Benemérita
Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

"Sistema de consulta de posicionamientos dentro de la Ciudad de Puebla"

Tesis Profesional

Que para obtener el título de:
Licenciado en Ciencias de la Computación

Presenta
Juan Peralta Trinidad

Asesor
Dr. Abraham Sánchez López

Puebla, Pue. Otoño 2007

Índice

Dedicatoria

Cuando inicie éste documento, una de las persona a la que yo debo mi carrera aún estaba con migo; ha la fecha he culminado y ése ser ya no está con migo; pero donde quiera que estés, quiero que sepa que te agradezco todo lo que soy, ¡gracias Padre!

Dedico a todos mis maestros no sólo éste documento, si no más que eso, toda mi carrera, porque de una forma u otra me han forjado para llegar hasta donde hoy estoy.

En una forma muy especial, agradezco mi Madre, porque además de darme la vida, me ha dado su apoyo y su Amor tan grande.

A mis hermanos agradezco todo su cariño que ha sido el motor de mi vida.

A mi esposa y mis dos hijos, porque me han motivado intensamente a culminar éste documento.

A mi asesor, por la motivación para culminar ésta etapa profesional.

Introducción

Propósito del proyecto

El propósito del proyecto es el de desarrollar una herramienta útil parra el control y toma de decisiones dirigida a empresas que se dedican a la distribución de mercancía dentro de la ciudad de Puebla.

Dado que hoy en día, se cuenta con poco información respecto al recorrido que hizo una determinada unidad durante la repartición de mercancía, ésta herramienta nos ayudará a tener un panorama general y aproximado sobre el kilometraje recorrido, los consumos de gasolina promedio y verificar en un momento dado si la unidad excedió los limites de velocidad o si se salió completamente del área de entrega.

Con la investigación obtenida sobre bases de datos espaciales y sistemas de información geográfica, podría ampliarse para un sistema que pueda monitorear en tiempo real, la localización de una unidad que se encuentra fuera de la empresa.

Descripción general del sistema

El sistema lleva el control de las unidades de transporte de mercancía, los productos de almacén, los conductores, pedidos y recepción de productos. Cuenta con un módulo donde se dan de alta las estrategias de reparto. Existe una opción que se encarga de descargar los puntos que tiene almacenado el módulo que va montado a la unidad de reparto.

El sistema también cuenta con un módulo de consulta de recorrido, el cuál emite el promedio de kilometraje, las velocidades para cada punto de la ruta, el consumo promedio y se visualiza en un mapa, el recorrido realizado para una unidad determinada.

Objetivo general del proyecto

El sistema podrá llevar el control de los productos que distribuye y mantendrá actualizada las existencias a la hora de recibirlos. Podrá también dar de altas, bajas y modificaciones a los diferentes catálogos de conductores, productos, unidades y clientes. También permitirá registrar los movimientos de pedidos, repartos, registros de salidas a reparto y llegadas de reparto.

Objetivos específicos del proyecto

1.-Que el usuario pueda visualizar los recorridos de cada unidad en un mapa que trace solo aquellas calles, colonias y manzanas por donde paso la unidad.
2.-Poder dar de alta y baja las diferentes entidades como son productos, unidades, conductores y clientes.
3.-Llevar un control de los pedidos que hace el cliente.
4.- Armar estrategias de reparto.
5.-Simular la descarga de los datos del recorrido provenientes del modulo montado en cualquier unidad con la ayuda de un archivo que contenga posiciones capturadas.
6.-Actualizar la existencia de productos a la hora de hacer una recepción de productos.

Justificación del proyecto

La toma de decisiones y estrategias de una empresa que en su infraestructura cuenta con unidades móviles, es de considerable importancias para optimizar gastos. La planeación con ayuda de información centralizada de sus móviles ayudará a la toma de decisiones que contribuirá a la competitividad y control de la empresa.

Limitantes del proyecto

Para la llegada de reparto, la unidad descargará los datos al sistema, por lo que solo se simulará con la ayuda de un archivo que contendrá puntos GPS.
Se cuenta con una pequeña parte del mapa de mancha urbana de puebla, que se obtuvo como un archivo de texto en catastro y que será descargado a la base de datos para trazar el mapa.

Contenido

Éste documento se encuentra organizado en capítulos; el capítulo uno trata de las sistemas de información geográfica (GIS por sus siglas en ingles), que son sistemas que tratan de resolver problemas de toma de decisiones. Anteriormente solo los usaban las dependencias de gobierno, pero actualmente se han venido utilizando en la iniciativa privada.
El capítulo dos trata sobre las bases de datos espaciales, que tienen el objetivo de representar situaciones del mundo real y almacenarlos como una base de datos. Actualmente no se han podido resolver en su totalidad ese objetivo, debido a que en el mundo real existe una sin fin de variables que hacen difícil de plasmarla en una base de datos.
El capítulo tres trata lo relacionado con el análisis del sistema de control de recorridos para la ciudad de puebla, que tiene que ver principalmente sobre la problemática de no tener un control del recorrido que siguió una unidad de reparto durante la entrega de mercancía.
En el capítulo cuatro se documenta el diseño de la base de datos que tiene que ver con las tres etapas de diseño conceptual, lógico y el físico.
Por último se documenta el capítulo cinco que tiene que ver exclusivamente con la implementación del sistema, las herramientas, lenguajes e interfaces de usuario que se usaron.

Capítulo 1

Sistemas de Información Geográfica (GIS)

1.1. GIS

Un Sistema de Información Geográfica (SIG o GIS, en su acrónimo inglés) es un sistema integrado compuesto por hardware, software, personal, información espacial y procedimientos computarizados, que permite y facilita la recolección, el análisis, gestión o representación de datos espaciales.[3]

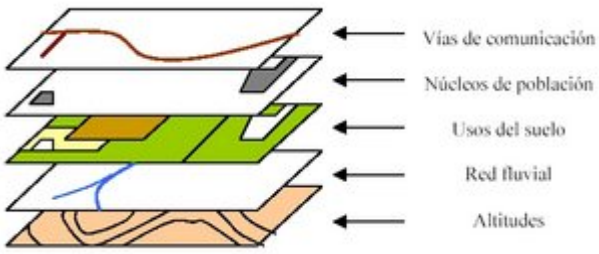
1.2. Descripción

El pionero de la epidemiología, el Dr. John Snow proporcionaría, allá por 1854, el clásico ejemplo de este concepto cuando cartografió la incidencia de los casos de cólera en un mapa del distrito de SoHo en Londres. Este protoSIG permitió a Snow localizar con precisión un pozo de agua contaminado como fuente causante del brote.

El SIG funciona como una base de datos con información geográfica (datos alfanuméricos) que se encuentra asociada por un identificador común a los objetos gráficos de un mapa digital. De esta forma, señalando un objeto se conocen sus atributos e, inversamente, preguntando por un registro de la base de datos se puede saber su localización en la cartografía.

El Sistema de Información Geográfica separa la información en diferentes capas temáticas y las almacena independientemente,

permitiendo trabajar con ellas de manera rápida y sencilla, y facilitando al profesional la posibilidad de relacionar la información existente a través de la topología de los objetos, con el fin de generar otra nueva que no podríamos obtener de otra forma.[1]



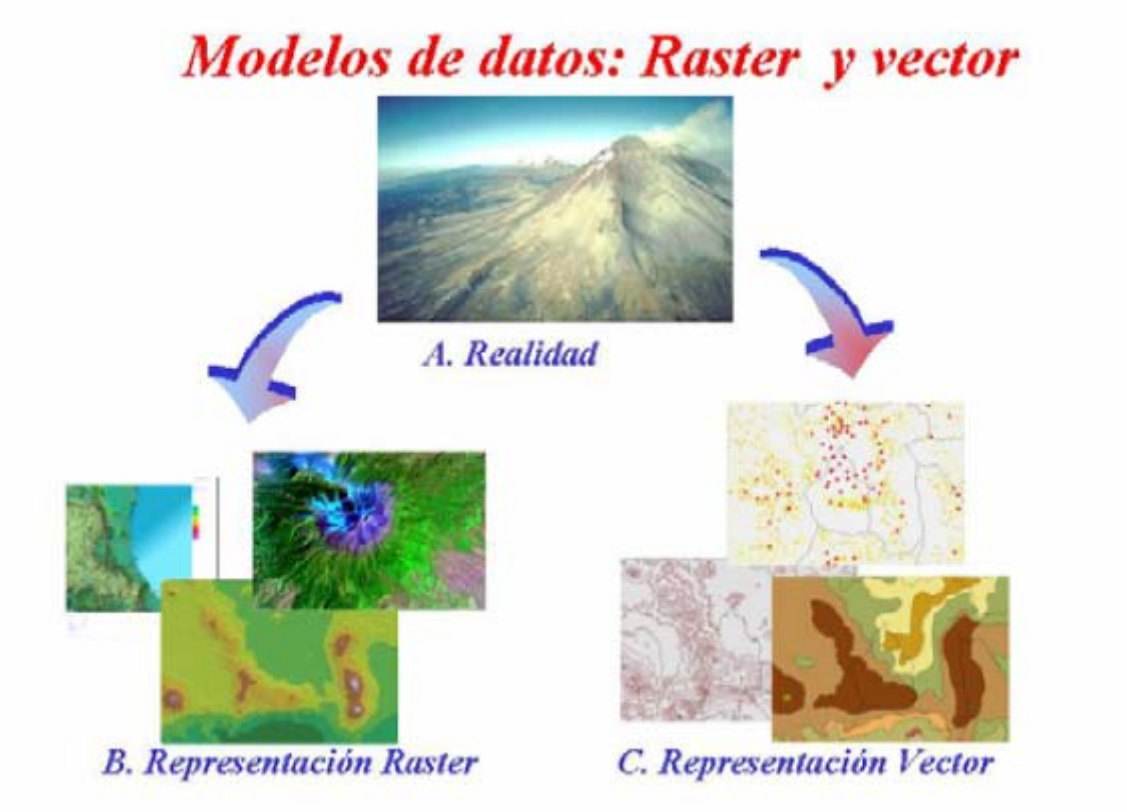
Un Sistema de Información Geográfica puede mostrar la información en capas temáticas para realizar análisis multicriterio complejos.[3]

1.3. Rastreo y Vectorial

Los software SIG pueden ser de rastreo o vectoriales. El modelo de SIG rastreo se centra en las propiedades del espacio más que en la precisión de la localización. Compartimenta el espacio en celdas regulares donde cada una de ellas representa un único valor. Cuanto mayores sean las dimensiones de las celdas (resolución) menor es la precisión o detalle en la representación del espacio geográfico. En el caso del modelo de SIG vectorial, el interés de las representaciones se centra en la precisión de localización de los elementos sobre el espacio. Para modelizar digitalmente las entidades del mundo real se utilizan tres objetos espaciales: el punto, la línea y el polígono.

Los SIG vectoriales son más populares en el mercado. No obstante, los SIG rastreo son muy utilizados en estudios medioambientales donde la precisión espacial no es muy requerida (contaminación

atmosférica, distribución de temperaturas, localización de especies pesqueras, análisis geológicos, etc.).[3]



1.4. Cuestiones a las que puede dar respuesta un SIG

Las principales cuestiones que puede resolver un Sistema de Información Geográfica son:

1.- **Localización**: Preguntar por las características de un lugar concreto

2.- **Condición:** El cumplimiento o no de unas condiciones impuestas al sistema.

3.- **Tendencia:** Comparación entre situaciones temporales o espaciales distintas de alguna característica.

4.- **Rutas:** Cálculo de rutas óptimas entre dos o más puntos.

5.- **Pautas:** Detección de pautas espaciales.

6.- **Modelos:** Generación de modelos a partir de fenómenos o actuaciones simuladas.

Por ser tan versátiles los sistemas de información geográfica, su campo de aplicación es muy amplio, pudiendo utilizarse en la mayoría de las actividades con un componente espacial. La profunda revolución que han provocado las nuevas tecnologías ha incidido de manera decisiva en su evolución.[3]

1.5. Listado de software SIG

Software no libre comercial: ArcGIS (Arcview, ArcInfo), Mapinfo, Maptitude, Geomedia, GenaMap, Autodesk Map, MicroStation Geographics, GeoWeb Publisher, SmallWorld, Manifold, Idrisi, MapPoint, TatukGIS, TNT mips, MiraMon.

Software libre: GRASS GIS, JUMP, MapServer, Quantum GIS, gvSIG, SAGA GIS.

Software no libre freeware: Spring, FGIS [3]

1.6. ArcGis

Es una colección integrada de herramienta desarrolladas por ESRI(Environmental Systems Research Institute, empresa dedicada a distribuir y desarrolla Sistemas de Información Geográfica); de fácil

uso que permite visualizar, explorar, consultar y analizar información geográfica. En ArcGIS no se necesita conocer como se crean los datos geográficos, de hecho solo los usa. ArcGIS carga fácilmente datos tabulares, mapas vectoriales (polígonos líneas y puntos) y mapas rastreo (fotografía aérea e imágenes de satélite)[6]

1.7. Modelo de datos GIS

El Modelo de datos se compone de 2 tipos

- **Modelo vector**

Éste modelo esta representado por puntos, líneas y poligonos

- **Modelo Rastreo**

Almacena datos numéricos en celdas o píxeles dentro de una cuadricula.[4]

1.8. Formato Shapefiles

El formato SapeFile fue desarrollado por la compañía ESRI.El formato Shapefile no es topológico, almacena localización geométrica e información de atributos de los elementos geográficos.

ESRI tiene una evolución respecto a la forma en que almacenar datos GIS que son:

Coberturas

Shapefiles

Geodatabases

Siendo el formato Coberturas el usado en los inicios, hasta llegar a la forma de almacenamiento en geodabases.[5]

1.9. ArcMap

ESRI ofrece un si número de productos para implementar GIS que van desde sistemas monousuario, hasta llegar a sistemas de aplicaciones web.

ArcMap Permite la visualización y consulta de varias capas de forma simultánea, gracias a herramientas como la ventana de aumento, la ventana de situación o los marcadores espaciales, así como la posibilidad de aplicar porcentajes de transparencia a las capas tanto vectoriales como rastreo.

ArcMap incorpora numerosas herramientas de edición de Geodatabases monousuario y ficheros Shapefile. Con estas herramientas se asegura la creación y el mantenimiento de la integridad de la información geográfica de forma rápida y sencilla.

Mediante la topología implícita o topología de mapa se controlan las relaciones espaciales existentes entre los elementos elegidos, las cuales se mantienen durante el proceso de edición.

Junto con las operaciones de generación de zonas de influencia y geoprocesamiento, ArcMap incorpora innumerables funciones para el análisis SIG.

La multitud de librerías de simbología especializada, herramientas de etiquetado y plantillas hacen de ArcMap la aplicación ideal para la producción cartográfica de alta calidad.[6]

1.10. ArcCatalog

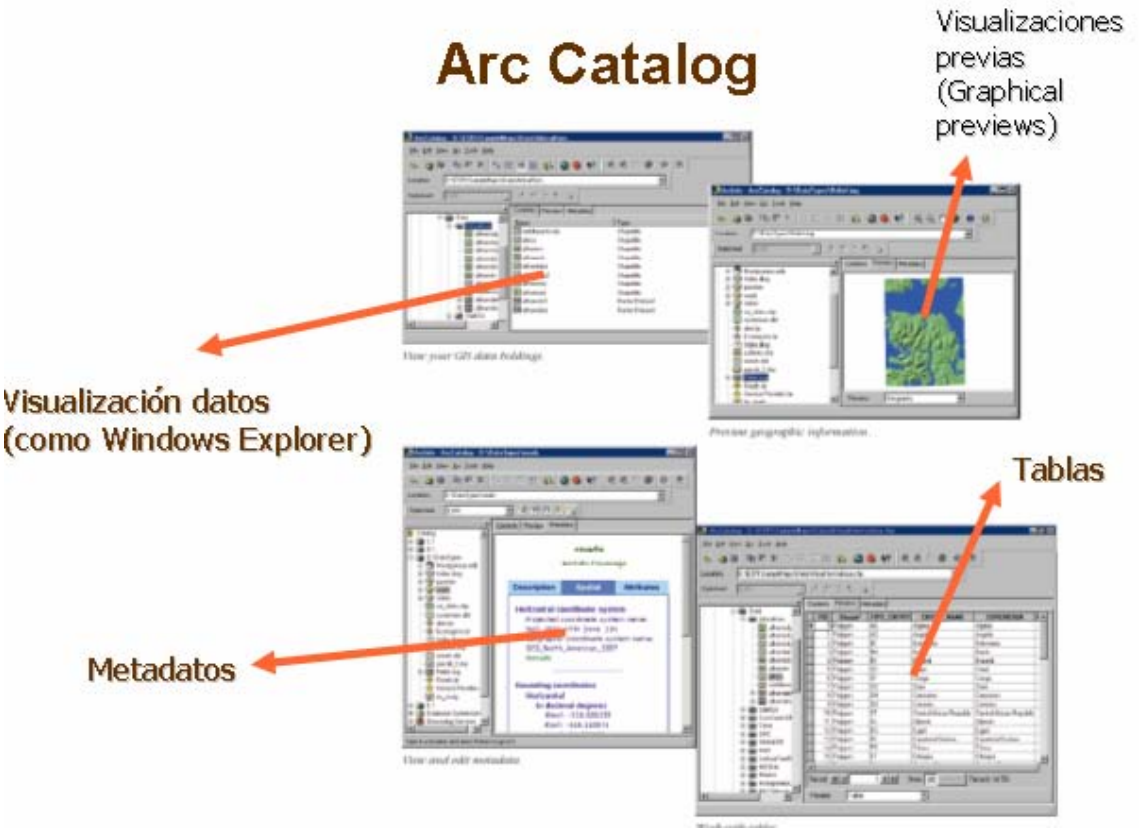
ArcCatalog permite administrar, organizar, crear y previsualizar tanto datos geográficos como alfanuméricos.

La funcionalidad que incorpora ArcCatalog consta de varias herramientas para:Definir reglas topológicas y elementos geográficos a los que se aplicarán dichas reglas, en la generación de topología de Geodatabase.

- Definición de subtipos en las entidades presentes en la Geodatabase.

- Generación de redes geométricas
- Carga de datos (vectoriales y rastreo) en bases de datos corporativas.
- Almacenar datos rastreo en bases de datos corporativas con posibilidad de hacerlo como imágenes independientes, catálogos embebidos o referenciados, y como mosaicos.

Todas las herramientas son de funcionamiento muy sencillo gracias a intuitivos asistentes y formularios.[6]



Capítulo 2

Base de datos espaciales

2.1. Definición

Base de datos espacial (spatial database) es un sistema administrador de bases de datos que maneja datos existentes en un espacio o datos espaciales. El espacio establece un marco de referencia para definir la localización y relación entre objetos. El que normalmente se utiliza es el espacio físico que es un dominio manipulable, perceptible y que sirve de referencia. La construcción de una base de datos geográfica implica un proceso de abstracción para pasar de la complejidad del mundo real a una representación simplificada que pueda ser procesada por el lenguaje de las computadoras actuales. Este proceso de abstracción tiene diversos niveles y normalmente comienza con la concepción de la estructura de la base de datos, generalmente en capas; en esta fase, y dependiendo de la utilidad que se vaya a dar a la información a compilar, se seleccionan las capas temáticas a incluir.

La estructuración de la información espacial procedente del mundo real en capas conlleva cierto nivel de dificultad. En primer lugar, la necesidad de abstracción que requieren los computadores implica trabajar con primitivas básicas de dibujo, de tal forma que toda la complejidad de la realidad ha de ser reducida a puntos, líneas o polígonos. En segundo lugar, existen relaciones espaciales entre los

objetos geográficos que el sistema no puede obviar; la topología, que en realidad es el método matemático-lógico usado para definir las relaciones espaciales entre los objetos geográficos puede llegar a ser muy compleja, ya que son muchos los elementos que interactúan sobre cada aspecto de la realidad.[7]

2.1. Datos espaciales

Un modelo de datos geográfico es una abstracción del mundo real que emplea un conjunto de objetos dato, para soportar el despliegue de mapas, consultas, edición y análisis. Los datos geográficos, presentan la información en representaciones subjetivas a través de mapas y símbolos, que representan la geografía como formas geométricas, redes, superficies, ubicaciones e imágenes, a los cuales se les asignan sus respectivos atributos que los definen y describen.

Un dato espacial es una variable asociada a una localización del espacio. Normalmente se utilizan datos vectoriales, los cuales pueden ser expresados mediante tres tipos de objetos espaciales.

Puntos

Se encuentran determinados por las coordenadas terrestres medidas por latitud y longitud. Por ejemplo, ciudades, accidentes geográficos puntuales, hitos.

Líneas

Objetos abiertos que cubren una distancia dada y comunican varios puntos o nodos, aunque debido a la forma esférica de la tierra

también se le consideran como arcos. Líneas telefónicas, carreteras y vías de trenes son ejemplos de líneas geográficas.

Polígonos

Figuras planas conectadas por distintas líneas u objetos cerrados que cubren un área determinada, como por ejemplo países, regiones o lagos.

De esta forma la información sobre puntos, líneas y polígonos se almacena como una colección de coordenadas (x,y). La ubicación de una característica puntual, pueden describirse con un sólo punto (x,y). Las características lineales, pueden almacenarse como un conjunto de puntos de coordenadas (x,y). Las características poligonales, pueden almacenarse como un circuito cerrado de coordenadas. La otra forma de expresar datos espaciales es mediante rasterización, la cual, a través de una malla que permite asociar datos a una imagen; es decir, se pueden relacionar paquetes de información a los píxeles de una imagen digitalizada.[7]

Los datos espaciales además se caracterizan por su naturaleza georreferenciada y multidireccional. La primera se refiere que la posición relativa o absoluta de cualquier elemento sobre el espacio contiene información valiosa, pues la localización debe considerarse explícitamente en cualquier análisis. Por multidireccional se entiende a que existen relaciones complejas no lineales, es decir que un elemento cualquiera se relaciona con su vecino y además con regiones lejanas, por lo que la relación entre todos los elementos no

es unidireccional. Es decir, todos los elementos se relacionan entre si, pero existe una relación más profunda entre los elementos más cercanos.[7]

2.2. Álgebra

El álgebra utilizada se denomina álgebra ROSE (RObust Spatial Extension) el cual está basada en los tipos de datos espaciales reales (STD - spatial data types), pero en este caso, los objetos no están definidos en el espacio Euclidiano continuo sino que en términos de la malla que discretiza el espacio, esto debido a que los cálculos computacionales son discretos. Los operadores espaciales se definen de la siguiente forma.

Operadores de selección

Point Query (PQ)

Dado un punto p, encontrar todos los objetos espaciales O que lo contienen.

PQ(p)={O|p pertenece a O.G ≠ Ø}

Range or region query (WQ)

Dado un polígono P de consulta, encuentre todos los objetos O que intersectan P. Cuando P es rectangular, se llama windows query.

WQ(P)={O|O.G □ P.G ≠ Ø}

Agregación espacial

Es una variante de búsqueda por vecino más cercano. Dado un objeto O', encuentre los objetos o que tiene una mínima distancia de o'.

NNQ(o')={o|para todo o: dist(o'.G,o.G) ≤ dist(o'.G,o.G)}

Join espacial

Es uno de los más importantes operadores. Cuando dos tablas R y S son unidas basado en un predicado espacial θ, la unión de las tablas es llamada espacial. Una variante de este operador en SIG es la superposición de mapas (map overlay). Este operador combina dos conjuntos de objetos espaciales para formar un nuevo conjunto. Las fronteras de este conjunto son determinadas por los atributos no espaciales asignados por la operación de superposición. Por ejemplo, si la operación asigna un mismo valor de un atributo no espacial a dos objetos vecinos, ellos se juntan o mezclan.[7]

2.3. Métodos de acceso espacial

Para evitar la revisión exhaustiva de los datos en una base de datos, se crean índices que reducen el número de elementos a visitar en la base de datos en un procesamiento de consulta. La clásica indexación por B-tree no es aplicable en el caso espacial donde no existe un orden único de los valores de claves. Es por este motivo que existen tres categorías de métodos de acceso espacial, las PAM (Point Access Method), R-Tree, las SAM (Spatial Access Method), los cuales se utilizan de acuerdo al tipo de dato en el que está la base de datos espacial ya sea raster o vectorial. Aunque se han creado los benchmarks que comparan diferentes métodos, los resultados no son concluyentes, pero se recomienda utilizar cualquiera de ellos. Un índice R-tree aproxima cada geometría en un único rectángulo que la acota minimizando los espacios llamado MBR (Minimal Bounding Rectangle) y organiza una colección de objetos espaciales en una

jerárquica donde las hojas contienen punteros a los datos y los nodos intermedios contienen el rectángulo mínimo que contiene a sus sub-hojas. Todas las hojas aparecen al mismo nivel. Cada entrada a una hoja es una tupla (R,O), donde R es el MBR y O es el objeto. Cada nodo intermedio es un tupla (R,P), donde R es el MBR que contiene los rectángulos hijos apuntados por P.[7]

2.4. Lenguajes de consulta espacial

Las bases de datos espaciales no tienen un conjunto de operadores que sirvan como elementos básicos para la evaluación de consultas ya que estas manejan un volumen extremadamente grande de objetos complejos no ordenados en una dimensión. Es por esto que existen algoritmos complejos para evaluar predicados espaciales. Las consultas son realizadas generalmente en SSQL (Spatial SQL), el cual introduce, mediante extensiones, los distintos conceptos del álgebra ROSE dentro del lenguaje SQL estándar, es decir, utiliza las cláusulas SELECT-FROM-WHERE para las tres operaciones en el álgebra relacional (proyección algebraica, producto cartesiano y selección). Las tres categorías fundamentales de consultas en un sistema de información espacial son:

Consultas exclusivamente de propiedades espaciales. Ejemplo: "Traer todos los pueblos que son cruzados por un río".

Consultas sobre propiedades no espaciales. Ejemplo: "Cuantas personas viven en Valdivia".

Consultas que combinan propiedades espaciales con no espaciales. Ej: "Traer todos los vecinos de un cuadra localizada en Puerto Varas"

En el lenguaje SSQL, el ejemplo del segundo punto se escribiría de la siguiente forma.

SELECT poblacion FROM ciudades WHERE nombre= "Valdivia"

EL otro tipo de consultas, para los datos obtenidos mediante rasterización, es llamado PSQL(Pictoral SQL) donde cada objeto espacial se extiende mediante un atributo loc (localización) el cual es referenciado en la cláusula SELECT para una salida gráfica y una cláusula específica para tratar relaciones espaciales. También se destaca en los lenguajes de modelado de la información espacial a GML que es una estructura para almacenar y compartir datos geográficos. Es una codificación del modelo geométrico de rasgo simple del OGC (Open Geospatial Consortium simple feature) usando XML. Un rasgo geográfico (geographic feature) es definido por el OGC como "una abstracción del fenómeno del mundo real, si este está asociado con una posición relativa a la Tierra". Por tanto, es posible hacer una representación del mundo real con un conjunto de rasgos. La especificación de un rasgo viene dada por sus propiedades, las que pueden pensarse definidas como un triple (nombre, tipo, valor). Si este rasgo es geográfico entonces la propiedad tendrá un valor geométrico. Por tanto, un rasgo simple del OGC es aquel cuya propiedad geométrica está restringida a una geometría simple en la

que sus coordenadas estén definidas en dos dimensiones y en el caso de existir una curva, esta es sujeta a una interpolación lineal.[7]

2.5. Aplicaciones

Normalmente las bases de datos espaciales están asociadas a sistemas SIS (Sistemas de Información Estratégicos) o SIG (Sistemas de Información Geográfica). La información geográfica contiene una referencia territorial explicita como latitud y longitud o una referencia implícita como domicilio o código postal. Las referencias implícitas pueden ser derivadas de referencias explicitas mediante geocodificación. La información geográfica es a su vez el elemento diferenciador de un Sistema de Información Geográfica frente a otro tipo de Sistemas de Información; así, la particular naturaleza de este tipo de información contiene dos vertientes diferentes: por un lado está la vertiente espacial y por otro la vertiente temática de los datos. Mientras otros Sistemas de Información contienen sólo datos alfanuméricos (nombres, direcciones, números de cuenta, etc.), las bases de datos de un SIG integran además la delimitación espacial de cada uno de los objetos geográficos. Las implementaciones de bases de datos espaciales se dividen en tres campos.

SIG Puros

Son bases de datos espaciales sin ninguna capa intermedia, realizan las operaciones de selección espacial de manera nativa. Son modulares, extensibles y normalmente con una interfaz amigable. Aunque también son capaces de generar una interfaz gráfica

amigable para las bases de datos comunes, de tal manera de utilizar datos espaciales ya almacenados en estas tecnologías.

Ad-hoc

Son sistemas desarrollados para alguna aplicación determinada, que utilizan un sistema de manejo de archivos propio y por ende un sistema de administración de datos propio. Es por eso que no son modulares, ni reutilizables. La ventaja es que son muy eficientes.

Bases de datos con extensiones para bases de datos espaciales

Son sistemas de bases de datos normales a los cuales se les agrega una capa para el manejo de la geometría y hacer el "traspaso" desde datos comunes a datos espaciales transparente al usuario.[7]

Capítulo 3

Análisis requerimientos

3.1. Descripción general del problema

En el almacén de una empresa, donde se guardan diversos productos para ser distribuidos en la ciudad de puebla, solicita que se implante un sistema por computadora que lleve el control de los productos que salen y entran del almacén; cada producto se identifica por un código. Es importante que dicho sistema, permita saber el recorrido que siguió cada unidad durante el reparto de los pedidos. El recorrido se visualizará como puntos unidos por líneas según la trayectoria dentro de un mapa. Para todo ello se cuenta con información de unidades, clientes, productos y conductores.

De las unidades se cuenta con el número de placa, capacidad de carga y consumo de combustible.

Con lo que respecta a los clientes, se tiene información de nombre, dirección, teléfono, código postal, municipio y su dirección de correo web(email).

Los conductores están registrados con su nombre, dirección, teléfono, fecha de nacimiento y licencia.

Los productos se identifican con código único, su descripción, precio y la cantidad en existencia.

Con toda la información anterior, se pretendiendo sirva como apoyo al nuevo sistema de rutas que deberá implementarse bajo el siguiente esquema:

El encargado del sistema deberá actualizar el sistema cada vez que halla entrada de mercancía.

El encargado del sistema también deberá dar de alta los pedidos que el cliente solicite, le asignará un folio de pedido y los registrará como “pedidos pendientes”. Después de tener una cantidad considerable de pedidos pendientes, se procederá a organizarlos, de tal forma que se obtengan diversas rutas de reparto o bitácoras de reparto. A cada bitácora de reparto se le asignará una unidad y un conductor. La bitácora de reaparto será entonces entregada al conductor para que la satisfaga.

Después la unidad es cargada con los productos para iniciar el recorrido, sin antes notificar su salida.

Cabe señalar que cada unidad estará equipada con un dispositivo identificado por una llave única. Éste dispositivo almacenará las posiciones cada determinado periodo de tiempo. Las posiciones se componen de coordenadas GPS, de hora-fecha y velocidad que registra en esemomento.

Cuando la unidad salga de la empresa, empezará a tomar lecturas de su posición; cuando termine el reparto, se deberán descargar los datos al sistema a través del puerto serie y se actualizarán las entregas completadas como “pedidos no pendientes”.

En la base de datos, se deberá contar con un mapa de la ciudad, organizado en calles y colonias; todos éstos por separado.

Cada calles tendrá su nombre. Es importante precisar que cada calles puede estar compuesta de segmentos, debido a que una calle puede empezar en un punto e interrumpirse para posteriormente continuar.

De forma similar a las calles, sucede con las colonias, las cuales podrían estar fragmentadas, incluirá nombre, localización y su área que la compone.

Con todo esto se deberá dibujar un trazado cada vez que el administrador haga una consulta de la ruta que siguió la unidad en

alguna bitácora de reparto. El trazado solo se hará de las colonias y calles y manzanas que intercepte a la ruta en un diámetro de de 1KM de radio para cada punto de la ruta. La consulta deberá mostrar además el kilometraje, el consumo promedio, que unidad (placas) hizo el recorrido que conductor hizo el recorrido y que tiempo duró el mismo.

3.2. Diagramas de flujo

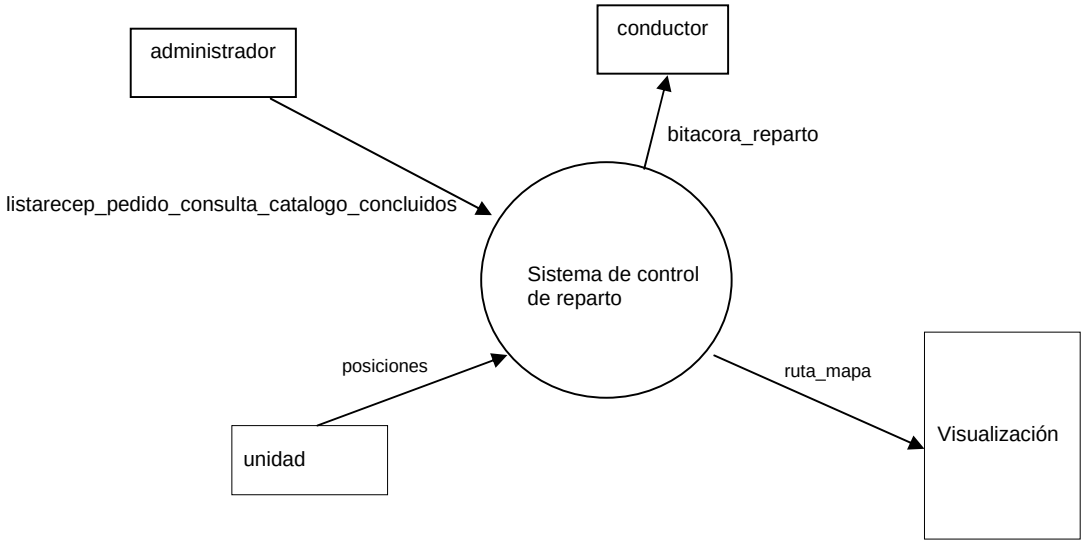


Figura 3.1 DFD contextual del sistema de rutas

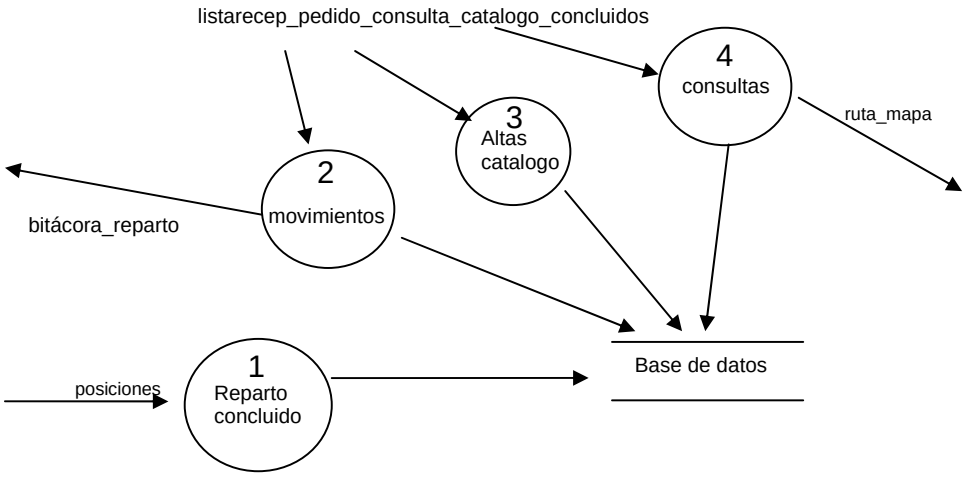


Figura 3.2 Sistema de control de rutas

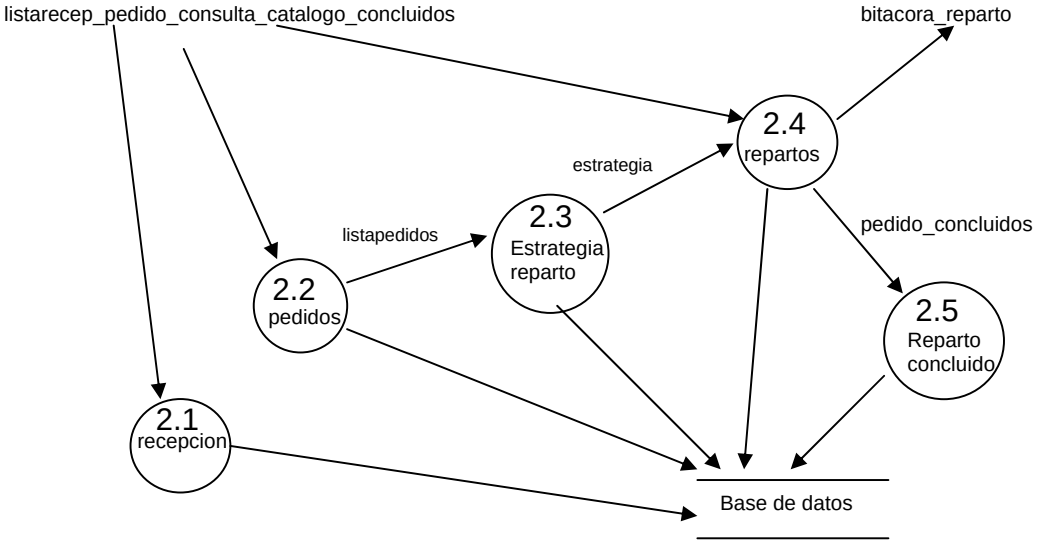


Figura 3.3 Movimientos

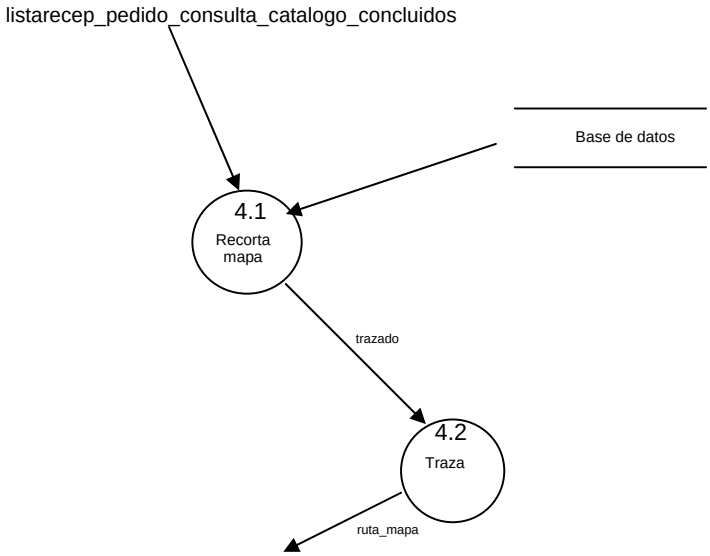


Figura 3.4 Consulta

3.3. Diccionario de datos

listarecep_pedido_consulta_catalogo_concluidos=[{catalogo} | {pedido} | {listarecep} | {consulta} | {concluidos}]

catalogo	=	[{unidad} | {conductor} | {cliente} | {producto}]
unidad	=	llave+modelo+placas+marca+consumo
conductor	=	nombre+apellidos+licencia+comentarios+fechanac+telefono+unidadasig
cliente	=	nocliente+rfc+email+telefono+nombre+apellidos+direccion
producto	=	codigo+descripcion+precio+cantidad
pedido	=	1{producto}N+nocliente+fecha+folio_pedido
listarecep	=	1{producto}N+fecha+folio_recep
consulta	=	folio_reparto
concluidos	=	1{pedido}N
bitácora_reparto	=	1{pedido}N
estrategia	=	bitácora_reparto
pedido_concluidos	=	1{pedido}N
posiciones	=	1{posicion}N
ruta_mapa	=	ruta+mapa
ruta	=	1{posiciones}N
mapaRecorte	=	{calle}+ {colonia}+{municipio}
calle	=	{poliline}+{nombre}
pililine	=	{punto}
punto	=	x+','+y
x	=	numero
y	=	numero
colonia	=	{poligono}+{nombre}
poligono	=	{punto}
punto	=	x+','+y
x	=	numero
y	=	numero
municipio	=	{poligono}+{nombre}
poligono	=	{punto}
punto	=	x+','+y

x	=	numero
y	=	numero
posición	=	{ llave} +{latitud}+{longitud}+{velocidad}+{fecha}+{hora}
latitud	=	[0-360]
longitud	=	[0-360]
velocidad	=	1{digito}5+'.'+1{digito}2
folio_recep	=	numero
folio_pedido	=	numero
codigo	=	4{alfanum}4
descripción	=	{alfanum}
precio	=	*es un real*
cantidad	=	numero
nocliente	=	numero
rfc	=	4{alfanum}4+2{año}2+2{mes}2+2{dia}2+"-"+3{alfanúm}3
email	=	1{ alfanum }20+@+1{alfanum}20+1{.1{ alfanum }20}3
direccion	=	{alfanum}
nombre	=	['A'-'Z'|'a'-'z'|'_'| '0'-'9' | ESPACIOENBLANCO]
apellidos	=	['A'-'Z'|'a'-'z'|'_'| '0'-'9' | ESPACIOENBLANCO]
licencia	=	{alfanum}
comentarios	=	{alfanum}
fechanac	=	2{dia}2+2{mes}2+4{año}4
telefono	=	1{lada}3+1{numero}
unidadasig	=	placas
modelo	=	numero
placas	=	{alfanum}
marca	=	{alfanum}
consumo	=	*es un real*
llave	=	20{alfanum}20
alfanum	=	['A'-'Z'|'a'-'z'|'_'| '0'-'9']
lada	=	numero
numero	=	[0-9]
dia	=	[0-9]
mes	=	[0-9]
año	=	[0-9]
hora	=	2{hr}2+2{min}2+2{seg}2
hr	=	[0-9]
min	=	[0-9]
seg	=	[0-9]

Capítulo 4

Diseño de la Base de Datos

4.1. Introducción

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico. [7]

4.2. Modelo de datos

Un modelo de datos es una serie de conceptos que puede utilizarse para describir un conjunto de datos y las operaciones para manipularlos. Hay dos tipos de modelos de datos: los modelos conceptuales y los modelos lógicos. Los modelos conceptuales se utilizan para representar la realidad a un alto nivel de abstracción. Mediante los modelos conceptuales se puede construir una descripción de la realidad fácil de entender. En los modelos lógicos, las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos.

En general, un modelo no es capaz de expresar todas las propiedades de una realidad determinada, por lo que hay que añadir aserciones que complementen el esquema.[7]

4.3. Diagrama de Entidad-Relación

Denominado por sus siglas como: E-R; Este modelo representa a la realidad a través de un esquema gráfico empleando los terminología de **entidades**, que son objetos que existen y son los elementos

principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas **atributos**, el enlace que rige la unión de las entidades esta representada por la **relación** del modelo.

Recordemos que un rectángulo nos representa a las entidades; una elipse a los atributos de las entidades, y una etiqueta dentro de un rombo nos indica la relación que existe entre las entidades, destacando con líneas las uniones de estas y que la llave primaria de una entidad es aquel atributo que se encuentra subrayado.[2]

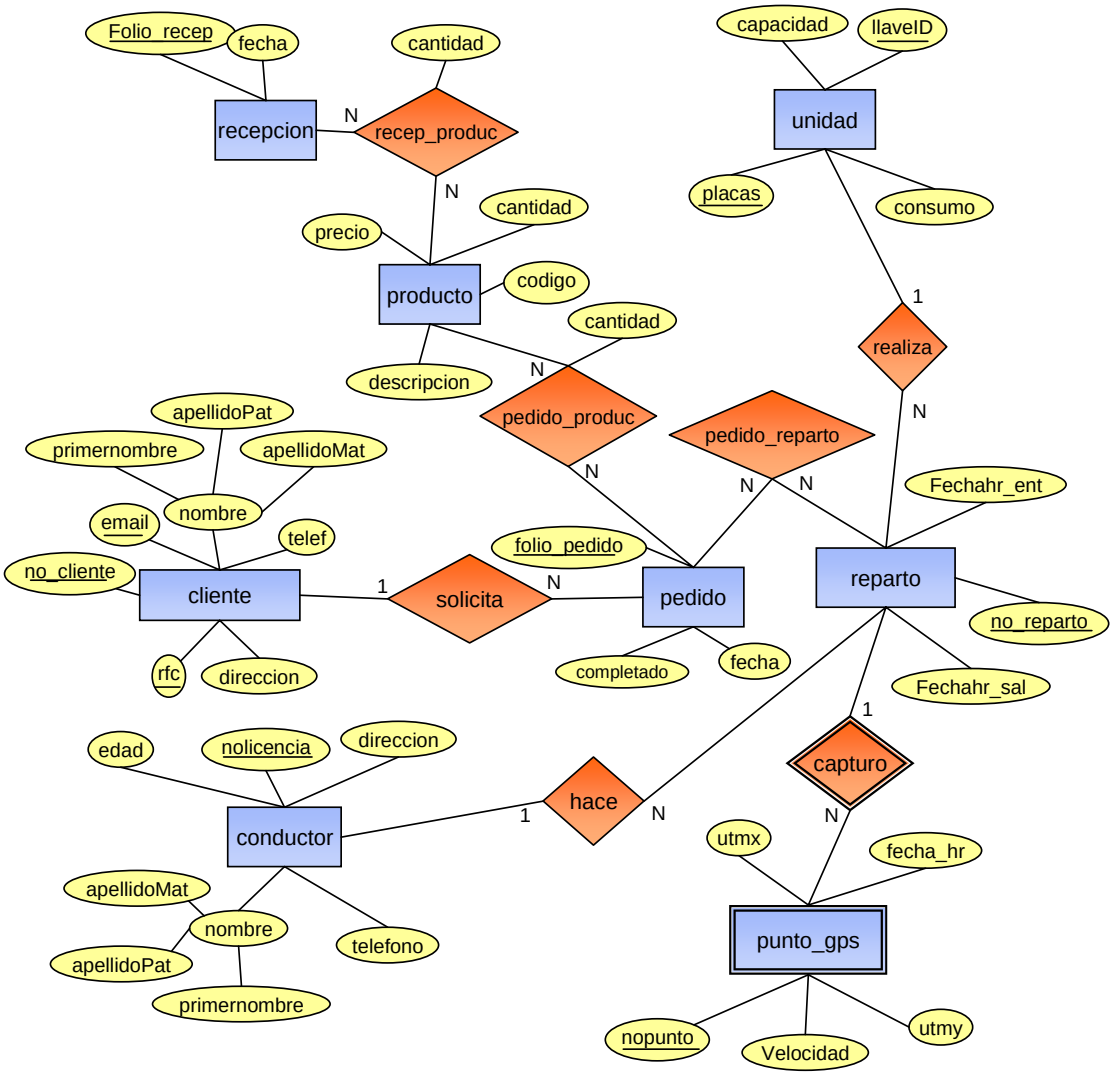


Figura 4.1 Diagrama E-R de la base de datos

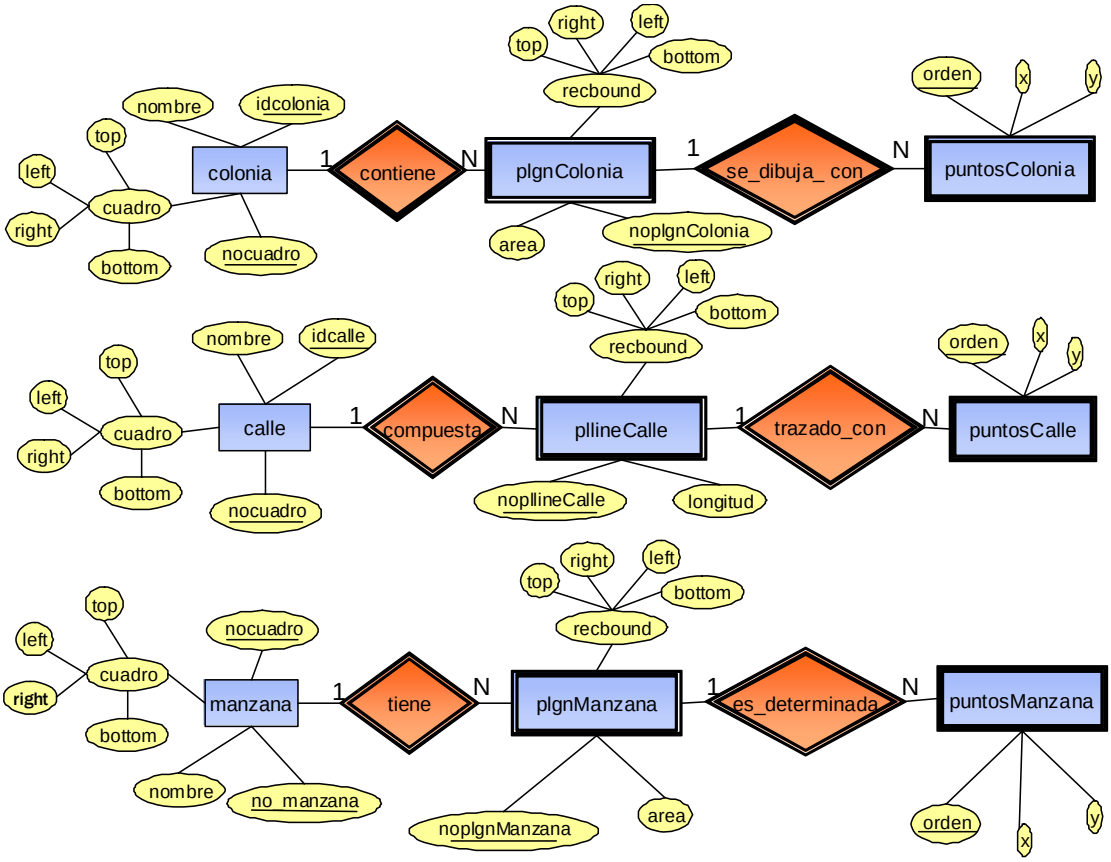


Figura 4.2 Continuación de diagrama E-R

Es importante señalar todo el diagrama se divide en lo que es el sistema y lo que concierne al trazado del mapa.

La figura 4.1 describe la forma de cómo el cliente puede realizar varios pedidos. Los pedidos están constituidos por una serie de productos.

Con lo que respecta a las recepciones, éstas cuentan con una lista de productos, muy similar a los pedidos.

Al llevar a cabo un reparto, a éste se le asigna un conductor, un conjunto de pedidos y una unidad. También se muestra como se le

asocia un conjunto de puntos GPS que son los que se descargan al término del recorrido a través del puerto serie o USB.

Con lo que respecta a la figura 4.2, muestra lo concerniente al trazado del mapa. El mapa esta partido en cuadros de 1km cuadrado, por lo que las calles, colonias y manzanas, estas integradas en dicho cuadrado determinado por los atributos left, top, right y bottom.

Para poder tazar las colonias y las manzanas, se hace necesario saber su localización como un polígono, y cada polígono esta determinado por coordenadas (x,y).

Similarmente ocurre con las calles, la diferencia estriba en que las calles son determinadas no por polígonos, si no por de líneas conectadas denominadas polilineas.

4.4. Diseño lógico de la base de datos

El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un esquema lógico es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. Un modelo lógico es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto.

Los siguientes pasos se han de seguir para llevar a cabo la transformación de modelo ER a esquema relacional.[2]

(a) *Entidades fuertes*. Crear una relación para cada entidad fuerte que incluya todos sus atributos simples. De los atributos compuestos incluir sólo sus componentes.

Cada uno de los identificadores de la entidad será una clave candidata. De entre las claves candidatas hay que escoger la clave primaria; el resto serán claves alternativas. Para escoger la clave primaria entre las claves candidatas se pueden seguir estas indicaciones:

- Escoger la clave candidata que tenga menos atributos.
- Escoger la clave candidata cuyos valores no tengan probabilidad de cambiar en el futuro.
- Escoger la clave candidata cuyos valores no tengan probabilidad de perder la unicidad en el futuro.
- Escoger la clave candidata con el mínimo número de caracteres (si es de tipo texto).
- Escoger la clave candidata más fácil de utilizar desde el punto de vista de los usuarios.

(b) *Entidades débiles*. Crear una relación para cada entidad débil incluyendo todos sus atributos simples. De los atributos compuestos incluir sólo sus componentes. Añadir una clave ajena a la entidad de la que depende. Para ello, se incluye la clave primaria de la relación que representa a la entidad padre en la nueva relación creada para la entidad débil. A continuación, determinar la clave primaria de la nueva relación.

(c) *Relaciones binarias de uno a uno*. Para cada relación binaria se incluyen los atributos de la clave primaria de la entidad padre en la

relación (tabla) que representa a la entidad hijo, para actuar como una clave ajena. La entidad hijo es la que participa de forma total (obligatoria) en la relación, mientras que la entidad padre es la que participa de forma parcial (opcional). Si las dos entidades participan de forma total o parcial en la relación, la elección de padre e hijo es arbitraria. Además, en caso de que ambas entidades participen de forma total en la relación, se tiene la opción de integrar las dos entidades en una sola relación (tabla). Esto se suele hacer si una de las entidades no participa en ninguna otra relación.[2]

(d) *Relaciones binarias de uno a muchos*. Como en las relaciones de uno a uno, se incluyen los atributos de la clave primaria de la entidad padre en la relación (tabla) que representa a la entidad hijo, para actuar como una clave ajena. Pero ahora, la entidad padre es la de "la parte del muchos" (cada padre tiene muchos hijos), mientras que la entidad hijo es la de "la parte del uno" (cada hijo tiene un solo padre).

(e) *Relaciones binarias de muchos a muchos*. En ésta caso se debe crear una nueva tabla que contenga las claves primarias que participan en la interrelación, incluyendo en la tabla, los atributos de las interrelaciones.

(f) *Jerarquías de generalización*. En las jerarquías, se denomina entidad padre a la entidad genérica y entidades hijo a las

subentidades. Hay tres opciones distintas para representar las jerarquías. La elección de la más adecuada se hará en función de su tipo (total/parcial, exclusiva/superpuesta).[2]

1. Crear una relación por cada entidad. Las relaciones de las entidades hijo heredan como clave primaria la de la entidad padre. Por lo tanto, la clave primaria de las entidades hijo es también una clave ajena al padre. Esta opción sirve para cualquier tipo de jerarquía, total o parcial y exclusiva o superpuesta.
2. Crear una relación por cada entidad hijo, heredando los atributos de la entidad padre. Esta opción sólo sirve para jerarquías totales y exclusivas.
3. Integrar todas las entidades en una relación, incluyendo en ella los atributos de la entidad padre, los atributos de todos los hijos y un atributo discriminativo para indicar el caso al cual pertenece la entidad en consideración. Esta opción sirve para cualquier tipo de jerarquía. Si la jerarquía es superpuesta, el atributo discriminativo será multivaluado.[2]

Ahora se aplicarán todos los pasos anteriores al diagrama E-R de la figura 4.1 de la siguiente forma:
Se toman las entidades fuertes para crear relaciones, que son RECEPCION, PRODUCTO, UNIDAD, CLIENTE, PEDIDO, CONDUCTOR ,REPARTO, COLONIA, CALLE y MANZANA con las llave primarias foliorecep, codigo, placas, nocliente, foliopedido, nolicencia, noreparto, idcolonia, idcalle e no_manzana respectivamente.

Relación	Clave primaria	Atributos					
RECEPCION	foliorecep	fecha					
PRODUCTO	codigo	descripcion	cantidad	precio			
UNIDAD	placas	llaveid	capacidad	consumo	marca	modelo	
CLIENTE	nocliente	primnom	apepat	apemat	telef	email	
PEDIDO	foliopedido	fecha	completado				
CONDUCTOR	nolicencia	primnom	apepat	apemat	telef	direc	
REPARTO	noreparto	Fechahr_ent	Fechahr_sal				
COLONIA	idcolonia	nombre	left	top	right	bottom	
CALLE	idcalle	nombre	left	top	right	bottom	
MANZANA	nomanzana		left	top	right	bottom	

Las entidades débiles en éste caso son las relaciones PUNTOS_GPS, PLGNCOLONIA, PLLINECALLE, PLGNMANZANA, PUNTOSCOLONIA, PUNTOSCALLE y PUNTOSMANZANA; añadiendo las claves de las relaciones de la que dependen.

Obsérvese que PLGNCOLONIA, PLLINECALLE y PLGNMANZANA son entidades débiles, pero también son entidades fuertes, por lo que generan las siguientes relaciones:

Relacion	Clave Primaria	Clave Ajena	Atributos					
PUNTOS_GPS	nopunto	no_reparto	utmx	utmy	velocidad	Fecha_hr		
PLGNCOLONIA	noplgncolonia	idcolonia	area	left	top	right	bottom	
PLLINECALLE	nopllineccalle	idcalle	longitud	left	top	right	bottom	
PLGNMANZANA	noplgnmanzana	no_manzana	area	left	top	right	bottom	
PUNTOSCOLONIA	orden	noplgncolonia	x	y				
PUNTOSCALLE	orden	nopllineccalle	x	y				
PUNTOSMANZANA	orden	noplgnmanzana	x	y				

Como no existen relaciones uno a uno, el inciso c no aplica.

El inciso d se aplica a las relaciones uno a muchos en la que se determinan las llaves foráneas de las relaciones, por lo que el resultado al aplicar ese inciso, queda como sigue:

Relación	Clave primaria	Clave foranea	Clave foranea	Atributos					
RECEPCION	foliorecep			fecha					
PRODUCTO	codigo			descripcion	cantidad	precio			
UNIDAD	placas			llaveid	capacidad	consumo	marca	modelo	
CLIENTE	nocliente			primnom	apepat	apemat	telef	email	
PEDIDO	foliopedido	nocliente		fecha	completado				
CONDUCTOR	nolicencia			primnom	apepat	apemat	telef	direc	
REPARTO	noreparto	placas	nolicencia	fechahrini	fechahrfin				
COLONIA	idcolonia			nombre	left	top	right	bottom	
CALLE	idcalle			nombre	left	top	right	bottom	
MANZANA	idmanzana			numero	left	top	right	bottom	
PUNTOS_GPS	nopunto	noreparto		no_reparto	utmx	utmy	velocidad	Fecha_hr	
PLGNCOLONIA	noplgncolonia	idcolonia			area	left	top	right	bottom
PLLINECALLE	nopllineccalle	idcalle			longitud	left	top	right	bottom
PLGNMANZANA	noplgnmanzana	no_manzana			area	left	top	right	bottom
PUNTOSCOLONIA	orden	noplgncolonia	idcolonia		x	y			
PUNTOSCALLE	orden	nopllineccalle	idcalle		x	y			
PUNTOSMANZANA	orden	noplgnmanzana	nomanzana		x	y			

En seguida se aplica el inciso (e) que tiene que ver con las relaciones de muchos a muchos, en la que se deben crear tablas para las relaciones RECEP_PRODUC, PEDIDO_PRODUC, PEDIDO_REPARTO; quedando como sigue:

Relación	Clave foránea	Clave foránea	Atributo(s)
RECEP_PRODUC	foliorecep	codigo	cantidad
PEDIDO_PRODUC	Foliopedido	codigo	cantidad
PEDIDO_REPARTO	Foliopedido	noreparto	

4.5. Diseño físico de la base de datos.

El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un esquema físico es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos. [2]

Capítulo 5

Implementación

5.1. Descripción del Software.

Para desarrollar el sistema, se usó Visual Foxpro, debido a la facilidad de desarrollar sistemas de base de datos.

Se usaron en su mayoría vistas locales y parametrizadas en las operaciones de consultas, altas, bajas y modificaciones.

Con lo que respecta al trazado del mapa, se implementó un control ActiveX en Visual C++ .Net y éste fue incrustado en una Form de Visual Foxpro.

En la dependencia de catastro de la ciudad de puebla, se consiguieron los archivos que contienen el trazado del mapa. Las dimensiones del mapa son de aproximadamente 25 km^2 ubicada cerca de Ciudad Universitaria.

5.2. Uso de Visual Foxpro.

Microsoft Visual FoxPro 9.0 es una herramienta extremadamente poderosa que permite construir rápidamente aplicaciones y componentes de base de datos. Centrado en la data y orientado a objetos, su lenguaje soporta un robusto set de herramientas que permiten construir aplicaciones de bases de datos para computadoras de escritorio, servidores, o servicios Web basados en XML y servicios Web convencionales.

Como Visual Foxpro soporta el Lenguaje de Definición de Datos (LDD), la creación de tablas se realizó con sentencias SQL de la siguiente forma:

```
CREATE TABLE unidad (;
  placas C(10) NOT NULL PRIMARY KEY,;
  llaveid C(10) NULL,;
```

```
  capacidad N(6,2) NOT NULL DEFAULT 0.0,;
  consumo N(6,2) NOT NULL DEFAULT 0.0,;
  marca C(45) NULL,;
  modelo I NOT NULL DEFAULT 1900)

CREATE TABLE producto (;
  codigo C(10) NOT NULL PRIMARY KEY,;
  descripcion C(45) NOT NULL,;
  precio N(6,2) NOT NULL DEFAULT 0.0,;
  cantidad I NOT NULL DEFAULT 0)

CREATE TABLE recepcion (;
  foliorecep I NOT NULL AUTOINC PRIMARY KEY,;
  fecha DATE NOT NULL)

CREATE TABLE conductor (;
  licencia C(20) NOT NULL PRIMARY KEY,;
  primnom C(45) NOT NULL,;
  apemat C(45) NOT NULL,;
  apepat C(45) NOT NULL,;
  telef C(20) NULL,;
  direccion C(60) NOT NULL)

CREATE TABLE cliente (;
  nocliente I NOT NULL AUTOINC PRIMARY KEY,;
  primnom C(45) NOT NULL,;
  apepat C(45) NOT NULL,;
  apemat C(45) NOT NULL,;
  telef C(45) NULL,;
  email C(45) NULL)

CREATE TABLE pedido (;
  foliopedid I NOT NULL AUTOINC PRIMARY KEY,;
  nocliente I NOT NULL,;
  fecha DATE NOT NULL,;
  completado INT NOT NULL DEFAULT 0,;
  FOREIGN KEY nocliente TAG nocliente ;
    REFERENCES cliente ;
```

```
)

CREATE TABLE reparto (;
  noreparto I NOT NULL AUTOINC PRIMARY KEY,;
  licencia C(20) NOT NULL,;
  placas C(10) NOT NULL,;
  fechahrini T ,;
  fechahrfin T ,;
  FOREIGN KEY placas TAG placas ;
    REFERENCES unidad ,;
  FOREIGN KEY licencia TAG licencia;
    REFERENCES conductor ;
)

CREATE TABLE recep_prod (;
  orden I NOT NULL AUTOINC PRIMARY KEY,;
  codigo C(10) NOT NULL,;
  folioreceP I NOT NULL,;
  cantidad I NOT NULL DEFAULT 0,;
  FOREIGN KEY folioreceP TAG folioreceP ;
    REFERENCES recepcion, ;
  FOREIGN KEY codigo TAG codigo ;
    REFERENCES producto ;
)


CREATE TABLE pedido_produc (;
  orden I NOT NULL AUTOINC PRIMARY KEY,;
  codigo C(10) NOT NULL,;
  foliopedid I NOT NULL,;
  cantidad I NOT NULL DEFAULT 0,;
  FOREIGN KEY foliopedid TAG foliopedid ;
    REFERENCES pedido, ;
  FOREIGN KEY codigo TAG codigo ;
    REFERENCES producto ;
)
```

```
CREATE TABLE pedido_reparto (;
  orden I NOT NULL AUTOINC PRIMARY KEY,;
  foliopedid I NOT NULL,;
  noreparto I NOT NULL,;
  FOREIGN KEY noreparto TAG noreparto ;
    REFERENCES reparto,;
  FOREIGN KEY foliopedid TAG foliopedid ;
    REFERENCES pedido;
)
```

Y las vistas se crearon con las siguientes instrucciones propias de foxpro:

```
CREATE VIEW viewlistcliente AS SELECT Cliente.nocliente,
Cliente.primnom, Cliente.apepat,;
  Cliente.apemat, Cliente.telef, Cliente.email, Cliente.direc;
 FROM ;
     rutas!cliente

DBSetProp(“viewlistcliente”,"View","SendUpdates",.F.)
DBSetProp(“viewlistcliente”,"View","BatchUpdateCount",1)
DBSetProp(“viewlistcliente”,"View","CompareMemo",.T.)
DBSetProp(“viewlistcliente”,"View","FetchAsNeeded",.F.)
DBSetProp(“viewlistcliente”,"View","FetchMemo",.T.)
DBSetProp(“viewlistcliente”,"View","FetchSize",100)
DBSetProp(“viewlistcliente”,"View","MaxRecords",-1)
DBSetProp(“viewlistcliente”,"View","Prepared",.F.)
DBSetProp(“viewlistcliente”,"View","UpdateType",1)
DBSetProp(“viewlistcliente”,"View","UseMemoSize",255)
DBSetProp(“viewlistcliente”,"View","WhereType",3)

DBSetProp(“viewlistcliente”+".nocliente","Field","DataType","I")
DBSetProp(“viewlistcliente”+".nocliente","Field","UpdateName","rutas!c
liente.nocliente")
DBSetProp(“viewlistcliente”+".nocliente","Field","KeyField",.T.)
DBSetProp(“viewlistcliente”+".nocliente","Field","Updatable",.F.)

DBSetProp(“viewlistcliente”+".primnom","Field","DataType","C(45)")
DBSetProp(“viewlistcliente”+".primnom","Field","UpdateName","rutas!cli
ente.primnom")
DBSetProp(“viewlistcliente”+".primnom","Field","KeyField",.F.)
DBSetProp(“viewlistcliente”+".primnom","Field","Updatable",.F.)

DBSetProp(“viewlistcliente”+".apepat","Field","DataType","C(45)")
DBSetProp(“viewlistcliente”+".apepat","Field","UpdateName","rutas!clie
nte.apepat")
DBSetProp(“viewlistcliente”+".apepat","Field","KeyField",.F.)
DBSetProp(“viewlistcliente”+".apepat","Field","Updatable",.F.)

DBSetProp(“viewlistcliente”+".apemat","Field","DataType","C(45)")
DBSetProp(“viewlistcliente”+".apemat","Field","UpdateName","rutas!clie
nte.apemat")
```

```
DBSetProp(“viewlistcliente”+".apemat","Field","KeyField",.F.)
DBSetProp(“viewlistcliente”+".apemat","Field","Updatable",.F.)

DBSetProp(“viewlistcliente”+".telef","Field","DataType","C(45)")
DBSetProp(“viewlistcliente”+".telef","Field","UpdateName","rutas!clien
te.telef")
DBSetProp(“viewlistcliente”+".telef","Field","KeyField",.F.)
DBSetProp(“viewlistcliente”+".telef","Field","Updatable",.F.)

DBSetProp(“viewlistcliente”+".email","Field","DataType","C(45)")
DBSetProp(“viewlistcliente”+".email","Field","UpdateName","rutas!clien
te.email")
DBSetProp(“viewlistcliente”+".email","Field","KeyField",.F.)
DBSetProp(“viewlistcliente”+".email","Field","Updatable",.F.)

DBSetProp(“viewlistcliente”+".direc","Field","DataType","C(100)")
DBSetProp(“viewlistcliente”+".direc","Field","UpdateName","rutas!clien
te.direc")
DBSetProp(“viewlistcliente”+".direc","Field","KeyField",.F.)
DBSetProp(“viewlistcliente”+".direc","Field","Updatable",.F.)



CREATE VIEW viewlistconductor AS SELECT Conductor.licencia,
Conductor.primnom, Conductor.apemat,;
  Conductor.apepat, Conductor.telef, Conductor.direccion;
 FROM ;
     rutas!conductor

DBSetProp(“viewlistconductor”,"View","SendUpdates",.F.)
DBSetProp(“viewlistconductor”,"View","BatchUpdateCount",1)
DBSetProp(“viewlistconductor”,"View","CompareMemo",.T.)
DBSetProp(“viewlistconductor”,"View","FetchAsNeeded",.F.)
DBSetProp(“viewlistconductor”,"View","FetchMemo",.T.)
DBSetProp(“viewlistconductor”,"View","FetchSize",100)
DBSetProp(“viewlistconductor”,"View","MaxRecords",-1)
DBSetProp(“viewlistconductor”,"View","Prepared",.F.)
DBSetProp(“viewlistconductor”,"View","UpdateType",1)
DBSetProp(“viewlistconductor”,"View","UseMemoSize",255)
DBSetProp(“viewlistconductor”,"View","WhereType",3)

DBSetProp(“viewlistconductor”+".licencia","Field","DataType","C(20)")
DBSetProp(“viewlistconductor”+".licencia","Field","UpdateName","rutas!
conductor.licencia")
DBSetProp(“viewlistconductor”+".licencia","Field","KeyField",.T.)
DBSetProp(“viewlistconductor”+".licencia","Field","Updatable",.F.)

DBSetProp(“viewlistconductor”+".primnom","Field","DataType","C(45)")
DBSetProp(“viewlistconductor”+".primnom","Field","UpdateName","rutas!c
onductor.primnom")
DBSetProp(“viewlistconductor”+".primnom","Field","KeyField",.F.)
DBSetProp(“viewlistconductor”+".primnom","Field","Updatable",.F.)

DBSetProp(“viewlistconductor”+".apemat","Field","DataType","C(45)")
DBSetProp(“viewlistconductor”+".apemat","Field","UpdateName","rutas!co
nductor.apemat")
DBSetProp(“viewlistconductor”+".apemat","Field","KeyField",.F.)
DBSetProp(“viewlistconductor”+".apemat","Field","Updatable",.F.)

DBSetProp(“viewlistconductor”+".apepat","Field","DataType","C(45)")
```

```
DBSetProp(“viewlistconductor”+".apepat","Field","UpdateName","rutas!co
nductor.apepat")
DBSetProp(“viewlistconductor”+".apepat","Field","KeyField",.F.)
DBSetProp(“viewlistconductor”+".apepat","Field","Updatable",.F.)

DBSetProp(“viewlistconductor”+".telef","Field","DataType","C(20)")
DBSetProp(“viewlistconductor”+".telef","Field","UpdateName","rutas!con
ductor.telef")
DBSetProp(“viewlistconductor”+".telef","Field","KeyField",.F.)
DBSetProp(“viewlistconductor”+".telef","Field","Updatable",.F.)

DBSetProp(“viewlistconductor”+".direccion","Field","DataType","C(60)")
DBSetProp(“viewlistconductor”+".direccion","Field","UpdateName","rutas
!conductor.direccion")
DBSetProp(“viewlistconductor”+".direccion","Field","KeyField",.F.)
DBSetProp(“viewlistconductor”+".direccion","Field","Updatable",.F.)



CREATE VIEW viewlistproducto AS SELECT Producto.codigo,
Producto.descripcion, Producto.precio,;
  Producto.cantidad;
 FROM ;
     rutas!producto

DBSetProp(“viewlistproducto”,"View","SendUpdates",.F.)
DBSetProp(“viewlistproducto”,"View","BatchUpdateCount",1)
DBSetProp(“viewlistproducto”,"View","CompareMemo",.T.)
DBSetProp(“viewlistproducto”,"View","FetchAsNeeded",.F.)
DBSetProp(“viewlistproducto”,"View","FetchMemo",.T.)
DBSetProp(“viewlistproducto”,"View","FetchSize",100)
DBSetProp(“viewlistproducto”,"View","MaxRecords",-1)
DBSetProp(“viewlistproducto”,"View","Prepared",.F.)
DBSetProp(“viewlistproducto”,"View","UpdateType",1)
DBSetProp(“viewlistproducto”,"View","UseMemoSize",255)
DBSetProp(“viewlistproducto”,"View","WhereType",3)

DBSetProp(“viewlistproducto”+".codigo","Field","DataType","C(10)")
DBSetProp(“viewlistproducto”+".codigo","Field","UpdateName","rutas!pro
ducto.codigo")
DBSetProp(“viewlistproducto”+".codigo","Field","KeyField",.T.)
DBSetProp(“viewlistproducto”+".codigo","Field","Updatable",.F.)

DBSetProp(“viewlistproducto”+".descripcion","Field","DataType","C(45)"
)
DBSetProp(“viewlistproducto”+".descripcion","Field","UpdateName","ruta
s!producto.descripcion")
DBSetProp(“viewlistproducto”+".descripcion","Field","KeyField",.F.)
DBSetProp(“viewlistproducto”+".descripcion","Field","Updatable",.F.)

DBSetProp(“viewlistproducto”+".precio","Field","DataType","N(6,2)")
DBSetProp(“viewlistproducto”+".precio","Field","UpdateName","rutas!pro
ducto.precio")
DBSetProp(“viewlistproducto”+".precio","Field","KeyField",.F.)
DBSetProp(“viewlistproducto”+".precio","Field","Updatable",.F.)

DBSetProp(“viewlistproducto”+".cantidad","Field","DataType","I")
DBSetProp(“viewlistproducto”+".cantidad","Field","UpdateName","rutas!p
roducto.cantidad")
DBSetProp(“viewlistproducto”+".cantidad","Field","KeyField",.F.)
```

```
DBSetProp(“viewlistproducto”+".cantidad","Field","Updatable",.F.)
```

```
CREATE VIEW viewlistunidad AS SELECT Unidad.placas, Unidad.llaveid,
Unidad.capacidad, Unidad.consumo,;
  Unidad.marca, Unidad.modelo;
 FROM ;
     rutas!unidad

DBSetProp(“viewlistunidad”,"View","SendUpdates",.F.)
DBSetProp(“viewlistunidad”,"View","BatchUpdateCount",1)
DBSetProp(“viewlistunidad”,"View","CompareMemo",.T.)
DBSetProp(“viewlistunidad”,"View","FetchAsNeeded",.F.)
DBSetProp(“viewlistunidad”,"View","FetchMemo",.T.)
DBSetProp(“viewlistunidad”,"View","FetchSize",100)
DBSetProp(“viewlistunidad”,"View","MaxRecords",-1)
DBSetProp(“viewlistunidad”,"View","Prepared",.F.)
DBSetProp(“viewlistunidad”,"View","UpdateType",1)
DBSetProp(“viewlistunidad”,"View","UseMemoSize",255)
DBSetProp(“viewlistunidad”,"View","WhereType",3)

DBSetProp(“viewlistunidad”+".placas","Field","DataType","C(10)")
DBSetProp(“viewlistunidad”+".placas","Field","UpdateName","rutas!unida
d.placas")
DBSetProp(“viewlistunidad”+".placas","Field","KeyField",.T.)
DBSetProp(“viewlistunidad”+".placas","Field","Updatable",.F.)

DBSetProp(“viewlistunidad”+".llaveid","Field","DataType","C(10)")
DBSetProp(“viewlistunidad”+".llaveid","Field","UpdateName","rutas!unid
ad.llaveid")
DBSetProp(“viewlistunidad”+".llaveid","Field","KeyField",.F.)
DBSetProp(“viewlistunidad”+".llaveid","Field","Updatable",.F.)

DBSetProp(“viewlistunidad”+".capacidad","Field","DataType","N(6,2)")
DBSetProp(“viewlistunidad”+".capacidad","Field","UpdateName","rutas!un
idad.capacidad")
DBSetProp(“viewlistunidad”+".capacidad","Field","KeyField",.F.)
DBSetProp(“viewlistunidad”+".capacidad","Field","Updatable",.F.)

DBSetProp(“viewlistunidad”+".consumo","Field","DataType","N(6,2)")
DBSetProp(“viewlistunidad”+".consumo","Field","UpdateName","rutas!unid
ad.consumo")
DBSetProp(“viewlistunidad”+".consumo","Field","KeyField",.F.)
DBSetProp(“viewlistunidad”+".consumo","Field","Updatable",.F.)

DBSetProp(“viewlistunidad”+".marca","Field","DataType","C(45)")
DBSetProp(“viewlistunidad”+".marca","Field","UpdateName","rutas!unidad
.marca")
DBSetProp(“viewlistunidad”+".marca","Field","KeyField",.F.)
DBSetProp(“viewlistunidad”+".marca","Field","Updatable",.F.)

DBSetProp(“viewlistunidad”+".modelo","Field","DataType","I")
DBSetProp(“viewlistunidad”+".modelo","Field","UpdateName","rutas!unida
d.modelo")
DBSetProp(“viewlistunidad”+".modelo","Field","KeyField",.F.)
DBSetProp(“viewlistunidad”+".modelo","Field","Updatable",.F.)
```

```
CREATE VIEW viewrowconductor AS SELECT Conductor.licencia,
Conductor.primnom, Conductor.apemat,;
  Conductor.apepat, Conductor.telef, Conductor.direccion;
 FROM ;
     rutas!conductor;
 WHERE  Conductor.licencia = ( ?param_licencia )

DBSetProp(“viewrowconductor”,"View","SendUpdates",.T.)
DBSetProp(“viewrowconductor”,"View","BatchUpdateCount",1)
DBSetProp(“viewrowconductor”,"View","CompareMemo",.T.)
DBSetProp(“viewrowconductor”,"View","FetchAsNeeded",.F.)
DBSetProp(“viewrowconductor”,"View","FetchMemo",.T.)
DBSetProp(“viewrowconductor”,"View","FetchSize",100)
DBSetProp(“viewrowconductor”,"View","MaxRecords",-1)
DBSetProp(“viewrowconductor”,"View","Prepared",.F.)
DBSetProp(“viewrowconductor”,"View","UpdateType",1)
DBSetProp(“viewrowconductor”,"View","UseMemoSize",255)
DBSetProp(“viewrowconductor”,"View","Tables","rutas!conductor")
DBSetProp(“viewrowconductor”,"View","WhereType",3)

DBSetProp(“viewrowconductor”+".licencia","Field","DataType","C(20)")
DBSetProp(“viewrowconductor”+".licencia","Field","UpdateName","rutas!c
onductor.licencia")
DBSetProp(“viewrowconductor”+".licencia","Field","KeyField",.T.)
DBSetProp(“viewrowconductor”+".licencia","Field","Updatable",.F.)

DBSetProp(“viewrowconductor”+".primnom","Field","DataType","C(45)")
DBSetProp(“viewrowconductor”+".primnom","Field","UpdateName","rutas!co
nductor.primnom")
DBSetProp(“viewrowconductor”+".primnom","Field","KeyField",.F.)
DBSetProp(“viewrowconductor”+".primnom","Field","Updatable",.T.)

DBSetProp(“viewrowconductor”+".apemat","Field","DataType","C(45)")
DBSetProp(“viewrowconductor”+".apemat","Field","UpdateName","rutas!con
ductor.apemat")
DBSetProp(“viewrowconductor”+".apemat","Field","KeyField",.F.)
DBSetProp(“viewrowconductor”+".apemat","Field","Updatable",.T.)

DBSetProp(“viewrowconductor”+".apepat","Field","DataType","C(45)")
DBSetProp(“viewrowconductor”+".apepat","Field","UpdateName","rutas!con
ductor.apepat")
DBSetProp(“viewrowconductor”+".apepat","Field","KeyField",.F.)
DBSetProp(“viewrowconductor”+".apepat","Field","Updatable",.T.)

DBSetProp(“viewrowconductor”+".telef","Field","DataType","C(20)")
DBSetProp(“viewrowconductor”+".telef","Field","UpdateName","rutas!cond
uctor.telef")
DBSetProp(“viewrowconductor”+".telef","Field","KeyField",.F.)
DBSetProp(“viewrowconductor”+".telef","Field","Updatable",.T.)

DBSetProp(“viewrowconductor”+".direccion","Field","DataType","C(60)")
DBSetProp(“viewrowconductor”+".direccion","Field","UpdateName","rutas!
conductor.direccion")
DBSetProp(“viewrowconductor”+".direccion","Field","KeyField",.F.)
DBSetProp(“viewrowconductor”+".direccion","Field","Updatable",.T.)



CREATE VIEW viewrowcliente AS SELECT Cliente.nocliente,
Cliente.primnom, Cliente.apepat,;
```

```
  Cliente.apemat, Cliente.telef, Cliente.email, Cliente.direc;
 FROM ;
     rutas!cliente;
 WHERE  Cliente.nocliente = ( ?param_noclient )

DBSetProp(“viewrowcliente”,"View","SendUpdates",.T.)
DBSetProp(“viewrowcliente”,"View","BatchUpdateCount",1)
DBSetProp(“viewrowcliente”,"View","CompareMemo",.T.)
DBSetProp(“viewrowcliente”,"View","FetchAsNeeded",.F.)
DBSetProp(“viewrowcliente”,"View","FetchMemo",.T.)
DBSetProp(“viewrowcliente”,"View","FetchSize",100)
DBSetProp(“viewrowcliente”,"View","MaxRecords",-1)
DBSetProp(“viewrowcliente”,"View","Prepared",.F.)
DBSetProp(“viewrowcliente”,"View","UpdateType",1)
DBSetProp(“viewrowcliente”,"View","UseMemoSize",255)
DBSetProp(“viewrowcliente”,"View","Tables","rutas!cliente")
DBSetProp(“viewrowcliente”,"View","WhereType",3)

DBSetProp(“viewrowcliente”+".nocliente","Field","DataType","I")
DBSetProp(“viewrowcliente”+".nocliente","Field","UpdateName","rutas!cl
iente.nocliente")
DBSetProp(“viewrowcliente”+".nocliente","Field","KeyField",.T.)
DBSetProp(“viewrowcliente”+".nocliente","Field","Updatable",.F.)

DBSetProp(“viewrowcliente”+".primnom","Field","DataType","C(45)")
DBSetProp(“viewrowcliente”+".primnom","Field","UpdateName","rutas!clie
nte.primnom")
DBSetProp(“viewrowcliente”+".primnom","Field","KeyField",.F.)
DBSetProp(“viewrowcliente”+".primnom","Field","Updatable",.T.)

DBSetProp(“viewrowcliente”+".apepat","Field","DataType","C(45)")
DBSetProp(“viewrowcliente”+".apepat","Field","UpdateName","rutas!clien
te.apepat")
DBSetProp(“viewrowcliente”+".apepat","Field","KeyField",.F.)
DBSetProp(“viewrowcliente”+".apepat","Field","Updatable",.T.)

DBSetProp(“viewrowcliente”+".apemat","Field","DataType","C(45)")
DBSetProp(“viewrowcliente”+".apemat","Field","UpdateName","rutas!clien
te.apemat")
DBSetProp(“viewrowcliente”+".apemat","Field","KeyField",.F.)
DBSetProp(“viewrowcliente”+".apemat","Field","Updatable",.T.)

DBSetProp(“viewrowcliente”+".telef","Field","DataType","C(45)")
DBSetProp(“viewrowcliente”+".telef","Field","UpdateName","rutas!client
e.telef")
DBSetProp(“viewrowcliente”+".telef","Field","KeyField",.F.)
DBSetProp(“viewrowcliente”+".telef","Field","Updatable",.T.)

DBSetProp(“viewrowcliente”+".email","Field","DataType","C(45)")
DBSetProp(“viewrowcliente”+".email","Field","UpdateName","rutas!client
e.email")
DBSetProp(“viewrowcliente”+".email","Field","KeyField",.F.)
DBSetProp(“viewrowcliente”+".email","Field","Updatable",.T.)

DBSetProp(“viewrowcliente”+".direc","Field","DataType","C(100)")
DBSetProp(“viewrowcliente”+".direc","Field","UpdateName","rutas!client
e.direc")
DBSetProp(“viewrowcliente”+".direc","Field","KeyField",.F.)
DBSetProp(“viewrowcliente”+".direc","Field","Updatable",.T.)
```

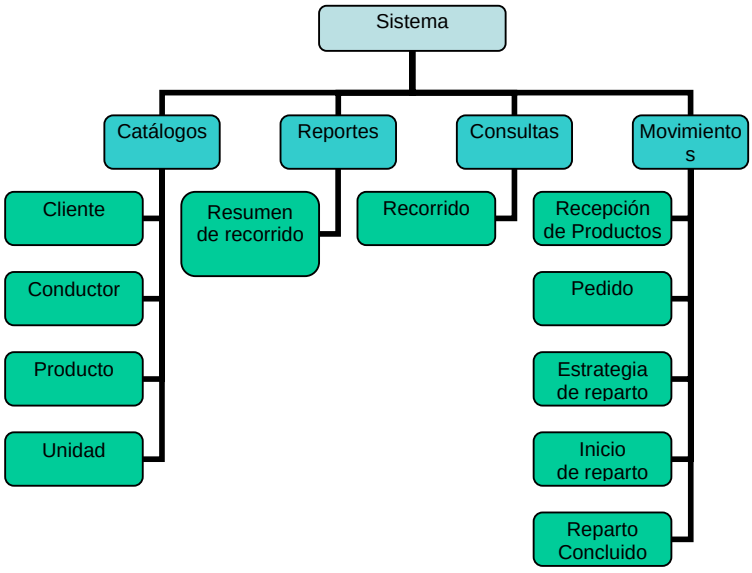
```
CREATE VIEW viewrowproducto AS SELECT Producto.codigo,
Producto.descripcion, Producto.precio,;
  Producto.cantidad;
 FROM ;
     rutas!producto;
 WHERE  Producto.codigo = ( ?param_codigo )

DBSetProp(“viewrowproducto”,"View","SendUpdates",.T.)
DBSetProp(“viewrowproducto”,"View","BatchUpdateCount",1)
DBSetProp(“viewrowproducto”,"View","CompareMemo",.T.)
DBSetProp(“viewrowproducto”,"View","FetchAsNeeded",.F.)
DBSetProp(“viewrowproducto”,"View","FetchMemo",.T.)
DBSetProp(“viewrowproducto”,"View","FetchSize",100)
DBSetProp(“viewrowproducto”,"View","MaxRecords",-1)
DBSetProp(“viewrowproducto”,"View","Prepared",.F.)
DBSetProp(“viewrowproducto”,"View","UpdateType",1)
DBSetProp(“viewrowproducto”,"View","UseMemoSize",255)
DBSetProp(“viewrowproducto”,"View","Tables","rutas!producto")
DBSetProp(“viewrowproducto”,"View","WhereType",3)

DBSetProp(“viewrowproducto”+".codigo","Field","DataType","C(10)")
DBSetProp(“viewrowproducto”+".codigo","Field","UpdateName","rutas!prod
ucto.codigo")
DBSetProp(“viewrowproducto”+".codigo","Field","KeyField",.T.)
DBSetProp(“viewrowproducto”+".codigo","Field","Updatable",.F.)

DBSetProp(“viewrowproducto”+".descripcion","Field","DataType","C(45)")
DBSetProp(“viewrowproducto”+".descripcion","Field","UpdateName","rutas
!producto.descripcion")
DBSetProp(“viewrowproducto”+".descripcion","Field","KeyField",.F.)
DBSetProp(“viewrowproducto”+".descripcion","Field","Updatable",.T.)

DBSetProp(“viewrowproducto”+".precio","Field","DataType","N(6,2)")
DBSetProp(“viewrowproducto”+".precio","Field","UpdateName","rutas!prod
ucto.precio")
DBSetProp(“viewrowproducto”+".precio","Field","KeyField",.F.)
DBSetProp(“viewrowproducto”+".precio","Field","Updatable",.T.)

DBSetProp(“viewrowproducto”+".cantidad","Field","DataType","I")
DBSetProp(“viewrowproducto”+".cantidad","Field","UpdateName","rutas!pr
oducto.cantidad")
DBSetProp(“viewrowproducto”+".cantidad","Field","KeyField",.F.)
DBSetProp(“viewrowproducto”+".cantidad","Field","Updatable",.T.)



CREATE VIEW viewrowunidad AS SELECT Unidad.placas, Unidad.llaveid,
Unidad.capacidad, Unidad.consumo,;
  Unidad.marca, Unidad.modelo;
 FROM ;
     rutas!unidad;
 WHERE  Unidad.placas = ( ?param_placas )

DBSetProp(“viewrowunidad”,"View","SendUpdates",.T.)
DBSetProp(“viewrowunidad”,"View","BatchUpdateCount",1)
DBSetProp(“viewrowunidad”,"View","CompareMemo",.T.)
DBSetProp(“viewrowunidad”,"View","FetchAsNeeded",.F.)
DBSetProp(“viewrowunidad”,"View","FetchMemo",.T.)
DBSetProp(“viewrowunidad”,"View","FetchSize",100)
```

```
DBSetProp(“viewrowunidad”,"View","MaxRecords",-1)
DBSetProp(“viewrowunidad”,"View","Prepared",.F.)
DBSetProp(“viewrowunidad”,"View","UpdateType",1)
DBSetProp(“viewrowunidad”,"View","UseMemoSize",255)
DBSetProp(“viewrowunidad”,"View","Tables","rutas!unidad")
DBSetProp(“viewrowunidad”,"View","WhereType",3)

DBSetProp(“viewrowunidad”+".placas","Field","DataType","C(10)")
DBSetProp(“viewrowunidad”+".placas","Field","UpdateName","rutas!unidad
.placas")
DBSetProp(“viewrowunidad”+".placas","Field","KeyField",.T.)
DBSetProp(“viewrowunidad”+".placas","Field","Updatable",.F.)

DBSetProp(“viewrowunidad”+".llaveid","Field","DataType","C(10)")
DBSetProp(“viewrowunidad”+".llaveid","Field","UpdateName","rutas!unida
d.llaveid")
DBSetProp(“viewrowunidad”+".llaveid","Field","KeyField",.F.)
DBSetProp(“viewrowunidad”+".llaveid","Field","Updatable",.T.)

DBSetProp(“viewrowunidad”+".capacidad","Field","DataType","N(6,2)")
DBSetProp(“viewrowunidad”+".capacidad","Field","UpdateName","rutas!uni
dad.capacidad")
DBSetProp(“viewrowunidad”+".capacidad","Field","KeyField",.F.)
DBSetProp(“viewrowunidad”+".capacidad","Field","Updatable",.T.)

DBSetProp(“viewrowunidad”+".consumo","Field","DataType","N(6,2)")
DBSetProp(“viewrowunidad”+".consumo","Field","UpdateName","rutas!unida
d.consumo")
DBSetProp(“viewrowunidad”+".consumo","Field","KeyField",.F.)
DBSetProp(“viewrowunidad”+".consumo","Field","Updatable",.T.)

DBSetProp(“viewrowunidad”+".marca","Field","DataType","C(45)")
DBSetProp(“viewrowunidad”+".marca","Field","UpdateName","rutas!unidad.
marca")
DBSetProp(“viewrowunidad”+".marca","Field","KeyField",.F.)
DBSetProp(“viewrowunidad”+".marca","Field","Updatable",.T.)

DBSetProp(“viewrowunidad”+".modelo","Field","DataType","I")
DBSetProp(“viewrowunidad”+".modelo","Field","UpdateName","rutas!unidad
.modelo")
DBSetProp(“viewrowunidad”+".modelo","Field","KeyField",.F.)
DBSetProp(“viewrowunidad”+".modelo","Field","Updatable",.T.)
```

En gran parte de la implementación se usaron vistas parametrizadas para las operaciones de eliminar, actualizar y modificar. Para listar los diferentes catálogos se usaron las vistas de solo lectura; de tal forma que para consultar un catálogo, se lista primero para que el usuario pueda seleccionar alguno, después de haber seleccionado alguno, se toma el id del catálogo y se asigna ese di al parámetro de la vistas para en seguida realizar un requery() de la misma; y como la vista esta ligada a los controles de texto, no es necesario hacer mas

operación que un refresh() de la ventana para que presenta los datos de la consulta en los textos de la ventana.

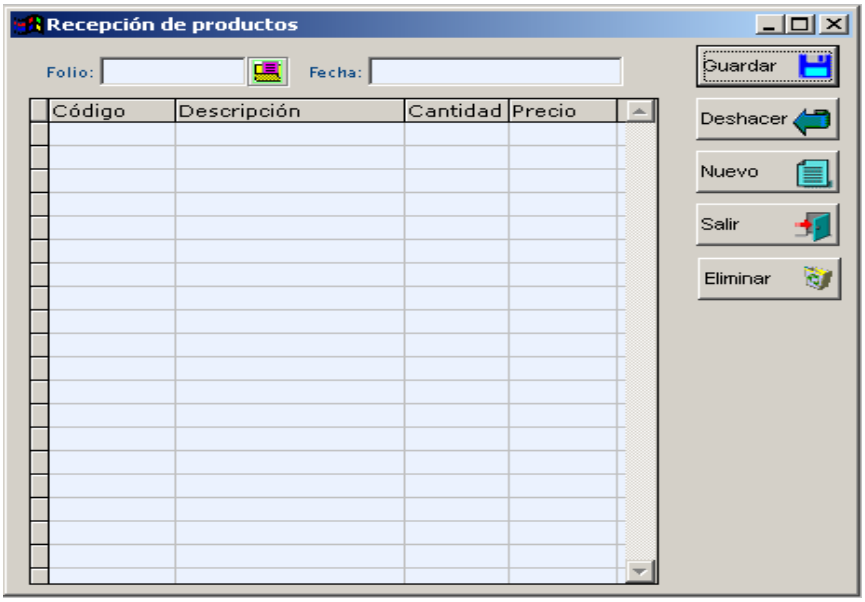
5.3. Descripción de Modulos del software.



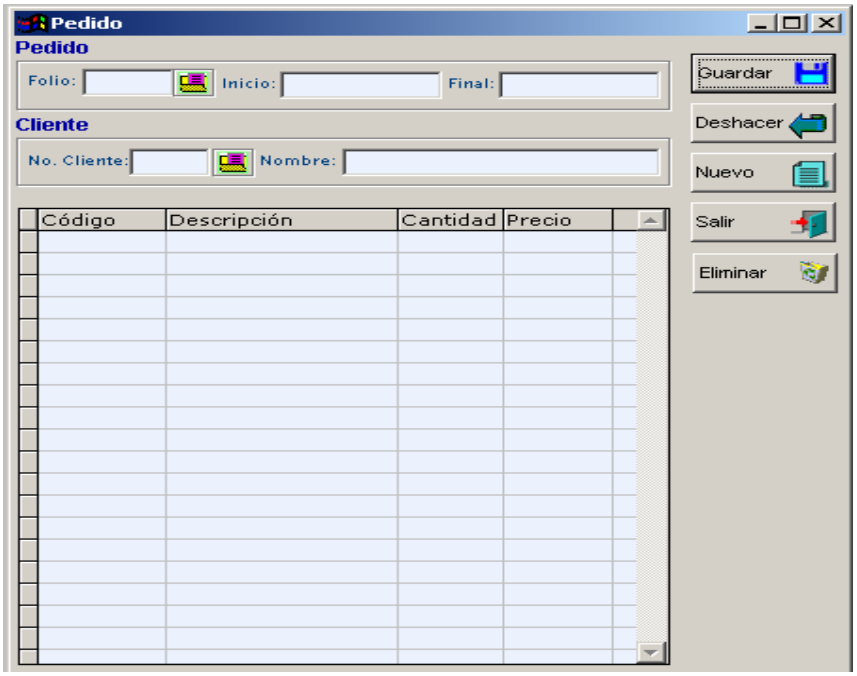
5.4. Descripción de las interfaces de usuario.

En la figura se muestra la ventana de catálogos de clientes, conductores, productos y unidades. Aquí es donde se podrán dar de alta, consultar, eliminar y modificar dichos catálogos.

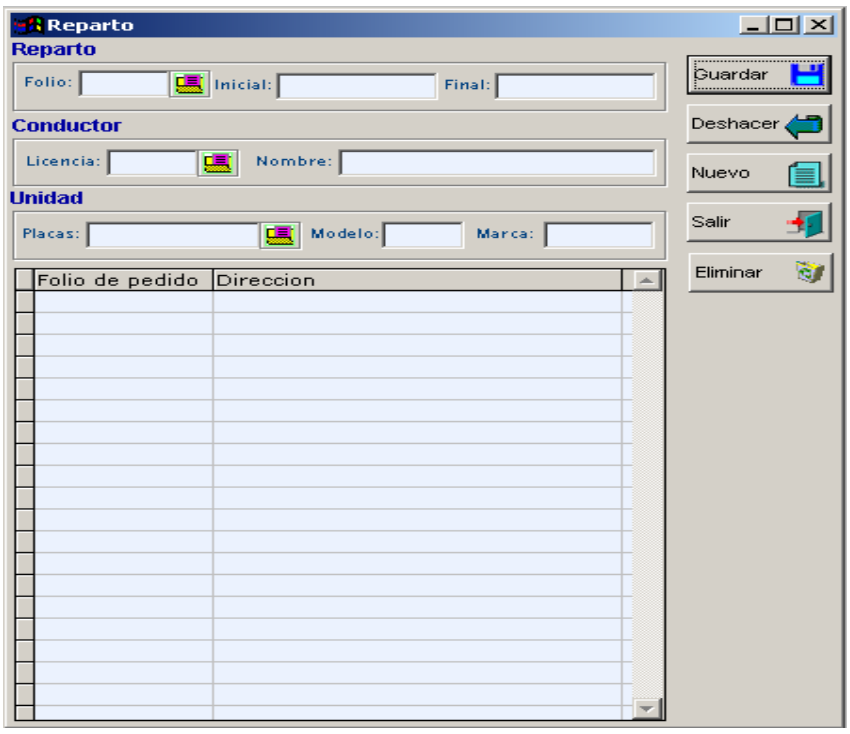
En la figura siguiente se muestra la ventana en donde se captura una recepción de productos diversos, que al efectuarse dicha operación, modificará la existencia de dichos productos de una manera de aumento en la existencia.



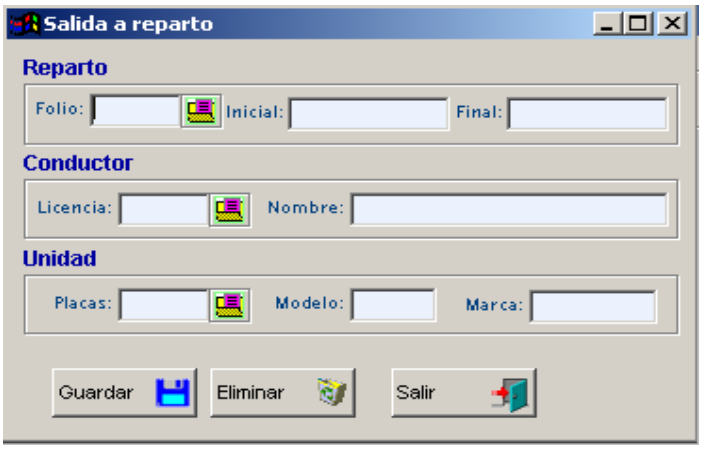
En seguida se muestra la ventana donde se capturan los diferentes pedidos de productos de los clientes.



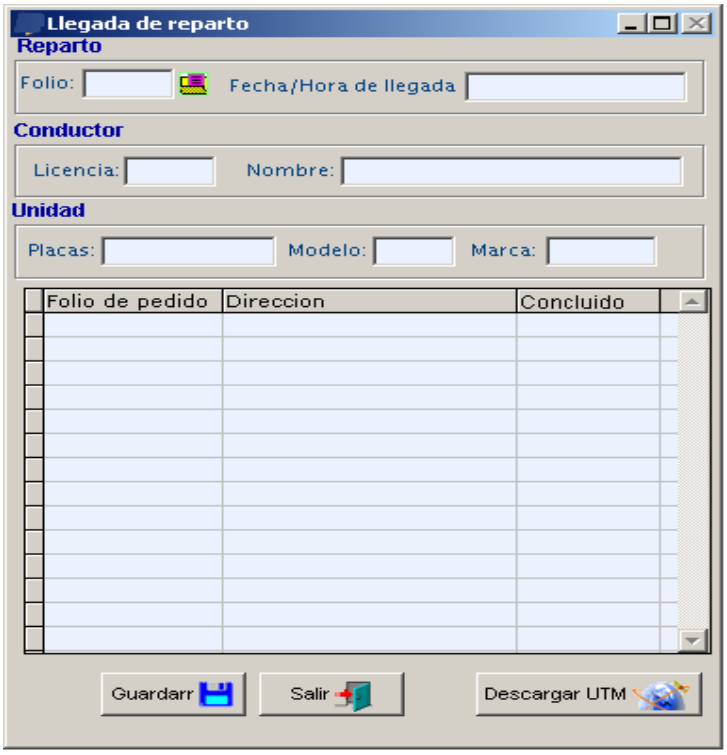
La operación de cómo se llevará acabo el reparto de tal forma que los usuarios son los responsables de que los pedidos se concentren un una zona determinada, se lleva acabo en la siguiente ventana.



Después de determinar la forma en que se van a llevar a cabo la entrega de pedidos, es decir la estrategia de reparto, se puede dar salida a la unidad y esto lo permite la siguiente ventana.



La parte donde se descargan los datos provenientes del módulo montado en la unidad móvil y el registro de los lugares donde la entrega no se llevo o no a cabo, se muestra en la ventana siguiente:



La ventana que consulta la trayectoria que siguió la unidad durante el recorrido, se muestra en seguida.

CONCLUSIONES Y PERSPECTIVAS

En gran parte de toda la labor de éste sistema, es importante considerar lo complicado que es obtener un mapa cartográfico actualizado de la ciudad de Puebla. Conforme se fue indagando en la forma de cómo obtenerlo, se torno el contrincante a vencer, por una parte el costo que esto implicaba; y por otra, la forma de cómo organizar la información en la base de datos, para que fuera lo suficientemente eficiente.

Respecto a la forma de obtener el mapa, existen empresas que cuentan con bases de datos cartográficas, que se dedican a rentar, por otra lado, las dependencias de gobierno cuentan con información de ese tipo; por lo que se optó por la segunda opción y se acudió al departamento de catastro de la Ciudad de Puebla, para después adquirir una pequeña parte de la mancha urbana.

Haciendo investigación de el mejor DBMS para almacenar información cartográfica, se verificó que existen pocos DBMS que la soportan, algunos son: Oracle y PostgreSQL. Tomando en consideración la flexibilidad para instalar el sistema se tomó la decisión de resolver la forma de de almacenarlo en tablas de foxpro con un visión relacional.

La idea de que la base de datos se organizara en calles, colonia y manzanas surgió por la forma en que arcamap organiza la información por capas en los archivos shp.

Finalmente con lo que respecta a la implementación, se utilizaron vistas, que es la forma de hacer aplicaciones muy rápido.

Con lo que respecta al graficado del mapa, se implemento un activeX hecho en Visual C++, para que pueda ser incrustado en cualquier herramienta de desarrollo que soporte controles ActiveX.

Con esto se ha tenido panorama general sobre la dificultad de plasmar la realidad en una base de datos , aunque como Oracle y PostgreSQL son herramientas para base de datos geografías, falta mucho por investigar en este campo. La captura de datos espaciales, es una tarea ardua y laboriosa, que involucra mucha gente, razón por la cual, la obtención de información espacial es costosa.

Futuras Investigaciones

El sistema es solo una pequeña muestra del reto que implica las consultas cartográficas alrededor de un punto; enfocada a la visualización de trayectorias seguidas por una unidad móvil a través de colonias y calles más importantes de la ciudad de Puebla.

Aunque la captura de posiciones globales de la unidad se van almacenando, es posible transmitirlas a una centra con tecnología SMS, la cual expandiría el sistema de una simple consulta de trayectorias, a un sistema que localiza la unidad móvil en tiempo real.

Por otra parte, se podría llevar el sistema a un ambiente web, en la cual se consultaría en cualquier parte donde haya acceso a Internet, la posición en tiempo real de una unidad móvil.

De todo lo anterior podríamos hablar de un sistema de Rastreo Móvil en tiempo real. Esto se podría aplicar para localizar unidades móviles robadas, Sistemas de seguridad, Sistemas de Paquetería, Sistema envíos y distribución de mercancías etc. A todo ello que aplique los puntos que son:
Monitoreo de trayectorias.
Monitoreo de Velocidad.
Estadísticas de consumo de gas, gasolina o cualquier otro combustible.

Localización puntual en tiempo real en todo el mundo.

Estadísticas de tiempos de trayectorias seguidas por el móvil.

Sistema integrado GPS.

Referencias bibliográficas

[1] Dr. Sánchez López Abraham. Ingenieria del Software. Facultad de Ciencias de la Computación. Av. San Claudio, San Manuel Puebla, Puebla Mexico, Agosto 2005.

[2] Dra. Somodevilla García Maria Josefa. Análisis y Diseño de Base de Datos. Facultad de Ciencias de la Computación. Av. San Claudio, San Manuel Puebla, Puebla Mexico, Otoño 2005.

[3] http://es.wikipedia.org/wiki/SIG

[4] http://www.ots.ac.cr/~pcambientales/documentos/manual/SIG.pdf

[5] http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

[6] http://www.esri-es.com

[7] http://es.wikipedia.org/wiki/Base_de_datos_espacial

[8] http://www3.uji.es/~mmarques/f47/apun/node81.html