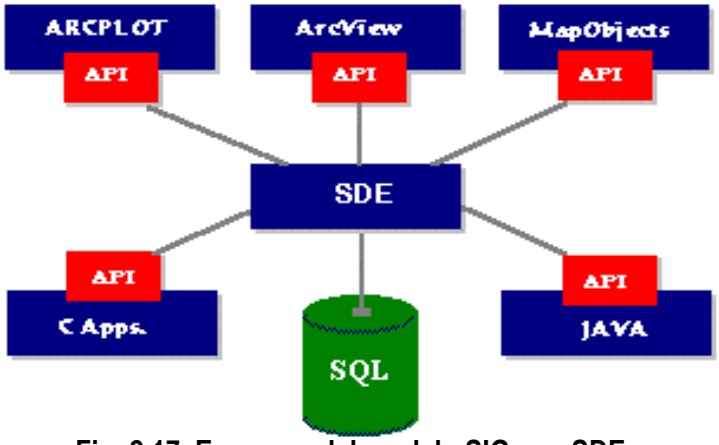


Fig. 2.17 Esquema del modelo SIG con SDE

SDE proporciona herramientas de acceso a datos geográficos utilizando complejas sentencias de búsquedas espaciales sobre bases de datos de millones de registros. Gracias a su estructura de geoprocesamiento compartido, SDE es más rápido que cualquier otro sistema de acceso a datos geográficos y proporciona una mínima degradación de rendimiento aún cuando cientos de clientes trabajen simultáneamente con la Base de Datos. SDE brinda un set de herramientas de procesamiento espacial para realizar todo tipo de análisis geográfico.

La comunicación con SDE se realiza utilizando TCP/IP y XDR (eXternal Data Representation), estándares del mercado que permiten a los clientes de cualquier plataforma (UNIX, Windows 3.1, Windows NT, y Windows 95) acceder a SDE mediante redes de área local (LAN) o redes distribuidas (WAN).

SDE está totalmente integrado con los productos de ESRI,[31] ya que tanto ArcView GIS, Map Objects[32] y ARC/INFO 7.1.2 pueden ser clientes de SDE.

Existe también al nivel de investigación (Tesis de Grado) la implementación de clases espaciales para la visualización de mapas en el mundo WWW. Contiene la extensión de las características de un mapa, adicionándole propiedades interactivas de un hiperdocumento con una organización jerárquica, permitiendo navegación no necesariamente lineal, sino de tipo aleatorio, para accesar información de tipo gráfico, texto, video, sonido, etc.

Como se observa el panorama todavía debe ser explorado y esta investigación pretende, finalmente ofrecer la adaptabilidad de un SMBDOO a los SIG's y evaluar sus ventajas y desventajas comparadas con los SIG's actuales y con los que ya se empiezan a proyectar con las nuevas herramientas a pesar de continuar en ambiente relacional.

2.31 UML *(*Lenguaje de Modelado Unificado*)*

El tamaño y complejidad de los nuevos sistemas de software, conllevan a la búsqueda de nuevas técnicas y/o herramientas que sirvan de apoyo a las actividades de análisis, diseño y documentación de estos. Una de estas herramientas es el Lenguaje de Modelado Unificado o UML (Unified Modeling Language), el cual fue creado y mantenido por la compañía Rational Software y que en 1999 fue aceptado por el Object Management Group (OMG) como un estándar de modelado de sistemas de software orientados por objeto.

El Lenguaje de Modelado Unificado (UML) es un lenguaje para especificación, visualización, construcción y documentación de elementos de sistemas de software, modelado de negocios; y además puede ser usado para modelar otros sistemas que distintos al software[33] . El UML representa una colección de las mejores prácticas que han probado ser exitosas para el modelamiento de sistemas grandes y complejos.

Los beneficios más importantes que se pueden obtener con el uso del UML son:

1. Suministra, a los desarrolladores, un lenguaje de modelado visual expresivo y de fácil uso para desarrollar modelos. El UML consolida un conjunto de conceptos de modelado básicos que han sido aceptados en varios métodos de desarrollo y herramientas de modelado.
2. Provee una base formal para el entendimiento del lenguaje de modelado.
3. Posee independencia de cualquier lenguaje de programación o proceso de desarrollo de software específico.
4. Soporta conceptos de desarrollo de alto nivel como patrones, componentes, marcos de trabajo y colaboraciones. Al incorporar estos conceptos, se obtienen los beneficios de la orientación a objetos y la reutilización.

5. Provee mecanismos de extensión y especialización para extender los conceptos básicos del lenguaje.
6. Integra las mejores prácticas, al abarcar una gran variedad de vistas basadas en niveles de abstracción, dominios, arquitecturas, fases del ciclo de vida, tecnologías de implementación, entre otras.

Por otra parte, el Lenguaje de Modelado Unificado o UML es un lenguaje estándar de modelado y no un proceso estándar de desarrollo. El UML debe ser aplicado en el contexto de un proceso de desarrollo, de acuerdo a la experiencia de la organización y el tipo de problema; porque en ocasiones, distintos problemas, requieren distintos procesos de desarrollo.

2.32 Diagramas del UML

La selección de modelos y diagramas que se crean, tienen profunda influencia en cómo atacar un problema y cómo es concebida una solución.

En términos de vistas del modelo, el UML define los siguientes diagramas gráficos[34]:

- Diagramas de casos de uso
- Diagramas de clases
- Diagramas de comportamiento
    - Diagramas de estados
    - Diagramas de actividades
    - Diagramas de interacción
        - Diagramas de secuencia
        - Diagramas de colaboración
- Diagramas de implementación
    - Diagramas de componentes

- Diagramas de distribución

Estos diagramas proveen múltiples perspectivas del sistema a analizar o desarrollar; a continuación se presenta una breve descripción de cada uno de ellos[35] , [36].

2.32.1 Diagramas de casos de uso[37]

Un diagrama de casos de uso (Fig. 2.18) muestra actores y casos de uso, unidos a través de relaciones. Los casos de uso son utilizados para definir el comportamiento (funcionalidad) de un sistema o entidad sin revelar la estructura interna de éstos. Cada caso de uso especifica una secuencia de acciones, incluyendo variantes que la entidad puede ejecutar interactuando con actores de la entidad o sistemas.

Una actor define un conjunto coherente de roles que juegan los usuarios de una entidad o sistema cuando interactúan con éste. Un actor puede ser tanto una persona como otro sistema de software o dispositivo externo.

La relación más común que se encuentran en los diagramas de casos de uso son las relaciones de asociación, las cuales describen la participación de un actor en un caso de uso.

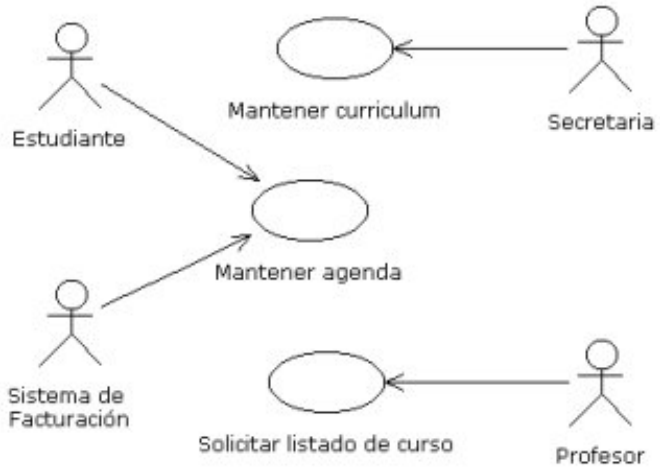


Fig. 2.18. Casos de Uso

2.32.2 Diagrama de clases

Un diagrama de clases (Fig. 2.19) es una vista gráfica que muestra la estructura estática del modelo.

Un diagrama de clases es un diagrama que consta de elementos clasificadores (clases, paquetes e interfaces) conectados a través de varias relaciones estáticas (asociaciones, herencia, agregación). En un diagrama de clases se pueden encontrar tanto clases, como interfaces, paquetes e instancias como objetos y enlaces. Una relación define una relación semántica entre clasificadores.

Una clase es un descriptor (clasificador) de un conjunto de objetos, los cuales poseen una estructura, comportamiento, relaciones y semántica común. La colección de métodos, operaciones y atributos describen completamente los objetos.

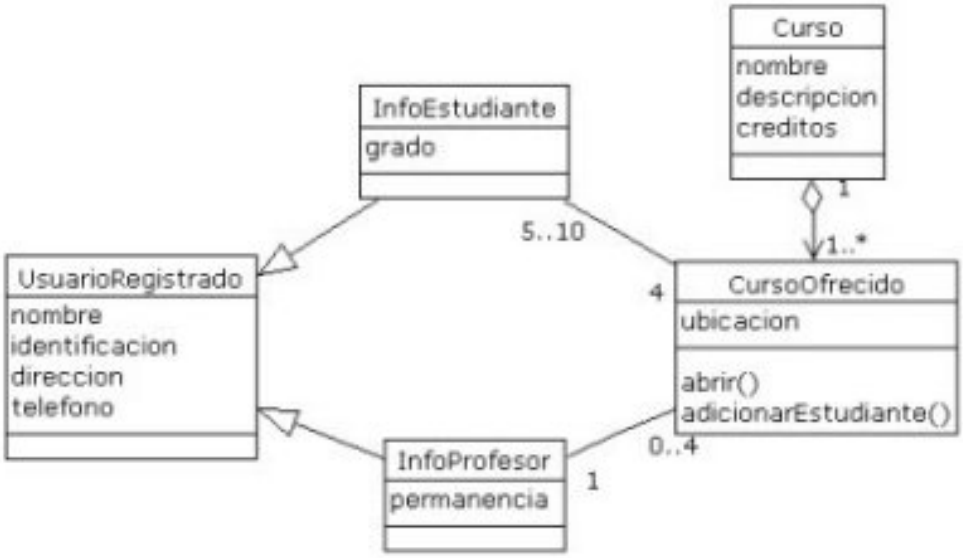


Fig. 2.19. Diagrama de clases

2.32.3 Diagramas de Estados

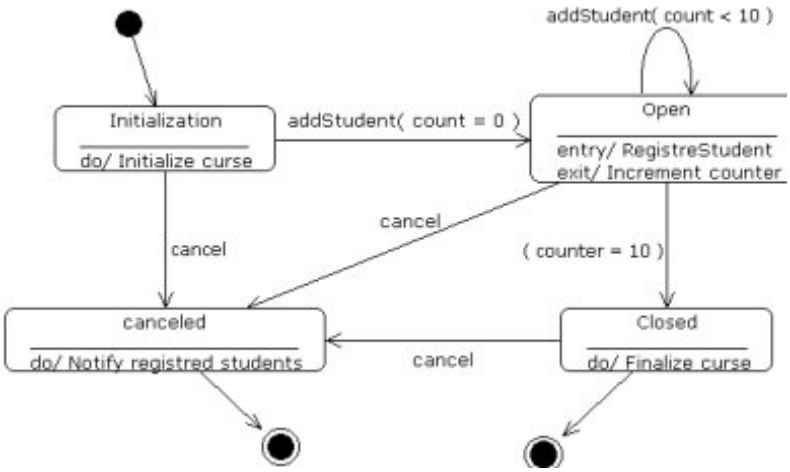


Fig. 2.20. Diagrama de Estados

Un diagrama de estados (Fig. 2.20) es utilizado para describir el comportamiento de un elemento del modelo; típicamente son utilizados para describir el comportamiento dinámico de una clase, aunque pueden ser utilizados para describir el comportamiento de otras entidades como actores, casos de uso, subsistemas, operaciones o métodos.
Un diagrama de estados (máquina de estados) es un grafo con estados y transiciones que describen la respuesta de un objeto a estímulos recibidos.

Un estado es una condición de un objeto o interacción durante el cual se satisface una condición, se ejecuta una acción o se espera algún evento.

Una transición es una relación entre dos estados que indican que un objeto que está en el primer estado, después de ejecutar una acción, pasa al segundo estado cuando ocurre un evento específico y se satisface una condición.

2.32.4 Diagramas de Secuencia

Un diagrama de secuencia (Fig. 2.21) es una interacción ordenada en el tiempo. En particular, éstos muestran la participación de los objetos en una interacción y los mensajes que éstos intercambian (no muestran las asociaciones entre objetos).

El propósito de una interacción es el especificar la comunicación entre un conjunto de objetos que interactúan en la ejecución de una tarea específica.

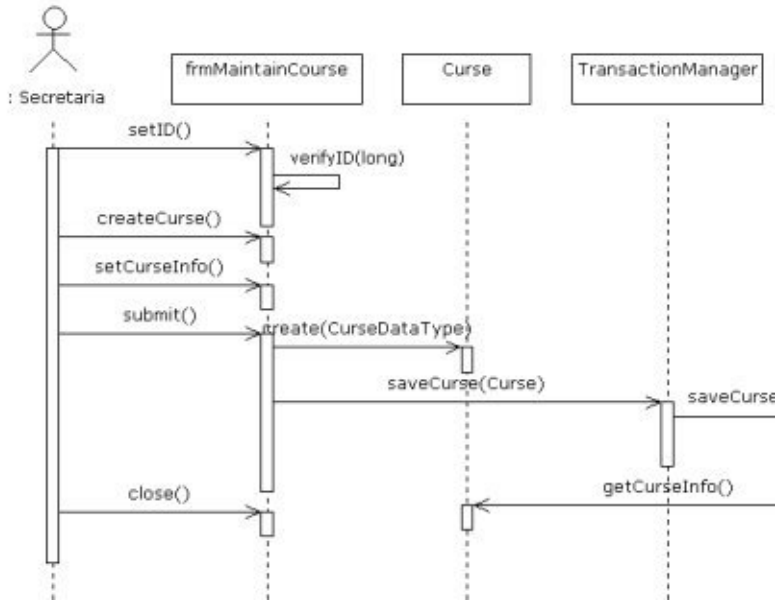


Fig. 2.21. Diagrama de secuencia

2.32.5 Diagramas de Colaboración

Una diagrama de colaboración (Fig. 2.22) muestra una interacción organizada alrededor de los objetos y sus enlaces con otros objetos. A diferencia de los diagramas de secuencia, los diagramas de colaboración muestran las relaciones entre los objetos y no la relación del paso de mensajes a través del tiempo.

La colaboración puede estar asociada a una operación o caso de uso, para describir el contexto en el cual ocurre el comportamiento.

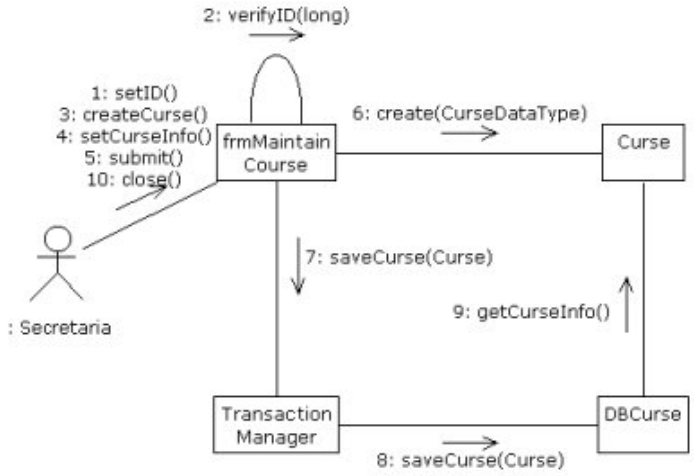


Fig. 2.22. Diagrama de colaboración

2.32.6 Diagramas de Actividades

Un diagrama de actividad (Fig. 2.23) es una variación de una máquina de estados en el cual los estados son actividades representando la ejecución de operaciones y las transiciones son disparadas al concluir las operaciones.

Estos diagramas son asociados a las clases, a la implementación de una operación o a un caso de uso.

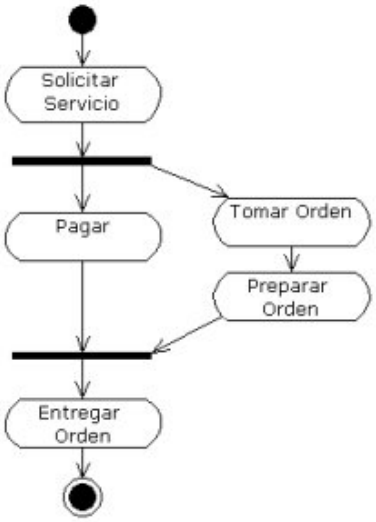


Fig. 2.23. Diagrama de actividades

2.32.7 Diagramas de componentes

Los diagramas de componentes (Fig. 2.24) muestran las dependencias entre componentes de software, incluyendo código fuente, código binario y componentes ejecutables.

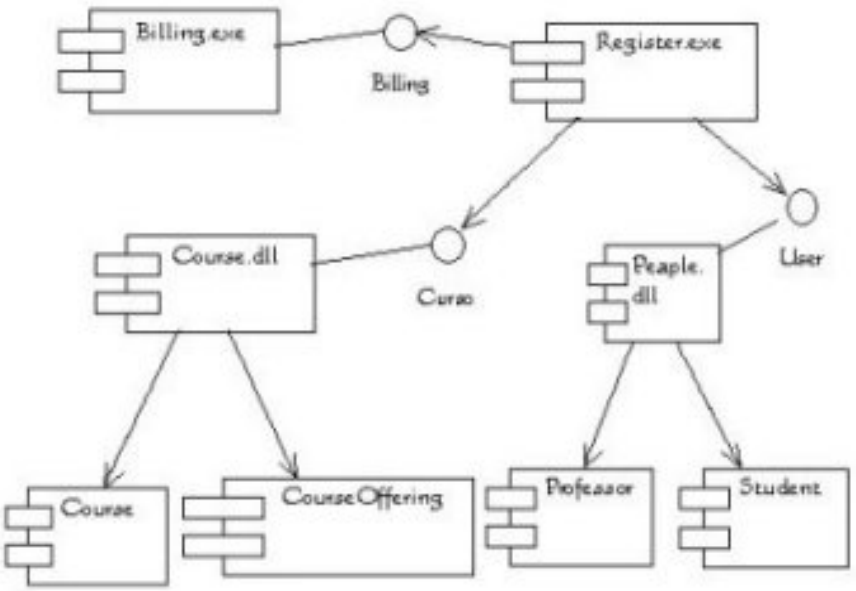


Fig. 2.24
Diagrama de componentes

2.32.8 Diagramas de Distribución

Los diagramas de distribución (Fig. 2.25) muestran la configuración de los elementos en el proceso de ejecución y los componentes de software.

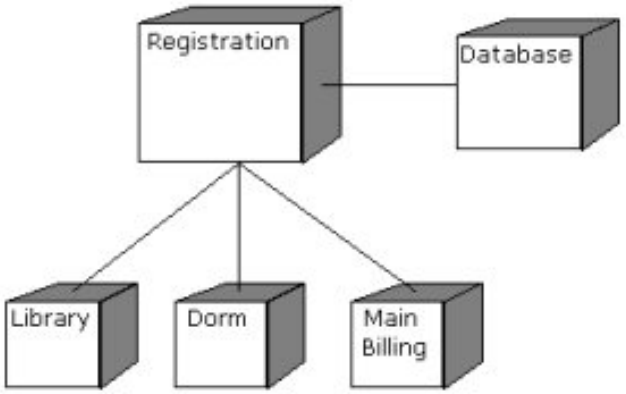


Fig. 2.25
Diagrama de distribución

2.33 Estado del Arte

La aplicación del modelo Objeto fue implementada en los sistemas de Información Geográfica por Luis A. Colmenares G.[38] en su trabajo de Tesis: "Conversión e implementación en lenguaje java de la librería de clases de objetos espaciales, para su presentación y manipulación en visualizadores del tipo WWW " como parte del proyecto "Hipermapas y Grafos Espaciales", el cual tiene por objetivo el establecimiento de un sistema orientado a objetos para crear, manipular, acceder y consultar Hipermapas, en aplicaciones multimedia y geográficas.

El objetivo principal de este trabajo, en consecuencia, es el de convertir, implementar y mejorar dicha librería de clases en Lenguaje Java. Para ello se usó un diseño de acuerdo a Técnica de Modelado por Objetos (OMT). Dichas clases serán la plataforma base de un Sistema de Información Geográfica (SIG), el cual está relacionado con el procesamiento de datos geográficos referenciados; esto es, datos que emplean un sistema de referencia que determina la localización en el espacio de las entidades que ellos representan.

La noción de objeto espacial es particularmente útil para representar entidades espaciales en un SIG orientado por objetos. El objeto espacial es un constructor de modelado que captura explícitamente las propiedades geométricas y topológicas de una entidad, en un espacio determinado. Las propiedades geométricas están relacionadas con la localización de la entidad en un sistema coordenado dado, con la forma o geometría (área, perímetro, centro, etc.); mientras que las propiedades topológicas determinan sus relaciones de conexión, adyacencia e inclusión que la entidad mantiene, con respecto a otras entidades, en un espacio determinado.

Un hipermapa es la extensión de las características de un mapa, adicionándole a éste propiedades interactivas de un hiperdocumento con una organización no jerárquica, permitiendo una navegación no necesariamente lineal, sino más bien de tipo aleatorio, para accesar una información ya sea del tipo gráfica, textos, animación, vídeo, sonido, etc.; acerca de una región geográfica determinada. Se trata de un mapa multimedia interactivo, con hiperenlaces que permiten extender el conocimiento tan extensivamente como se quiera.[39]

Entiéndase por navegación, el proceso de accesar la información por medio de clicks del mouse sobre cualquier objeto con capacidad de producir un enlace a otro objeto multimedia (texto, gráfico, sonido, etc.) o hipermedia (con iguales capacidades de generar enlaces).

Para ello se han desarrollado una serie de pruebas, direccionadas por la página principal de Hipermapas. Para la implementación del prototipo de demostración final, se modeló el Mapa de la división política territorial para 1996 del Estado Mérida, Venezuela. Para ello, se utilizó como fuente las coordenadas de los puntos estimados por el Grupo GIDYC, en trabajos anteriores.

Es importante destacar que en la Librería de Clases Espaciales implementada, incorpora el cambio de escala, característica (opcional) que presenta la nueva implementación, por lo que se aprovecha este recurso para escalar la representación. El modelo desarrollado y orientado a objetos, contiene, no solamente, las características geográficas comunes que posee cualquier mapa, sino que también tiene la capacidad de contener información topológica, referente al objeto representado. Obteniéndose una plataforma especializada para cualquier Sistema de Información Geográfica que se quiera desarrollar.

La librería de clases lograda, es una poderosa herramienta de apoyo para el desarrollo de SIG; además, tiene la capacidad para contener nuevas clases especializadas en el manejo de objetos multimedia no triviales, tales como el video. Cabe destacar que estas clases han sido implementadas para objetos presentados en dos dimensiones 2D, por lo tanto, no modelan objetos tridimensionales.

La librería de clases proporcionada, resuelve la mayoría de los obstáculos que se presentan en lo referente a la portabilidad, robustez y seguridad de ejecución. Queda, entonces, por extender su radio de aplicación a otros objetos multimedia, como el sonido y el video. Igualmente, se recomienda el desarrollo de una librería que suministre la propiedad de persistencia a la librería implementada; es decir, que tenga la capacidad de almacenar y recuperar la información, en una base de datos especializada para tal fin.

Finalmente, se recomienda el desarrollo de una interfaz gráfica (GUI), la cual, junto con las librerías que efectúen la persistencia, permitirá una rápida expansión y almacenaje de datos, para obtener un completo Sistema de Información Geográfico. También sería deseable que el SIG tuviese la capacidad de responder interrogantes acerca del objeto que, en un momento determinado, se esté visualizando, tales como: saber cual es el área que ocupa, o la densidad de población, etc.

3 EL MEDIO ECOSISTÉMICO[40]

3.1 Geología Regional

La geología del área ha sido estudiada previamente por autores como Hubach y Alvarado 1934; Reiser 1954, Van Der Hammen 1958, Nelson 1962, Misión Belga 1963, Cucalón y Schwinn 1969, Orrego 1975, Borrero 1972 y la firma IGLA Consultores S.A.. Un resumen cronológico se sintetiza a continuación:

- Las rocas más antiguas corresponden a la edad cretácica superior, donde sucedió una intensa actividad volcánica que dio origen al grupo diabásico que corresponde a una serie de rocas diabásicas y basálticas, con intercalaciones de sedimentario liditas y arcillolitas, que se encuentran en la zona más alta y occidental de las cuencas en estudio.

- Debido a que las erupciones volcánicas no fueron continuas en los períodos intermedios a esta actividad, se depositaron materiales sedimentarios clásticos de arenas, arcillas y chert que a su vez se intercalaron con material diabásico y basáltico. Este material sedimentario y volcánico está localizado en la parte media y baja de las cuencas afectadas por erosión antrópica, debido entre otras razones, al aprovechamiento inadecuado de los pastos.

- A finales del cretáceo y coincidiendo con la disminución de la actividad volcánica, se suceden las intrusiones tonalíticas y de tipo gabroide que se encuentran a lo largo de la Cordillera Occidental. En la región estudiada se identificaron en inmediaciones de Mediacanoa y en forma de intrusiones en la quebrada Mulaló.

- Durante el terciario reciente (Oligoceno-Mioceno) se presentaron movimientos tectónicos orogénicos, acompañados de transgresiones y regresiones del mar. Este fenómeno originó los depósitos sedimentarios de tipo continental como areniscas y conglomerados con intercalaciones de arcillas, calizas y tobas volcánicas. Estos materiales se localizan en inmediaciones del Municipio de Vijes y la quebrada Mulaló, presentando problemas graves de erosión,

especialmente por el aprovechamiento inadecuado de la caliza.

- Durante el plioceno-pleistoceno, se presentó el levantamiento final de las cordilleras y el retiro definitivo del mar y se activaron los fenómenos de erosión y sedimentación.

- Con la orogénesis de la Cordillera Occidental, las rocas fueron sometidas a tensiones y comprensiones originando diaclasamientos, fallamientos y plegamientos; los cuales normalmente tienen orientación Norte - Sur, con huellas bien marcadas por la sucesión de fenómenos erosivos paralelos al sistema y de los cuales se hace una narración con mayor detalle en el capítulo de factores y procesos de la erosión.

- El cuaternario reciente está representado por pequeños valles, conos terrazas y abanicos que se observan en la zona de transición hacia el plano aluvial del valle geográfico del río Cauca.

La formación estructural y estratigráfica de la zona de estudio está ligada a la evolución de los Andes Colombianos, cuyo desarrollo es el resultado de la interrelación de las placas de Nazca, Caribe y Suramericana.

La conformación estructural de la zona se encuentra enmarcada por un sistema de fallas (Santa Ana y de Cali) cuya mayoría son fracturas de carácter regional, este rasgo estructural bajo los esfuerzos derivados del movimiento de las placas dan como resultado diversos plegamientos. En la región de Yumbo afloran las siguientes unidades litológicas, de más antigua a más joven.

- Formación Volcánica de la Cordillera Occidental, constituida por basaltos y diabasas (rocas de corteza oceánica), conforman la mayor parte de la Cordillera Occidental y por tanto de su flanco oriental, son de edad Cretácea superior (mayor de 60 millones de años).

- Rocas Sedimentarias del Grupo Cauca, predominantemente de origen continental y en menor cantidad de origen marino, tienen sus mayores espesores en el sur del departamento y adelgazan hasta desaparecer totalmente en la zona próxima a Yumbo.

El Grupo Cauca está integrado por tres Formaciones: Chimborazo, Guachinte y Ferreira.

La Formación Guachinte está conformada por rocas sedimentarias principalmente de origen continental, en contacto discordante con los basaltos sobre los cuales reposa directamente al norte de Pance, disminuyendo su espesor y desapareciendo en la zona de Yumbo.

La Formación Guachinte contiene carbones, los que han sido explotados desde comienzos del siglo en el sector occidental de Cali; algunos mantos de menor espesor y menores propiedades térmicas existen hacia las proximidades de Yumbo, lo que explica que allí se haya localizado una pequeña minería asociada a este combustible fósil.

La edad de las rocas sedimentarias del Grupo Cauca es Mioceno inferior (unos 25 millones de años).

- Formación calcárea de Vijes: rocas sedimentarias de origen marino que afloran al norte de Yumbo, conformada por calizas y areniscas calcáreas, de edad Oligoceno superior (mayor de 25 millones de años). Estas rocas que desde el punto de vista de geología regional tienen poca extensión, son la fuente de la materia prima para Cementos del Valle.

- Depósitos Cuaternarios: Conos aluviales de los drenajes procedentes del flanco oriental de la Cordillera Occidental (entre ellos el río Yumbo), con edades no determinadas con precisión, pero posiblemente más jóvenes que 10.000 años. Gran parte de la zona urbana del Municipio se localiza sobre esta unidad litológica.

- Depósitos Cuaternarios: relleno aluvial por parte del río Cauca del valle geográfico del mismo nombre, cuyas edades no se han determinado con precisión, pero los estratos superiores pueden ser más jóvenes que 10.000 años. Parte de la zona urbana y la totalidad de la zona industrial se encuentran localizadas sobre esta unidad geológica.

- Depósitos Cuaternarios: depósitos asociados a movimientos de masa de diversos tipos (deslizamientos, caídas de roca, flujos de lodo locales, etc.) que recubren parcialmente la zona

de ladera de Yumbo y derivados de las unidades litológicas situadas inmediatamente encima. Sobre este tipo de materiales están localizados los barrios de ladera, entre ellos Las Cruces, Nuevo Horizonte y Bellavista.

3.2 Geomorfología y Relieve

En el Municipio de Yumbo se encuentran geoformas con variaciones que comprenden desde planas o casi planas hasta escarpadas, presentándose en forma drástica y no gradual. En términos generales la conformación del terreno divide al Municipio de Yumbo en dos áreas: una plana al oriente, que hace parte del fértil Valle del río Cauca y representa el 26,1% del Municipio y un área montañosa, al occidente, perteneciente al relieve de la Cordillera Occidental y que corresponde al 73,9% de su extensión total.

El Municipio está dividido en cinco unidades geográficas las cuales se han tenido en cuenta para determinar la topografía y pendientes del Municipio en general. La unidad que presenta mayor pendiente (51,9%) es la cuenca de la Quebrada Mulaló con vertientes laterales empinadas y un cauce de drenaje rápido. El sector Bermejal con una pendiente de 45.7% y que, a pesar de tener una extensión considerable de terreno plano, presenta una zona bastante escarpada. Le siguen en su orden, la cuenca del río Yumbo con una pendiente de 41.4%, el Sector Guabinas con 32.0% y la Cuenca de Arroyohondo con 30.6%.

Las elevaciones del área urbana de Yumbo varían entre las cotas 950 a 1100 m sobre el nivel del mar. Las cuencas que integran la hidrografía de los cerros aledaños a la ciudad, fluyen a través de los drenajes de la zona plana hacia el río Cauca.

De 10.171.932 m^2 de área urbana, 7.902.939 m^2 corresponden a zonas planas, 1.357.666 m^2 a zonas empinadas (zona residencial estrato I) y 911.334 m^2 a zonas inclinadas (zona residencial estrato II: 526.334 m^2 y 385.000 m^2 a lotes). Las zonas planas se encuentran repartidas de la siguiente manera: 168.500 m^2 en zonas de uso comercial, 1.958.758 m^2 uso residencial estrato II, 263.666 m^2 uso residencial estrato III, 4.272.000 m^2 uso industrial, 348.000 m^2 uso institucional y 892.008 m^2 son lotes.

Desde el punto de vista de las geoformas, la zona urbana e industrial de Yumbo se localiza así:

- En el cono aluvial del río Yumbo: zona urbana
- En el cono aluvial del río Yumbo, parte distal: una porción de la zona industrial
- Zona de ladera, sobre rocas meteorizadas (basaltos) y sobre depósitos de vertiente: zona urbana, barrios de ocupación posterior al año 1970.
- En los depósitos aluviales asociados al río Cauca: zona industrial

La zona industrial, en su mayoría plana, con una leve pendiente hacía el río Cauca, está conformada por tres corredores:

- El comprendido entre el pie de monte de la Cordillera Occidental, cota 1000, y la antigua carretera que de Cali conduce a Yumbo.

- El corredor central, casi plano con pendiente menor del 7% donde se ha desarrollado principalmente la industria, comprendido entre la antigua vía Cali-Yumbo y la cota de inundación del río Cauca.

- La zona más baja correspondiente al área inundable del río Cauca, considerada como zona de protección del mismo.

3.3 Hidrografía

El sistema hidrográfico del Municipio de Yumbo está conformado por tres cuencas mayores:

- Quebrada Arroyohondo,
- Cuenca superior del Río Yumbo,
- la Cuenca de la Quebrada Mulaló

y tres menores o subcuencas:

- Sector Guabinas,
- Sector del Bermejal y
- Cuenca Quebrada San Marcos

La cuenca de la Quebrada Arroyohondo, está formada por cuatro sectores:

- Alto Arroyohondo, donde caen varias Quebradas pequeñas como la Sonora, Otobal y el Rincón.
- Sector Medio Arroyohondo
- Subcuenca de la Quebrada Pérez compuesta por la Quebrada Juanambú y Pedregal
- Sector Bajo Arroyohondo surtida por la Quebrada Dapa

La Cuenca superior del Río Yumbo atraviesa todo el casco urbano en dirección occidente - oriente, formada por:

- Subcuenca Quebrada Yumbillo donde desembocan las Quebradas Los Monos, San Antonio, Salazar, Del Aguacate y San Pedro
- Subcuenca Quebrada Santa Inés compuesta por las Quebradas Carbonero, Nacederos y El Chocho
- Subcuenca de la Buitrera compuesta por varias pequeñas vertientes

La Cuenca de la Quebrada Mulaló por las Quebradas los Chancos, Crestagallo, Piedra Grande, Trapo Viejo y el Cangrejo

Las subcuencas se configuran de la siguiente manera:

- Sector Guabinas compuesta por la Quebrada Guabinas, la Rafaela y Quebrada del Guayabo

- Sector del Bermejal

- Cuenca Quebrada San Marcos, compuesta por las Quebradas la Esmeralda, Guadualito, Peñaliza, Salado Blanco y los Indios

CUENCA	QUEBRADA	CAUDAL PROMEDIO (l/s)
Arroyohondo	Pérez La Sonora El Rincón Dapa	27,0 11,0 21,0 17,5
Yumbo	La Buitrera Yumbillo Santa Inés	22,0 59,0 88,0
Mulaló	Chancos Mulaló	4,0 13,0
Bermejal	Bermejal	1,3
Guabinas	Guabinas	0,1

Tabla No 3.1. Hidrografía[41]

FUENTE	CAUDAL ASIGNADO Lts/seg.	USO DOMESTICO Lts/seg.	USO AGRICOLA	USO INDUSTRIAL	No. USUARIOS	CAUDAL ASIGNADO J.A.C.
Río Arroyohondo	235.00	30.0	165.00	40	54	8.50
Río Arroyohondo	6.49	3.49	3.00		5	0.45
Q. La Sonora	22.60	19.02	3.58		30	5.40
Q. La Tranquilidad	4.65	1.00	3.65		16	
Q. El Champan	6.45	6.45			24	
Q. La Olga	3.38	2.20	1.18		14	2.20
Q. El Rincón	11.75	7.69	4.06		30	0.19
Q. Pilas Dapa	18.00	18.00			6	3.00
Q. La Menga	4.70	4.7			7	
Q. Las Brisas	2.76	1.53	1.24		10	0.86
Q. Mulaló	10.10	8.95	1.15		9	5.00
Q. Miravalle	0.30	0.3			1	
Q. La Buitrera	4.93	2.9	2.03		9	0.60
Río Yumbo	30.28	12.16	10.12	8.00	10	12.16
Q. S/n 05	1.00	1.00			1	
Q. S/n 06	0.85	0.85			116	
Q. S/n 07	0.15	0.15			1	
Q. S/n 08	0.10	0.10			2	
Q. S/n 09	0.06	0.06			1	
Q. S/n 10	0.05	0.05			1	
Q. S/n 12	1.00	1.00			1	1.00
Q. S/n 13	1.19	1.19			3	
Q. S/n 14	3.00	3.00			2	3.00
Q. Pérez	3.34	2.00	0.34		9	2.00

FUENTE	CAUDAL ASIGNADO Lts/seg.	USO DOMESTICO Lts/seg.	USO AGRICOLA	USO INDUSTRIAL	No. USUARIOS	CAUDAL ASIGNADO J.A.C.
Q. Santa Clara	2.10	2.10			13	
Q. Santa Inés	3.44	3.44			5	
Q. La María	3.34	2.34	1.00		5	
Q. Yumbillo	11.02	1.02	10.00		5	
Q. Zanja Honda	2.90	2.90			2	
Q. El Otobal	2.03	2.03			2	
Q. El Diamante	4.14	4.14			4	
Q. El Higuerón	0.48	0.48			2	
Nto. Comando Beverly	3.60	3.60			4	
Q. La Culebra	0.52	0.52			1	0.52
Q. Cantarrana	1.70	1.70			5	1.00
Río Cauca	4745.70		1046.00	3766.00	19	
Q. El Estobacal	1.00	1.00			1	
Q. La Fontana	0.02	0.02			1	
Q. La Guaca	0.71	0.71			3	
Q. Guabinas	0.50	0.50			1	
Q. Juanambú	1.62	0.62	1.00		6	
Q. El Muerto	0.20	0.20			2	
Q. La Mina	0.03	0.03			1	
Q. La Nidia	1.00	1.00			4	
Q. Honda Yumbo	10.50	10.50			1	10.50
Q. La Cortina	2.25	2.25			1	2.25
Nto. El Barcino	0.20	0.20			2	
Q. Seca Menga	0.04	0.04			1	
Q. San Marcos	45.02		45.02		3	
Q. El Sinaí	2.85	2.85			3	2.25
Q. San Antonio	1.40	1.40			1	1.40
Q. Salazar	6.00	6.00			2	6.00
Q. S/n Dapa	0.10	0.10			2	
Q. La Baija	8.25	1.00	7.25		1	
Q. Nto. Vafore	0.02	0.02			2	
Q. Nto. No. 2	3.20				17	
Nto. S/n Yumbo 3	2.04	2.04			14	
Nto. S/n Yumbo 4	1.00	1.00			8	
Q. S/n Yumbo 11	0.86	0.86			12	
TOTAL FUENTES: 59	5242.70	120.05	1308.65	3814.00	418	68.28

Tabla No 3.2. Cuenca Yumbo – Arroyohondo. Caudales Asignados (Diferentes Usos) en su Jurisdicción[42]

4 ETAPAS Y ACTIVIDADES EN EL DESARROLLO DE SOFTWARE ORIENTADO POR OBJETOS USANDO UML

En el desarrollo del SIG propuesto, se deben realizar las tres etapas clásicas de un proceso de desarrollo de software que son: análisis de requerimientos, análisis y diseño, e implementación. A continuación se describen cada una de estas etapas.

4.1 Etapa de análisis de requerimientos

El objetivo de esta actividad: describir qué debe hacer el sistema, delimitarlo, identificar qué usuarios son fuente importante de información, entre otros.

En esta etapa se realizaron las siguientes tareas: analizar el problema (desarrollar el plan de requerimientos, encontrar actores y casos de uso), entender las necesidades del usuario final (los usuarios del sistema tienen diferentes perspectivas y necesidades del problema que deben tenerse en cuenta para la solución), definir el sistema (realizar un buen análisis sobre las solicitudes de los usuarios del sistema, hacer el modelo de casos de uso y mostrar los resultados en modelos y documentos) , definir el alcance del sistema (determinar el alcance del proyecto, los límites del sistema, identificar usuarios (representados por actores) que trabajarán con el sistema y definir el conjunto de casos de uso (o escenarios)) y refinar la definición del sistema (Describir en detalle el flujo de eventos de los casos de uso, describir más detalladamente, si es necesario, las especificaciones de los requerimientos de software, hacer un modelo y un prototipo de la interfaz de usuario y describir brevemente a los actores).

A continuación se muestra la documentación originada en esta etapa:

4.1.1. Identificación de Requerimientos

Se requiere implementar un Sistema de Información Geográfico para el manejo de los recursos hídricos de un municipio, como prototipo se propone crear un SIG para el municipio de Yumbo, que permita consultar la información de cuencas, subcuencas, estaciones, entre otros.

4.1.2. Recolección de Información

El primer paso para iniciar el análisis del sistema de información, consiste en recopilar toda la información requerida con el fin de conocer el mundo real en el cual se desenvolverá el sistema, además del dominio del problema a solucionar.

La información recolectada serán las bases de datos construidas en el PAAL[43], en formatos de Shape file y los mapas del Municipio de Yumbo en un formato estándar de un SIG.

4.1.3. Especificación de requerimientos

En adelante se presentaran los requerimientos en cuanto a la funcionalidad del proyecto y las necesidades técnicas para el cumplimiento de los alcances establecidos.

El usuario potencial de esta aplicación serían las personas interesadas en obtener información georeferenciada sobre la hidrografía del Municipio de una manera fácil de acuerdo a su poco entrenamiento con el sistema.

4.1.4. Requerimientos Funcionales

El producto debe cumplir funcionalmente con los siguientes requisitos:

Se debe tener un registro de la información que pertenece a cada cobertura y que se desplegará geográficamente o que servirá como dato de consulta como coordenadas, tipo de dato espacial y atributos respectivos por ejemplo estación meteorológica, distancias, descripción, dirección, gráficos alusivos o fotos, sonidos, videos, clima, etc.

Las coberturas empleadas para este Sistema De Información Geográfica son: Cuencas, Subcuencas, Ríos, Quebradas, Estaciones metereológicas.

La aplicación debe permitir:

Mostrar un mapa con múltiples coberturas, tales como Cuencas, Subcuencas, Estaciones, Ríos y Quebradas

Permitirá hacer acercamiento y alejamiento del mapa, además de desplazarse por toda su área.

- Dibujar elementos gráficos tales como puntos, líneas, círculos y polígonos, utilizándolos como una forma gráfica de selección.
- Permitirá consultar la información de algún punto seleccionado desplegándola automáticamente.
- Identificar elementos de un mapa al apuntarlos.
- Seleccionar elementos con una expresión SQL y OQL.
- Seleccionar elementos que se encuentran dentro de un perímetro alrededor de otro elemento.
- Consultar y actualizar los atributos de los elementos seleccionados.
- Mostrar los atributos de los elementos con texto de alguno de sus campos.
- Mostrar dinámicamente datos de tiempo real o de periodos de tiempo.
- Permitirá involucrar variables de tipo ambiental (como caudal, temperatura, longitud, entre otros)
- Permitirá la impresión de la información que se obtiene como resultado de las consultas implementadas.
- Mostrar en el mapa las Cuencas y Subcuencas, los ríos y quebradas que corresponden con una determinada descripción, caracterizada por sus atributos.

4.1.5. Requerimientos No Funcionales

El sistema deberá tener una gran capacidad de manejo y almacenamiento de datos, dado que además del manejo de datos multimedia que son de gran tamaño, los datos georeferenciados aumentan una carga grande al sistema. Deberá tener un ágil conjunto de rutinas de manejo gráfico y georeferenciado para el despliegue amable y rápido de mapas, videos, etc.

4.2 Etapa de análisis y diseño

El principal objetivo de esta etapa es el trasladar las necesidades de los usuarios a elementos o componentes de software que permitan la implementación de dichas necesidades.

En dicho proceso, se construyen una serie de diagramas que permiten ilustrar la forma como los distintos componentes (clases) cooperan para satisfacer los requerimientos tanto de los clientes como de los usuarios. De acuerdo a los requerimientos la aplicación tendrá la siguiente arquitectura:

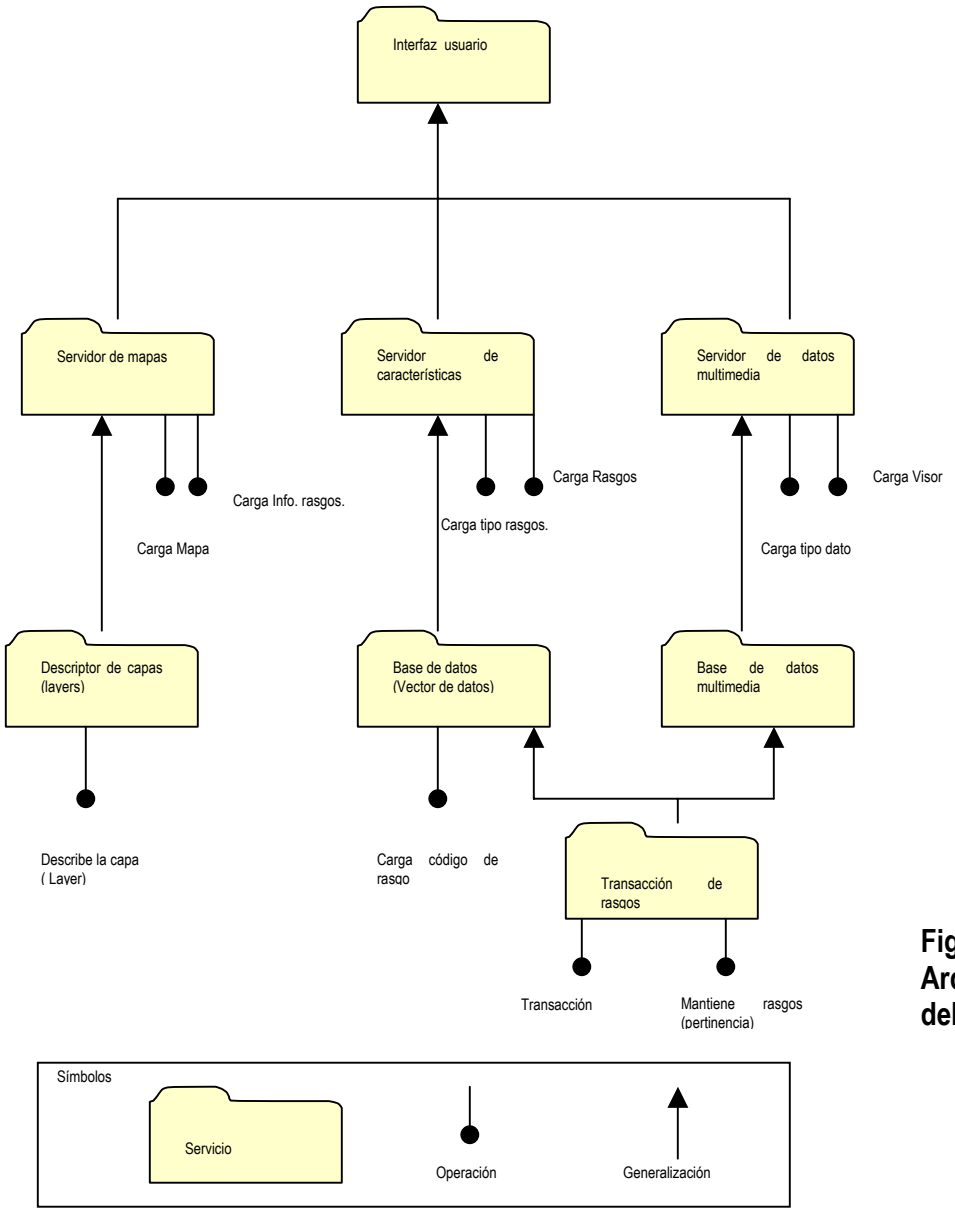


Fig. 4.1 Arquitectura del SIG

Arquitectura de la aplicación

El usuario debe poder controlar las funciones principales a través de barras de herramientas y menú de contexto incluidos en la interfaz, como se ilustra en el siguiente gráfico:

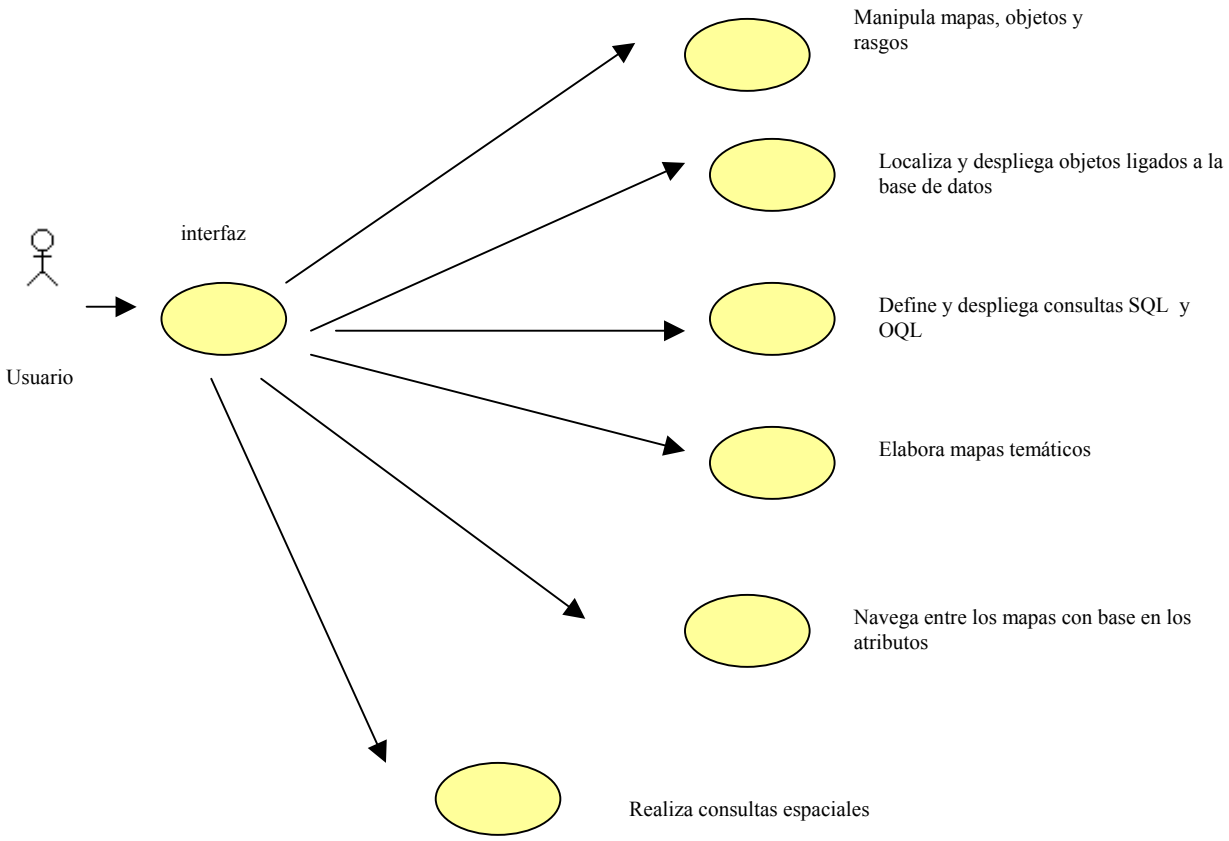


Fig. 4.2 Manejo de la aplicación por parte del usuario

4.2.1 Construcción del Modelo Objeto

Con este modelo se pretende mostrar la estructura estática del Sistema De Información Geográfica en lo referente a los datos que representan los objetos del mundo real y las relaciones que existen entre ellos.

Se modela primero la parte estática debido a que las estructuras estáticas tienen la posibilidad de estar mejor definidas y ser menos dependientes de los detalles de la aplicación, más fáciles de controlar en el desarrollo de la solución y más comprensibles.

El modelo de objeto se genera a partir de la definición del problema y del trabajo de campo que se ha realizado para la recolección de la información, en este caso en el Plan de Acción Ambiental Local.

Para la construcción del modelo objeto se identifican los siguientes pasos que posteriormente se desarrollan:

- Identificación de Clases de Objetos
- Identificación de Asociaciones
- Identificación de Atributos
- Descripción de Clases
- Esquema del Modelo Objeto

4.2.2 Identificación de clases de objetos

Como primera instancia para la modelación e implementación de un Sistema De Información Geográfica y más específicamente de su modelo objeto, identificamos las clases de objetos que se pueden observar en el

enunciado del problema.

La óptima descripción estática del sistema la componen las siguientes clases.

Vista	Subcuencas
Mapa	Ríos
Forma	Estaciones
Polígono	Línea
M_Cuencas	Punto
M_subcuencas	Medio
M_Ríos	Vídeo
M_Quebradas	Sonido
Cuencas	Imagen

A continuación se pueden apreciar las clases mencionadas y la forma como se relacionan

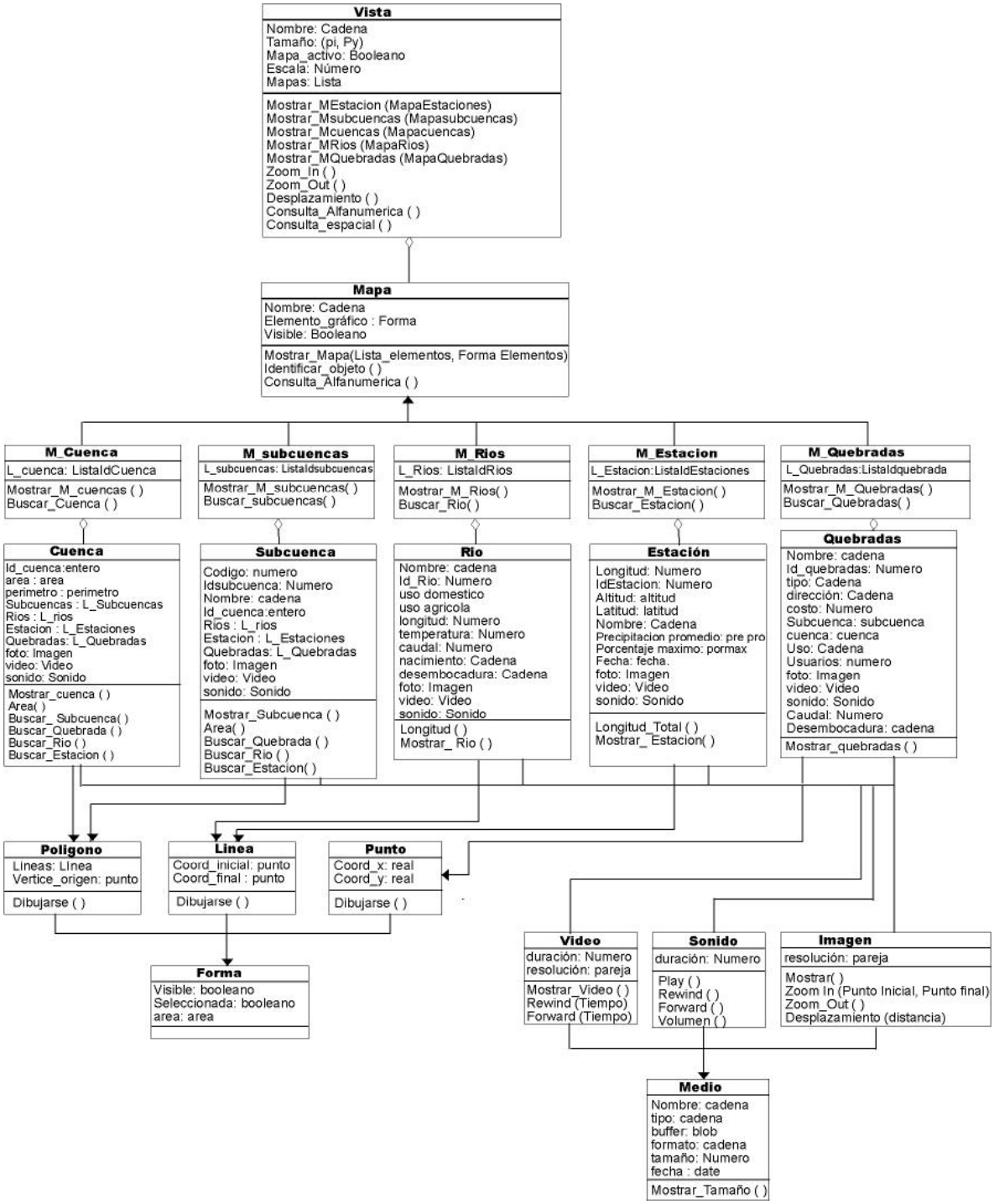


Fig. 4.3 Modelo Objeto

Una vez identificadas las clases que hacen parte del sistema se deben establecer las asociaciones existentes. Estas se determinan cuando existe una dependencia entre dos o más clases, de igual forma se considera a las referencias que hay de una clase a otra.

Las asociaciones muestran dependencias entre clases con el mismo nivel de abstracción que las clases en sí; estas se pueden implementar de diferentes maneras que deben mantenerse fuera del modelo de análisis, para conservar la libertad de diseño.

4.2.3 Descripción de clases

- Vista: Ventana para desplegar una colección de mapas.
- Mapa: Representación geográfica de la Tierra o parte de ella en una superficie plana. Conjunto de elementos de un mismo tipo o categoría que tienen una distribución espacial determinada.
- Forma: Es una clase que define las características genéricas de una forma geométrica. Tipo de elemento que usa el Sistema De Información Geográfica para representar los datos en este mapa. Ej: Cuencas se representa mediante polígonos, puntos de interés se representa mediante puntos y estaciones mediante puntos.
- Punto: Es una combinación de dos números o representación plana (X-Y) o esférica (longitud - latitud). Elemento con el que se marca la ubicación de algo sobre un sistema de coordenadas y posee algunos atributos adicionales.
- Línea: Extensión continua de una sola dimensión, definido por dos puntos y se puede considerar como una sucesión de puntos.
- Polígono: Un Polígono es una representación de un conjunto de partes (arcos). Cada parte (arco) es un conjunto de vértices conectado por segmentos que se cierra en el primer vértice. Un arco no puede cruzar consigo mismo.
- M_Cuencas: (mapa de cuencas)Clase que agrupa y dibuja todas las cuencas.
- M_Subcuencas: (mapa de subcuencas) Clase que agrupa y dibuja todos las subcuencas.

- M_Ríos: (mapa de ríos) Clase que agrupa y dibuja todos los ríos.
- M_Quebradas: (mapa de quebradas) Clase que agrupa y dibuja todos los quebradas.
- Cuenca: Es la unidad administrativa más grande en que se divide el municipio en cuanto al manejo de recursos hídricos
- Subcuenca : Las Subcuencas conforman la cuenca.
- Estación: Estación meteorológica.
- Río: Serán las corrientes de agua que atraviesan las distintas Cuencas y Subcuencas.
- Quebrada: Serán las corrientes de agua, de menor caudal y longitud, que atraviesan las distintas Cuencas y Subcuencas
- Medio: Todo elemento que se va a mostrar de forma visual o sonora.
- Imagen: Este será un medio que almacenará y mostrará las imágenes estáticas necesarias.
- Vídeo: Este será un medio que almacenará y mostrará los videos necesarios.
- Sonido: Este será un medio que almacenará y mostrará los sonidos necesarios.

4.2.4 Identificación de atributos

Los atributos son propiedades de objetos individuales, como el nombre, la velocidad o color. Los atributos no deben ser objetos y solo deben considerarse aquellos que estén relacionados directamente con la aplicación. A diferencia de las clases y las asociaciones, es menos probable que los atributos se describan por completo en la definición del problema. Es preciso recurrir al conocimiento, que se tiene, del dominio de la aplicación, y del mundo real, para encontrarlos. Los atributos solo afectan en raras ocasiones a la estructura básica del problema.

Los atributos identificados por clase son, (de acuerdo al Modelo Objeto):

- ***Vista:*** (Nombre, Tamaño, MapaActivo, Escala, Mapas)
  - *Nombre*: Identificación que se le asigna a la ventana donde se exhibirán los mapas en conjunto.

- *Tamaño*: Dimensión en la que la vista se mostrara al iniciarse la aplicación.
- *MapaActivo:* Indica cual es el mapa que se encuentra activo en el momento actual para ser utilizado en el momento de la realización de las consultas.
- *Escala:* Es la relación entre las dimensiones de una mapa y las dimensiones de la tierra. Esta es usualmente expresada como un radio entre una distancia sobre el mapa y una distancia sobre la tierra, tal como: 1:20,000, lo que significa que una unidad sobre el mapa representa 20.000 unidades sobre la tierra.

- *Mapas*: Lista de mapas a ser desplegados en la ventana de la aplicación, tales como:
  - *MapaCuencas*
  - *MapaSubcuencas*
  - *MapaRios*
  - *MapaQuebradas*
  - *MapaEstaciones*

❖ ***Mapa:*** (Nombre, Elemento_gráfico, Visible)

- *Nombre*: Nombre que se le asigna al mapa.
- Elemento_gráfico: Forma geométrica usado para dibujar los elementos que constituyen al mapa.
- Activo: Informa si el mapa se encuentra desplegado en la vista de la aplicación.

❖ ***Forma:*** (*Area, Visible, Seleccionada*)

- *Area:* Magnitud del espacio ocupado por el elemento que la figura representa en el mundo real.
- *Visible:* Campo que especifica si la figura se encuentra visible en el mapa que la está desplegando.
- *Seleccionada:* Campo que informa si el elemento ha sido seleccionado para efectos de alguna operación.

❖ ***Punto:*** (coordenada_x, coordenada_y)

- Coordenada_x: Posición sobre el eje horizontal.
- Coordenada_y: Posición sobre el eje vertical.

❖ ***Línea:*** (coord_inicial, coord_final).

- Coord_inicial: Punto en el que inicia la linea.
- Coord_final: Punto en el que finaliza la linea.

❖ ***Polígono*** (Líneas, vertice_origen)

- Líneas: Conjunto de líneas que conforman el polígono.
- Vertice_origen: Punto desde el cual se empieza a crear el polígono y donde se supone debe cerrarse.

❖ ***M_Cuenca***: (Lcuenca)

- Lcuencas: Lista de cuencas que constituyen el mapa de cuencas, cabe recordar que el objeto cuenca es una forma de tipo polígono esto se puede ver al analizar la herencia en el diagrama de clases.

❖ ***M_Subcuenca***: (Lsubcuencas)

- Lsubcuencas: Lista de Subcuencas que constituyen el mapa de Subcuencas. Esta clase está representada por una lista de polígonos.

❖ ***M_Ríos:*** (Lrios)

- Lrios**:** Lista de líneas que representan los ríos

❖ ***M_Quebradass:*** (LQuebradas)

- Lquebradas: Lista de Quebradas.

❖ ***Cuenca***: (IdCuenca, Nombre, Área, Perímetro, Subcuencas, Ríos, Quebradas, Estaciones)

- *IdCuenca*: Número que identifica a las cuencas.
- *Nombre*: Identificación de la cuenca.
- *Area*: Dimensión de la superficie que ocupa la cuenca.
- *Perímetro*: Longitud del borde de la superficie ocupada por la cuenca.
- *Subcuenca*: Conjunto de las subcuencas que contiene esta cuenca.

- *Ríos*: Conjunto de ríos que pasan por la cuenca.
- Estación. Nombre de la estación ubicada en la cuenca
- Quebradas. Conjunto de quebradas que pasan por la cuenca

❖ ***Subcuenca***: (Código, Id_Cuenca, Nombre, Ríos, Quebradas)

- *Código*: Número que identifica la Subcuenca.
- *Id Cuenca*: Identificador de la cuenca a la que pertenece.
- *Nombre*: Nombre de la Subcuenca.
- *Ríos*: Conjunto de ríos que pasan por la subcuenca.
- *Quebradas*: Conjunto de Quebradas que pasan por la subcuenca

❖ ***Río***: (nombre, altitud, temperatura, longitud, caudal, nacimiento, desembocadura)

- *Nombre*: Identificación común del río.
- *Altitud*: Altura con respecto al nivel del mar a la que se encuentra el río en cuestión.
- *Temperatura*: Grados Centígrados a los que se encuentra el río durante su travesía por la cuenca.
- *Longitud*: Extensión del río desde su nacimiento hasta su desembocadura.
- *Caudal*: Caudal que trae el río a la altura de la cuenca.
- *Nacimiento*: Sitio donde nace el río.
- *Desembocadura*: Sitio al que el río desemboca que puede ser otro río o el mar.
    - *Descripción*: Texto descriptivo de las cualidades del río, uso principal, etc.

❖ ***Quebrada***: (nombre, longitud, caudal, nacimiento, desembocadura, Uso, Usuarios)

- *Nombre*: Identificación común de la Quebrada
- *Longitud*: Extensión de la Quebrada desde su nacimiento hasta su desembocadura.
- *Caudal*: Caudal que trae el Quebrada a la altura de la cuenca.
- *Nacimiento*: Sitio donde nace la Quebrada.
- *Desembocadura*: Sitio al que la Quebrada desemboca.
    - *Uso*: Uso principal de el agua de la Quebrada

- *Usuarios*: Número de usuarios.

❖ ***Estación***: (nombre, longitud, caudal, nacimiento, desembocadura, Uso, Usuarios)

- *Nombre*: Identificación común de la Quebrada
- *Longitud*: Extensión de la Quebrada desde su nacimiento hasta su desembocadura.
- *Caudal*: Caudal que trae el Quebrada a la altura de la cuenca.
- *Nacimiento*: Sitio donde nace la Quebrada.
- *Desembocadura*: Sitio al que la Quebrada desemboca.
    - *Uso*: Uso principal de el agua de la Quebrada
    - *Usuarios*: Número de usuarios.

❖ ***Medio***: (Nombre, tipo, buffer, formato, tamaño, fecha)

- *Nombre*: Titulo del documento.
- *Buffer*: Espacio donde se almacena el documento.
- *Tipo*: Clase de medio que se almacena (vídeo, sonido, imagen).
- *Formato*: Forma en la que viene almacenado el documento, que lo asocia a algún método de visualización o apertura.
- *Tamaño*: Extensión en Kilobytes que ocupa el documento.
- *Fecha*: día mes y año en que se creó el documento.

❖ ***Vídeo***: (duración, resolución)

- Duración: Tiempo que se tarda el vídeo en ser exhibido a una velocidad normal desde el principio hasta el final.

❖ ***Sonido***: (duración).

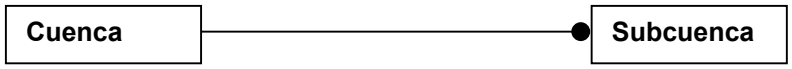
- Duración: Tiempo que se tarda el sonido en ser ejecutado a una velocidad normal desde el principio hasta el final.

❖ **Imagen**: (resolución)

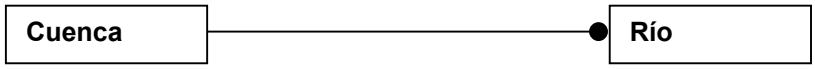
- Resolución: Calidad en pixeles de alto por pixeles de ancho a la que esta almacenada la imagen.

4.2.5 Descripción de asociaciones

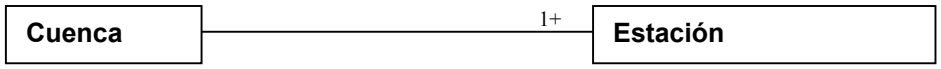
Una cuenca **contiene** una o más subcuencas



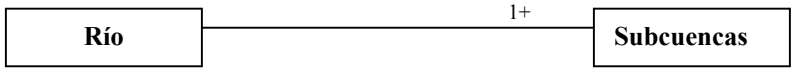
Una cuenca es **atravesada** por 1 o más Ríos



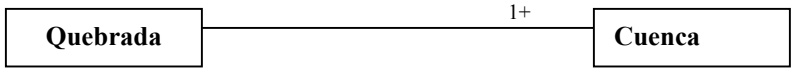
Una Cuenca **posee** 1 o más Estaciones



Un río **cruza** uno o más subcuencas.

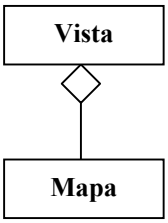


Un Quebrada **cruza** uno o más Cuencas.

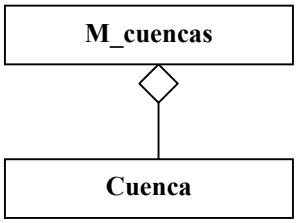


4.2.6 Agregaciones

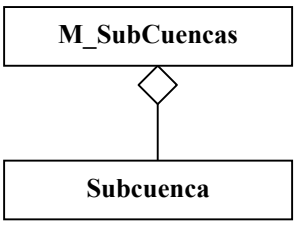
Agregación Vista



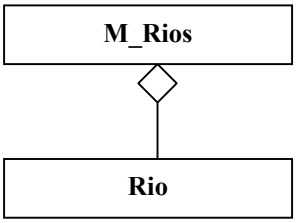
Agregación M_Cuencas ➔(Cuenca)



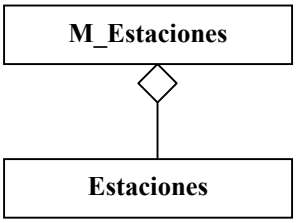
Agregación M_Subcuencas ➔(Subcuenca)



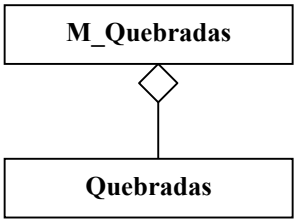
Agregación M_Rios ➔ (Rio)



Agregación M_Estaciones ➔ (Estaciones)

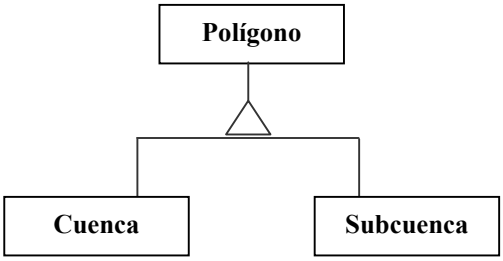


Agregación M_Quebradas➔(Quebradas)

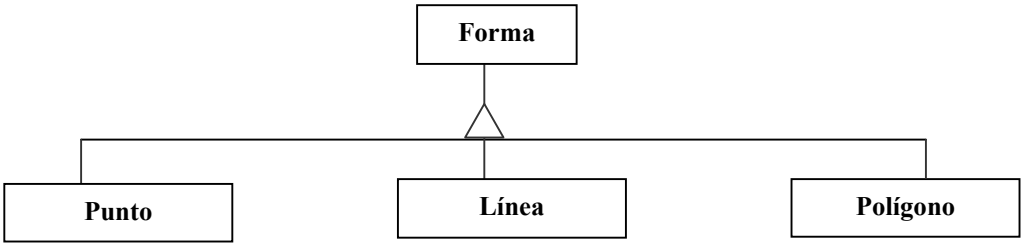


4.2.7 Herencias

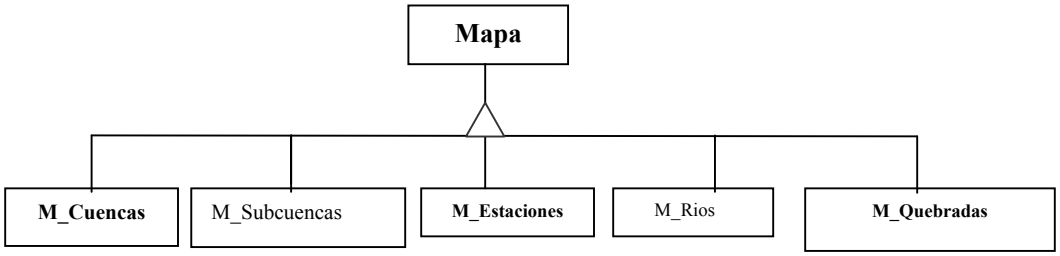
Herencia Polígono ➔ (Cuenca, Subcuenca)



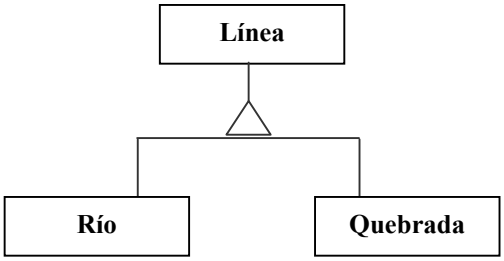
Herencia Forma ➔ (puntos, líneas, polígonos).



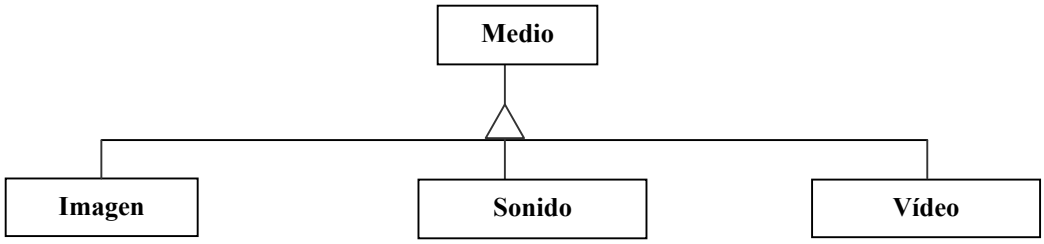
Herencia Mapa →(Mcuencas, Msubcuencas, Mrios, Mestaciones, Mquebradas)



Herencia Línea ➔(Río)



Herencia Medio ➔ (Imagen, sonido, video)



4.2.8 Definición de Operaciones

En general, las operaciones que se realizan sobre las clases corresponden a la operación fundamental de consulta sobre las instancias de cada clase.

Las siguientes operaciones despliegan gráficamente cada uno de los mapas que contienen los elementos que intervienen en el sistema. El despliegue se realiza usando la operación de cada clase de mapa.

- ♦ Mostrar_MCuencas (MapaCuencas)
- ♦ Mostrar_MSubcuencas (MapaSubcuencas)
- ♦ Mostrar_MEstaciones (Mapaestaciones)
- ♦ Mostrar_MRios(MapaRios)
- ♦ Mostrar_MQuebrada (MapaQuebradas)

Las siguientes operaciones amplían y disminuyen el tamaño de despliegue de la vista.

- ♦ Zoom_In(Punto inicial, punto final, factor)
- ♦ Zoom_Out(Punto inicial, punto final, factor)
- ♦ Desplazamiento(distanciaX, distanciaY)

Descripción: Corre la parte visible de la vista para poder ver la parte que se encuentra oculta. El desplazamiento es de las unidades de distancia que su parámetro indica.

♦ Consulta_alfanumerica (Mapa Activo, operando1, operador, operando2)

Descripción: Expresión del lenguaje de selección de consultas donde el operador puede ser aritmético, booleano, binario, etc. Y cuyo resultado es una selección de elementos sobre un mapa.

♦ Consulta_espacial (Mapa Activo, clase1, clase2, tipo_operación, criterio)

Descripción: Expresión que permite la interacción de varias instancias de dos clases diferentes. Donde criterio puede ser la distancia a la cual se desea identificar la operación.

Tipos de consultas espaciales: El sistema tendrá la habilidad para seleccionar los elementos sobre el mapa. Seleccionando elementos en esta forma es uno de los métodos de análisis espacial que usaremos para responder consultas que envuelven características de proximidad, adyacencia e inclusión.

- Mapa
  - Mostrar_Mapa(Lista_Elementos, FormaElementos)
  - Identificar_objeto(Posición)
  - Descripción: Esta función Identifica el objeto del mapa que ha sido seleccionado y muestra la información que este contiene.
  - Consulta_alfanumerica (operando1, operador, operando2)
  - Descripción: Este es el prototipo de una operación que realiza la consulta para cada mapa concreto de la aplicación.
- Forma
  - No hay operaciones necesarias.
- Punto
  - Dibujarse()
- Línea
  - Dibujarse()
- Poligono
  - Dibujarse()

- M_Cuencas
    - Mostrar_M_Cuencas(Lista_cuencas)
    - Descripción: Muestra el mapa de cuencas usando la operación de la clase mapa.
    - Buscar_Cuenca()
    - Descripción: Busca una cuenca.
    - Consulta_afanumerica (operando1, operador, operando2)
    - Descripción: Expresión del lenguaje de selección de consultas donde el operador puede ser aritmético, booleano, binario, etc. Los operadores pueden ser atributos de esta clase o valores y cuyo resultado es una selección de elementos sobre un mapa.
- M_Subcuenca
    - Mostrar_M_subcuencas(Lista_Subcuencas)
    - Descripción: Muestra el mapa de subcuencas usando la operación de la clase mapa.
    - Buscar_subcuenca()
    - Descripción: Busca una subcuenca usando como criterio cualquiera de sus atributos.
    - Consulta_afanuemrica (operando1, operador, operando2)
    - Descripción: Expresión del lenguaje de selección de consultas donde el operador puede ser aritmético, booleano, binario, etc. Los operadores pueden ser atributos de esta clase o valores y cuyo resultado es una selección de elementos sobre un mapa.
- M_Estaciones
    - Mostrar_M_Estaciones(Lista_Estaciones)
    - Descripción: Muestra el mapa de estaciones meteorológicas usando la operación de la clase mapa.
    - Buscar_Estaciónl()
    - Descripción: Busca una estación usando como criterio cualquiera de sus atributos.
    - Consulta_alfanumerica (operando1, operador, operando2)

- Descripción*:* Expresión del lenguaje de selección de consultas donde el operador puede ser aritmético, booleano, binario, etc. Y los operadores pueden ser atributos de esta clase o valores. Y cuyo resultado es una selección de elementos sobre un mapa.
- Longitud()
- Descripción: Halla la longitud
- Latitud()
- Descripción: Halla la latitud
- precipitación()
- Descripción: Halla la precipitación máxima
- Mostrar_Estación()
- Altura()
- Descripción: halla la altura.
- promedioanual()
- Descripción: Halla el promedio anual de pluviosidad.

- M_Ríos
  - Mostrar_M_Ríos(Lista_Ríos)
  - Descripción: Muestra el mapa de ríos usando la operación de la clase mapa.
  - Buscar_Río()
  - Descripción: Busca un río usando como criterio cualquiera de sus atributos.
  - Consulta_alfanumerica (operando1, operador, operando2)
  - Descripción*:* Expresión del lenguaje de selección de consultas donde el operador puede ser aritmético, booleano, binario, etc. Y los operadores pueden ser atributos de esta clase o valores. Y cuyo resultado es una selección de elementos sobre un mapa.
- M_Quebradas
  - Mostrar_M_Quebradas(Lista_Quebradas)

- Descripción: Muestra el mapa de quebradas usando la operación de la clase mapa.
- Buscar_Quebradas(nombre, cuenca)
- Descripción: Busca una quebrada usando como criterio cualquiera de sus atributos.
- Consulta_alfanumerica (operando1, operador, operando2)
- Descripción*:* Expresión del lenguaje de selección de consultas donde el operador puede ser aritmético, booleano, binario, etc. Y los operadores pueden ser atributos de esta clase o valores. Y cuyo resultado es una selección de elementos sobre un mapa.

- Medio
- Mostrar_tamaño()
- Vídeo
  - Mostrar_Video()
  - Rewind(Tiempo)
  - Descripción: Devuelve el video en el tiempo establecido.
  - Forward(Tiempo)
  - Descripción: Adelanta el video en el tiempo establecido.
- Sonido
  - Play()
  - Descripción: Ejecuta el sonido.
  - Rewind()
  - Descripción: Devuelve el sonido en el tiempo establecido.
  - Forward()
  - Descripción: Adelanta el sonido en el tiempo establecido.
  - volumen()
- Imagen
  - Mostrar()

- Zoom_In(Punto inicial, punto final)
- Zoom_Out()
- Desplazamiento(distancia)
- Descripción: Corre la parte visible de la imagen para poder ver la parte que se encuentra oculta. El desplazamiento es de las unidades de distancia que su parámetro indica.

4.2.9 Identificación y Descripción de Actores

En el proceso de definición del sistema (etapa de análisis de requerimientos) se identifico un solo actor: el Usuario del SIG; posteriormente, en el proceso de descripción detallada del sistema se identifica un segundo actor: el Mplayer2.exe (en este caso una aplicación externa a la que se desarrolla y que es utilizada para la reproducción de videos).

Actor	Descripción	Características
Usuario SIG	Persona interesada en la búsqueda de información sobre recursos hídricos y de erosión de las cuencas, del municipio de Yumbo (Valle del Cauca - Colombia)	Persona con manejo de los conceptos básicos de un SIG y con conocimiento en la utilización de software para el manejo de información geográfica.
Mplayer2.exe	Aplicación para la reproducción de videos.	

4.2.10 Descripción de Casos de Uso y Modelo de Casos de Uso.

A continuación se presenta la descripción de los casos de uso hallados en el proceso de definición del sistema y la relación entre los actores y los casos de uso.

4.2.10.1 Caso de Uso: Gestión de Mapas

Este caso de uso es iniciado por el Usuario del SIG, y en el se permite:

- **Cargar mapas.** Consiste en la selección (por parte del Usuario del SIG) de los distintos mapas o layers (un mapa esta constituido por uno o más layers, que se superponen, dependiendo de la información que requiera el usuario) que se desean ver en el SIG. Una vez el Usuario del SIG ha

seleccionado el mapa (con sus layers) que desea sea incluido, el sistema procede a cargar dicho mapa.

- **Descargar mapas:** Consiste en la selección (por parte del Usuario del SIG) de los distintos mapas (con sus layers) que no se desean ver en el SIG. Una vez el Usuario del SIG ha seleccionado el mapa (con sus layers) que desea sea eliminado de la vista, el sistema procede a descargar dicho mapa (lo elimina de la vista del SIG, no de la base de datos de mapas contenidos en el SIG).

4.2.10.2 Caso de Uso: Búsqueda de Información Geográfica

Este caso de uso es iniciado por el Usuario del SIG, y en el se permite la realización de consultas alfanuméricas y búsquedas espaciales (búsquedas geográficas).

- **Búsquedas Alfanuméricas**. Son búsquedas de la información alfanumérica contenida en el SIG. Para la realización de este tipo de búsquedas, el Usuario del SIG entra (digita) la(s) palabra(s) que se desean buscar y selecciona los mapas (con sus layers) sobre los cuales se desea hacer la búsqueda. El sistema realiza la búsqueda requerida y muestra la información encontrada (dicha información depende del tipo de layer sobre el cual de realiza la búsqueda)

- **Búsquedas Espaciales.** Son búsquedas de la información alfanumérica contenida en el SIG, pero a diferencia de la búsqueda alfanumérica, en este caso el Usuario del SIG selecciona de forma visual el elemento geográfico (río, quebrada, etc.) sobre el cual desea información. Una vez seleccionado dicho elemento, el sistema lanza la consulta SIG y muestra el resultado de la misma (al igual que en las búsquedas alfanuméricas, la información resultado de dicha consulta depende del elemento seleccionado).

4.2.10.3 Caso de Uso: Despliegue de Información Multimedial

Este caso de uso es iniciado por el Usuario del SIG, y en él se permite mostrar la información multimedial contenida en la base de datos del SIG, a saber: imágenes y videos.

Una vez realizada una consulta espacial, en la información que se muestra como resultado de la ejecución de la misma, se especifica si existe una imagen o video de la cuenca, quebrada, río, etc. Si esta existe y el Usuario del SIG selecciona la opción para ver la imagen (o video), el sistema procede a mostrar dicho recurso multimedia.

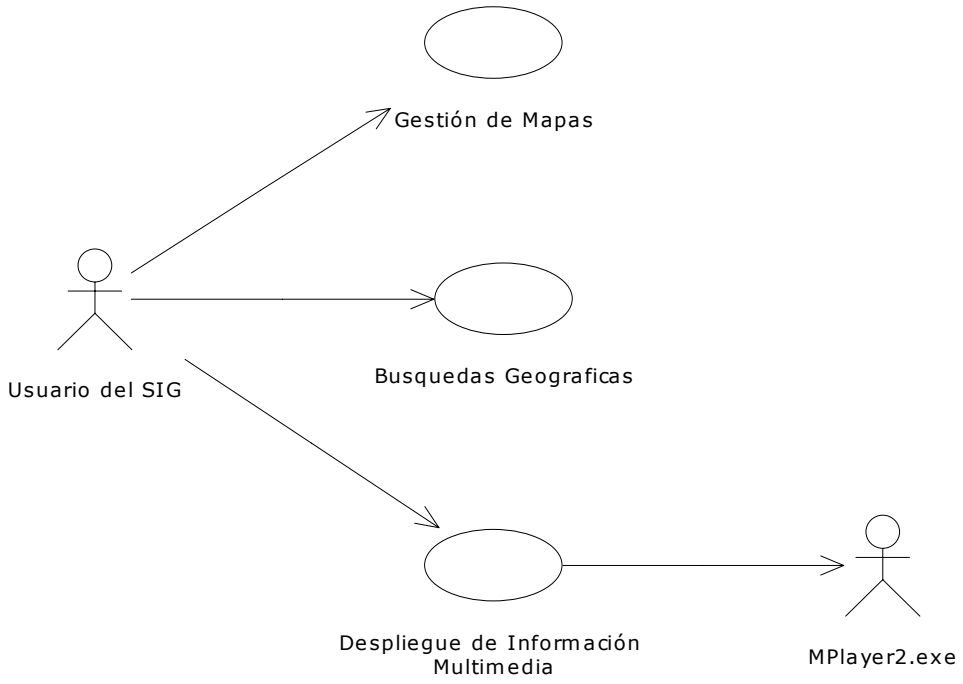


Fig. 4.4 Modelo de casos de uso. Esta figura muestra de forma gráfica, la relación existente entre los actores y los distintos casos de uso hallados en la descripción

4.2.11 Diagramas de Secuencia

Un paso importante en el proceso de desarrollo de software orientado por objetos, es el distribuir el comportamiento del software en componentes (clases). Esta distribución de comportamiento es ilustrada con ayuda de los diagramas de secuencia. A continuación se muestran los diagramas de secuencia para cada uno de los casos de uso identificados.

4.2.11.1 Diagramas de secuencia para el caso de uso: Gestión de Mapas

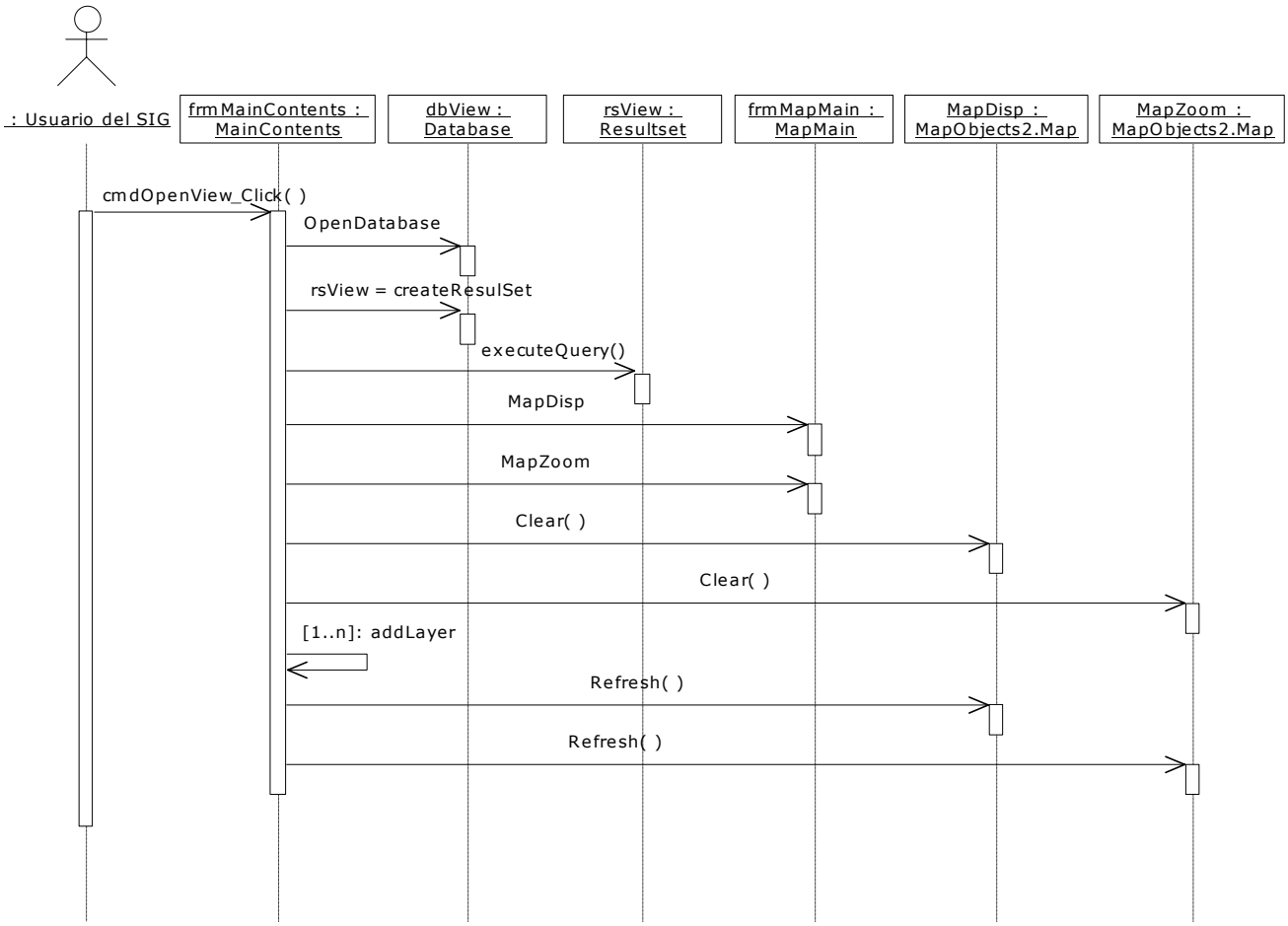


Fig. 4.5 Diagrama de secuencia para la tarea "Cargar mapas".

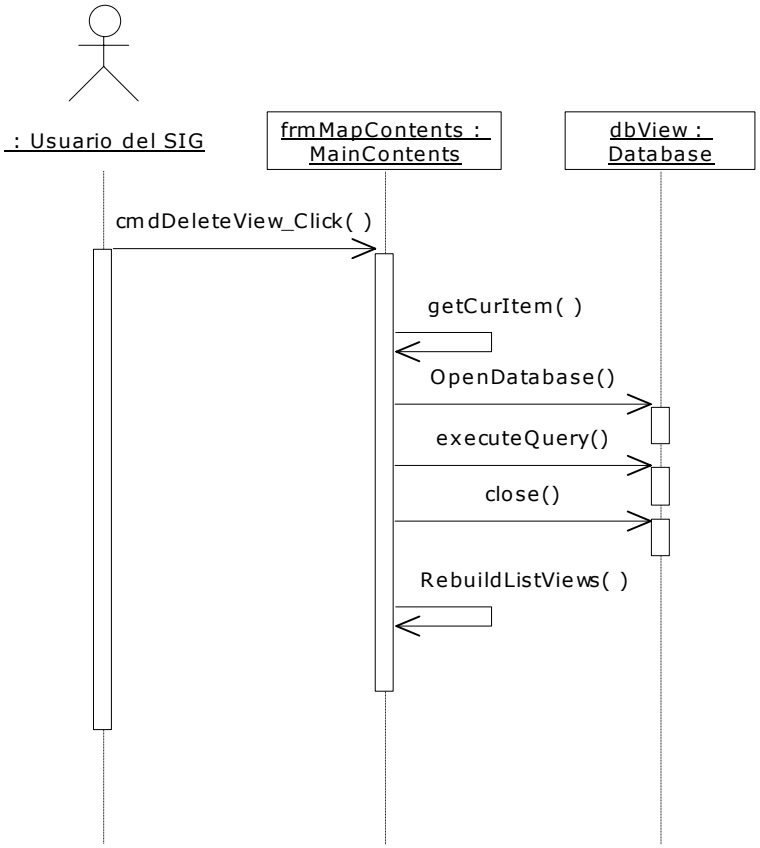


Fig. 4.6 Diagrama de secuencia para la tarea “Descargar mapas”.

4.2.11.2 Diagramas de secuencia para el caso de uso: Búsqueda de Información Geográfica

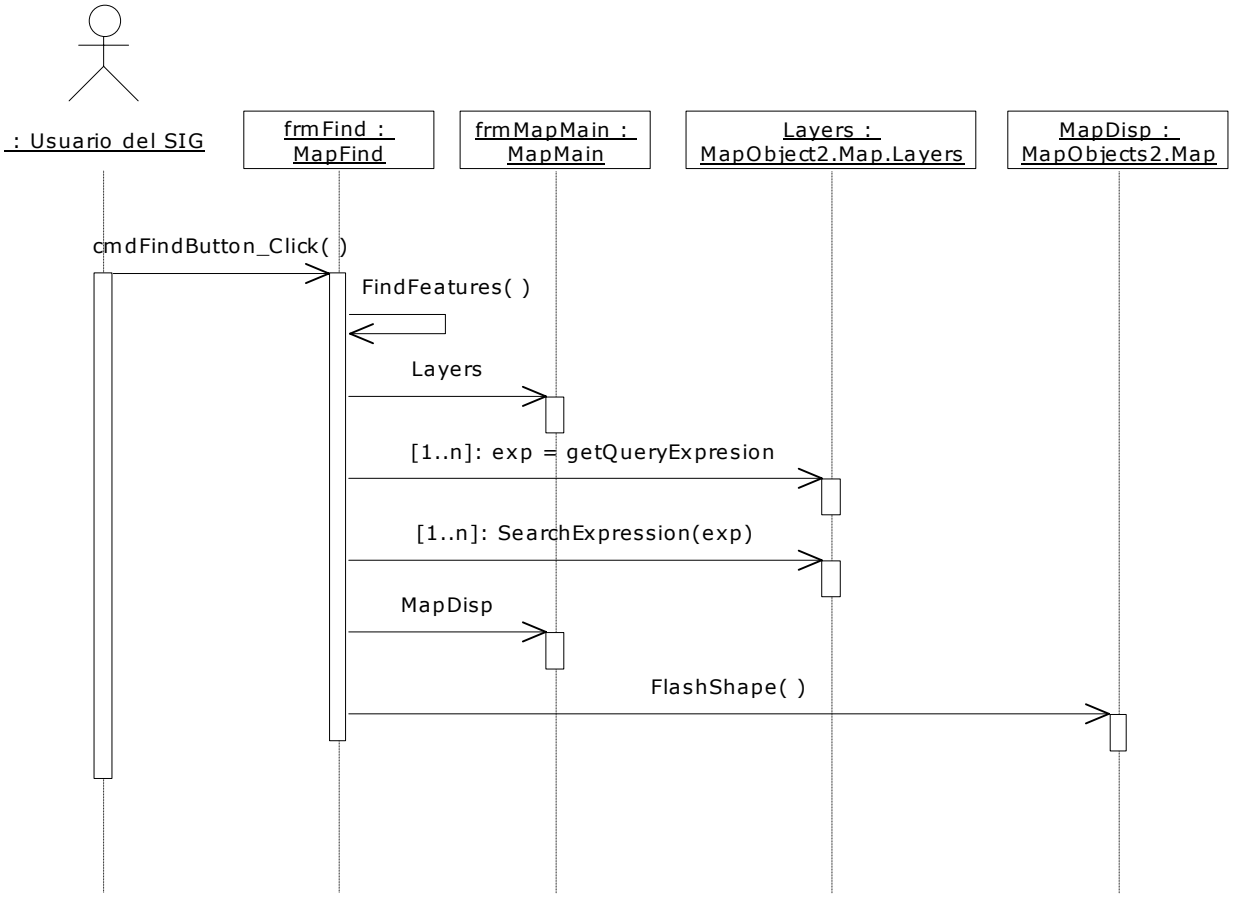


Fig. 4.7 Diagrama de secuencia para la tarea "Búsqueda Alfanumérica".

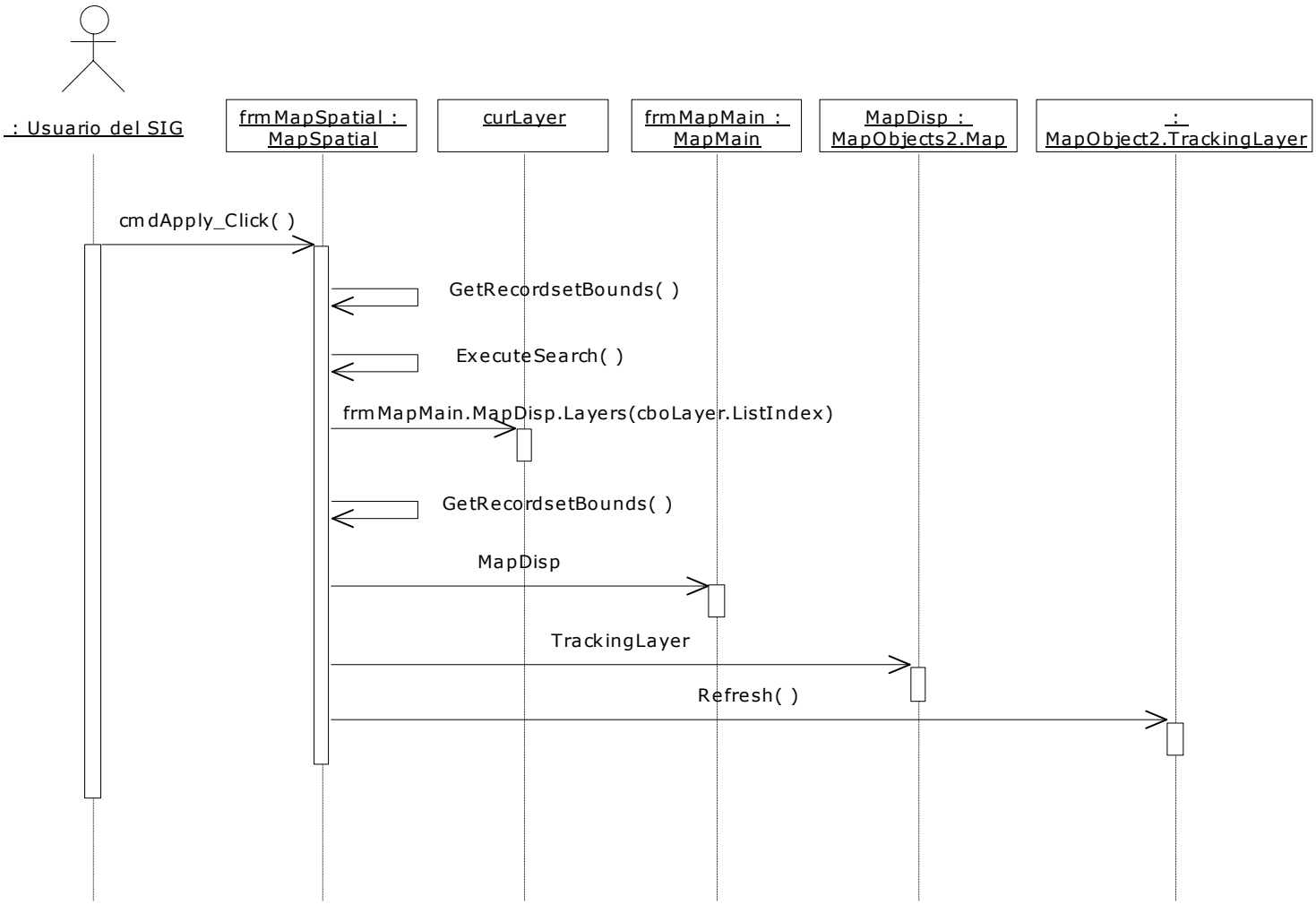


Fig. 4.8 Diagrama de secuencia para la tarea "Búsqueda Espacial".

4.2.11.3 Diagramas de secuencia para el caso de uso: Despliegue de Información Multimedial

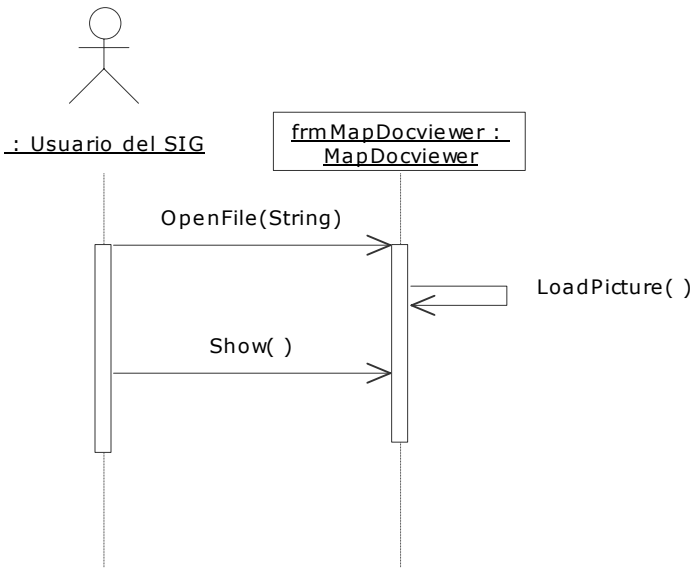


Fig. 4.9 Diagrama de secuencia para la tarea "Despliegue de Información Multimedial - Foto".

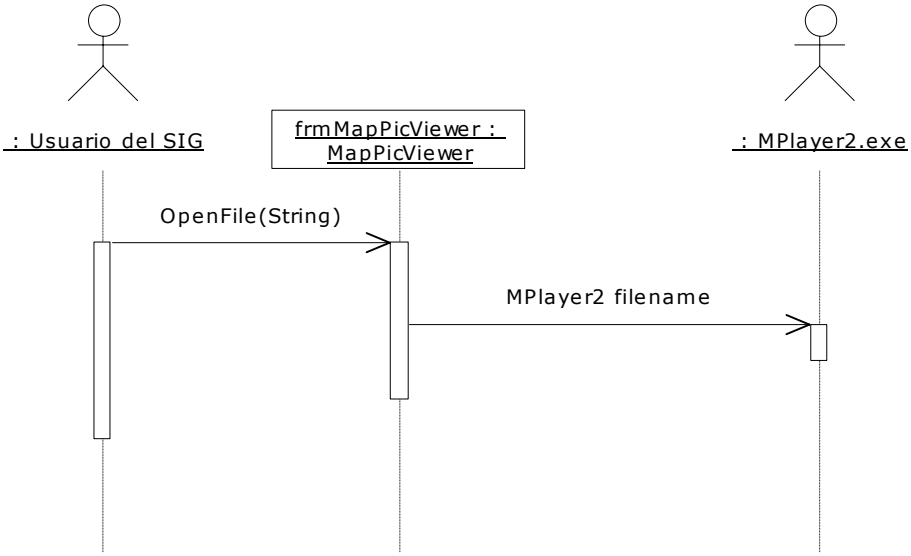
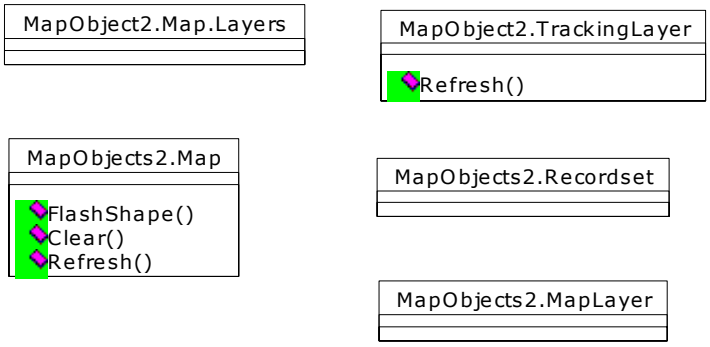


Fig. 4.10 Diagrama de secuencia para la tarea "Despliegue de Información Multimedial - Video".

4.2.12 Diagramas de Clases de MapObject

A continuación se presenta un diagrama de clases que muestra las clases utilizadas en la implementación del SIG y que hacen parte de la biblioteca de clase MapObject.



4.2.13 Diagramas de clases de la aplicación

El diagrama que se muestra a continuación, muestra las distintas clases que componen la

aplicación, y las relaciones existentes entre ellas. Más adelante se encuentra la descripción de cada una de las operaciones.

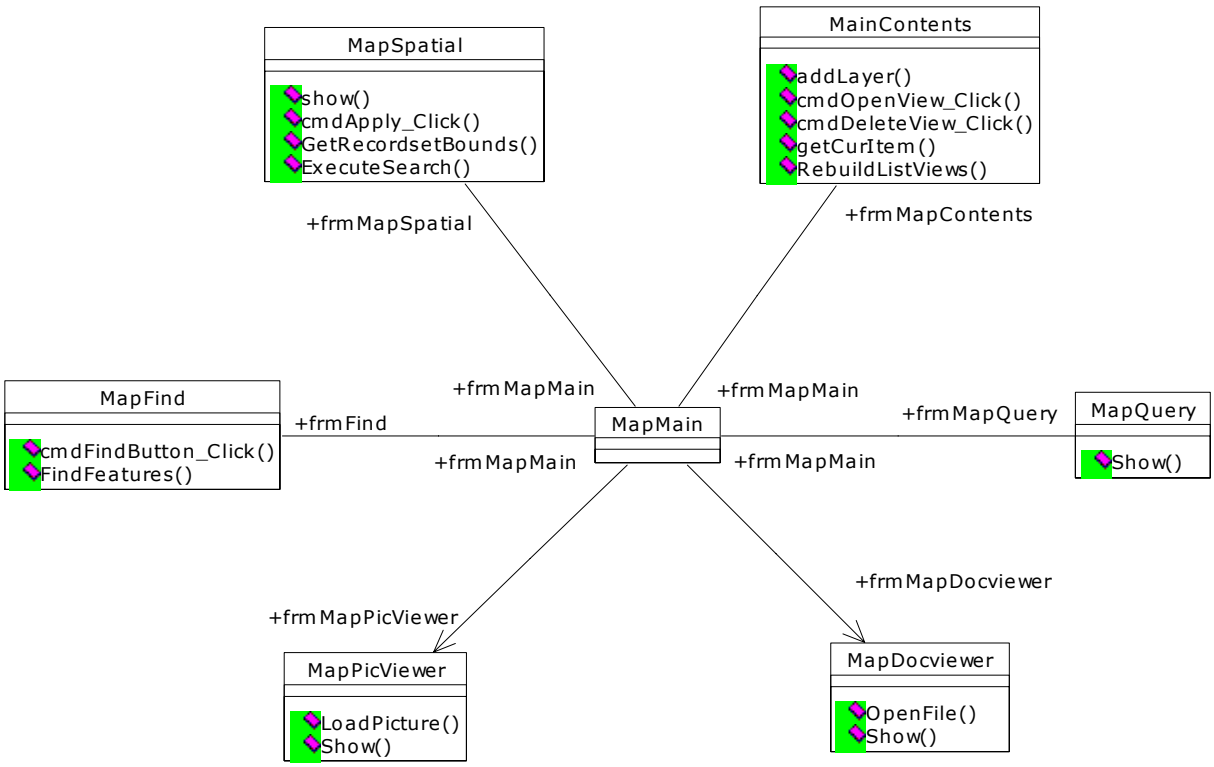


Fig. 4.11 Clases que componen la aplicación

4.2.14 Descripción de Clases y Operaciones

- **MapMain.** Ventana principal de la aplicación, contiene el área que muestra el mapa (compuesto de layers), y el area que permite seleccionar los layes que conforman el mapa. Además, contiene la barra de herramientas, la barra de menú y la barra de estado.

- **MapPicviewer.** Ventana que muestra las imágenes de los recursos hídricos manejados en el SIG.

- **LoadPicture(String name):** carga la imagen en la ventana. El parámetro “name” corresponde al nombre de la imagen que se desea mostrar.
- **Show():** Muestra la ventana.

- **MapDocviewer.** Ventana que muestra los videos de los recursos hídricos manejados en el SIG.
  - **LoadFile(String name):** carga el video en la ventana. El parámetro “name” corresponde al nombre del video que se desea mostrar.
  - **Show():** Muestra la ventana.

- **MainContents:** Área de trabajo que muestra la lista de capas (layers) mostrados en la ventana principal.
  - **addLayer():** permite adicionar un nuevo layer al mapa.
  - **cdmOpenView_Click():** método que permite responder a l evento de clic para seleccionar un layer a adicionar.
  - **cmdDeleteView_Click():**método que permite responder a l evento de clic para seleccionar un layer a remover.
  - **getCurItem():** método que permite conocer el layer seleccionado.
  - **RebuilListViews():_**método que reconstruye la lista de layers disponibles en la aplicación.

- **MapSpatial:** Área de trabajo que muestra y permite la manipulación de mapas.
  - **Show():** muestra el mapa cargado.

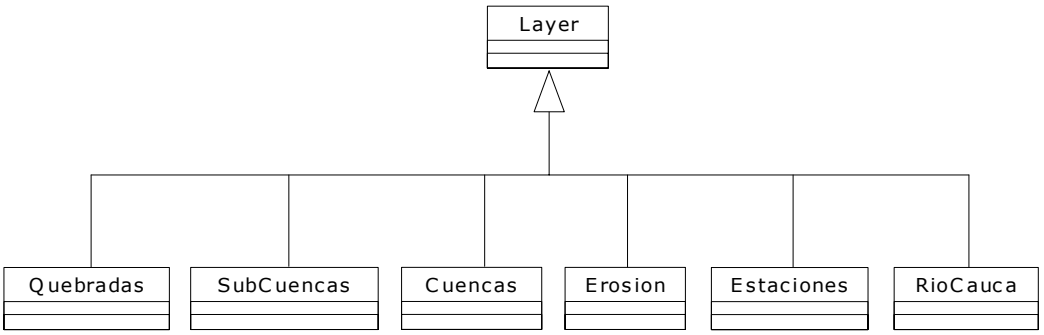
- **cmdApply_Click():** método que responde a las operaciones de clic sobre el mapa.
- **getRecordsetBounds():** método que retorna las coordenadas del área seleccionada para la realización de las búsquedas espaciales.
- **executeSearch(RecordsetBounds rsb):** método que realiza la búsqueda especial con base en las coordenadas devueltas por el método getRecordsetBounds()

- **MapFind:** Ventana utilizada para solicitar los datos de entrada para la realización de las consultas alfanuméricas.
  - **CmdFindButton_Click():** Método que reconoce el evento clic sobre el botón que lanza la ejecución de la consulta.
  - **FindFeatures():** Procesa la consulta con base en los parámetros de búsqueda digitados por el usuario.

- **MapQuery:** Ventana que muestra el resultado de las consultas alfanuméricas.
  - **Show():** método que muestra la ventana.

4.2.15 Esquema de mapas incorporados en el SIG

Los mapas dentro del SIG son manejados como capas (layers) los cuales conforman las vistas en el SIG. Estos archivos están en formato ShapeFile (shx y shp). A continuación se muestra un diagrama que representa los distintos capas (layers) incorporados en el prototipo del SIG para el manejo de los recursos hídricos, algunos de ellos los podrá cargar el usuario.



Capas (layers) incorporados en el SIG

4.2.16 Esquemas de las bases de datos

La información alfanumérica dentro del SIG es almacenada por ArcView en archivos de bases de datos en formato DBF. A continuación se muestra la estructura de cada una de las tablas definidas

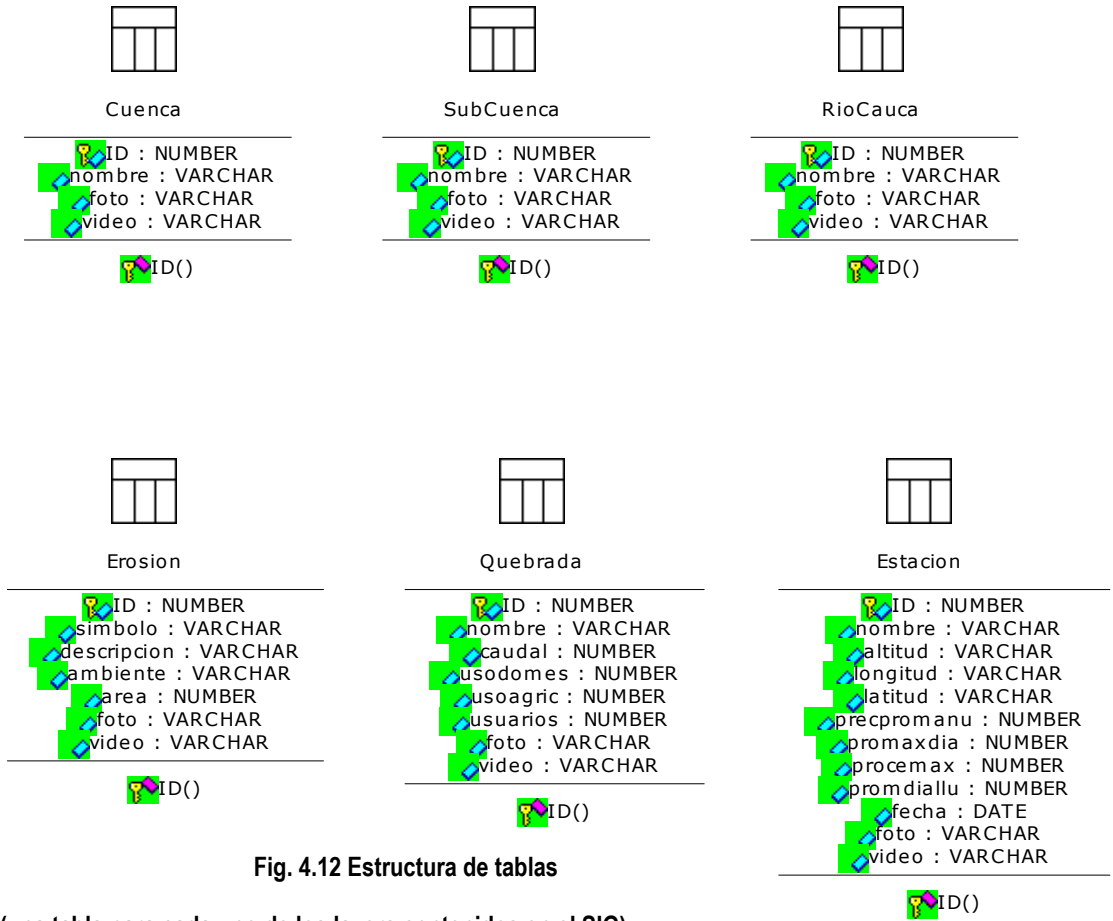


Fig. 4.12 Estructura de tablas

(una tabla para cada uno de los layers contenidos en el SIG).

(una tabla para cada uno de los layers contenidos en el SIG).

El diagrama que se muestra a continuación, presenta la forma como ArcView representa la información sobre los mapas a ser mostrados. Cada Mapa (View) es la composición de capas (Layer) cuya información de registra en archivos tipo shx (la tabla files, registra los arhivos shx). Estas dos tablas son creadas por ArcView en una base de datos llamada "Views"

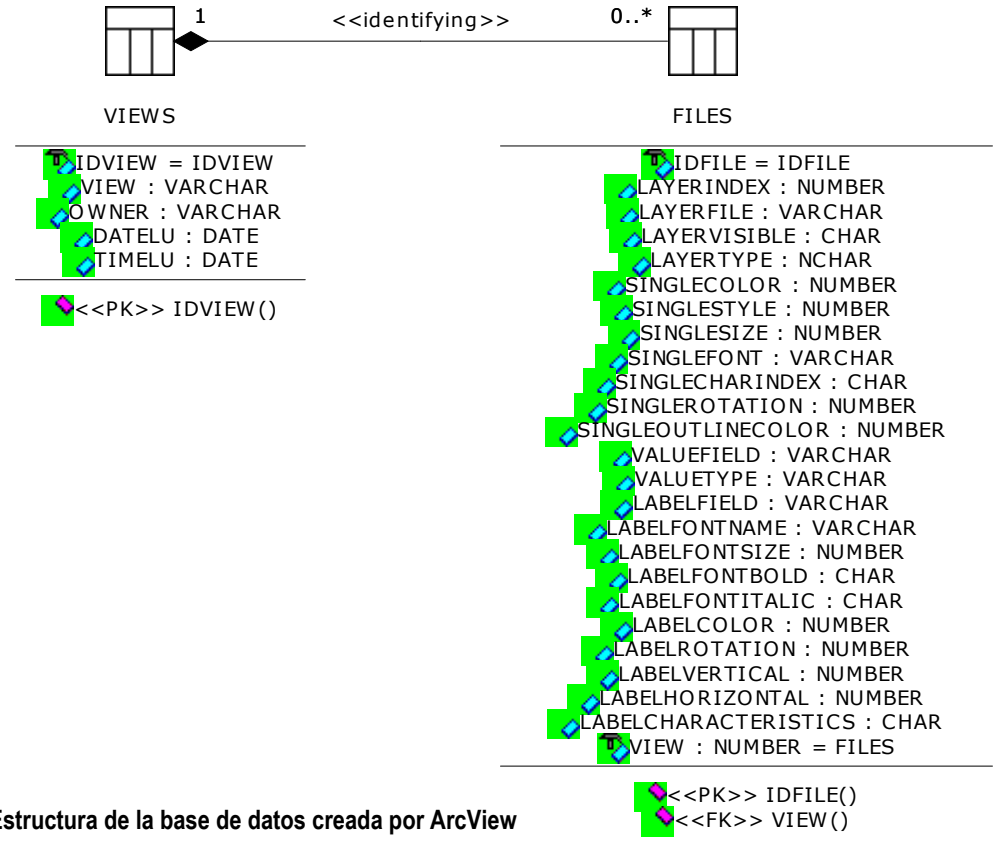


Fig. 4.13 Estructura de la base de datos creada por ArcView

A continuación se presenta el diagrama de flujo de trabajo para realizar una consulta espacial.

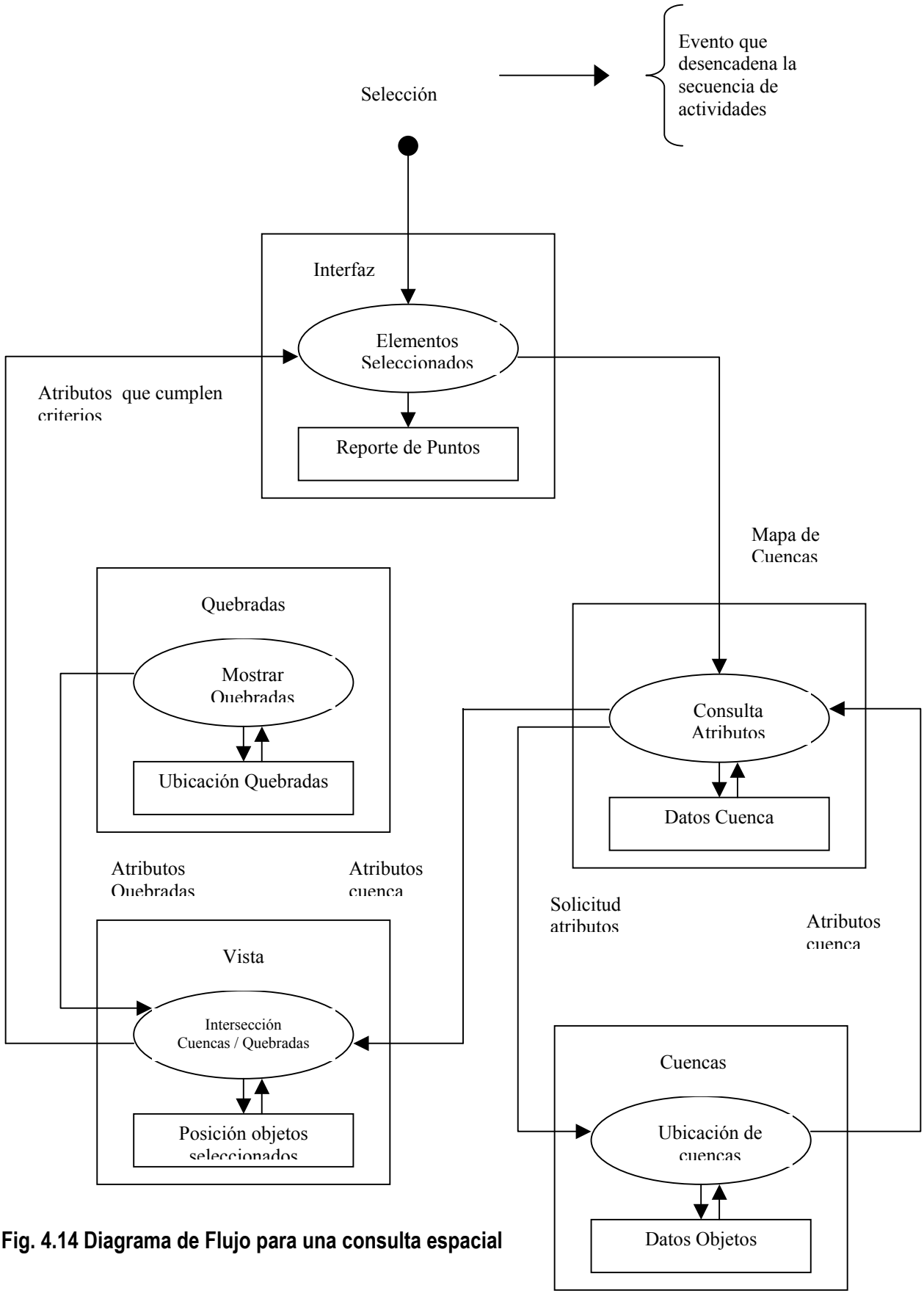


Fig. 4.14 Diagrama de Flujo para una consulta espacial

4.3. Implantación

El principal objetivo de esta etapa es implantar como producto un prototipo del software definido en la etapa de análisis de requerimientos y diseñado en la etapa de análisis y diseño. En este apartado se indicará el proceso que se llevó a cabo para desarrollar esta aplicación desde el análisis por medio del cual se preparan los mapas del municipio de Yumbo para ser publicados, hasta los procesos relacionados con la implantación, éstos últimos se explican en mayor detalle en el proceso correspondiente.

Para llevar a cabo esta tarea se realizaron los siguientes pasos:

- Conseguir en Planeación Municipal de Yumbo y la Corporación Autónoma Regional del Valle del Cauca – CVC, los mapas del municipio de Yumbo actualizados a la fecha de Junio de 1998. En este caso se emplearon los mapas de estaciones de monitoreo, quebradas, cuencas y subcuencas. Otra información fue adquirida a través del PAAL (Plan de Acción Ambiental Local).

- Hacer la conexión de la base de datos del sistema hidrográfico de Yumbo que se encuentra en Word con la información geográfica para convertirlos al formato shapefile que maneja MapObject.

- Definir la forma de mostrar los datos dentro de la interfaz, de tal forma que las operaciones básicas se realicen en una sola pantalla y para cada tipo de operación haya un menú diferente y una ventana en la cual se pueden

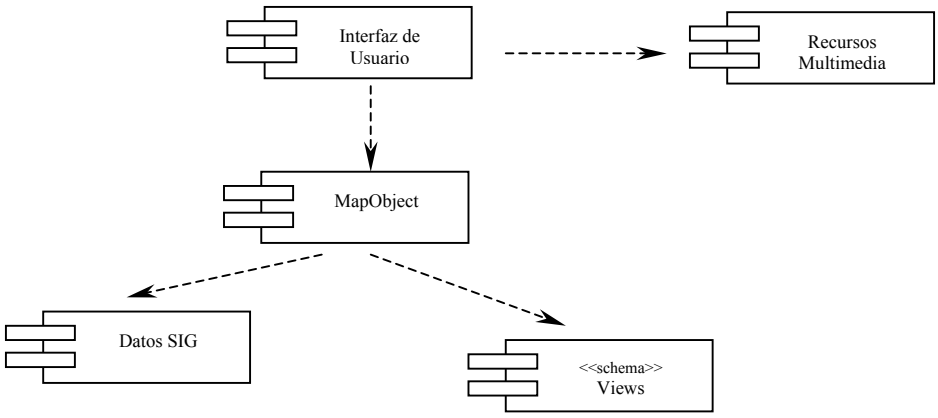


Fig. 4.15 Diagrama de componentes que muestra la estructura de la aplicación.

ejecutar operaciones especiales.

- Adicionar algunos atributos a la información hidrográfica que muestran datos multimedia, como fotos, videos y sonidos.

En el proceso de implementación del software, se hace necesaria la construcción de la interfaz de usuario en una herramienta de programación de alto nivel como lo es Visual Basic. La razón de dicha selección radica en la necesidad de hacer uso de la biblioteca de objetos conocida como MapObject la cual fue construida en este lenguaje de programación. A continuación de muestra fragmentos del código de la clase que permita la realización de las búsquedas espaciales en el sistema.

```
Private Sub cmdApply_Click()
	If txtDistance.Visible Then
		g_searchDistance = txtDistance.text
	Else
		Set g_searchSet = g_selectedFeatures
		Set g_searchBounds = GetRecordsetBounds(g_selectedFeatures)
		Set g_searchShape = Nothing
	End If
	ExecuteSearch
End Sub

Sub ExecuteSearch()
	If cboLayer.ListIndex = -1 Then Exit Sub
	Dim shapes As Object, curIndex As Integer
	Set shapes = Nothing
	If Not g_searchShape Is Nothing Then Set shapes = g_searchShape
	If Not g_searchSet Is Nothing Then Set shapes = g_searchSet
	If shapes Is Nothing Then Exit Sub

	'Reset the selection and execute the search
	Screen.MousePointer = vbHourglass
	Set g_selectedFeatures = Nothing
	Dim curLayer As MapObjects2.MapLayer
	Set curLayer = frmMapMain.MapDisp.Layers(cboLayer.ListIndex)
If StrComp(cboMethod.List(cboMethod.ListIndex), "La Forma está dentro de la distancia de búsqueda de la Característica") = 0

Then

		  Set g_selectedFeatures
		       = curLayer.SearchByDistance(shapes, g_searchDistance,"")
	Else
		'Execute the selected SearchByShape method on selected layer.
		Set g_selectedFeatures
			= curLayer.SearchShape(shapes, cboMethod.ListIndex, "")
	End If
	Set g_selectedBounds = GetRecordsetBounds(g_selectedFeatures)
	'Remove any existing search by layer selection in the combo box.
	curIndex = cboUsing.ListIndex
	If cboUsing.listCount = 5 Then cboUsing.RemoveItem 4
	cmdApply.Enabled = False
	'Add search by layer expression if there is a valid selected set, and the
	selected set is not from an SDE layer.
	If Not g_selectedFeatures Is Nothing Then
		If g_selectedFeatures.Count > 0 Then
			cboUsing.AddItem "Formas de la cobertura " & curLayer.Name
			If curIndex = 4 Then
				cboUsing.ListIndex = 4
				cmdApply.Enabled = True
			End If
		End If
	Else
		If curIndex = 4 Then cboUsing.ListIndex = 0
	End If
	frmMapMain.MapDisp.TrackingLayer.Refresh True
```

Fragmento de código de la clase MapSpatial correspondiente a la realización de una búsqueda espacial. Este fragmento de código corresponde con el diagrama de secuencia titulado “diagrama de secuencia para la tarea – Búsqueda Espacial”. Las líneas que aparecen en negrilla corresponden a la lógica principal de la aplicación

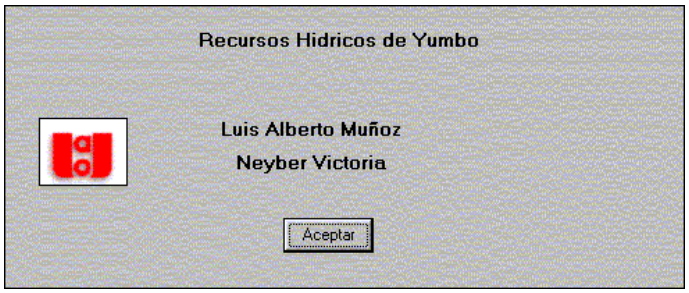
la cual es ilustrada en el diagrama de interacción en mención; las líneas adicionales corresponden al código de la interfaz de usuario (cambio del cursor, etc.)

4.3.1. Pantallas del sistema

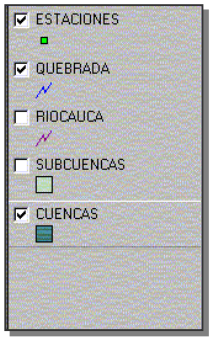
A continuación se describen las pantallas de manejo del sistema y se da una inducción acerca de las funciones que maneja

4.3.1.1. Ventana de presentación

En esta ventana se presenta el nombre del proyecto y los autores del proyecto.



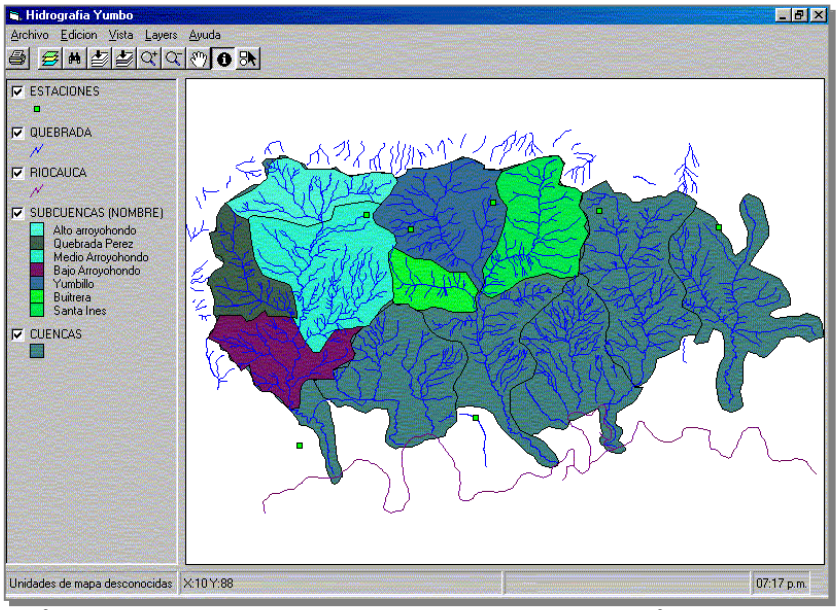
4.3.1.2. Áreas de trabajo



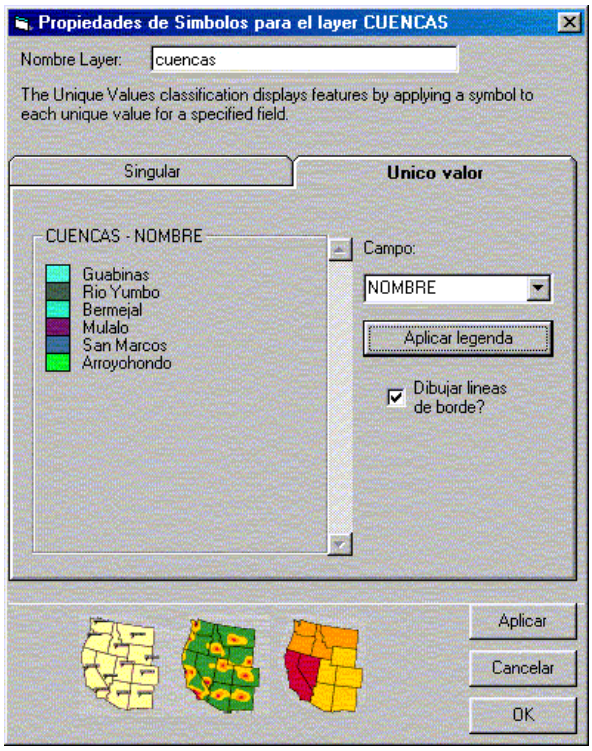
Leyenda. En esta área aparecen las etiquetas de los mapas que se despliegan en el area de vista de mapas.

Cargue y descargue de mapas: Esta función se utiliza chequeando o deschequeando las cajas que aparecen en la parte izquierda de la etiqueta de los mapas.

Despliegue de clasificaciones: En esta area se muestra la clasificación de los mapas de acuerdo al campo que el usuario seleccione. Esta clasificación se representa en el mapa de acuerdo a la leyenda escogida. En este caso se hace una clasificación de las subcuencas de acuerdo a su nombre.

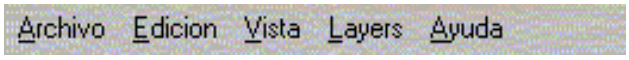


Estas clasificaciones se pueden realizar haciendo doble click sobre el mapa a clasificar, aparece una ventana en la cual se pueden definir colores de representación, campo de clasificación y forma de relleno.



Barra de herramientas. En esta area se encuentran las herramientas disponibles para ejecutar por el programa. Se identifican operaciones como: Impresión, Administración de mapas, Buscar, Acercamiento al mapa completo, Acercamiento al mapa activo, Acercamiento, Alejamiento, Desplazamiento, Identificación de elementos y Consultas espaciales.

Barra de Menu. Son las mismas operaciones que se pueden ejecutar por la barra de herramientas.



Barra de estado. En esta area se muestran las unidades en las cuales se encuentra el mapa y la posición X,Y del cursor del mouse sobre el mapa.

Ventana principal. Esta pantalla muestra las capas de información de estaciones, quebradas, cuencas y subcuencas cargadas en la leyenda en la parte izquierda, con la opción de cargar las capas de erosión, uso actual, uso potencial (que por efectos de visualización no aparecen cargadas en el mapa) también puede descargar alguno de ellos con solo marcar la caja de chequeo en la parte izquierda. En la parte superior se muestra la barra de herramientas con la cual se pueden ejecutar las operaciones básicas con que cuenta el SIG. Además de alguna información del proyecto en el menú Ayuda.

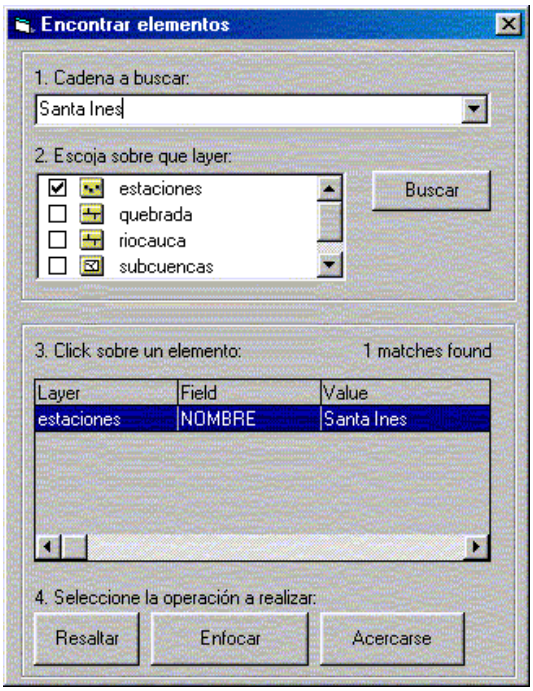
4.3.1.3.1. Propiedades del mapa

En esta ventana aparecen las opciones de:

- Adicionar layer:
- Permite adicionar mapas adicionales a los cargados originalmente.
- Permite definir el estilo del borde, las unidades y el fondo del mapa.

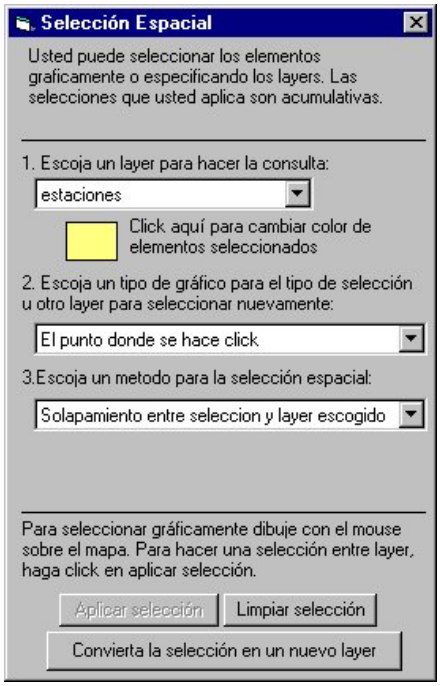
4.3.1.3.2. Encontrar elementos

Esta ventana se utiliza para hacer busquedas alfanumericas sobre los mapas, el resultado de las consultas se muestra en un panel de resultados y se puede seleccionar, enfocar y acercar el resultado de la busqueda en el mapa haciendo click sobre el botón correspondiente, esta búsqueda se puede filtrar chequeando el mapa sobre el cual se quiere realizar la consulta especificada.

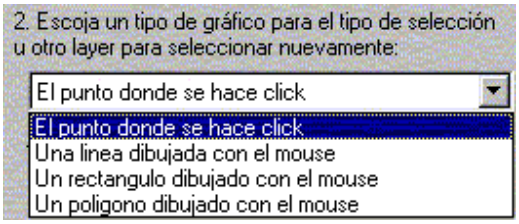


4.3.1.3.3. Consultas espaciales

Esta ventana permite hacer selecciones espaciales en el mapa con base en diferentes criterios de busqueda. Esta busqueda se representa en el mapa despues de ejecutada y muestra los elementos encontrados en una ventana adicional.



Esta selección espacial interactua con el usuario por medio de cuatro tipos de operaciones: Por el punto donde haga click, o dibuje una linea, rectangulo o poligono.

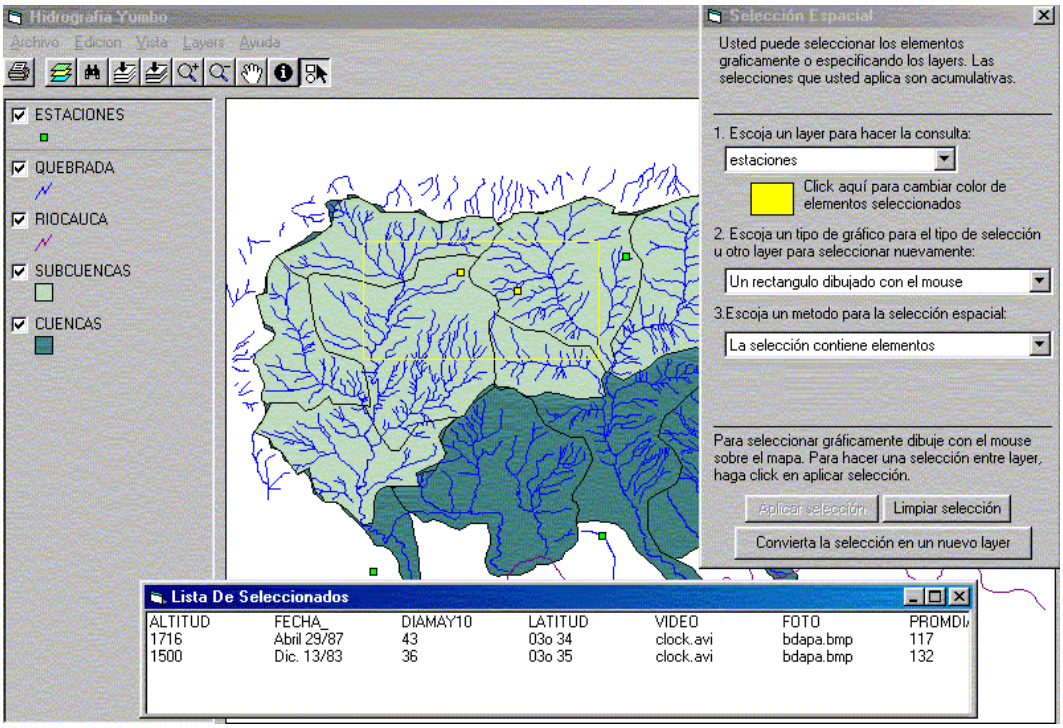


A partir de ahí se definen los tipos de busqueda a ejecutar.

Los siguientes son los posibles metodos de selección que se permiten. Estas operaciones estan basadas en metodos de operaciones de conjuntos que solo se permiten ejecutar sobre información geográfica con base en la modelación realizada.

La siguiente imagen es el resultado de una consulta espacial basado en una selección de un rectángulo

dibujado por el usuario. El mapa de estaciones es el mapa activo.

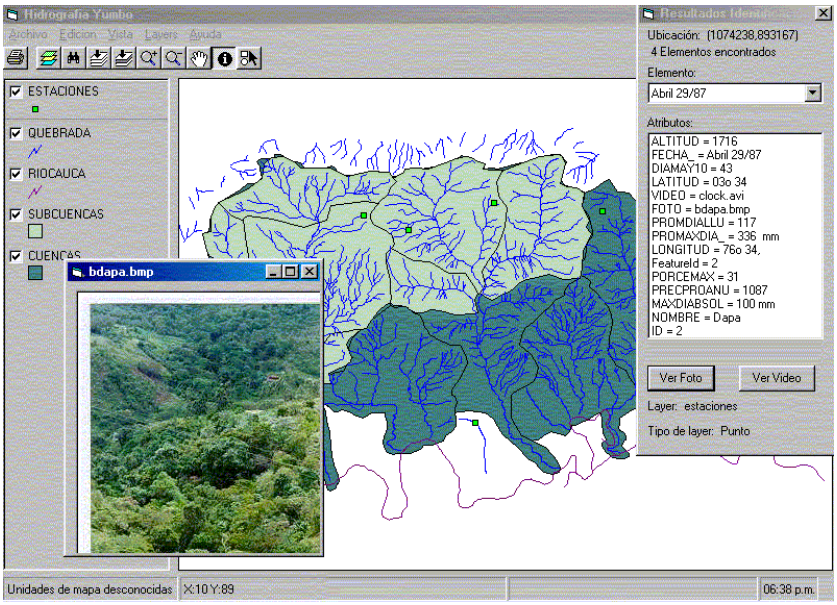


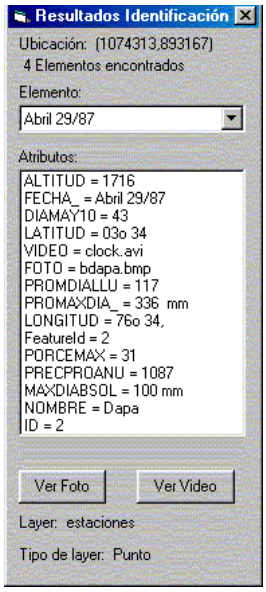
4.3.1.3.4. Identificación de elementos

Esta herramienta se utiliza para el despliegue de la información alfanumérica de los elementos del mapa. Funciona haciendo click sobre el elemento a identificar.

4.3.1.3.5. Mostrar Foto y Video

Con base en la identificación del elemento sobre el mapa se puede mostrar la imagen y el video correspondiente al objeto seleccionado haciendo click sobre el botón correspondiente.





4.3.2. Criterios para la elección de la Herramienta de Desarrollo

El trabajo que construye un “Sistema de Información Geográfico de los Recursos Hídricos del Municipio de Yumbo", se compone fundamentalmente de dos partes, unidas entre sí:

- Manipulación de Sistemas de Información Geográfica
- Un manejador de bases de datos o lenguaje de programación que permita consultas eficientes sobre información geográfica.

De acuerdo con lo anterior se necesitan dos herramientas para su desarrollo, un Sistema de Información Geográfico que provea toda la fortaleza de manipular la información geográfica y un lenguaje de programación que permita tecnología abierta incluyendo la potencialidad de un SIG.

4.3.2.1. Características de las herramientas.

4.3.2.1.1. SIG

- El SIG debe ofrecer la funcionalidad básica, como proyección de mapas, manipulación y manejo de datos, despliegue de imágenes y manejo de datos tabulares.

- El acceso a bases de datos SQL debe ser completo para el SIG.

- El manejo de datos geográficos – Almacenamiento efectivo y manejo de grandes bases de datos multiusuarias que estén distribuidas dentro de una organización.

- Debe tener una interfaz gráfica productiva dado por un ambiente manejado por menús que permiten ejecutar todas las herramientas productivas y funciones SIG.

- Deben estar en capacidad para trabajar como generador de reportes, compatible con otras herramientas, establecer seguridad, gran capacidad de almacenamiento de datos, número y tamaño máximo de registros, extendibilidad, distribución de datos, integridad, validaciones, rendimiento.

- El desarrollo de sistemas y personalización debe proveer un ambiente de desarrollo que permita a los usuarios la creación de menús, automatizar procesos.

4.3.2.1.2. Lenguaje de desarrollo

- La tecnología empleada para el desarrollo del proyecto debe ser de arquitectura abierta, permitiendo importar objetos externos.

4.3.2.2. Prioridades para el desarrollo

Para el mejor desempeño en el manejo del lenguaje, los desarrolladores deben tener una experiencia en la programación en el lenguaje escogido.

Luego de analizar estas características se determinó estudiar e investigar específicamente dos herramientas que cumplían con varios criterios:

- ArcView y MapObjects como SIG

- Visual Basic y Java como lenguajes de desarrollo

Nombre	Tipo	Equipo	# Registros	Pseudo lenguaje	SQL	Reportes
ArcView	Relacional	Pentium II 300 MHz	10.000	Si	Si	Si

MapObjects	Relacional	Pentium II 300 MHz	10.000	Si	Si	Si
	Seguridad	**Flexibilidad**	**Compatibilidad**	**Manejo Multiusuario**		**Precio**
	Buena	Buena	Buena	Si		500US

Originalmente se había tomado ArcView como la plataforma a utilizar, dado que fue la primera herramienta que estuvo disponible para publicar mapas y además proveía las herramientas básicas del SIG en una interfaz gráfica, pero surgió la posibilidad de trabajar con MapObjects que no daba las bondades de proveer las herramientas automáticamente, pero la implementación de estas se facilitaba con un amplio manejo del lenguaje de programación y librerías provistas por MapObjects y en pruebas de velocidad da respuesta dio mejores resultados.

Por otra parte, para escoger la herramienta de desarrollo se realizó la siguiente comparación.

ArcView: "Es un completo sistema de acceso, despliegue, consulta y análisis de datos organizados. Conjuga las capacidades de las herramientas tradicionales de análisis como las hojas electrónicas de cálculo o las gráficas de negocios con mapas, para obtener un sistema integrado de análisis. Puede ser usado para un proyecto independiente o para una organización completa. Arcview integra todas las herramientas posibles de administración y visualización de datos, encadenando información de diferentes fuentes como son formatos de bases de datos con datos cartográficos en formatos vectorizados y raster"[44]

MapObjects: MapObjects es un conjunto de componentes de software de cartografía que permiten, al programador, agregar mapas a su aplicación. [45]

MapObjects incluye un control OLE (OCX) llamado el Map Control y un conjunto de más de treinta objetos de automatización OLE. Este es para uso en industrias de entorno de programación estándar tales como Visual Basic 4.0+, Delphi 2.0, PowerBuilder, Microsoft Access y otros. No es para usuario final. El desarrollo de aplicaciones usando MapObjects brinda todas las ventajas de un entorno de programación orientado a objetos, trabajando íntegramente en tecnología Windows de 32 bits. Se pueden construir aplicaciones basadas sobre MapObjects, y desarrollar esos programas para usuario final.

MapObjects posibilita, acceder a diversos formatos geográficos (shapefiles, ARC/INFO, SDE), utilizar

una amplia gama de imágenes, realizar mapeo temático, acceder a bases de datos relacionales (RDBMS), georeferenciar eventos por direcciones y desplegar eventos en tiempo real (GPS)

Al final, se escogió MapObjects debido a diferentes factores, como son:

- El lenguaje de desarrollo que más se conocía era Visual Basic, además
- la arquitectura de hardware más flexible era la de Windows.

En las tablas siguientes se evalúan las características principales de cada software:

Nombre	Tipo	Ultima Versión	Equipo	Software Preinstalado	Características Claves
Arc/View	SIG	3.1	Procesador Pentium Ram 32 DD libre 60 mb Monitor SVGA	Windows 95 Windows NT UNIX	Manejo de mapas: presentación en diversos colores Usa diferentes tipos de clasificaciones como igual área, igual intervalo, tamaño, etc. Integra muchos tipos de mapas. Incorpora a los mapas símbolo prediseñados. Visualiza mapas usando proyecciones. Importa y exporta en formatos estándar. Otras características: Interfaz de fácil uso. Actualiza gráficos cuando se actualiza tablas. Integra gráficos estadísticos, mapas, tablas y otro tipo de gráficas. Dinámico sistema de actualización. Geocodificación. Ambiente Robusto de Edición. Análisis y modelamiento Espacial. Amplios criterios de selección para mapas. Realiza Join entre tablas.

Nombre	Tipo	Ultima Versión	Equipo	Software Preinstalado	Características Claves
MapObjects	SIG	2.0	Procesador Pentium II 300 Ram 64 DD libre 30 Mb	Windows 95 Windows NT Visual Basic (o cualquier lenguaje que acepte insertar objetos COM)	Manejo de Mapas Dibujar elementos geográficos tales como puntos, líneas, círculos y polígonos. Dibujar texto descriptivo. Seleccionar elementos a lo largo de líneas, contenido de selecciones, áreas, polígonos y círculos. Potentes consultas con expresiones SQL. Fácil compatibilidad Análisis espacial. Integra las funciones de un SIG a cualquier aplicación de base de datos. Aprovecha toda la potencialidad de los lenguajes con respecto a la conectividad a bases de datos externas.

	SQL	Reporte Gráficos	Pseudo lenguaje	Idioma	Compatible con formatos	Modo operación	Precio
ArcView	Si	Si	Avenue	Inglés	ASCII, INFO, DLL, RPC,DDE, ArcCad, AutoCad, Atlas, MapInfo, TIFF, JPEG, EPS, BMP, LAN, BIP, RLC, Image, LandSat, SPOT, SunRaster, CGM, WMF, GIF, PICT, y con BD:DB2 ASSCII, Ingres y cualquier Bd con ODBC/SQL	Monousuario y red local	ArcView 1842 US

	SQL	Reporte Gráficos	Pseudo Lenguaje	Idioma	Compatible con formatos	Modo operación	Precio
MapObject	Si	Si	Visual Basic Visual C++ Visual Java Power Builder Delphi	Inglés	ARC/INFO, TIFF, GIF, PCX, HRF, dBASE y cualquier BD compatible OBDC 2.0.	Monousuario y red Local	MapObject Developer Kit 4500 US

4.4. Análisis comparativo de Productos GIS

Se analizarán los productos más relevantes dentro del mercado: ArcView, MapInfo y MicroStation GeoGraphis.

4.4.1. ArcView

El ArcView, en su versión 3.0, no cabe duda que se trata de un producto GIS Desktop con funciones de análisis topológico, edición de mapas, gráficos estadísticos y con posibilidad de adquirir módulos adicionales que le proporcionan una potencia que solo se puede encontrar en los GIS de alto valor, de ahí que sea el producto GIS de mayor usos.

Organiza la información en capas de datos, que son conjuntos de elementos geográficos que representan información del mundo real. Las clasificaciones pueden ser organizadas de acuerdo con diferentes factores: cartografía básica, vías de comunicación, clases de suelos, cobertura vegetal, divisiones administrativas, etc. Esta información puede ser representada gracias a que dispone de siete categorías divididas en cerca de un centenar de tipos diferentes de proyecciones.

La ventana de proyecto es la que se utiliza para organizar y gestionar el trabajo de una forma eficiente y organizada, a partir de ella se pueden abrir todos los componentes del proyecto como vistas, tablas, mapas, gráficos y scripts.

Para visualizar la información gráfica se abre una vista y los temas que la componen, un tema representa elementos de determinado tipo: puntos, como pueden ser las ciudades; líneas, por ejemplo las carreteras o los ríos o bien polígonos, como los tipos de terreno, las divisiones geográficas. La información asociada a los elementos de un tema la podemos visualizar seleccionando en la ventana de proyecto los componentes de la tabla, en donde aparecerá una lista completa de todas las tablas asociadas a los temas. Así mismo, se realiza la creación de un mapa o un gráfico estadístico.

Pueden realizarse personalizaciones gracias que dispone del lenguaje Avenue, orientado a objetos, lo cual permite automatizar determinadas tareas.

El editor de texto, es una herramienta que proporciona funciones para la generación de mapas temáticos, donde se define toda la utilería para su elaboración: colores, rangos de valores, personalización de la simbología, símbolos de gráficas estadísticas y funciones estadísticas, entre otras posibilidades.

Para las consultas de los datos se dispone de sentencias SQL, con asistentes para su creación. También posee facilidades para la realización de consultas gráficas: la opción selección por tema proporciona una forma eficaz y sencilla de análisis topológico. Permite las operaciones del tipo: "elemento contenido en", "intersección de", "a una distancia de", "completamente contenido en", etc. Las consultas son acumulables de forma que podemos ir haciendo selecciones sucesivas entre temas hasta obtener el resultado deseado.

Permite dibujar los elementos fundamentales y su agregación al tema que se esté editando. Se dispone de posibilidades de edición de nodos y vértices tanto desde ratón como desde tableta digitalizadora, así como herramientas para partir y combinar elementos, intersección, unión, extracción, etc. Para poder editar elementos de un tema éste debe ser un ShapeFile, o un formato compatible de ESRI.

No dispone de creación de áreas de influencia, pero se pueden realizar selecciones del tipo elemento a distancia determinada. Sin embargo, no permite obtener los elementos que cumplan la condición definida. Para un tratamiento avanzado de zonas de influencia se debe adquirir como herramienta el módulo Network Analyst.

ArcView puede soportar la geocodificación de datos de tablas externas que contengan direcciones de calles en diversos formatos; definido el tipo de geocodificación, en las propiedades del tema para el cual se va a enlazar la tabla de datos externa, luego se pueden realizar todo tipo de ajustes.

La visualización de imágenes raster, requiere el registro de la imagen para que se pueda visualizar en las coordenadas correctas y que los temas se superpongan correctamente con la imagen. La información geográfica viene normalmente almacenada junto con la imagen en la cabecera del archivo en los siguientes formatos, ERDAS, IMAGINE, BSQ, BIL, BIP, GeoTIFF y grids. Otros, sin embargo, almacenan esta información en archivos ASCII separados, generalmente con el nombre World. La impresión de las vistas, mediante la creación de mapas, se ve facilitada por la existencia de plantillas predefinidas que, automáticamente, presentan un mapa de la vista actual, donde también se puede modificar cualquier elemento.

4.4.2. MapInfo Professional

El Mapinfo, en su última versión presenta novedades importantes que mejoran la forma de trabajo, con la introducción de herramientas adicionales. Como por ejemplo el cursor informativo sensible al elemento por el que pasa, que muestra el valor del campo que se haya seleccionado previamente, o el nuevo asistente de informes que permite crear listados tabulares usando los datos existentes en tablas tipo MapInfo. Además, en esta versión se puede guardar una consulta creada previamente con las herramientas SQL y el límite de nodos para regiones y se han incrementado las polilíneas

notablemente.

MapInfo permite organizar la información en el concepto de mapa que engloba a su vez el concepto de capa. Los mapas pueden incluir información de diferentes características, contenida en capas de información específica. Posee un convertidor universal, herramienta que tiene el objetivo de transformar los datos geográficos, permitiendo de esta forma el intercambio entre varios formatos vectoriales. Soporta AutoCad DWG/DXF, ESRI Shape, Intergraph/MicroStation Design y MapInfo MID/MIF/TAB.

Para la representación dispone de un conjunto de sistemas de coordenadas y proyecciones. MapInfo posee funciones de geocodificación para una tabla externa de datos. Con la opción de geocodificación automática, MapInfo procesa toda la tabla, sin detenerse en los registros no coincidentes. Se puede también seleccionar elementos mediante herramientas diversas, como el selector radial, poligonal y rectangular.

Realiza las consultas mediante el asistente SQL Selección. Se puede gestionar con el cuadro de diálogo de control de capas, la visualización, edición, selección y etiquetado de las capas.

Para la creación de objetos ya existentes o su modificación, se pueden dibujar los elementos básicos de GIS y crear nuevas capas de datos, agrupar y combinar polígonos para crear otros, o bien seleccionar elementos contenidos en polígonos. También se dispone de un asistente para la generación de informes de datos tabulares cuyo origen puede ser cualquiera de las tablas disponibles, una consulta o una selección.

El cálculo de zonas o áreas de influencia dispone de la opción de generar una única zona para todos los objetos seleccionados o bien una zona para cada objeto. Como resultado de estas operaciones se obtiene dibujada la zona o zonas de influencia, con la que se puede hacer una selección poligonal de elementos de otras capas.

Igualmente se puede agregar una imagen ráster como una capa más del mapa, de forma que permite tenerla como fondo de las capas de datos y utilizarla para digitalizar nuevos elementos, tomándola como referencia o simplemente como información visual adicional. Lógicamente, como una imagen no posee ningún tipo de sistema geográfico de coordenadas, se debe geocodificar mediante un conjunto de puntos de control en ambas capas.

4.4.3. MicroStation GeoGraphics

Aunque es cierto que GeoGraphics es un producto con entidad propia, también lo es, como su nombre indica, que necesita a MicroStation instalado, del cual es una extensión. La consecuencia inmediata es que tanto el interfase como la forma genérica de funcionamiento será, por tanto, la de MicroStation.

El diseño basado en características es un concepto propio. Se trata de un tipo de diseño que permite definir

características gráficas y atributos no-gráficos para todos los objetos de un mapa.

Este sistema también es útil para la creación de mapas temáticos o de propósito espacial. Las herramientas temáticas utilizan los atributos de la base de datos para controlar la asignación de símbolos, rellenar áreas y marcar patrones. Igualmente existen herramientas de anotación para la generación automática de etiquetas (texto) a partir de la información de la base de datos.

La limpieza de Geometría y Validación, consiste en la capacidad de depurar los datos de forma interactiva, junto con herramientas de validación que localizan y solucionan problemas respecto la entrada de datos. En efecto, en la digitalización de mapas es muy común el desplazar ligeramente algún punto, haciendo necesarias correcciones para que los nodos de los arcos sean coherentes. Otros errores clásicos pueden ser las líneas similares, los fragmentos lineales, colgantes, elementos solapados, etc...

En lo referente al análisis topológico y espacial, las herramientas proporcionadas realizan las operaciones más clásicas, es decir operaciones sobre puntos, líneas, conjuntos de polígonos y desarrollo de intersecciones y uniones espaciales. Es de destacar que las operaciones de análisis espacial utilizan funciones de topología que permiten inferir relaciones espaciales de los datos sin la necesidad de almacenar esta información de forma permanente. Es decir, los elementos poligonales pueden ser creados bajo requerimiento.

Las bases de datos a gestionar se manipulan mediante un acceso ODBC lo que asegura una conectividad cliente/servidor. Existe una utilidad de extracción de datos que permite la creación de nuevos conjuntos de datos a partir de una librería de mapas utilizando criterios combinados de filtrado, información espacial y base de datos.

Es posible utilizar imágenes georeferenciadas como complemento a los mapas vectoriales. Particularmente interesante la metodología interna de manipulación de grandes imágenes en memoria así como mosaicos de las mismas. En la salida por plotter es posible combinar imágenes vectoriales (mapas) con imágenes binarias (fotografías), dado su excelente gestor de impresión. Finalmente, la personalización de la aplicación o bien la automatización de tareas corre a cargo de un lenguaje propio, el Microstation Basic.

4.4.4. Cuadro comparativo de productos GIS comerciales.

	ArcView	**MapInfo**	**MicroStation Geographics**
Versión	3.0	4.5	5.5
Fabricante	E.S.R.I.	MapInfo	Bentley Systems
Idioma	Castellano	Castellano	Castellano
Interfaces	Aspecto y funcionalidad algo anticuada pero resulta muy útil y esta estructurado eficazmente.	Sencillez con detalles útiles y aspecto y funcionalidad actual.	El que conozca el microstation 95 no va a tener problemas.

	ArcView	MapInfo	MicroStation Geographics
Documentación	Suficiente para comenzar a trabajar, buenos ejemplos ilustrativos, aunque en inglés. Datos de ejemplo.	Todo tipo de documentación impresa, prácticas, de usuario, referencia, y traducidos al castellano. Datos de ejemplo.	Suficiente, se hecha en falta datos de ejemplo.
Formatos gráficos y GIS soportados Importación /exportación.	SHP(Propio),DXF,DWG,DGN, ARCINFO,MIF(intercambio de mapinfo).	El conversor Universal translator permite transformar archivos de DWG,DXF,DGN,SHP,MIF,TAB, a formatos SHP,DGN,TAB,MIF	DGN(Propio),Dwg,CGM,IGES . Para intercambio con MapInfo, ArcView, ArcInfo y otros se tiene que hacer con el producto independiente GeoExchange
Formatos de datos no gráficos	Dbase, Info (ESRI), ASCII, soportados por Jet Database, ODBC.	TAB, Dbase, ASCII, Lotus, Excel, ODBC	Oracle, INFORMIX, ODBC
Salidas de datos y mapas temáticos.	Suficiente para la edición de mapas GIS en los que los retoques estéticos no son prioritarios. Temáticos de degradado por rangos, valor único, con gráficas barras y tarta, densidad de puntos y graduado.	Suficiente para editar mapas GIS. Asistentes para generación de informes. Temáticos de degradado y por rangos, con gráficos barras y tarta, densidad de puntos y graduado.	Excelente, toda la potencia de Microstation 95 a nuestro servicio. Mapas con todo tipo de detalles.
Autoetiquetado de los elementos gráficos.	Si.	Si. Visualización de etiqueta sensible al cursor.	Si.
Análisis Topológico	Todas las operaciones típicas GIS son realizadas con eficacia y sencillez mediante constructor de consultas. Probablemente sea el mejor en esta faceta	Como en los demás productos se resuelve mediante asistentes de consultas. En este caso bien diseñados.	Las operaciones fundamentales se realizan de mediante asistente de consultas.
Selección Polígono con Polígono	Si	Si	Si
Selección Punto con Poligono	Si	Si	Si
Selección a distancia de	Si	Si	Si
Selección radial	Si	Si	Si
Crea zonas de influencia/buffering	No. Se puede hacer mediante programación en Avenue y con el módulo Network Analyst	Si	Si
Generador avanzado de informes	No	Si. Asistente para su generación.	No
Otros productos GIS	Muy diversos	Muy diversos	Soluciones Frames.
Conversor de proyecciones	Si	Si	Producto independiente. *GeoExchange*.
Módulos adicionales.	Todo tipo de extensiones. Destacar los módulos Spatial Analyst, Network Analyst, 3D y Arcpress, para impresión ,excelentes herramientas aunque el coste sea superior al del producto base.	Modulo de terceros de análisis espacial. Para problemas de redes se proporcionan soluciones concretas.	Bentley ofrece soluciones especificas de análisis GIS para problemas concretos.

4.4.5. Conclusión final sobre algunos productos SIG's comerciales

La colección de productos de que dispone ESRI, así como mapas digitales de todo tipo, es absolutamente completa. Lo cual, todo sea dicho, justifica que el binomio ArcView/ArcInfo sea, probablemente, la solución GIS más extendida a nivel mundial.

El MapInfo no se queda muy a la zaga en determinadas áreas, concretamente en aquellas donde compite directamente

con ArcView. Se trata de un producto muy veterano, con bastante introducción en sectores comerciales.

5 CONCLUSIONES

- Se lograron establecer los requerimientos necesarios para el desarrollo de un Sistema de Información Geográfica. Se pudieron establecer lo correspondiente a la descripción de los datos, descripción de consultas (tanto de tipo gráfico como de tipo escrito), se logro conceptualizar y establecer los requerimientos técnicos necesarios, los tiempos de respuesta, el volumen de datos y lo correspondiente al hardware, software y sistema operativo necesario para la implementación de un SIG.

- El diseño e implementación de un SIG genérico es fortalecido con la utilización del UML.

- El desarrollo de aplicaciones usando MapObjects brinda todas las ventajas de un entorno de programación orientado a objetos, trabajando íntegramente en tecnología Windows de 32 bits. Se tiene las posibilidades de acceder a diversos formatos geográficos (shapefiles, ARC/INFO, SDE) y una gran variedad de formatos de imágenes raster como formatos .bmp, .tif y otros), producir mapas temáticos, utilizar si se requieren, bases de datos relacionales (RDBMS) , georeferenciar eventos por direcciones y desplegar eventos en tiempo real utilizando tecnología GPS.

- La tecnología orientada a objeto- han probado ser una buena solución para el diseño de aplicaciones no - convencionales, donde la complejidad de los datos y las relaciones subyacentes son críticas. Al usar esta tecnología se puede obtener no sólo software reusable y modificable, sino que también se consiguen sistemas interoperables, como objetos de conocimiento encapsulado, que pueden ser adaptados a diferentes necesidades.

6 BIBLIOGRAFIA

Aronoff, S. 1989. Geographical Information System. WDL Publications Otawa Canadá.

Aronoff S. 1989. Los Sistemas Geográficos de Información: Una Perspectiva de Gestión". WDL de Publicaciones. Ottawa. Canadá.

Bakker X., Pérez F. 1997. Conceptos de análisis y modelamiento. Software Ilwis 2.0 for Windows. Santafé de Bogotá.

Booch, G., Rumbaugh J., Jacobson I., 1999. El lenguaje unificado de modelado, Editorial Addison Wesley, Series Editors.

Bosque Sendra, J. 1992. Sistemas de Información Geográfica, Ed.Rialp, Madrid, España.

Burrough, P.A.1988. Principles of Geographical Information Systems for land resources assesment, Oxford, Oxford University Press.

Catell R.G.G. 1994. Object Data Management Object-Oriented and Extended Relational Database Systems. Addison-Wesley Plublishing Company United States.

Kim W. 1995. Observations on the ODMG-93 Proposal for an Object-Oriented Database Language UniSQL, Inc. 9390 Research Blud. Austin, Texas 78759.

López J., Medeiros C.B. A Direct Manipulation User for Querying Geographic Databases. Universidad de Campinas SP Brasil.

Meaden G. J., Kapetsky J.M. 1992. Sistemas de Información Geográficos y la Telepercepción en la pesca continental y la acuicultura. FAO documento técnico de pesca. Roma.

Medeiros, C.B., Pires F. Databases for SIG. Universidad de Campinas SP Brasil.

Pérez U. 1993. Fundamentos de un SIG. Instituto Colombiano Agustín Codazzi. Santafé de Bogotá, Colombia.

Pires f., Medeiros, C.B. Un ambiente computacional de modelagem de aplicacoes geográficas. Universidad de Campinas SP Brasil.

Pressman, E.S. Ingeniería de Software. Editorial Prentice Hall, 1990.

Valenzuela, C.R., Belward, A.S., 1991. Remote Sensig and Geographical Information Systems for Resource Management in Developing Countries. Netherlands.

7 NOTAS

[1] Ministerio del Medio Ambiente - IDEAM U. Nacional 1997.

[2] Recursión. Disponible en:http://www.itlp.edu.mx/publica/tutoriales/estru1/34.htm

[3] History of GIS. Disponible en: http://feature.geography.wisc.edu/sco/gis/gis_hist.html

[4] Sistemas de Información Geográfica. Disponible en: http://www.geocon.hn/espanol/sig.html

[5] Aronoff S. 1989. Los Sistemas Geográficos de Información: Una Perspectiva de Gestión". WDL de Publicaciones. Ottawa. Canadá.

[6] National Center for Geographic Information & Analysis (NCGIA).
Disponible en: http:// www.ncgia.ucsb.edu/

[7] Análisis de los sistemas de información geográfica:
Disponible en: http://www.map.es/csi/silice/Sisinfgeo.html

[8] Pérez U. 1993. Fundamentos de un SIG. Instituto Colombiano Agustín Codazzi. Santafé de Bogotá, Colombia.

[9] Antecedentes: La FAO y las bases de datos de recursos naturales, Aspectos técnicos del SIG, Algunas funciones básicas del SIG.
Disponible en http://www.fao.org/WAICENT/FAOINFO/SUSTDEV/spdirect/gis/chap3.htm.
Aplicaciones de la tecnología del SIG.
Disponible en http://www.fao.org/WAICENT/FAOINFO/SUSTDEV/spdirect/gis/chap4.htm

[10] Universitat de Girona. La nueva forma de aprender GIS. .http://giscampus.com/

[11] AM-FM. Disponible en: www.aeroterra.com/HTMs/SrvAMFM.htm

[12] Centro de Investigación de las Telecomunicaciones. Disponible en: http://www.cintel.org.co/

[13] Catell R.G.G. 1994. Object Data Management Object-Oriented and Extended Relational Database Systems. Addison-Wesley Plublishing Company United States.

[14] Medeiros, C.B., Pires F. Databases for SIG. Universidad de Campinas SP Brasil.

[15] CORBA, ORBit y Baboon. Disponible en: http://www.hispafuentes.com/manuales/7.0/hf-guia-gnome/corba.html

[16] dcom. Disponible en: http://www.terra.com.ve/informatica/que-es/dcom.cfm

[17] JDBC. Disponible en: http://www.vistoynovisto.com/vistoynovisto/Tutorial/Javatut2/jdcbook/jdbc.html

[18] The Object-Oriented Database System Manifesto, Malcolm Atkinson, University of Glasgow, François Bancilhon, Altaïr, David DeWitt, University of Wisconsin, Klaus Dittrich, University of Zurich, David Maier, Oregon Graduate Center, Stanley Zdonik, Brown University

[19] Manejador de Bases de Datos. Disponible en: http://www.itlp.edu.mx/publica/tutoriales/basedat1/tema1_9.htm

[20] Bertino E. y Martino L. *Object-Oriented Database Systems: Concepts and Architectures*. Addison-Wesley, 1993. Traducción castellano: *Sistemas de Bases de Datos Orientadas a Objetos. Conceptos y Arquitecturas*. Addison-Wesley/Diaz de Santos, 1995

[21] ORION/ITACSA consiste de una serie de manejadores de bases de datos de una generación superior al modelo relacional cuyo desarrollo fue iniciado por el Advanced Computer Technology (ACT) Program, en el Microelectronics and Computer Technology Corporation

(MCC) en 1985. ORION fue desarrollado como un vehículo para la integración de lenguajes de programación orientados a objetos y bases de datos avanzadas para aplicaciones en Inteligencia Artificial, CAD/CAM y sistemas de automatización de oficinas. La serie ORION incluye ahora tres sistemas, ORION-1, es un sistema monousuario, multitarea; ORION-1SX está diseñado para redes de área local en una arquitectura cliente - servidor; y ORION-2 es un manejador de bases de datos distribuidas. ORION "per se", está sólo disponible a miembros patrocinadores de ACT. La versión comercialmente disponible en el mercado se llama ITACSA y la distribuye Itacsa Systems Inc.

22 ONTOS es la BDOO que ofrece la compañía Ontologic, Inc. La característica principal de ONTOS es que incluye una interfaz totalmente transparente con C++ (esto es, transparentemente activa objetos en la memoria como se requiere dinámicamente). Además, permite, transparentemente, activar objetos desde el servidor de la base de datos y proporciona una interfaz muy versátil con otros productos y herramientas de ONTOS como ONTOS SQL, Ontos Studio, Ontos Shorthand y Ontos Ptech. ONTOS está completamente basado en el modelo cliente- servidor, proporciona soporte para datos distribuidos y aplicaciones distribuidas en redes cliente - servidor. El protocolo de transacciones distribuido soporta "two-pahse commits" atómicos. El producto ONTOS está disponible para Sun 3, 4 y SPARC, y las series Apollo 3000 y 4000 y múltiples equipos en OS/2.

23 GemStone es el producto distribuido por Servio Corporation. Gemstone está desarrollado en C y está basado en un diseño sobre una arquitectura cliente - servidor con múltiples hilos de ejecución. La mayor virtud es su soporte para múltiples plataformas, estaciones de trabajo y servidores que van desde IBM, Digital y Sun. Además puede implantarse en PCs IBM y compatibles, así como Macintosh II.

24 Versant es la BDOO de tipo distribuido y multiusuario, en un modelo multicliente/multiservidor que desarrolló Versant Object Technology. Está disponible para la Silicon Graphics IRIS 4D y todas las estaciones Sun. Versant es el producto que soporta más estándares, incluyendo X Window, OSF/Motif, ANSI C, AT&T C++2.0, Smalltalk-80 y TCP/IP

[25] El sistema manejador de bases de datos O2 es producido por la firma francesa Altair y es un producto comercial bastante conocido. Los expertos consideran a O2 como un manejador de BDOO del mismo desarrollo que ITACSA. O2 es el resultado de un proyecto de investigación a largo plazo y constituye un sistema avanzado, tanto desde el punto de vista de la tecnología de bases de datos, como del ambiente de desarrollo. O2 no sigue la filosofía de diseñar una extensión de un lenguaje OO para incluir persistencia (como puede pensarse de Gemstone a partir de Smalltalk); tiene su propio lenguaje CO2 para implantar métodos, el cual es una extensión de C.

[26] ObjectStore es el producto desarrollado y comercializado por Object Design. ObjectStore es un manejador de BDOOs para las estaciones de trabajo SUN3 y Sun Sparc. Existen versiones para Microsoft Windows 3.0 y otras estaciones de trabajo en UNIX. ObjectStore, y sus herramientas de desarrollo, está basado en C++. El preprocesador proporciona una interfaz de desarrollo que permite comprimir código de C++ en enunciados optimizados. Además, el manejador de ObjectStore permite tipos parametrizados. Los tipos parametrizados son comúnmente utilizados para definir clases que contienen e incorporan código de C++ y que permiten el desarrollo de código C++ reutilizable.

[27] Architecture of the ORION Next-Generation Databases System. Disponible en: http://www.computer.org/tkde/tk1990/k0109abs.htm

[28] OMG, disponible en: http://www.omg.org/, Object Management Group

[29] Kim W. 1995. Observations on the ODMG-93 Proposal for an Object-Oriented Database Language UniSQL, Inc. 9390 Research Blud. Austin, Texas 78759.

[30] Base de datos orientadas por objetos: http://sistemas.ing.ula.ve/sistemas/bd/odmg.html

[31] ESRI, GIS y mapping software.

Proveedor mundial de software para sistemas de información geográfica. Disponible en: http://www.esri.com

[32] Proveedor de Software, en Colombia. Productos ESRI. Documento sobre Sistemas de Información Geográfica. Disponible en: http://www.prosis.com/solucion/sig/SIG.html

[33] UML Summary. Rational Software Corp. Septiembre de 1997. Disponible en: http://www.rational.com/uml

[34] UML Summary. Rational Software Corp. Septiembre de 1997. Disponible en http://www.rational.com/uml

[35] UML Notation Guide. Rational Software Corp. Septiembre de 1997. Disponible en http://www.rational.com/uml

[36] UML Semantic. Rational Software Corp. Septiembre de 1997. Disponible en http://www.rational.com/uml

[37] Booch, G., Rumbaugh J., Jacobson I., El lenguaje unificado de modelado, Ed. Addison Wesley, Series Editors, 1999.

[38] Luis A. Colmenares G., Tesista de la Escuela de Ingeniería de Sistemas de la Universidad de los Andes. Mérida - Venezuela, realizó la implementación de clases espaciales para la visualización de mapas en el mundo WWW. El proyecto es auspiciado por el Grupo de Investigación de Ingeniería de Datos y de Conocimientos (GIDYC)

[39] JMC/Y&R2. Disponible en: http://www.2punto1.com/glosario/e-h.html

[40] PAAL (Plan de Acción Ambiental Local) de Yumbo

41 CVC, 1997.

42 CVC, 1997.

43 En este caso la información fue recolectada por el Plan de Acción Ambiental Local del Municipio de Yumbo.

44 ESRI, GIS y mapping software.

Proveedor mundial de software para sistemas de información geográfica. Disponible en: http://www.esri.com

45 ESRI de Venezuela. MapObjects. Disponible en:

http://www.esriven.com/paginas/productos/Prod_Mapobjects.htm