UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

“ANÁLISIS Y DISEÑO DEL SISTEMA QUE GENERA ENCUESTAS ELECTRÓNICAS VÍA INTERNET”

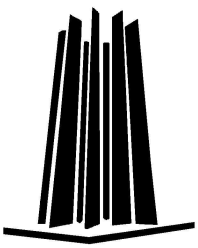
T R A B A J O E S C R I T O
EN LA MODALIDAD DE INFORME TESIS PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A N :

J O S É M A R T Í N E Z M O R E N O

F R A N C I S C O J A V I E R C O R O N A O L V E R A

ASESOR: M. EN E. IMELDA DE LA LUZ FLORES DÍAZ



MÉXICO, 2011.

AGRADECIMIENTOS

A mi hijo:

Agradezco a mi hijo por ser mi fuente de inspiración y motivo para poder salir adelante ante cualquier adversidad, el motor que me permitió llegar a esta etapa de mi vida y poder concluirla satisfactoriamente.

A mi esposa:

Agradezco a mi esposa por ser un gran apoyo en esta aventura y estar a mi lado sufriendo mis caídas y alegrándose en mis triunfos. Por darme su amor.

A mis padres:

Agradezco a mis padres por su cariño, comprensión y apoyo sin condiciones ni medida. Gracias por guiarme sobre el camino de la educación. Por haberme dado las herramientas necesarias y enseñarme los valores que fueron mi guía para forjarme como hombre de bien. Gracias por su dedicación y apoyo recibidos.

A mis hermanos y sobrinos:

Agradezco a mis hermanos y sobrinos por el cariño, apoyo y comprensión que me brindaron.

A mi abuela:

Agradezco a mi abuela Lucrecia por ser una parte importante en este logro obtenido. Ahora entiendo porque la rigidez con la que me educó. ¡Dónde quiera que te encuentres este logro es dedicado a ti!

A mi asesor de tesis y profesores:

Agradezco a todos los profesores porque con cada uno de los conocimientos que nos transmitieron y el compartir la experiencia con sus alumnos, hicieron de la enseñanza algo más que el formar ingenieros, formando hombres y mujeres preparados para enfrentar los retos que la vida diaria presenta.

A mis amigos:

A los que estuvieron conmigo y compartimos tantas aventuras, experiencias y desveladas. Gracias a cada uno por hacer que mi estancia en la Universidad fuera amena.

Gracias a todas estas personas que fueron parte importante de mi vida, que me ayudaron a conseguir este logro que no solo es mío, sino también de cada uno de ellos.

José Martínez Moreno.

A Dios.
Por darme la oportunidad de comenzar, concluir esta etapa de mi vida y poder disfrutarla plenamente.

A mis padres.
Por el amor, apoyo incondicional y moral que me han brindado a cada paso de mi vida.

A Yaretzi y a mi Esposa.
Porque cambiaron mi mundo y se volvieron el motor de mi existir, este esfuerzo es con amor para mi hija y mi esposa.

A Salvador Corona.
Porque aprendí de ti a caminar y no detenerme, porque siempre hay que levantarse y seguir adelante; para ti mi gran Tío y amigo.

A mi asesor
Que con paciencia supo guiar y dirigir este proyecto hasta la culminación de esta etapa tan importante de mi vida

A mis amigos.
Porque siempre estuvieron ahí y nunca me negaron su apoyo para llegar a culminar una meta más; con cariño este triunfo que es de todos.

A mis hermanos:
Edgar, Dulce y Flavio, que siempre han estado ahí cuando más lo he necesitado.
Este triunfo también es suyo.

A Adrian y familia:
Sin lugar a duda gran persona que sabe ser amigo y del cual siempre he recibido un gran apoyo y a su familia que siempre me tendieron la mano.

Francisco Javier Corona Olvera.

ÍNDICE

INTRODUCCIÓN

El objetivo principal que queremos alcanzar con la presente tesis es analizar, diseñar y realizar un ejemplo de desarrollo de una herramienta, la cual realizará encuestas electrónicas de vía Internet. Estas encuestas nos permitirán recopilar información acerca de los entrevistados/beneficiarios a nivel nacional, la cual culminará en la integración de bases de datos organizadas e información relevante que permitirá identificar tendencias, así como evaluar el impacto y los resultados de las intervenciones de programas y acciones de salud. Esta entrega será en un tiempo muy corto, siendo casi en tiempo real la entrega de estas.

Esta información se hará llegar a los investigadores, clientes y/o los profesionales de la salud, para que puedan evaluar el impacto de los programas sociales o programas de salud en el beneficiario/entrevistado, el impacto de estos en las familias del beneficiario/entrevistado. Así como también evaluar estos programas para determinar que tan eficiente y eficaces son, con la finalidad de determinar qué cambios o mejoras hacerles a estos.

Es necesario el desarrollo de esta herramienta, para que pueda hacerse una investigación y entrega de resultados mucho más rápida de lo que se hace con los métodos o procedimientos convencionales.

A partir de la recopilación de la información (Entidad federativa en turno) que actualmente se hace por medio de encuestas en papel, la entrega de estos cuestionarios es en el centro de operación (D.F.) que se hace vía paquetería. Posteriormente se realiza un proceso de validación de estos cuestionarios y al mismo tiempo se realizan los programas de captura, para finalmente capturarlos. No sin antes contar con el tiempo de impresión de los cuestionarios, así como el traslado de estos a las entidades correspondientes. De igual manera la logística del operativo en cuanto a movilizar mucha gente (entrevistadores, supervisores, coordinadores) hacia distintas entidades del país es bastante complejo.

Estos procedimientos son tardados y costosos, empero al cambiarlos con el uso de la nueva herramienta, únicamente se utilizará tiempo de programación de la encuesta. Esto ayudará a que la logística del operativo se haga más sencilla.

Pasando a otro punto importante, para la obtención de información diversa de avance del operativo, tanto para supervisores como para coordinadores e investigadores. Esta información se cataloga en porcentaje de avance, productividad y otros tipos de reportes necesarios que permitirán observar el desarrollo de avance del proyecto en curso. Esta información ya no dependerá únicamente de los reportes dados por los encuestadores a los supervisores y/o coordinadores, así como estos a su vez reportados al director de encuestas. Pasando todo un proceso de validación de estos reportes, corroborando que sean totalmente verídicos, corrigiendo los posibles errores habidos como encuestas no realizadas, reportadas como hechas o encuestas hechas por dos personas, Etc.

Todo el flujo de esta información tanto de ida como de vuelta, por los procedimientos actuales es tardado, con la herramienta propuesta se podrán obtener casi en línea todos estos reportes. Por lo que facilitará la corrección de algunos detalles que permitan mejorar el levantamiento.

Se necesita analizar, diseñar y desarrollar una herramienta la cual ayudará a los investigadores o profesionales de la salud a obtener información eficaz, eficiente y oportuna. Esta herramienta servirá para poder hacer investigaciones y evaluaciones de cualquier tipo de programa social o de salud, así como también para aplicar cualquier tipo de encuesta vía Internet. La forma de hacer aplicable a cualquier tipo de encuesta será sólo modificando algunos parámetros iniciales, y automáticamente se generará la aplicación de la encuesta para el levantamiento de la información.

La herramienta constará de 4 etapas o módulos principales:

1. La creación de la encuesta electrónica.
2. La captación de los datos a través de la encuesta electrónica contestada por el beneficiario/entrevistado e introducida al sistema por medio del mismo beneficiario/entrevistado o por un entrevistador.
3. La consulta y creación de reportes de los avances a nivel nacional, de la encuesta por parte de los supervisores y/o investigadores.
4. Por último será la entrega final de la base de datos que tendrán un formato específico.

Esta herramienta de la encuesta electrónica deberá ser dinámica, es decir, en base al programa social o de salud que se necesite evaluar y por consiguiente levantará información de sus beneficiarios/entrevistados, podrá ser modificable para realizar diferentes tipos de reactivos. Esto se realizará sin tener que reprogramar toda la encuesta, únicamente modificando algunos parámetros que nos indicarán el tipo de programa social, qué reactivos son los que se aplicarán, así como también que respuestas y tipos de respuestas a esperar.

Esta encuesta será de tipo autoaplicable o aplicable por medio de un entrevistador según sea la necesidad del proyecto en turno.

Si es el primer caso autoaplicable, entonces será contestada por los beneficiarios/entrevistados del programa social o de salud a evaluar. Al beneficiario/entrevistado del programa se le enviará vía Internet un correo electrónico el cual contendrá una liga que lo llevará al inicio de la encuesta. En el segundo caso, la encuesta será aplicada por medio de una figura llamada entrevistador hacia el beneficiario/entrevistado.

Al final de esta investigación trataremos un ejemplo de desarrollo, es decir, un prototipo de un caso práctico, el cual hará una encuesta en particular para dar

ejemplo de la utilización de esta herramienta. Con este caso práctico se podrá ver la funcionalidad de la misma, tomando un programa social existente.

En la presente investigación abordamos en el contenido del primer capítulo el marco teórico con el cual damos sustento formal a esta tesis. Dentro de este capítulo vemos conceptos básicos de bases de datos, así como del internet. Así mismo encontramos descripciones de los lenguajes de alto nivel y manejadores de bases de datos con los cuales nos apoyamos para la solución del problema. Para que posteriormente abordemos los conceptos de las metodologías de análisis y diseño, los cuales nos sirvieron para desmembrar en células pequeñas el problema y así poder darle una mejor solución. También se desarrolla el tema del cual es parte importante esta tesis, “la encuesta”, que es el parteaguas de la necesidad que originó la creación de esta tesis.

En el siguiente capítulo exponemos el planteamiento del problema; concretamente qué fue lo que nos dio la pauta para proponer la creación de esta herramienta; también se da la solución que nos permitirá dar mejoras a la forma actual de levantar grandes encuestas a nivel nacional. Así mismo planteamos los requerimientos y análisis de costos para que en función de esto se determine si es viable o no el desarrollo de la tesis.

Adentrándonos en el tercer capítulo vemos el desarrollo del análisis y diseño de la herramienta por medio de las diferentes metodologías convenientes para este tema, es decir, creación de diagramas de distinta índole que ayudan a realizar la descomposición del problema en partes mínimas para poderles dar una mejor solución. Así como también vemos los tiempos que nos llevará el realizar este análisis y diseño, así como la creación del ejemplo de desarrollo.

Por último, en el capítulo final vemos la manera en cómo se desarrolló la herramienta a nivel software, así como la parte de implementación para poder hacer el ejemplo de desarrollo.

CAPÍTULO 1: ANTECEDENTES Y MARCO TEÓRICO

1.1 INTRODUCCIÓN A INTERNET

Internet es una red mundial de computadoras con un conjunto de protocolos, el más destacado, el TCP/IP. Aparece por primera vez en 1969, cuando ARPANET[1] establece su primera conexión entre tres universidades en California y una en Utah. También se usa el término Internet como sustantivo común y por tanto en minúsculas para designar a cualquier red de redes que use las mismas tecnologías que Internet, independientemente de su extensión o de que sea pública o privada.

Cuando se dice red de redes se hace referencia a que es una red formada por la interconexión de otras redes menores.

1.1.1 Historia del Internet

En 1961, se publicó el primer documento sobre la teoría de conmutación de paquetes. Y otro paso fundamental fue hacer dialogar a los servidores entre sí. En 1965, se conectó una computadora TX2 en Massachusetts con un Q-32 en California a través de una línea telefónica conmutada de baja velocidad, creando así la primera red de computadoras de área amplia jamás construida.

En 1969, Defense Advanced Research Projects Agency (DARPA) comenzó a planificar la creación de una red que conectaba computadoras en caso de una eventual guerra atómica que incomunicara a los humanos sobre la Tierra, con fines principalmente de defensa.

En 1972, se realizó la primera demostración pública de ARPANET, una nueva Red de comunicaciones financiada por la DARPA, que funcionaba de forma distribuida sobre la red telefónica conmutada.

En 1973, se desarrollaron nuevos protocolos de comunicaciones que permitían el intercambio de información de forma "transparente" para las computadoras conectadas. Surgió el nombre de "Internet", que se aplicó al sistema de redes interconectadas mediante los protocolos TCP e IP.

En 1983, ARPANET cambió el protocolo Network Control Program (NCP) por TCP/IP. Se creó el Internet Architecture Board (IAB) con el fin de estandarizar el protocolo TCP/IP y de proporcionar recursos de investigación a Internet.

En 1986, la National Science Foundation (NSF) comenzó el desarrollo de NSFNET que se convirtió en la principal Red en árbol de Internet, complementada después con las redes NSINET y ESNET.

[1]Red de computadoras. (*Advanced Research Projects Agency Network*).

En 1989, con la integración de los protocolos Open Systems Interconnection (OSI) en la arquitectura de Internet, se creó la tendencia actual de permitir no sólo la interconexión de redes de estructuras dispares, sino también la de facilitar el uso de distintos protocolos de comunicaciones.

En 1989, en el CERN[2] de Ginebra, se crea el lenguaje HyperText Markup Language (HTML), basado en el Standard Generalized Markup Language (SGML).

1990, el mismo equipo del CERN construyó el primer cliente Web, llamado WorldWideWeb (WWW), y el primer servidor Web.

Cerca de 7,000 millones de internautas es la cifra de usuarios de Internet que se alcanzó el pasado mes de Junio (2010) en todo el mundo, según el estudio realizado por la consultora estadounidense InertnetWorldStats[3]. Sin duda, un hito significativo en la historia de Internet. Este hecho sitúa a los ciudadanos de las regiones de Asia y el Pacífico como las más conectadas a la Red (42%). El estudio se realizó durante el pasado mes de Junio, teniendo en cuenta todas las personas mayores de 15 años que se conectaron desde su hogar o bien desde el trabajo. En segunda posición se ubicó Europa, con un 24.2% de la cuota, y en la cola América del Norte (13.5%), Latinoamérica (10.4%) y Oriente Medio y África (8.8%). Se estima que en diez años, la cantidad de navegantes de la Red aumentará al doble.

MUNDIAL DE USO DE INTERNET Y LAS ESTADÍSTICAS DE LA POBLACIÓN

Regiones del mundo	Población (2010 est.)	Usuarios de Internet 31 de diciembre 2000	Usuarios de Internet Últimos datos	Penetración (Población%)	Crecimiento 2000-2010	% De usuarios de la tabla
África	1,013,779,050	4,514,400	**110,931,700**	10,9%	2,357.3%	5,6%
Asia	3,834,792,852	114,304,000	**825,094,396**	21,5%	621,8%	42,0%
Europa	813,319,511	105,096,093	**475,069,448**	58,4%	352,0%	24,2%
Medio Oriente	212,336,924	3,284,800	**63,240,946**	29,8%	1,825.3%	3,2%
América del Norte	344,124,450	108,096,800	**266,224,500**	77,4%	146,3%	13,5%
América Latina y el Caribe	592,556,972	18,068,919	**204,689,836**	34,5%	1,032.8%	10,4%
Oceanía / Australia	34,700,201	7,620,480	**21,263,990**	61,3%	179,0%	1,1%
TOTAL MUNDIAL	6,845,609,960	360,985,492	**1,966,514,816**	**28,7%**	444,8%	100,0

Fig. 1.1 Estadística de usuarios de Internet en el mundo.

[2]Laboratorio Europeo de Física de Partículas (Conseil Européen pour la Recherche Nucléaire).
[3] http://www.internetworldstats.com/stats.htm.

La principal diferencia entre Internet y cualquier otra red informática reside en que esta no pertenece a ningún país, ni organismo oficial, ni a una empresa determinada, es decir, se trata de una red libre ya que cualquier persona puede acceder a ella desde cualquier punto del planeta, de la misma forma que no existe ningún tipo de restricción para toda la información que circula por la misma.

1.1.2 Direccionamiento IP y servicio DNS

El direccionamiento IP es un protocolo de la capa de red encargado del proceso de enrutamiento de los paquetes a través de la red. Todas las direcciones IP tienen un formato básico y no pueden ser tomadas arbitrariamente, deben ser asignadas por el Centro de Información de Red Internet (InterNIC) para evitar conflictos.

Cada dirección se divide en dos partes: el número de red y el número de host. El número de red es el que asigna la InterNIC e identifica a la red. El número de host es asignado por el Administrador de la red e identifica a un host (PC, servidor, ruteador, Etc.) en la red.

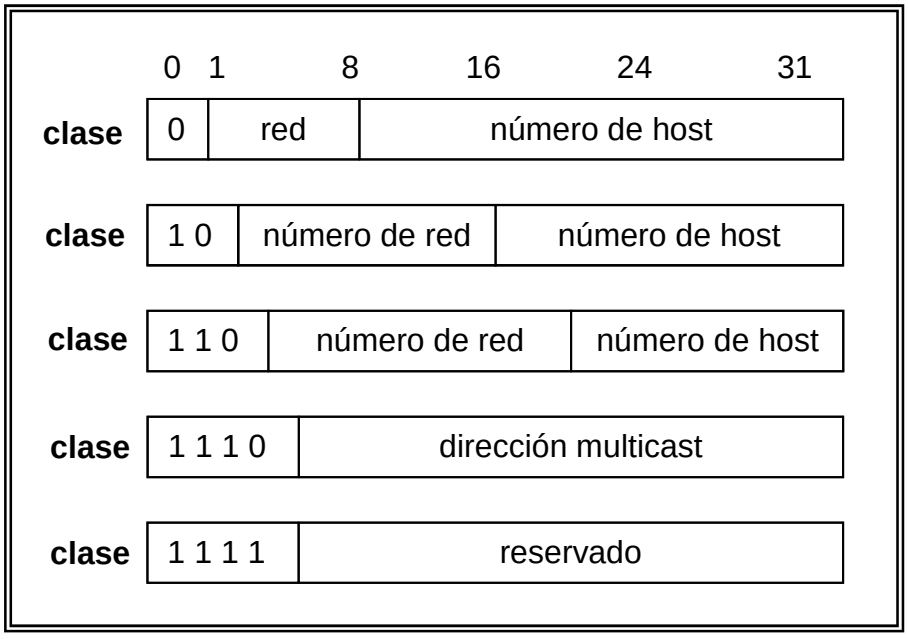


Fig. 1.2 Clases asignadas de direcciones de Internet.

Clase	Formato (r=red, h=host)	Número de redes	Número de hosts por red	Rango de direcciones de redes	Máscara de subred
A	r.h.h.h	128	16.777.214	0.0.0.0 - 127.0.0.0	255.0.0.0
B	r.r.h.h	16.384	65.534	128.0.0.0 - 191.255.0.0	255.255.0.0
C	r.r.r.h	2.097.152	254	192.0.0.0 - 223.255.255.0	255.255.255.0
D	grupo	-	-	224.0.0.0 - 239.255.255.255	-
E	no válidas	-	-	240	

Fig. 1.3 Rangos de Clases asignadas de direcciones de Internet.

Difusión (broadcast) y multidifusión (multicast).-- El término difusión (broadcast) se refiere a todos los hosts de una red; multidifusión (multicast) se refiere a varios hosts (aquellos que se hayan suscrito dentro de un mismo grupo). Siguiendo esta misma terminología, en ocasiones se utiliza el término unidifusión para referirse a un único host.

El Domain Name System (DNS) es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet. Aunque como base de datos el DNS es capaz de asociar distintos tipos de información a cada nombre, los usos más comunes son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico de cada dominio.

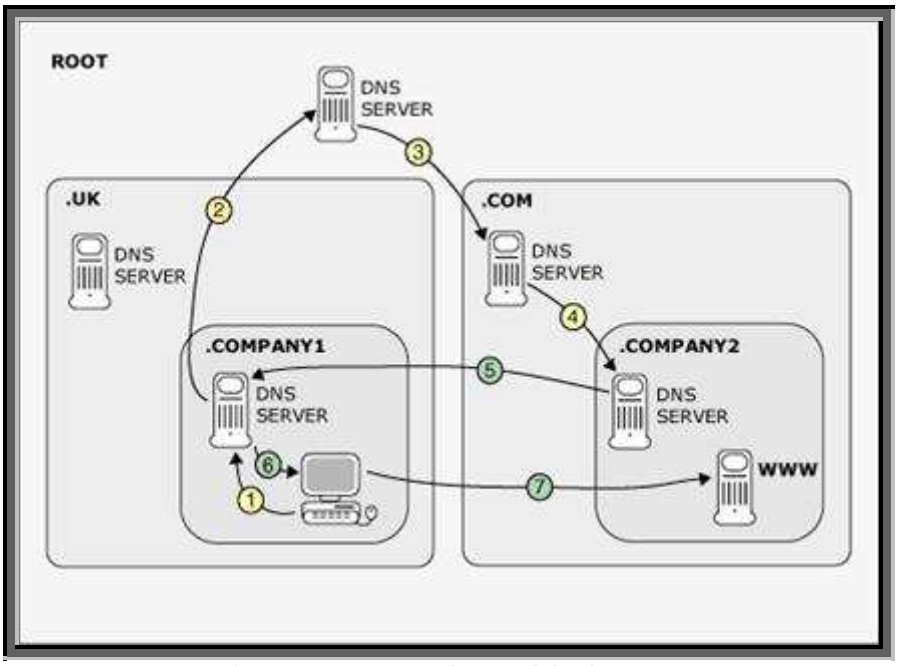


Fig. 1.4 Esquema de Servicio de DNS.

1.1.3 Servicios de Internet

Internet ofrece una serie de servicios creados para darle sentido a esa gran conexión mundial y que, a lo largo de los años, ha crecido y se ha fortalecido para bien de los usuarios.

La Internet comenzó a ser utilizada de muchas formas y con muchos propósitos. Y cada una de estas formas de uso es lo que se conoce como un servicio, una forma estandarizada de utilización, lo que a su vez requiere el uso de protocolos (estándares) universalmente aceptados tanto por los "clientes" como por los "proveedores" del servicio. Un caso paradigmático en este sentido, podría ser el correo electrónico. Resulta evidente que para poder usarlo, además de los protocolos mínimos de transmisión (TCP/IP)[4], hacen falta otros para que el servicio pueda ser universal. Es un hecho que los servicios no han sido siempre los mismos ni se han utilizado exactamente de la misma forma; algunos han aparecido nuevos (uno de los últimos y más espectaculares, la Web), otros han sufrido modificaciones y otros siguen prácticamente igual que en sus orígenes.

Aunque los "servicios de Internet" soportan los usos más comunes, navegación por páginas Web, correo, descarga de archivos, Etc., no se limitan a ellos ni son los más importantes (aunque sean los más aparentes). La Internet es utilizada cada día más como el nexo de unión de una gigantesca red de máquinas que dialogan e interactúan entre sí. La mayoría de las veces con poca o nula intervención humana en tales "conversaciones". Internet se parece cada día más a un sistema de cómputo distribuido de dimensiones planetarias.

Estos servicios son:
World Wide Web
Correo electrónico
Foros de discusión en tiempo real y no presenciales
- Foros
- Listas de discusión de correos
- Chat (MSN y Yahoo messenger, entre otros)

Servicio de transferencia de archivos: FTP
Servicio de trabajo compartido en servidores
Servicios de noticias
Blogs
Telefonía sobre IP

Correo electrónico

El correo electrónico es, junto con el WWW, uno de los servicios más utilizados en Internet. Consiste en el intercambio de mensajes, entre dos o más usuarios, que utilizan la red como plataforma.

[4] Protocolo de control de transmisión/Protocolo de Internet".

Funciona siguiendo el patrón de las cartas de correo ordinario. Las direcciones de correo de los usuarios en Internet son únicas. Para elaborarlas se requiere de un determinado dominio y de un identificador. Por ejemplo, si tenemos una cuenta de correo electrónico con Yahoo, nuestra dirección podría ser:

mi_nombre@yahoo.com

Así, "mi_nombre" sería nuestro identificador; la arroba (@) es el indicador de que estamos ante una dirección de correo electrónico y separa el nombre del usuario de los nombres de dominio del servidor de correo; por último, "kaixo.com" es el dominio, es decir, el lugar donde fue creada esa dirección.

Web

World Wide Web apareció a principios de los noventa y convirtió Internet en lo que hoy conocemos. Este sistema, cuyas iniciales se corresponden con las famosas WWW con las que empiezan las direcciones de las páginas web, permite intercambiar información en formato multimedia (texto, imágenes, video, sonido, Etc.).

La Web utiliza el sistema de hipertexto, que ofrece la posibilidad de ir de un documento a otro utilizando palabras enlazadas (también conocidas como *links*), lo que permite visitar un sinnúmero de páginas de diferente naturaleza. También es posible establecer enlaces entre páginas a través de imágenes o fotografías. En este caso se habla de hipermedia. El sistema que permite utilizar los hiperenlaces es el HTTP (*Hyper Text Transfer Protocol*).

Para poder visualizar las páginas web, es necesario tener instalado en la computadora un programa llamado navegador. Donde los más comunes son Netscape y Explorer.

En informática, World Wide Web (o la "Web") o Red Global Mundial es un sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas Web que pueden contener texto, imágenes, videos u otros contenidos multimedia, y navegar a través de ellas usando hiperenlaces.

La visualización de una página web de la World Wide Web normalmente comienza tecleando la URL[5] de la página en el navegador web, o siguiendo un enlace de hipertexto a esa página o recurso. En ese momento el navegador comienza una serie de comunicaciones, transparentes para el usuario, para obtener los datos de la página y visualizarla.

[5] *Uniform Resource Locator* (Localizador de Recurso Uniforme).

El primer paso consiste en traducir la parte del nombre del servidor de la URL en una dirección IP usando la base de datos distribuida de Internet conocida como DNS. Esta dirección IP es necesaria para contactar con el servidor web y poder enviarle paquetes de datos.

El siguiente paso es enviar una petición HTTP al servidor Web solicitando el recurso. En el caso de una página web típica, primero se solicita el texto HTML y luego es inmediatamente analizado por el navegador, el cual, después, hace peticiones adicionales para los gráficos y otros archivos que formen parte de la página. Las estadísticas de popularidad de un sitio web normalmente están basadas en el número de 'páginas vistas' o las 'peticiones' de servidor asociadas, o peticiones de fichero, que tienen lugar.

Al recibir los archivos solicitados desde el servidor web, el navegador renderiza la página tal y como se describe en el código HTML, el CSS[6] y otros lenguajes Web. Y al final se incorporan las imágenes y otros recursos para producir la página que ve el usuario en su pantalla.

La mayoría de las páginas Web contienen hiperenlaces a otras páginas relacionadas y algunas también contienen descargas, documentos fuente, definiciones y otros recursos Web.

Esta colección de recursos útiles y relacionados, interconectados a través de enlaces de hipertexto, es lo que ha sido denominado como 'red' (Web, en inglés) de información. Al trasladar esta idea a Internet, se creó lo que Tim Berners-Lee llamó WorldWideWeb en 1990.

Si un usuario accede de nuevo a una página después de un pequeño intervalo, es probable que no se vuelvan a recuperar los datos del servidor Web de la forma en que se menciona en el apartado anterior. Por defecto, los navegadores almacenan en una caché del disco duro local todos los recursos Web a los que el usuario va accediendo. El navegador enviará una petición HTTP sólo si la página ha sido actualizada desde la última carga, en otro caso, la versión almacenada se reutilizará en el paso de renderizado para agilizar la visualización de la página.

Esto es particularmente importante para reducir la cantidad de tráfico Web en Internet. La decisión sobre la caducidad de la página se hace de forma independiente para cada recurso (imagen, hoja de estilo, archivos JavaScript, Etc., además de ser para el propio código HTML). Sin embargo en sitios de contenido muy dinámico, muchos de los recursos básicos sólo se envían una vez por sesión. A los diseñadores de sitios Web les interesa reunir todo el código CSS y JavaScript en unos pocos archivos asociados a todo el sitio Web, de forma que pueden ser descargados en las cachés de los usuarios y reducir así el tiempo de carga de las páginas y las peticiones al servidor.

[6]Cascading Style Sheets (Hojas de estilo).

Hay otros componentes de Internet que pueden almacenar contenido Web. El más común en la práctica son los frecuentes firewalls de empresa y académicos donde se pueden almacenar los recursos Web solicitados por un usuario para el beneficio de todos los que estén conectados a ese firewall. Algunos buscadores como Google, Yahoo!, GlowBoom o AltaVista también almacenan contenidos de sitios Web.

Aparte de las utilidades creadas en los servidores Web que pueden determinar cuando los archivos físicos han sido actualizados, los diseñadores de páginas Web generadas dinámicamente pueden controlar las cabeceras HTTP enviadas a los usuarios, de forma que las páginas intermedias o sensibles a problemas de seguridad no sean guardadas en caché. Por ejemplo, en los bancos online y las páginas de noticias se utiliza frecuentemente este sistema.

Es común encontrar el prefijo "WWW" al comienzo de las direcciones Web debido a la costumbre de nombrar a los Host de Internet (los servidores) con los servicios que proporcionan. De esa forma, por ejemplo, el nombre de host para un servidor Web normalmente es "WWW", para un servidor FTP se suele usar "ftp", y para un servidor de noticias USENET, "news" o "nntp" (en relación al protocolo de noticias NNTP). Estos nombres de Host aparecen como subdominios de DNS, como en "www.example.com".

Empero el uso de estos prefijos no está impuesto por ningún estándar, de hecho, el primer servidor Web se encontraba en "nxoc01.cern.ch" e incluso hoy en día existen muchos sitios Web que no tienen el prefijo "WWW". Este prefijo no tiene ninguna relación con la forma en que se muestra el sitio Web principal. El prefijo "WWW" es simplemente una elección para el nombre de subdominio del sitio Web.

Existen algunos navegadores Web que añaden automáticamente "WWW." al principio, y posiblemente ".com"" al final, en las URLs que se teclean, si no se encuentra el host sin ellas. Internet Explorer, Mozilla Firefox y Opera también añaden "http://www." y ".com" al contenido de la barra de dirección si se pulsan al mismo tiempo las teclas de Control y Enter. Por ejemplo, si se teclea "ejemplo" en la barra de direcciones y luego se pulsa sólo Enter o Control+Enter normalmente buscará "http://www.ejemplo.com", dependiendo de la versión exacta del navegador y su configuración.

FTP

Otra de las posibilidades que ofrece Internet es la de descargar archivos de un servidor a otro. Este servicio permite acceder a gran cantidad de programas ubicados en cualquier parte del mundo y en cualquier formato, así como descargarlos directamente a nuestra computadora.

El sistema que facilita este servicio es el FTP o protocolo de transferencia de archivos. Para aprovechar las posibilidades de esta nueva tecnología se han creado páginas que reúnen, por sectores de interés, todos estos archivos.

Asimismo, gracias a este servicio, es posible publicar en otros servidores información que poseemos. Para ello, es necesario disponer de un programa de FTP que nos permita colocar nuestros archivos en una computadora remota. Este sistema de intercambio se ha popularizado enormemente, sobre todo gracias a la posibilidad de intercambio de archivos de música o imágenes, que son los que más interés tienen para los usuarios.

En Internet es posible encontrar grandes cantidades de programas y archivos almacenados en servidores accesibles mediante el protocolo FTP. Para acceder a estos archivos es necesario utilizar una aplicación que utilice este protocolo, como el Explorador de Windows, el conocido CuteFTP o el WSFTP. En la actualidad, desde el mismo navegador también se puede acceder a estos servidores, cambiando la etiqueta http:// por la de ftp://, aunque la velocidad y fiabilidad de la conexión es menor que utilizando programas específicamente diseñados con esta finalidad.

Con tantos archivos almacenados en computadoras diferentes, el problema puede ser encontrar aquello que se busca. Con la intención de solucionar este problema se creó Archie, una base de datos que dispone de información sobre los programas y su localización dentro de Internet. Sin embargo el WWW ha ido dejando en el olvido este tipo de transferencia de archivos.

Sesiones remotas de trabajo: SSH

SSH (Secure SHell, en español: intérprete de órdenes seguro) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos, y también puede redirigir el tráfico de X para poder ejecutar programas gráficos si tenemos un Servidor X (en sistemas Unix y Windows) corriendo.

Además de la conexión a otras máquinas, SSH nos permite copiar datos de forma segura (tanto archivos sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a las máquinas y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

SSH trabaja de forma similar a como se hace con telnet, la diferencia principal es que SSH usa técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible y ninguna tercera persona pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe

durante toda la sesión; aunque es posible atacar este tipo de sistemas por medio de ataques de REPLAY y manipular así la información entre destinos.

Telnet sólo sirve para acceder en modo terminal, es decir, sin gráficos, pero fue una herramienta muy útil para arreglar fallos a distancia, sin necesidad de estar físicamente en el mismo sitio que la máquina que los tenía. También se usaba para consultar datos a distancia, como datos personales en máquinas accesibles por red, información bibliográfica.

Aparte de estos usos, en general telnet se ha utilizado (y aún hoy se puede utilizar en su variante SSH) para abrir una sesión con una máquina UNIX, de modo que múltiples usuarios con cuenta en la máquina, se conectan, abren sesión y pueden trabajar utilizando esa máquina. Es una forma muy usual de trabajar con sistemas UNIX.

1.2 ARQUITECTURA DE DESARROLLO EN 3 CAPAS

La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño.

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, simplemente es necesario conocer la API[7] que existe entre niveles.

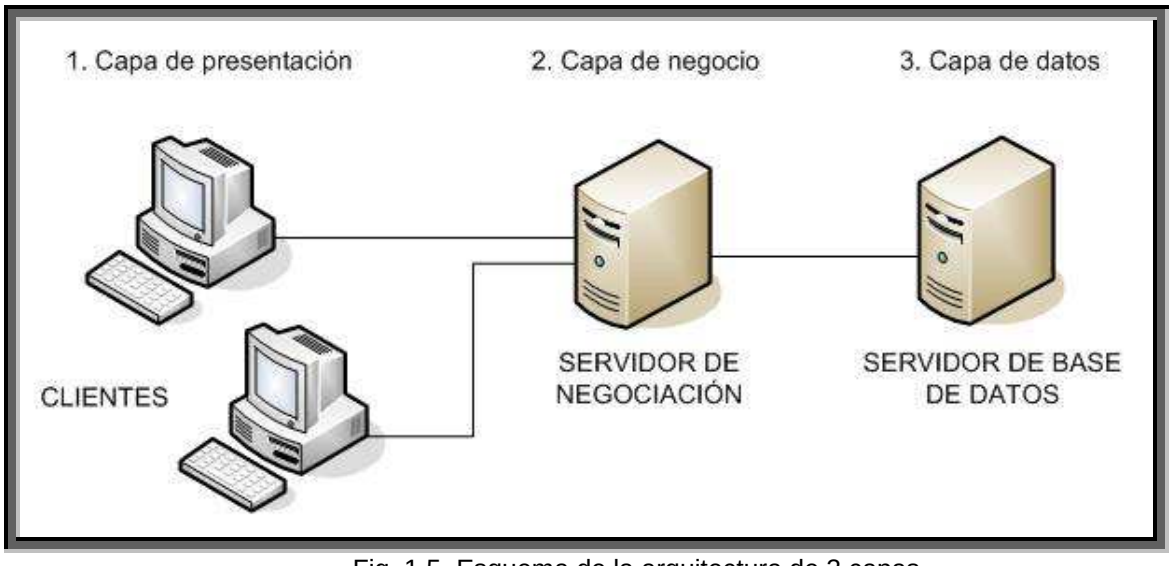


Fig. 1.5 Esquema de la arquitectura de 3 capas.

[7] interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*).

1.2.1 Capa de Presentación

La capa de presentación permite al usuario visualizar y acceder al sistema a través de una interfaz de usuario (GUI Graphic User Interface). También puede llamarse servicio de usuario, front-end, cliente, aplicación, o aplicación cliente.

Esta interfaz de usuario se compone, el 99% de los casos, de ventanas y de un conjunto de controles, esto es, botones, cajas de texto, menús desplegables, cajas de diálogo, entre otros.

Interfaces actuales

- En la actualidad se pueden distinguir 3 formas de interfaz de usuario:
- La interfaz clásica de ventanas.
- La interfaz integrada en programas de gestión.
- Los navegadores.

1.2.2 Capa de Negocio

En esta capa se encuentran las reglas y lógica de procedimientos necesarios para realizar las operaciones del sistema, esto es, las actividades operacionales, controles de consistencia, validaciones, cálculos.

Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

Aquí se encuentran también los distintos "servicios" relacionados con el sistema. Un servicio es un concepto lógico, un componente que describe un paquete físico de funcionalidad.

Un componente es un fragmento de código compilado, que reside en un archivo separado, y que se registra en la máquina que va a utilizarlo. Una vez registrado, el componente puede ser utilizado por cualquier aplicación, y no sólo por aquella para la que ha sido pensada originalmente.

Toda regla de negocio debe ser implementada en esta capa; la capa de negocios debería ser vista como un conjunto de servicios expuestos a las capas de presentación de las aplicaciones. Es muy importante definir y diseñar los componentes de esta capa en términos de negocios. Por ejemplo, un componente podría ser ServiciosFinancieros, una clase dentro de este componente podría ser ServiciosBancarios, y un método dentro de la clase podría ser ObtenerBalances. Estos componentes deberían estar orientados a las tareas y no orientados a las entidades, para facilitar el desarrollo y mantenimiento de las aplicaciones.

En las aplicaciones DNA, los componentes de negocio generalmente residen en COM+[8] (COM+ para Windows 2000 y servidor .NET con Microsoft Transaction Server para Windows NT). El motivo para esto es que COM+ provee la infraestructura necesitada por aplicaciones de alta disponibilidad, escalables, de buen comportamiento y desempeño; algunos de los servicios de infraestructura proporcionados por COM+ son grupos de threads (thread pooling), grupos de objetos (object pooling), sincronización, administración de transacciones distribuidas, invocaciones asíncronas, entre otros más.

Muchos de los problemas al interrumpir el servicio de una aplicación, con síntomas como que el IIS[9] deje de responder, degradación del servicio, Etc., recae en componentes que no cumplen con las mejores prácticas.

1.2.3 Capa de Datos

Es donde reside la información. Está formada por uno o más administradores de bases de datos que reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. La información se puede encontrar almacenada en bases de datos, sistemas de directorios, sistemas de archivos, entre otros.

Por lo que respecta a los mecanismos de acceso a los datos, el nivel de estandarización también es notable en la actualidad. El primer intento de unificación en este sentido fue ODBC (Open DataBase Connectivity). Se trata de un estándar para el acceso a los datos, en cual se crea un controlador específico para la base de datos a manejar en el servidor adecuado. Si se cambian los datos de un servidor a otro, la aplicación permanece intacta. Sólo hay que cambiar el controlador, y todo puede seguir funcionando como estaba.

Para el caso de Java, también existe un estándar de acceso a los datos, denominado JDBC (Java DataBase Connectivity), que se comporta de forma similar a como lo hacen los controladores ODBC para cualquier otro lenguaje.

El objetivo de la capa de acceso a datos es el de abstraer los detalles del formato y localización del almacén de datos, de las capas de presentación y negocios. La meta principal por la cual hacer esto es que provee la habilidad de cambiar a una fuente de datos distinta sin tener que impactar al resto de las capas de la aplicación. Esta capa usualmente está compuesta de componentes que corren en COM+ y de una fuente de datos, que muchas veces es una base de datos acceder a través de procedimientos almacenados.

Las siguientes mejores prácticas aplican a objetos de datos ActiveX (ADO) sin embargo algunos pueden aplicar también a otros modelos de conexión:

---

[8] Component Object Model (es una plataforma de Microsoft para componentes de software).
[9] Internet Information Services o IIS es un servidor Web.

1. Devolver siempre Recordsets desconectados.
2. Siempre abrir Recordsets con tipo de cursor Static.
3. Guardar una copia en el caché del servidor de los datos estáticos y usados frecuentemente.
4. Adquirir recursos tarde y liberarlos temprano.
5. No devolver un Recordset para los métodos que solo devuelven un registro.

NIVELES

Todas estas capas pueden residir en un único servidor (no sería lo normal), si bien lo más usual es que haya varios servidores donde residan las capas de presentación, negocio y datos (A cada servidor en el cual reside el sistema se le conoce como nivel).

Las capas de negocio y de datos pueden residir en el mismo servidor, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios servidores, los cuales recibirán las peticiones de la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligara a la separación, esta capa de negocio podría residir en uno o más servidores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de servidores sobre los cuales corre la capa de datos, y otra serie de servidores sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares.

El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación/ Lógica de Negocio/ Datos.

En cambio, el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en un solo servidor (Presentación+lógica+datos). Se dice que la arquitectura de la solución es de tres capas y *un nivel*.
- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en dos servidores (presentación+lógica, lógica+datos). Se dice que la arquitectura de la solución es de tres capas y *dos niveles*.

- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en tres servidores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y *tres niveles*.

1.3 BASES DE DATOS

Un sistema de gestión de bases de datos (DBMS database management system) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a ellos. La colección de datos se denomina base de datos (BD). El objetivo primordial de un DBMS es que a la vez sea conveniente y eficiente para ser utilizado al extraer o almacenar información en la BD.

Los sistemas de bases de datos están diseñados para gestionar grandes bloques de información, que implica tanto la definición de estructuras para el almacenamiento como de mecanismos para la gestión de la información.

Además los DBMS deben mantener la seguridad de la información almacenada pese a la caída del sistema o accesos no autorizados.

1.3.1 Introducción a las bases de Datos.

Existen diversas definiciones de bases de datos, y al analizarse detenidamente nos podemos dar cuenta que la mayoría de ellas coinciden, aunque en algunas les hacen falta detalles que son característicos de las bases de datos. Algunas de estas definiciones son:

"*Una base de datos es un conjunto, colección o depósito de datos almacenados en insoporte informático no volátil. Los datos están interrelacionados y estructurados de acuerdo con un modelo capaz de recoger el máximo contenido semántico"*

"*Colección no redundante de datos que son compartidos por diferentes sistemas de aplicación*[10]"

Por lo que se ha podido llegar a una definición de bases de datos que en la actualidad ha llegado a ser la siguiente:

"*Colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se ha de apoyar en un modelo de datos, el cual ha de permitir captar las interrelaciones y restricciones existentes en el mundo*

[10] Sowe. 1983

real. Los procedimientos de actualización y recuperación, comunes y bien determinados, facilitarán la seguridad del conjunto de los datos[11].”

Objetivos de los sistemas de bases de datos.

Al considerar que una empresa bancaria guarda su información sobre todos los clientes y cuentas en archivos de sistemas permanentes. Además, el sistema tiene diversos programas de aplicación que permiten al usuario manejar los archivos, como hacer abonos, añadir cuentas, gestionar el saldo, Etc. Según surge la necesidad se añaden nuevos programas de aplicación al sistema.

El típico sistema de procesamiento de archivos descrito está apoyado por un sistema operativo convencional. Los registros permanentes se almacenan en varios archivos, y se escribe un número de diferentes programas de aplicación para manipular los archivos apropiados. Este sistema tiene un número de desventajas importante, como:

1.- Redundancia e inconsistencia de los datos. Puesto que los archivos y los programas de aplicación son creados por distintos programas durante un periodo largo de tiempo, es probable que tengan distintos formatos, y pueden estar duplicados en varios sitios. Por ejemplo, la dirección de un cliente puede aparecer en más de un archivo. Esta redundancia aumenta los costes de almacenamiento y acceso, y puede llevar a la inconsistencia de los datos, esto es, las diversas copias de los mismos datos no concuerdan entre si.

2.- Dificultad para tener acceso a los datos. Suponiendo que se necesita averiguar los nombres de los clientes que viven en una ciudad. Puesto que esta solicitud no fue prevista, no hay ningún programa que la satisfaga. Hay dos opciones, agarrar la lista de todos los clientes y extraer la información manualmente, o escribir el programa de aplicación necesario. Obviamente ninguna de las dos opciones es satisfactoria. Lo que se trata de probar es que los entornos convencionales de procesamiento de archivos no permiten recuperar los datos necesarios de una forma conveniente y eficiente. Deben sistemas de recuperación de datos para uso general.

3.- Aislamiento de los datos. Puesto que los datos están repartidos en varios archivos, y estos pueden tener diferentes formatos, es difícil escribir nuevos programas para obtener los datos apropiados.

4.- Anomalías del acceso concurrente. Para mejorar el funcionamiento del sistema y obtener un tiempo de respuesta más rápido, muchos sistemas permiten que los usuarios actualicen los datos simultáneamente. En un entorno así, las actualizaciones concurrentes pueden dar por resultados datos inconsistentes. Si una cuenta tiene $500, y dos clientes retiran $50 y $100 casi al mismo tiempo, el

[11] Adoración de, Miguel Castaño. ”Fundamentos y modelos de datos”. 2005. Ed. Alfa Omega

resultado de las ejecuciones concurrentes puede dejar la cuneta en un resultado incorrecto o inconsistente, la cuenta puede contener $450 ó $400, en vez de $350. Para prevenir esta posibilidad, debe mantenerse alguna forma de supervisión en el sistema. Puesto que se puede acceder a los datos por medio de diversos programas de aplicación, esta supervisión es muy difícil de proporcionar.

5.- Problemas de seguridad. No todos los usuarios del sistema de BD deben poder acceder a todos los datos. Por ejemplo, el personal de nóminas no necesita acceder a la información sobre la dirección de los clientes. Puesto que los programas de aplicación se añaden al sistema de una forma precisa, es difícil implantar tales restricciones de seguridad.

6.- Problemas de integridad. Los valores de datos almacenados en la BD deben satisfacer ciertos tipos de restricciones de consistencia. Por ejemplo, el saldo de una cuenta no debe caer por debajo de una cantidad prefijada. Estas restricciones se hacen cumplir añadiendo códigos apropiados en los programas. Sin embargo, cuando se añaden restricciones nuevas, es difícil cambiar los programas para hacerlas cumplir. El problema se complica más cuando las restricciones implican varios elementos de información de distintos archivos. Estas dificultades, entre otras, han fomentado el desarrollo de DBMS.

Abstracción de datos.

Un DBMS es una colección de archivos interrelacionados y un conjunto de programas que permiten a los usuarios acceder y modificar esos archivos. Uno de sus objetivos más importantes es proporcionar a los usuarios una visión abstracta de los datos, es decir, el sistema esconde ciertos detalles de como se almacenan y mantienen los datos, sin embargo se deben extraer eficientemente. Este requerimiento ha llevado al diseño de estructuras de datos complejas para la representación de datos en la BD. Se esconde esta complejidad a través de distintos niveles de abstracción para simplificar la interacción con el sistema.

1.- Nivel físico. El nivel más bajo de abstracción describe como se almacenan realmente los datos, se describen en detalle las estructuras de datos complejas del nivel bajo.

2.- Nivel conceptual. Describe que datos son realmente almacenados en la BD y las relaciones que existen entre ellos. Aquí se describe la BD completa en términos de un número pequeño de estructuras sencillas. El nivel conceptual de abstracción lo usan los administradores de BD, quienes deben decidir que información se va a guardar en la BD.

3.- Nivel de visión. El nivel más alto de abstracción describe sólo parte de la BD completa. Muchos usuarios no se interesan por toda la información, sólo necesitan una parte de la BD. Para simplificar su interacción con el sistema se define el nivel de abstracción de visión. El sistema pude proporcionar muchas visiones para la

misma BD. En el nivel físico, un registro puede describirse como un bloque de posiciones de memoria consecutivas. En el nivel conceptual, cada registro se describe por medio de una definición de tipo, y se define la interrelación entre los tipos. Finalmente, en el nivel de visión, se definen varias visiones de la BD. Por ejemplo, los cajeros sólo ven la información sobre las cuentas de los clientes, sin poder acceder a los salarios de los empleados.

Modelos de datos.

Para describir la estructura de una BD es necesario definir el concepto de modelo de datos, una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia. Los diversos modelos de datos se dividen en tres grupos: modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos de datos.

1. **Modelos lógicos basados en objetos**

Los modelos lógicos basados en objetos se usan para describir datos en el nivel conceptual y de visión. Se caracterizan porque proporcionan capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Hay muchos modelos diferentes y aparecerán más. Algunos de los más conocidos son el modelo entidad-relación, el orientado a objetos, el binario, el semántico de datos, el infológico y el modelo funcional de datos.

El modelo entidad-relación ha ganado aceptación y se utiliza ampliamente en la práctica, el modelo orientado a objetos incluye muchos conceptos del anterior, y este ha ganado aceptación rápidamente.

- **El modelo entidad-relación.**

El modelo de datos entidad-relación (E-R) se basa en una percepción de un mundo real que consiste en una colección de objetos básicos llamados entidades, y relaciones entre estos objetos. Una entidad es un objeto distinguible de otros por medio de un conjunto de atributos. Por ejemplo, los atributos número y saldo describen una cuenta particular. Una relación es una asociación entre varias entidades. Por ejemplo, una relación Clicta asocia a un cliente con cada cuenta que posee. El conjunto de todas las entidades del mismo tipo y relaciones del mismo tipo se denomina conjunto de entidades y conjunto de relaciones.

Además el modelo E-R representa ciertas restricciones a las que deben ajustarse los contenidos de una BD. Una restricción importante es la cardinalidad de asignación, que expresa el número de entidades a las que puede asociarse otra entidad mediante un conjunto de relación.

La estructura lógica global de una BD puede expresarse gráficamente por un diagrama E-R que consta de:

1.- Rectángulos, que representan conjuntos de entidades.
2.- Elipses, que representan atributos.
3.- Rombos, que representan relaciones entre conjuntos de entidades.
4.- Líneas, que conectan atributos a conjuntos de entidades y conjuntos de entidades a relaciones.

Cada componente se etiqueta con el nombre de lo que representa.

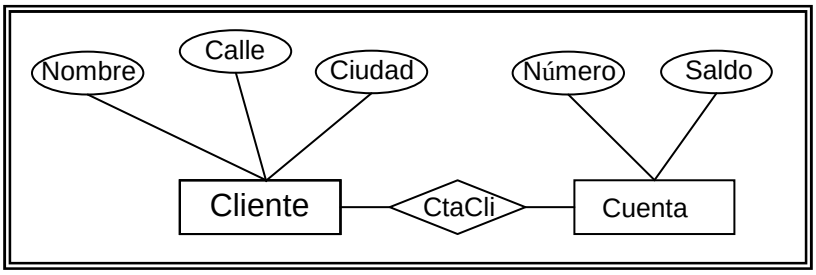


Fig. 1.6 Modelo entidad-relación.

- **El modelo orientado a objetos.**

Al igual que el modelo E-R, el modelo orientado a objetos se basa en una colección de objetos. Un objeto contiene valores acumulados en variables instancia dentro de él, y estos valores son objetos por si mismos. Así, los objetos contienen objetos a un nivel de anidamiento arbitrario. Un objeto también contiene partes de código que operan sobre el objeto, que se denominan métodos.

Los objetos que contienen los mismos tipos de valores y los mismos métodos se agrupan en clases. Una clase puede se vista como una definición de tipo para objetos.

La única forma en la que un objeto puede acceder a los datos de otro objeto es invocando a un método de ese otro objeto. Esto se llama envío de un mensaje al objeto. Así, la interfaz de llamada de los métodos de un objeto define su parte visible externamente, la parte interna del objeto (las variables de instancia y el código de método) no son visibles externamente. El resultado es dos niveles de abstracción de datos.

Para ilustrar el concepto, se puede considerar un objeto que representa una cuenta. Dicho objeto contiene las variables instancia número y saldo, y el método interés de pago, que añade interés al saldo. Suponiendo que se paga el 6% en todas las cuentas, pero ahora se va a pagar el 5% si el saldo es menor que $1000, y el 6% si es mayor o igual. Bajo la mayoría de los modelos de datos, esto implicaría cambiar de código en uno o más programas de aplicación. Bajo este modelo, sólo se hace un cambio dentro del método interés de pago. El interfaz

externo del objeto permanece sin cambios. A diferencia de las entidades en el modelo E-R, cada objeto tiene su propia identidad única independiente de los valores que contiene. Así dos objetos que contienen los mismos valores son distintos. La distinción entre objetos individuales se mantiene en el nivel físico por medio de identificadores de objeto.

2. Modelos lógicos basados en registros.

Los modelos lógicos basados en registros se utilizan para describir datos en los modelos conceptual y físico. A diferencia de los modelos lógicos basados en objetos, se usan para especificar la estructura lógica global de la BD y para proporcionar una descripción a nivel más alto de la implementación.

Los modelos basados en registros se llaman así porque la BD está estructurada en registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos, o atributos, y cada campo normalmente es de longitud fija. La estructura más rica de estas BD a menudo lleva a registros de longitud variable en le nivel físico.

Los modelos basados en registros no incluyen un mecanismo para la representación directa de código de la BD, en cambio, hay lenguajes separados que se asocian con el modelo para expresar consultas y actualizaciones.

Los tres modelos de datos más aceptados son los modelos relacional, de red y jerárquico. El modelo relacional ha ganado aceptación por encima de los otros.

- **Modelo relacional.**

El modelo relacional representa los datos y relaciones entre los datos mediante una colección de tablas, cuyas columnas tienen nombres únicos.

NOMBRE	CALLE	CIUDAD	NÚMERO
Laureano	Mapple	México	900
Sandro	Tlalpan	DF	556
Sandro	Tlalpan	DF	647
Hugo	Cedros	Veracruz	801
Hugo	Cedros	Veracruz	647

NÚMERO	SALDO
900	55
556	100,000
647	105,366
801	10,533

Fig. 1.7 Modelo Relacional.

- **Modelo de red.**

Los datos en el modelo de red se representan mediante colecciones de registros y las relaciones entre los datos se representan mediante enlaces, los cuales pueden verse como punteros.

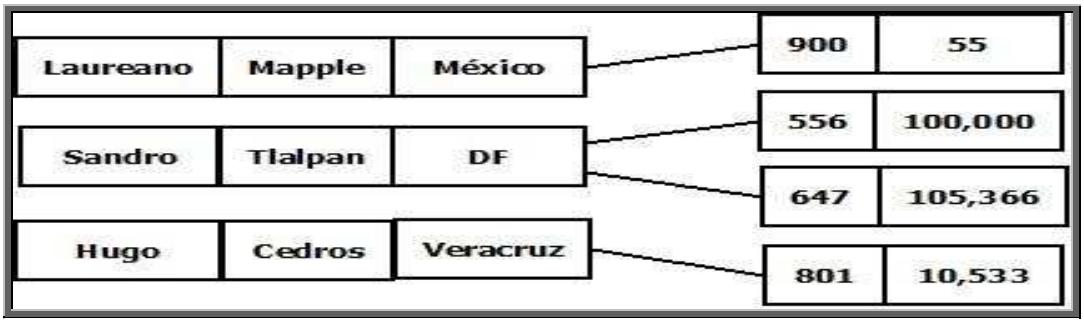


Fig. 1.8 Esquema Modelo de Red.

- **Modelo jerárquico.**

El modelo jerárquico es similar al modelo de red, los datos y las relaciones se representan mediante registros y enlaces. Se diferencia del modelo de red en que los registros están organizados como colecciones de árboles.

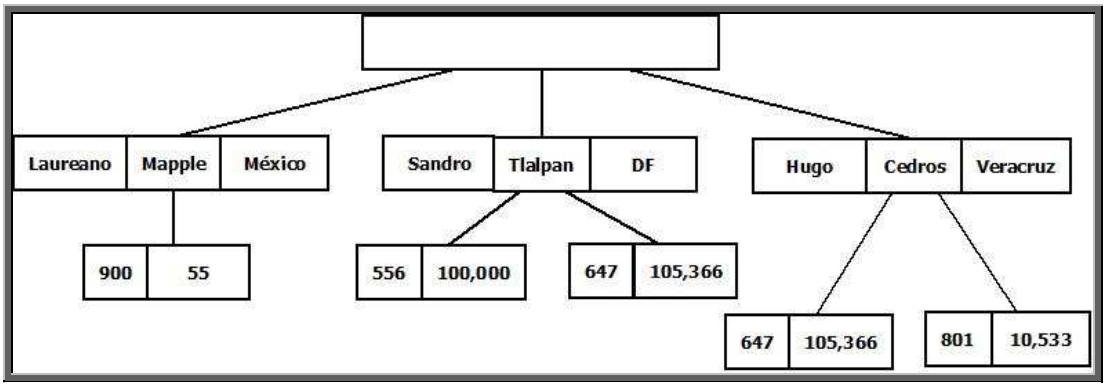


Fig. 1.9 Esquema Modelo Jerárquico.

Diferencias entre modelos.

Los modelos relacionales se diferencian de los modelos de red y jerárquico en que no usan punteros o enlaces. En cambio, el modelo relacional conecta registros mediante los valores que éstos contienen. Esta libertad del uso de punteros permite que se defina una base matemática formal.

3. Modelos físicos de datos.

Los modelos físicos de datos se usan para describir datos en el nivel más bajo. Hay muy pocos de modelos físicos de datos en uso, siendo los más conocidos el modelo unificador y de memoria de elementos.

Instancias y esquemas, independencia de datos.

Las BD cambian a lo largo del tiempo según se añade y se suprime información. La colección de información almacenada en un determinado

momento en el tiempo, se llama instancia de la BD. El diseño global de la BD se llama esquema de la BD, y se cambian muy raras veces.

El concepto de esquema corresponde a la noción de definición de tipo en un lenguaje de programación, y el concepto del valor de una variable corresponde al concepto de una instancia de un esquema de la BD.

Los sistemas de BD tienen varios esquemas, divididos de acuerdo a los niveles de abstracción tratados, el esquema físico, el esquema conceptual y los subesquemas. En general, los sistemas de BD soportan un esquema físico, otro conceptual y varios subesquemas.

La capacidad de modificar una definición de un esquema de un nivel sin afectar la definición de un esquema en el nivel superior siguiente se llama independencia de datos. Hay dos niveles de independencia de datos:

1.- Independencia física de datos. Es la capacidad de modificar el esquema físico sin provocar que se vuelvan a escribir los programas de aplicación.

2.-Independencia lógica de datos. Es la capacidad de modificar el esquema conceptual sin provocar que se vuelvan a escribir los programas de aplicación.

La independencia lógica de datos es más difícil de lograr, ya que los programas de aplicación son fuertemente dependientes de la estructura lógica de los datos a los que acceden.

El concepto de independencia de datos es similar al concepto de tipos abstractos de datos, ambos ocultan detalles de implementación.

Lenguaje de definición de datos, Lenguaje de manipulación de datos y Gestor de base de datos.

Un esquema de BD se especifica por medio de un conjunto de definiciones que se expresan mediante un lenguaje especial llamado lenguaje de definición de datos (data definition language, DDL). El resultado de la compilación de sentencias de DDL es un conjunto de tablas que se almacenan en un archivo especial llamado diccionario de datos o directorio.

Un directorio de datos es un archivo que contiene metadatos, es decir, datos sobre datos. Este archivo se consulta antes de leer o modificar los datos reales en el sistema de BD.

La estructura de almacenamiento y los métodos de acceso se especifican por medio de un conjunto de definiciones en un tipo especial de DDL llamado lenguaje de almacenamiento y definición de datos. El resultado de la compilación de estas

definiciones es un conjunto de instrucciones que especifican los detalles de implementación de los esquemas que normalmente se esconden a los usuarios.

Lenguaje de manipulación de datos.

Por manipulación de datos se entiende la recuperación y modificación de la información almacenada y la inserción y supresión de información.

A nivel físico, debemos definir algoritmos que permitan acceso eficiente a los datos. En los niveles de abstracción más altos, se pone énfasis en la facilidad de uso. El objetivo es proporcionar una interacción eficiente entre las personas y el sistema.

Un lenguaje de manipulación de datos (data manipulation lenguage, DML) es un lenguaje que capacita a los usuarios a acceder o manipular los datos. Existen básicamente dos tipos:

1.- Procedimentales: requieren que el usuario especifique qué datos se necesitan y cómo conseguirlos.

2.- No procedimentales: el usuario debe especificar qué datos se necesitan, pero no cómo obtenerlos.

Los DML no procedimentales normalmente son más sencillos de aprender y usar, sin embargo pueden generar código que no sea tan eficiente como el producido por los lenguajes procedimentales. Esta dificultad puede remediarse a través de varias técnicas de optimización.

Una consulta es una sentencia que solicita la recuperación de información. El trozo de un DML que implica recuperación de información se llama lenguaje de consultas. Aunque es incorrecto, suelen utilizarse los términos lenguaje de consultas y lenguaje de manipulación de datos como sinónimos.

Gestor de base de datos.

Generalmente, las BD requieren una gran cantidad de espacio de almacenamiento, se miden en gigabytes. Puesto que la memoria principal no puede almacenar toda esa información, se almacena en discos. Ya que el movimiento de los datos y el disco son lentos comparado con la UCP, es imperativo que el sistema de la BD estructure los datos de forma que minimice la necesidad de mover los datos entre el disco y la memoria.

El objetivo de un sistema de BD es simplificar y facilitar el acceso a los datos. Las vistas de alto nivel ayudan a lograrlo. Un factor para la satisfacción o insatisfacción del usuario con un sistema de BD es su funcionamiento. El funcionamiento de un sistema depende de la eficiencia de las estructuras de datos

usadas para representar los datos y de la capacidad de eficiencia de operar sobre estas estructuras de datos que el sistema tiene. Se debe llegar a un compromiso entre espacio, tiempo y eficiencia.

Un gestor de BD es un módulo de programa que proporciona el interfaz entre los datos de bajo nivel almacenados y los programas de aplicación y consultas, y es responsable de las siguientes tareas:

1.- Interacción con el gestor de archivos. El gestor de BD traduce las distintas sentencias DML a comandos del sistema de archivos de bajo nivel. Así es responsable del verdadero almacenamiento de los datos.

2.-Implantación de la integridad. Los valores de los datos que se almacenan deben satisfacer ciertos tipos de restricciones de consistencia, que debe especificar explícitamente el administrador de la BD. El gestor de la BD entonces puede determinar si se produce una violación de la restricción, si es así, se debe tomar la acción apropiada.

3.- Implantación de la seguridad. Es trabajo del gestor de la BD debe hacer que se cumplan los requisitos de seguridad.

4.- Copia de seguridad y recuperación. Un sistema informático, como cualquier otro dispositivo, esta sujeto a fallos. Es responsabilidad del gestor de BD detectar fallos y restaurar la BD al estado que existía antes de ocurrir el fallo. Esto se lleva a cabo normalmente con procedimientos de copias de seguridad y recuperación.

5.- Control de concurrencia. Cuando varios usuarios actualizan la BD de forma concurrente, es posible que no se conserve la consistencia de los datos. Controlar la interacción entre los usuarios concurrentes es otra responsabilidad del gestor de la BD.

Los sistemas de BD diseñados para computadoras personales pequeñas pueden no tener todas las características apuntadas.

Administrador de bases de datos y Usuarios de bases de datos

Una de las razones principales para tener DBMS es tener control central de los datos y de los programas que acceden a esos datos. La persona que tiene dicho control central sobre el sistema se llama administrador de la BD (database administrator, DBA). Las funciones del DBA son:

1.- Definición de esquema. El esquema original de la BD se crea escribiendo un conjunto de definiciones que son traducidas por el compilador de DDL a un conjunto de tablas, almacenadas permanentemente en el diccionario de datos.

2.- Definición de la estructura de almacenamiento y del método de acceso. Se crean escribiendo un conjunto de definiciones traducidas por el compilador del lenguaje de almacenamiento y definición de datos.

3.- Modificación del esquema y de la organización física. Las modificaciones son poco comunes, pero se logran escribiendo un conjunto de definiciones usadas por el compilador DDL para generar modificaciones a las tablas internas apropiadas.

4.- Concesión de autorización para el acceso a los datos. La concesión de diferentes tipos de autorización permite al DBA regular qué partes de la BD van a poder ser accedidas por varios usuarios.

5.- Especificación de las restricciones de integridad. Las restricciones de integridad se mantienen en una estructura especial del sistema que consulta el gestor de la BD cada vez que tiene lugar una actualización.

Usuarios de bases de datos.

Un objetivo principal de un DBMS es proporcionar un entorno para recuperar y almacenar información en la BD. Hay cuatro tipos de usuarios, diferenciados por la forma de interaccionar con el sistema:

1.- Programadores de aplicaciones. Los profesionales en computación interaccionan con el sistema por medio de llamadas en DML, incorporados en un programa escrito en un lenguaje principal (como Pascal o C). Estos programas se denominan comúnmente programas de aplicación.

Puesto que la sintaxis de DML es normalmente diferente de la del lenguaje principal, las llamadas en DML van normalmente precedidas de un caracter especial de forma que se pueda generar el código apropiado. Un preprocesador especial, llamado precompilador de DML, convierte las sentencias en DML a llamadas en el lenguaje principal. El programa resultante se ejecuta entonces por el compilador del lenguaje principal, que genera el código objeto apropiado.

Hay tipos especiales de lenguajes de programación que combinan estructuras de control de lenguajes como Pascal con estructuras de control para el manejo de un objeto de la BD, como relaciones. Estos lenguajes. A veces llamados lenguajes de cuarta generación, a menudo incluyen características especiales para facilitar la generación de formas y la presentación de datos en pantalla. La mayoría de los DBMS comerciales incluyen un lenguaje de cuarta generación.

2.- Usuarios sofisticados. Interaccionan con el sistema sin escribir programas, en cambio escriben sus preguntas en un lenguaje de consultas. Cada consulta se somete a un procesador de consultas, cuya función es tomar una sentencia en DML y descomponerla en instrucciones que entienda el gestor de la BD.

3.- Usuarios especializados. Algunos usuarios sofisticados escriben aplicaciones de BD especializadas que no encajan en el marco tradicional de procesamiento de datos, como diseño asistido por computadora, sistemas basados en el conocimiento.

4.- Usuarios ingenuos. Los usuarios no sofisticados interactúan con el sistema invocando a uno de los programas de aplicación permanentes que se han escrito anteriormente.

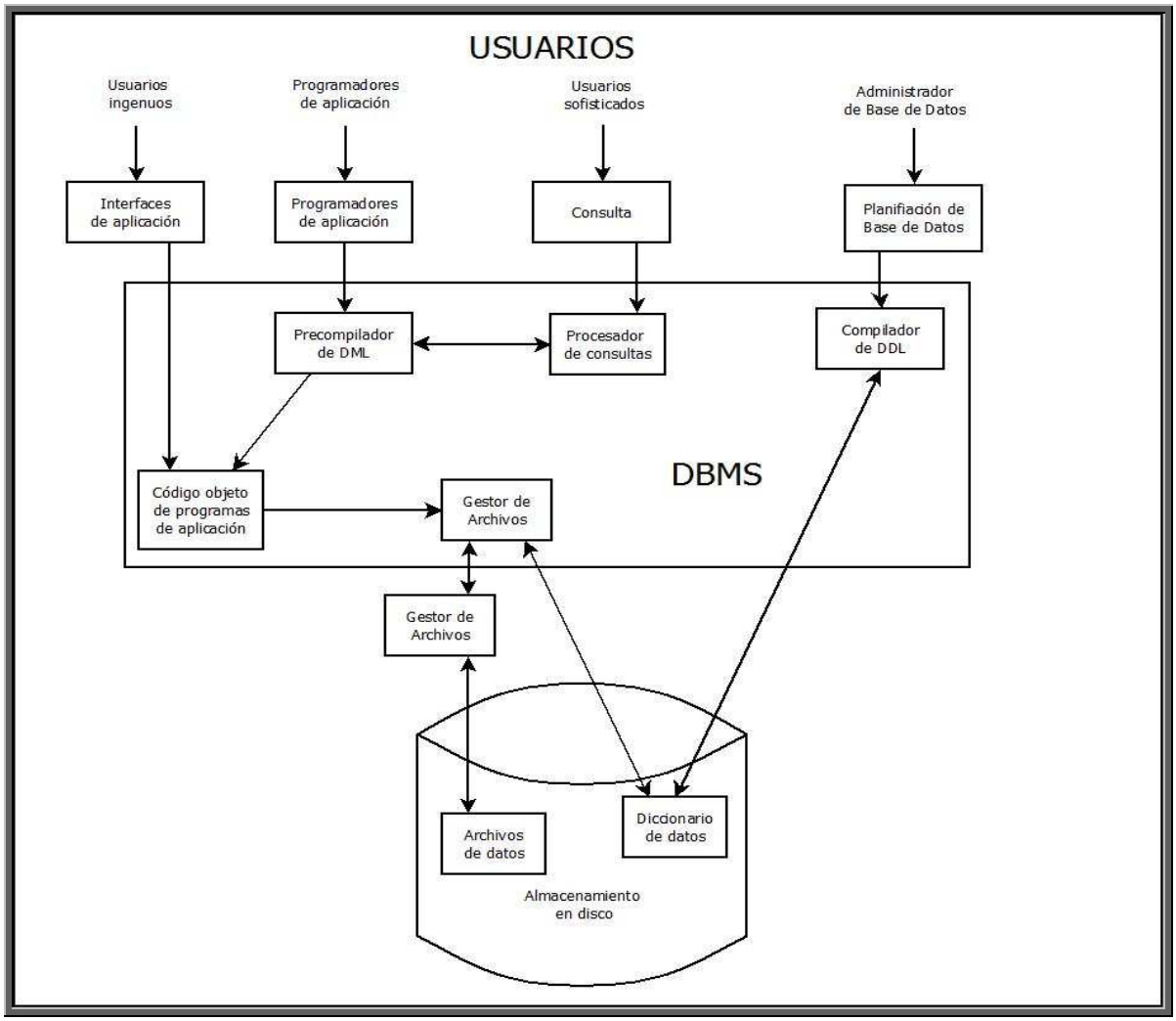


Fig. 1.10 Esquema de Usuarios de Bases de Datos

Estructura del sistema global.

Un DBMS se divide en módulos que tratan cada una de las responsabilidades del sistema general. En la mayoría de los casos, el sistema operativo proporciona únicamente los servicios básicos, y el sistema debe partir de esa base. Así, el diseño de un DBMS debe incluir la consideración del interfaz entre la BD y el sistema operativo.

Los componentes funcionales de un DBMS incluyen:

1.- Gestor de archivos. Gestiona la asignación de espacio en la memoria del disco y de las estructuras de datos usadas para representar la información almacenada en el disco.

2.- Gestor de base de datos. Proporciona el interfaz entre los datos de bajo nivel almacenados y los programas de aplicación y las consultas al sistema.

3.- Procesador de consultas. Traduce sentencias en un lenguaje de consultas a instrucciones de bajo nivel que entiende el gestor de la BD.

4.- Precompilador de DML. Convierte las sentencias DML incorporadas en un programa de aplicación en llamadas normales a procedimientos del lenguaje principal. Debe interaccionar con el procesador de consultas para generar el código apropiado.

5.- Compilador de DDL. Convierte sentencias DDL en un conjunto de tablas que contienen metadatos.

Además se requieren varias estructuras de datos como parte de la implementación del sistema físico, como:

A.- Archivos de datos. Para almacenar la BD.

B.- Diccionario de datos. Que almacena metadatos sobre la estructura de la BD. Se usa ampliamente, por lo que debe ponerse mucho énfasis en el desarrollo de un buen diseño y una implementación eficiente del diccionario.

C.- Índices. Proporcionan acceso rápido a los elementos de datos que contienen valores determinados.

1.3.2 Modelo entidad-relación.

El modelo de datos entidad-relación (E-R) se basa en una percepción de un mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos. Se desarrolló para facilitar el diseño de BD permitiendo la especificación de un esquema empresarial, que representa la estructura lógica global de la BD.

1.3.2.1 Entidades y conjuntos de entidades, Relaciones y conjuntos de relaciones, Atributos.

Una entidad es un objeto que existe y es distinguible de otros objetos, por ejemplo Juanito Garrison, con número de seguridad social 890-12-3456, es una

entidad, ya que indica únicamente una persona específica. Una entidad puede ser concreta, como un libro, o abstracta, como un concepto.

Un conjunto de entidades es un grupo de entidades del mismo tipo, por ejemplo, en un banco, el conjunto de entidades cliente. Los conjuntos de entidades no necesitan ser disjuntos, es posible definir los conjuntos de entidades de empleados y clientes de un banco, pudiendo existir una persona ser ambas o ninguna de las dos cosas.

Una entidad está representada por un conjunto de atributos, que formalmente es una función que asigna un conjunto de entidades a un dominio. Así, cada entidad se describe por medio de un conjunto de pares (atributo, valor de dato), uno para cada elemento del conjunto de entidades.

El concepto de un conjunto de entidades corresponde a la noción de definición de tipo en un lenguaje de programación.

Por tanto, una BD incluye una colección de conjuntos de entidades cada uno de los cuales contiene un número cualquiera de entidades del mismo tipo.

En este tema se tienen cinco conjuntos de entidades:

1.- Sucursal. El conjunto de todas las sucursales de un banco. Cada sucursal se describe por los atributos nombre_sucursal, ciudad_sucursal y activo.

2.- Cliente. El conjunto de todas las personas que tienen cuentas en el banco, y se describen por medio de los atributos nombre_cliente, seguridad_social, calle y ciudad_cliente.

3.- Empleado. El conjunto de todas las personas empleadas en el banco, que se describen por los atributos nombre_empleado y número_teléfono.

4.- Cuenta. El conjunto de todas las cuentas de banco, descritas por número_cuenta y saldo.

5.- Transacción. El conjunto de todas las transacciones de cuentas ejecutadas en le banco. Cada transacción se describe por los atributos número_transacción, fecha y cantidad.

Relaciones y conjuntos de relaciones.

Una relación es una asociación entre varias entidades, por ejemplo, podemos definir una relación que asocia al cliente Garrison con la cuenta 401.

Un conjunto de relaciones es un grupo de relaciones del mismo tipo. Formalmente es una relación de $n \geq 2$ conjuntos de entidades (posiblemente no

distintos). Si E_1, E_2,...,E_n son conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de

$$\{(e_1, e_2,..., e_n)|e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n\}$$

Donde $(e_1, e_2,..., e_n)$ es una relación.

La mayoría de los conjuntos de relaciones en un sistema de BD son binarios, aunque ocasionalmente hay conjuntos de relaciones que implican más de dos conjuntos de entidades, como la relación entre cliente, cuenta y sucursal. Siempre es posible sustituir un conjunto de relaciones no binario por varios conjuntos de relaciones binarias distintos. Así, conceptualmente, podemos restringir el modelo E-R para incluir sólo conjuntos binarios de relaciones, aunque no siempre es deseable.

La función que juega una entidad en una relación se llama papel, y normalmente son implícitos y no se suelen especificar. Sin embargo son útiles cuando el significado de una relación necesita ser clarificado. Tal es el caso cuando los conjuntos de entidades de un conjunto de relaciones no son distintos. Por ejemplo, el conjunto de relaciones trabaja_para podría modelarse por pares ordenados de entidades empleado. El primer empleado de cada par toma el papel de director, mientras el segundo toma el papel de trabajador.

Una relación también puede tener atributos descriptivos, por ejemplo fecha en el conjunto de relaciones cuenta_cliente que especifica la última fecha en la que el cliente tuvo acceso a su cuenta.

Atributos.

Es posible definir un conjunto de entidades y sus relaciones de varias formas diferentes. La principal diferencia está en la forma en que tratamos los diversos atributos. Considerando el conjunto de entidades empleado con atributos nombre_empleado y número_teléfono. Se puede argumentar fácilmente que un teléfono es una entidad en sí mismo con atributos número_teléfono y situación (la oficina donde está). Si se toma este punto de vista, el conjunto de entidades empleado debe redefinirse como sigue:

1.- El conjunto de entidades empleado con atributo nombre_empleado.

2.- El conjunto de entidades teléfono con atributo número_teléfono y situación.

3.- El conjunto de relaciones empltelf, que indica la asociación entre los empleados y los teléfonos que tienen.

En el primer caso, la definición implica que cada empleado tiene exactamente un número de teléfono asociado. En el segundo, la definición afirma que los

empleados pueden tener varios números de teléfono asociados. Así, la segunda definición es más general que la primera, y puede reflejar con más precisión la situación del mundo real.

No sería apropiado aplicar la misma técnica al atributo nombre_empleado, ya que es difícil argumentar que sea una entidad en si misma, así que es adecuado tener nombre_empleado como un argumento del conjunto de entidades empleado.

Surge entonces una pregunta ¿qué constituye un atributo y qué constituye un conjunto de entidades?. Desafortunadamente, no hay una respuesta sencilla, la distinción depende principalmente de la estructura de la empresa que se esté modelando y de la semántica asociada con el atributo en cuestión.

1.3.2.2 Restricciones de asignación (mapping) y Claves

Una planificación E_R de una empresa puede definir ciertas restricciones a las cuales deben ajustarse los contenidos de una BD. Una restricción importante es la de las cardinalidades de asignación, que expresa el número de entidades con las que puede asociarse otra entidad mediante un conjunto de relaciones.

Las cardinalidades de asignación son más útiles al describir conjuntos binarios de relaciones, por lo que nos abocaremos sólo en ellas.

Para un conjunto binario de relaciones R entre los conjuntos de entidades A y B, la cardinalidad de asignación debe ser una de las siguientes:

1.- Una a una. Una entidad en A está asociada a lo sumo con una entidad en B, y una de B a lo sumo con una de A.

2.- Una a muchos. Una entidad en A está asociada con un número cualquiera de entidades de B, pero una de B sólo puede estar asociada a una de A.

3.- Muchas a una. Una entidad en A está asociada con una sola de B, sin embargo, una entidad de B puede estar asociada con varias de A.

4.- Muchas a muchas. Una entidad en A está asociada con un número cualquiera en B, y una en B está asociada con un número cualquiera en A.

La cardinalidad de asignación adecuada para un conjunto de relaciones determinado es dependiente del mundo real que el conjunto de relaciones está modelando.

Las dependencias de existencia constituyen otra clase importante de restricciones. Específicamente, si la existencia de la entidad x depende de la existencia de la entidad y, entonces se dice que es dependiente por existencia de y. Operativamente, esto significa que si se suprime y, también se suprime x. La

entidad y se dice que es una entidad dominante, mientras que x se dice que es una entidad subordinada.

Para ilustrar esto, se consideran los conjuntos de entidades cuenta y transacción. Cada entidad transacción debe estar asociada con una entidad cuenta, si se suprime una entidad cuenta, entonces todas sus entidades transacción asociadas deben también suprimirse. Por el contrario, las entidades transacción pueden suprimirse de la BD sin afectar a ninguna cuenta. El conjunto de entidades cuenta es dominante, y transacción es subordinado.

Claves.

Es importante poder especificar cómo se distinguen las entidades y las relaciones. Conceptualmente, las entidades individuales y las relaciones son distintas, pero, desde la perspectiva de una BD, la diferencia entre ellas debe expresarse en términos de sus atributos. El concepto de superclave permite hacer dicha distinción.

Una superclave es un conjunto de uno o más atributos que, considerados conjuntamente, nos permiten identificar de forma única a una entidad dentro del conjunto de entidades. Por ejemplo, el atributo seguridad_social, del conjunto de entidades cliente es una superclave. Análogamente la combinación de nombre_cliente y seguridad_social es una superclave, así que si K es una superclave, entonces también lo será algún subconjunto de K. A menudo estamos interesados en superclaves mínimas para las cuales ningún subconjunto propio es superclave, y se denominan claves candidatas.

Emplearemos el término clave primaria para denotar una clave candidata que elige el diseñador de la BD como el medio principal de identificar entidades dentro de un conjunto de entidades.

Es posible que un conjunto de entidades no tenga atributos suficientes para formar una clave primaria. Un conjunto de entidades de este tipo se denomina conjunto de entidades débil, mientras que se denominará conjunto de entidades fuerte si tiene una clave primaria.

Para ilustrarlo considérese el conjunto de entidades transacción, distintas entidades asociadas a distintas cuentas pueden tener el mismo valor del atributo número_transacción. Así este conjunto de entidades es débil, ya que no tiene clave primaria. Para que un conjunto de entidades débil sea significativo, debe ser parte de un conjunto de relaciones una a muchas, y que no debe tener atributos descriptivos, ya que cualquier atributo que se necesite puede estar asociado con le conjunto de entidades débil.

Los conceptos de entidades fuertes y débiles están relacionados con las dependencias por existencia. Un miembro de un conjunto de entidades fuerte es,

por definición una entidad dominante, mientras que un miembro de un conjunto de entidades débil es una entidad subordinada por definición.

Aunque un conjunto de entidades débil no tiene clave primaria debe tener un medio de distinguir entre todas aquellas entidades en el conjunto de entidades que dependen de una entidad fuerte determinada.

El discriminador de un conjunto de entidades débil es un conjunto de atributos que permite que se haga esta distinción. Por ejemplo, el discriminador del conjunto de entidades débil transacción es el atributo número_transacción, ya que para cada cuenta un número de transacción identifica de forma única una entidad.

La clave primaria de un conjunto de entidades débil esta formada por la clave primaria del conjunto de entidades fuerte de la que depende su existencia y su discriminador. En el caso del conjunto transacción, la clave primaria es {número_cuenta, número-transacción}, donde número_cuenta identifica a la entidad dominante de una transacción y número_transacción distingue entidades transacción dentro de la misma cuenta.

Sea R un conjunto de relaciones que implica a los conjuntos E_1, E_2,..., E_n. Sea (E_i) la clave primaria que denota el conjunto de atributos que forma la clave primaria para el conjunto de entidades E_i. Supóngase que R no tiene atributos. Entonces los atributos que describen las relaciones individuales del conjunto R, denotadas por el atributo(R), son:

Clave_primaria (E_1)Ụ□Clave_primaria (E_n) Ụ... Ụ Clave_primaria (E_n)

En el caso de que R tenga atributos descriptivos, digamos {a_1, a_2,...,a_m}, entonces el conjunto atributo(R) consta de :

Clave_primaria (E_1) Ụ... Ụ □Clave_primaria (E_n) Ụ {a_1, a_2, ...,a_m}

Considérese el conjunto de relaciones CtaCli, que implica a los conjuntos de entidades cliente, con clave primaria seguridad_social, y cuenta, con clave primaria número_cuenta. Puesto que el conjunto de relaciones tiene el atributo fecha, el conjunto atributo(CtaCli) se compone de los tres atributos, seguridad_social, número_cuenta y fecha.

Ahora ya podemos explicar que constituye la clave primaria de un conjunto de relaciones R. La composición de la clave primaria depende de la cardinalidad de asignación y de la estructura de los atributos asociados con el conjunto de relaciones R.

Si el conjunto de relaciones R no tiene atributos asociados, entonces el conjunto atributo(R) forma una superclave. Esta superclave es una clave primaria si la cardinalidad de asignación es muchas a muchas.

Considérese el conjunto de relaciones CtaCli. Si el conjunto de relaciones es muchas a muchas su clave primaria será {seguridad_social, número_cuenta}, si es muchas a una, de cliente a cuenta, entonces su clave primaria será {seguridad_social}, ya que una persona puede tener una sola cuenta asociada.

Si el conjunto de relaciones R tiene varios atributos asociados con él, entonces una superclave está formada como antes, más la posible adición de uno o más de estos atributos. La estructura de la clave primaria depende tanto de la cardinalidad de asignación como de las semánticas del conjunto de relaciones.

Considérense los conjuntos de entidades cliente y banquero y un conjunto de relaciones BanqueroCli. Supóngase que este conjunto de relaciones tiene el atributo tipo asociado con el que representa la naturaleza de la relación con el cliente (prestamista o banquero). Si un banquero determinado puede representar dos papeles distintos en una relación con un cliente determinado, entonces la clave primaria de BanqueroCli consta de la unión de las claves primarias cliente y banquero, así como del atributo tipo. Sin embargo, si un banquero sólo puede tener una relación con un cliente determinado, entonces tipo no es parte de la clave primaria, que será únicamente la unión de las claves primarias cliente y banquero.

1.3.2.3 Diagrama entidad-relación.

La estructura global de una BD puede representarse gráficamente por medio de un diagrama de E-R, que consta de:

1.- Rectángulos, que representan conjuntos de entidades.
2.- Elipses, que representan atributos.
3.- Rombos, que representan conjuntos de relaciones.
4.- Líneas, que enlazan atributos a conjuntos de entidades y conjuntos de entidades a conjuntos de relaciones.

Para distinguir entre las diferentes cardinalidades, dibujaremos líneas, con o sin dirección entre los conjuntos de relaciones y el conjunto de entidades en cuestión. Una línea con dirección (→) marcará la cardinalidad una, mientras que una línea sin flecha (--) indicará la cardinalidad muchas.

Además, un conjunto de entidades débil se indica por medio de un rectángulo de doble contorno.

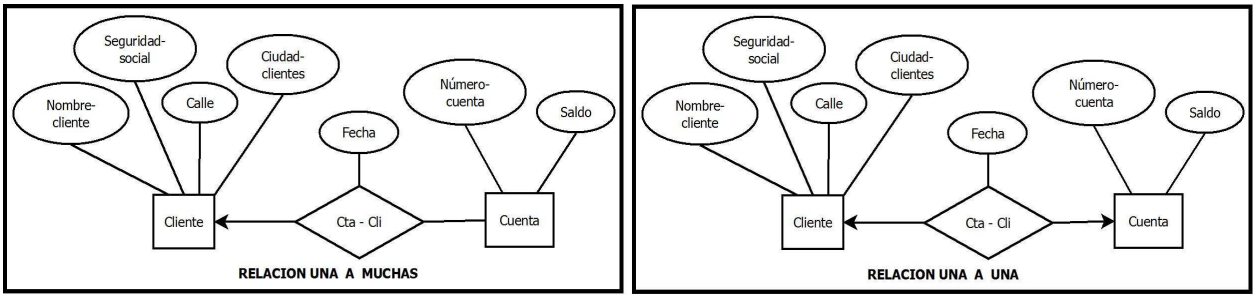


Fig. 1.11 Diagrama entidad-relación.

1.3.2.4 Reducción de los diagramas E-R a tablas.

Una BD que se ajusta a un diagrama E-R puede representarse por medio de una colección de tablas. Para cada conjunto de entidades y para cada conjunto de relaciones en la BD, existe una tabla única a la que se le asigna el nombre del conjunto correspondiente. Cada tabla tiene un número de columnas, que tienen nombres únicos.

Representación de conjuntos de entidades fuertes.

Sea E un conjunto de entidades fuerte con los atributos descriptivos a_1, a_2,...,a_n. Representamos esta entidad por medio de una tabla llamada E con n columnas distintas que corresponden a los n atributos de E. Cada fila de esta tabla corresponde a una entidad del conjunto de entidades E.

Por ejemplo, considérese el conjunto de entidades cuenta, con los atributos número_cuenta y saldo. Representaremos este conjunto mediante una tabla llamada cuenta, con dos columnas, llamadas número_cuenta y saldo. Podemos añadir una nueva cuenta a la BD con añadir una sola fila en la tabla.

Sea D_1 el conjunto de todos los números de cuenta, y sea D_2 el conjunto de todos los saldos. Cualquier fila de la tabla cuenta debe consistir en una tupla binaria (v_1, v_2), donde $v_1$es un número de cuenta, y v_2 es un saldo. En general la tabla cuenta contendrá únicamente un subconjunto de los conjuntos de todas las filas posibles. Nos referimos al conjunto de todas las filas posibles de cuenta como el producto cartesiano de D_1 y D_2, que se expresa: $D_1 \times D_2$

En general, si tenemos una tabla de n columnas, indicamos el producto cartesiano por: $D1 \times D_2 \times ... \times D_{n-1} \times D_n$

Representación de conjuntos de entidades débiles.

Sea A un conjunto de entidades débil con atributos a_1, a_2,..., a_r. Sea B el conjunto de entidades fuerte del que depende A. La clave primaria de B consta de los atributos b_1, b_2,...,b_s. Representamos el conjunto de entidades A mediante una tabla llamada A con una columna para cada atributo del conjunto:

$$\{a_1, a_2, ..., a_r\} \cup \{b_1, b_2, ..., b_s\}$$

Considérese el conjunto de entidades transacción, con atributos número_transacción, fecha y cantidad. La clave primaria del conjunto de entidades cuenta, de la que transacción es dependiente, es número_cuenta. Así, la tabla transacción tendrá cuatro columnas etiquetadas número_cuenta, número_transacción, fecha y cantidad.

Representación de conjuntos de relaciones.

Sea R un conjunto de relaciones que implica a los conjuntos de entidades E_1, E_2,..., E_m. Supongamos que atributo(R) consta de n atributos. Representamos este conjunto de relaciones mediante una tabla llamada R con n columnas distintas, cada una de las cuales corresponde a uno de los atributos de atributo(R).

Considérese el conjunto de relaciones CtaCli, que implica a los conjuntos de entidades cliente y cuenta, con claves primarias seguridad_social y número_cuenta. Puesto que el conjunto de relaciones tiene el atributo fecha, la tabla CtaCli tiene tres columnas etiquetadas seguridad_social, número_cuenta y fecha.

El caso de relaciones que conectan un conjunto de entidades débil con su correspondiente conjunto fuerte es especial. Estas relaciones son muchas a una y no tienen atributos descriptivos. Además la clave primaria de un conjunto de entidades débil incluye la del fuerte. Consideremos el conjunto de entidades débil transacción dependiente del conjunto de entidades fuerte cuenta a través del conjunto de relaciones log. La clave primaria de transacción es {número_cuenta, número_transacción}, y la de cuenta {número_cuenta}. Puesto que log no tiene atributos descriptivos, la tabla de log tendría dos columnas, número_cuenta y número_transacción. La tabla para el conjunto de entidades transacción tiene cuatro columnas, número_cuenta, número_transacción, fecha y cantidad. Así, la tabla log es redundante.

En general, la tabla para el conjunto de relaciones que conecta un conjunto de entidades débil con su correspondiente conjunto de entidades fuerte es redundante y no necesita mostrarse en una presentación tabular de un diagrama E-R.

1.3.2.5 Generalización.

Considerando el conjunto de entidades cuenta, con atributos número_cuenta y saldo. Ampliamos nuestro ejemplo anterior clasificando cada cuenta entre cuenta_ahorros y cuenta_cheques. Cada una de éstas se describe mediante un conjunto de atributos que incluye todos los atributos del conjunto de entidades cuenta más atributos adicionales.

Por ejemplo, las entidades cuenta_ahorros se describen además con el atributo tasa_interés, y las cuenta_cheques con el atributo saldo_deudor. Existen aspectos similares entre los conjuntos, en el sentido de que tienen varios atributos en común (los originales de cuenta).

Esto puede expresarse por generalización, que es una relación de inclusión que existe entre un conjunto de entidades de un nivel más alto y uno o más conjuntos de entidades de nivel más bajo. En el ejemplo anterior, cuenta es el conjunto de nivel más alto, y cuenta_ahorros y cuenta_cheques son los conjuntos de entidades de nivel más bajo.

En términos de un diagrama E-R, la generalización se representa por medio de un componente triángulo etiquetado ISA (is a, es un/a) y representa, por ejemplo, que una cuenta de ahorros es una cuenta.

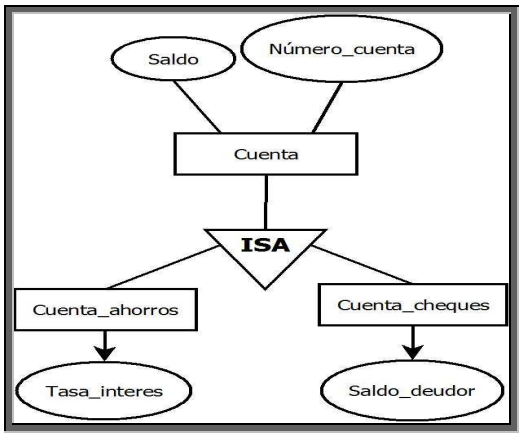


Fig. 1.12 Diagrama que muestra la Generalización.

La generalización se usa para hacer resaltar los parecidos entre tipos de entidades de nivel más bajo y ocultar sus diferencias. La distinción se hace a través de un proceso llamado herencia de atributos. Los atributos de entidades de nivel más alto se dice que son heredados por los conjuntos de entidades de nivel más bajo.

Por ejemplo, cuenta_ahorros hereda los atributos de cuenta, por lo que tendrá tres atributos, número_cuenta, saldo y tasa_interés.

Existen dos métodos diferentes para transformar un diagrama E-R que incluye generalización en una forma tabular:

1.- Crear una tabla para el conjunto de entidades de nivel más alto. Para cada conjunto de entidades de nivel más bajo, crear una tabla que incluya una columna para cada uno de los atributos de ese conjunto de entidades, más una columna para cada atributo de la clave primaria del conjunto de entidades del nivel más alto. Así, para el ejemplo anterior tendremos tres tablas: cuenta (número_cuenta,

saldo), cuenta_ahorros (número_cuenta, tasa_interés) y cuenta_cheques (número_cuenta, saldo_deudor).

2.- No crear una tabla para el conjunto de entidades de nivel más alto.

En cambio, para cada conjunto de entidades de nivel más bajo, crear una tabla que incluya una columna para cada uno de los atributos de ese conjunto de entidades, más una columna para cada atributo del conjunto de nivel más alto. Entonces, para el ejemplo, tendremos: cuenta_ahorros (número_cuenta, saldo, tasa_interés) y cuenta_cheques número_cuenta, saldo, saldo_deudor).

1.3.2.6 Agregación

Una limitación del modelo E-R es que no es posible expresar relaciones entre relaciones. Para ilustrar la necesidad de una construcción así, considérese una BD que describe información acerca de empleados que trabajan en un proyecto determinado y usan varias máquinas distintas.

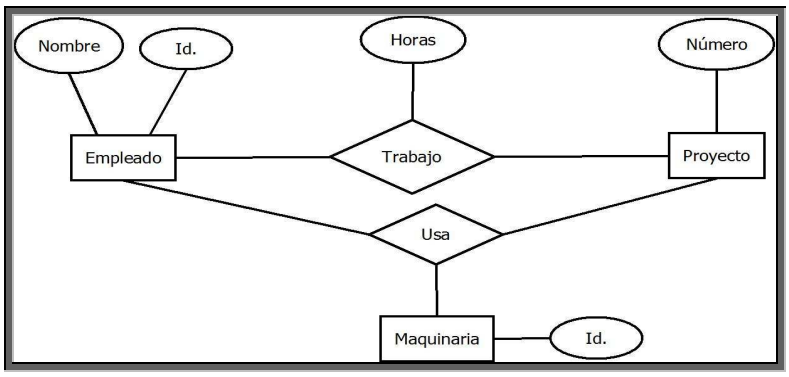


Fig. 1.13 Diagrama que muestra relación de relaciones.

Puede ocurrir que los conjuntos de relaciones Trabajo y Usa se combinen en un único conjunto de relaciones. Sin embargo, no deben combinarse, puesto que oscurecería la estructura lógica del esquema.

La solución es usar agregación. La agregación es una abstracción a través de la cual las relaciones se tratan como entidades de nivel más alto. Así, para nuestro ejemplo, recordamos el conjunto de relaciones Trabajo y los conjuntos de entidades Empleado y Proyecto como un conjunto de entidades de nivel más alto llamado Trabajo, así, se trata de la misma forma que cualquier otro conjunto de entidades.

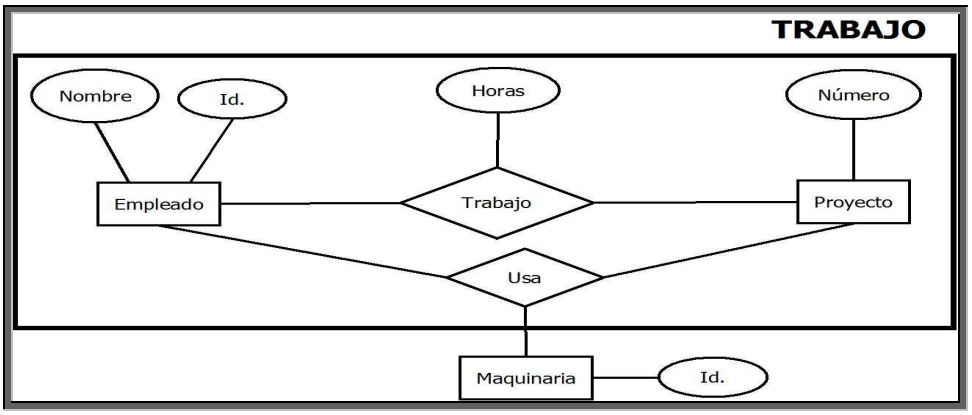


Fig. 1.14 Diagrama que muestra la Agregación.

La transformación de un diagrama E-R que incluya agregación a una forma tabular es directa, creando las tablas Empleado, Proyecto, Trabajo, Maquinaria y Usa. La tabla para el conjunto de relaciones Usa incluye una columna para cada atributo en la clave primaria del conjunto de entidades Maquinaria y del conjunto de relaciones Trabajo. También incluye una columna para cada atributo del conjunto de relaciones Usa.

1.3.2.7 Diseño de un esquema de base de datos E-R

El modelo de datos E-R proporciona un alto grado de flexibilidad en el diseño de un esquema de BD, entre una amplia variedad de alternativas, el diseñador de la BD debe tomar varias decisiones como:

1.- El uso de una relación ternaria o de un par de relaciones binarias.
2.- Si un concepto de un mundo real se expresa mejor como un conjunto entidades que como un conjunto de relaciones.
3.- El uso de un atributo o de un conjunto de entidades.
4.- El uso de un conjunto de entidades fuerte o débil.
5.- La oportunidad de usar agregación.
6.- La oportunidad de usar generalización.

Cardinalidades de asignación.

Considérese el conjunto ternario de relaciones de la figura. Esta relación especifica que un cliente puede tener varias cuentas, cada una en una sucursal, y que una cuenta puede pertenecer a varios clientes.

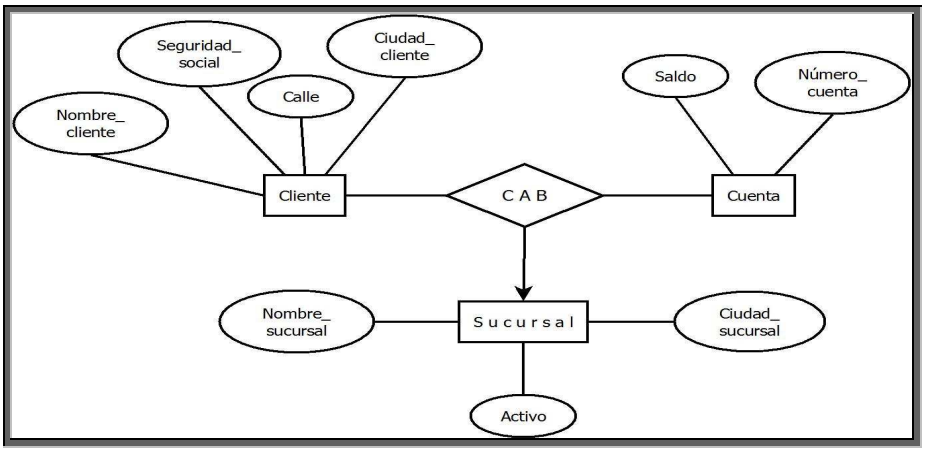


Fig. 1.15 Diagrama que muestra cardinalidades de agregación.

Este conjunto de relaciones podría sustituirse por un par de conjuntos de relaciones, como el de la siguiente figura, pero aquí, cada cuenta está situada en una sucursal. El conjunto de relaciones muchas a muchas CtaCli especifica que un cliente puede tener varias cuentas y que una cuenta puede pertenecer a varios clientes diferentes.

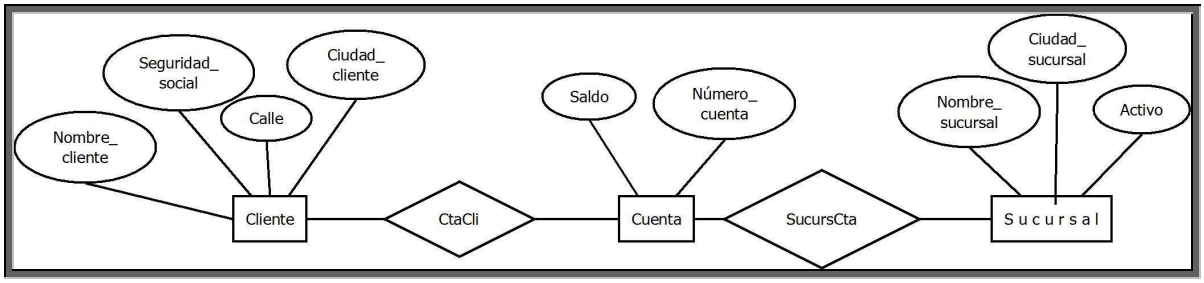


Fig. 1.16 Otra forma de realizar el diagrama entidad-relación.

Hay una sutil distinción entre ambos diagramas, en el primero, la relación entre un cliente y una cuenta puede representarse únicamente si hay una sucursal correspondiente, mientras que en el segundo, una cuenta puede estar relacionada con una sucursal sin el cliente correspondiente, o con un cliente sin la sucursal correspondiente.

Uso de conjuntos de entidades o de conjuntos de relaciones.

No siempre está claro si un objeto se expresa mejor mediante un conjunto de entidades o mediante un conjunto de relaciones. En el ejemplo desarrollado, las cuentas se pueden representar como relaciones entre clientes y sucursales, con número_cuenta y saldo como atributos descriptivos. En este diseño, un cliente puede tener una cuenta en muchas sucursales y una sucursal puede tener muchos clientes. Cada cuenta se representa mediante una relación entre un cliente y una sucursal.

Pero este diseño no puede representar convenientemente una situación en la varios clientes tienen una cuenta en común, se debe definir una relación separada para cada titular de la cuenta, con los mismos valores en los atributos descriptivos, desperdiciando espacio de almacenamiento. Sin embargo, si cada cuenta tiene exactamente un titular, el diseño será satisfactorio.

Uso de características de E-R ampliado.

El diseñador de un esquema de BD debe decidir cuando es conveniente el uso de conjuntos de entidades débiles, generalización y agregación para modelar una empresa con el modelo de datos E-R. Cada una de estas características contribuye a la modularidad del diseño:

1.- Un conjunto de entidades fuertes y sus conjuntos de entidades débiles dependientes pueden ser considerados como un objeto único en la BD, ya que las entidades débiles son dependientes por existencia de una entidad fuerte.

2.- La agregación agrupa una parte del diagrama E-R en un conjunto de entidades únicas. Es posible tratar el conjunto de entidades agregado como una unidad única sin preocuparse de los detalles de estructura interna.

3.- Generalización, o jerarquía de relaciones ISA, contribuye a la modularidad permitiendo que atributos comunes de conjuntos de entidades similares sean representados una sola vez en un diagrama E-R.

Sin embargo, el uso excesivo de estas características puede introducir una complejidad innecesaria en el diseño.

1.3.3 Modelo relacional.

El modelo de datos relacional es relativamente nuevo. Tras la introducción del modelo relacional se ha desarrollado una teoría esencial para las BD relacionales, que ayuda al diseño de BD relacionales y al procesamiento eficiente de solicitudes de información.

El modelo relacional se ha establecido como el principal modelo de datos para aplicaciones comerciales de procesamiento de datos.

1.3.3.1 Estructura de las bases de datos relacionales.

Una BD relacional consiste en una colección de tablas, a cada una de las cuales se les asigna un nombre único.

Una fila de la tabla representa una relación entre un conjunto de valores, puesto que una tabla es una colección de dichas relaciones, hay una estrecha correspondencia entre el concepto de tabla y el concepto matemático de relación, del cual toma su nombre el modelo de datos relacional.

Estructuras básicas.

Considérese la tabla depósito, con cuatro atributos: nombre_sucursal, número_cuenta, nombre_cliente y saldo. Para cada atributo hay un conjunto de valores permitidos, llamado dominio, de este atributo, por ejemplo, para nombre_sucursal, el dominio será el conjunto de todos los nombres de las sucursales.

Sea D_1 el dominio de nombre_sucursal, y D_2, D_3, D_4 los dominios de los otros atributos. Cada una de las filas de depósito debe constar de 4 tuplas (v_1, v_2, v_3, v_4), perteneciendo cada una a su dominio correspondiente. En general, depósito contendrá únicamente un subconjunto del conjunto de todas las filas posibles, por tanto, depósito es un subconjunto de:

$$D_1 \times D_2 \times D_3 \times D_4$$

En general, una tabla de n columnas debe ser un subconjunto de:

$$D_1 \times D_2 \times ... \times D_{n-1} \times D_n$$

Los matemáticos definen una relación como un subconjunto de un producto cartesiano de una lista de dominios, lo que se corresponde casi exactamente con nuestra definición de tabla.

Puesto que las tablas son esencialmente relaciones, usamos los términos matemáticos relación y tupla en lugar de los términos tabla y fila.

Sea la variable tupla t la primera tupla de la relación. Usamos la notación t[nombre_sucursal] para indicar el valor de t en el atributo nombre_sucursal, alternativamente, podemos escribir t[1] para indicarlo. Puesto que una relación es un conjunto de tuplas, usamos la notación matemática t∈r para indicar que la tupla t está en la relación r.

Necesitaremos que, para todas las relaciones r, los dominios de todos los atributos sean atómicos, entendiendo como tal a aquel en el que sus elementos se consideran unidades indivisibles, por ejemplo, los números naturales. En todos nuestros ejemplos supondremos dominios atómicos.

Esquema de la base de datos.

Cuando se habla de una BD debemos diferenciar entre el esquema de la BD o el diseño lógico de la BD, y una instancia de la BD, que son los datos en la BD en un instante de tiempo dado. El concepto de esquema de una relación corresponde a la noción de definición de tipo en los lenguajes de programación, mientras que el de instancia de una relación se corresponde con el de variable.

Es conveniente dar un nombre al esquema de una relación, por lo que adoptamos el convenio de usar nombres en minúsculas para relaciones y nombres, empezando por una letra mayúscula para los esquemas de relaciones.

Siguiendo esta notación usamos esquema_depósito para indicar el esquema de relación para la relación depósito. Así:

Esquema_depósito = (nombre_sucursal, número_cuenta, nombre_cliente, saldo)

Indicamos el hecho de que depósito es una relación sobre el esquema depósito por:

depósito (esquema_depósito)

En general, el esquema de una relación es una lista de atributos y sus correspondientes dominios. No nos preocuparemos, por ahora, de dar una definición precisa del dominio de cada atributo, sin embargo, cuando queramos definir nuestros dominios, para definir el esquema de relación para la relación depósito, usaremos la notación:

(nombre_sucursal: cadena, número_cuenta: entero, nombre_cliente: cadena, saldo: entero)

Considérese la relación cliente, con el esquema para esa relación que sigue:

Esquema_cliente = (nombre_cliente, calle, ciudad_cliente)

Obsérvese que el atributo nombre_cliente aparece en los dos esquemas de relaciones. Esto no es una coincidencia, el uso de atributos comunes en esquemas de relaciones es una forma de relacionar tuplas de distintas relaciones.

Se podría pensar en términos de un esquema de relaciones en vez de varios. Supóngase que usamos sólo una relación para nuestro ejemplo con el esquema:

Esquema_info_cuenta = (nombre_sucursal, número_cuenta, nombre_cliente, saldo, calle, ciudad_cliente)

Obsérvese que si un cliente tiene varias cuentas debemos listar su dirección una vez para cada cuenta, es decir, debemos repetir información, lo que supone un desperdicio y se evita mediante el uso de dos relaciones.

Además, si un cliente tiene una o más cuentas pero no ha dado una dirección, no podemos construir una tupla en esquema_info_cuenta, ya que no se conocen los valores para calle y ciudad_cliente. Para representar tuplas incompletas debemos usar valores nulos. Usando dos relaciones, podemos representar clientes cuya dirección se desconozca sin usar valores nulos, simplemente usamos una tupla en esquema_depósito y no creamos tupla en esquema_cliente hasta que la información esté disponible.

No siempre es posible eliminar los valores nulos, supóngase que incluimos el atributo número_teléfono en el esquema_cliente. Puede ocurrir que un cliente no tenga teléfono, entonces tendríamos que recurrir a los valores nulos.

Veremos más adelante que los valores nulos causan diversas dificultades en el acceso o actualización de la BD y, por tanto, deben eliminarse cuando sea posible.

Para el propósito de este capítulo supondremos una empresa bancaria con el diagrama E-R de la figura. Las claves primarias para los conjuntos entidad cliente y sucursal son nombre_cliente y nombre_sucursal. Los esquemas de relaciones para esta empresa son:

Esquema_sucursal = (nombre_sucursal, activo, ciudad_sucursal)
Esquema_cliente = (nombre_cliente, calle, ciudad_cliente)
Esquema_depósito = (nombre_sucursal, número_cuenta, nombre_cliente, saldo)
Esquema_préstamo = (nombre_sucursal, número_préstamo, nombre_cliente, cantidad)

Claves.

Las nociones de superclave, clave candidata y clave primaria también pueden aplicarse al modelo relacional, por ejemplo, en esquema_sucursal, {nombre_sucursal} y {nombre_sucursal, ciudad_sucursal} son superclaves, la primera es una clave candidata, no así la segunda, que contiene a la primara, por lo que la primera es la clave primaria, y la clave primaria para esquema_cliente es {nombre_cliente}.

Sea R un esquema de relación. Si decimos que un subconjunto K de R es una superclave para R estamos restringiéndonos a considerar las relaciones r(R), en las que no haya dos tuplas distintas que tengan los mismos valores en todos los atributos de K. Es decir, si t_1 y t_2 están en r y $t_1 \neq t_2$, entonces $t_1[K] \neq t_2[K]$.

Lenguajes de consulta.

Un lenguaje de consulta es un lenguaje en el que el usuario solicita información de la BD. Son, normalmente, de más alto nivel que los estándar de programación. Los lenguajes de consulta pueden clasificarse en lenguajes procedimentales y no procedimentales. En un lenguaje procedimental, el usuario da instrucciones al sistema para que realice una serie de operaciones en la BD para calcular el resultado deseado. En uno no procedimental, el usuario describe la información deseada sin dar un procedimiento específico para obtenerla.

La mayor parte de los sistemas comerciales de BD relacionales, ofrecen un lenguaje de consulta que incluye elementos de los dos enfoques.

En este capítulo examinamos dos lenguajes puros, el álgebra relacional es procedimental, mientras que el cálculo relacional de tuplas y el cálculo relacional

de dominios son no procedimentales. Estos lenguajes de consulta son concisos y formales, pero ilustran las técnicas fundamentales para extraer datos de la BD.

Inicialmente nos ocuparemos únicamente por las consultas, aunque un lenguaje de manipulación de datos completo incluye, además de un lenguaje de consulta, un lenguaje para la modificación de la BD.

1.3.3.2 El álgebra relacional.

El álgebra relacional es un lenguaje de consulta procedimental. Consta de un conjunto de operaciones que toman una o dos relaciones como entrada y producen una nueva relación como resultado. Las operaciones fundamentales en el álgebra relacional son seleccionar, proyectar, producto cartesiano, renombrar, unión y diferencia de conjuntos. Además, existen otras operaciones, intersección de conjuntos, producto natural, división y asignación, que se definirán en términos de las operaciones fundamentales.

Operaciones fundamentales.

Las operaciones seleccionar, proyectar y renombrar se llaman operaciones unitarias, ya que operan sobre una relación. Las otras tres operaciones operan sobre pares de relaciones y se llaman operaciones binarias.

a) La operación seleccionar.

La operación seleccionar selecciona tuplas que satisfacen un predicado dado. Usamos la letra griega minúscula sigma (**σ**) para indicar la selección. El predicado aparece como subíndice de **σ**. Así, para seleccionar aquellas tuplas de la relación préstamo en las que la sucursal es “Puebla”, escribimos:

$$\sigma_{\text{nombre\_sucursal=”Puebla”}}(\text{préstamo})$$

En general, permitimos las comparaciones usando =, ≠, <, ≤, >, ≥, en el predicado de selección. Además, pueden combinarse varios predicados en un predicado más complejo usando los conectores y(Λ) / o(V). Así:

$$\sigma_{\text{nombre\_sucursal=”Puebla” Λcantidad>1200}}(\text{préstamo})$$

El predicado de selección puede incluir comparaciones entre dos atributos, como: $\sigma_{\text{nombre\_cliente=nombre\_banquero}}(\text{servicio})$

b) La operación proyectar.

La operación proyectar es una operación unitaria que devuelve su relación argumento con ciertas columnas omitidas. Puesto que una relación es un conjunto, se eliminan todas las filas duplicadas. La proyección se indica con la letra griega pi (**π**). Listamos los atributos que queremos que aparezcan en el

resultado como subíndices de **π**. La relación argumento se escribe a continuación de **π** entre paréntesis. Así:

$$\pi_{\text{nombre\_sucursal, nombre\_cliente}} (\text{préstamo})$$

Mostrará solo los atributos nombre_sucursal y nombre_cliente.

En vez de dar una relación como argumento podemos dar el resultado de otra operación, como $\pi_{\text{nombre\_cliente}} (\sigma_{\text{nombre\_cliente=nombre\_banquero}} (\text{servicio}))$

c) La operación producto cartesiano.

Las operaciones que hemos tratado hasta ahora nos permiten extraer información únicamente de una relación cada vez. La operación producto cartesiano, representada por una cruz (x) es una operación binaria. Describimos el producto cartesiano de las relaciones r_1 y r_2 como r_1 x r_2. Recorando que una relación se define como un subconjunto de un producto cartesiano de un conjunto de dominios. Sin embargo, nos enfrentamos al problema de elegir los nombres de los atributos para la relación que resulta de un producto cartesiano.

Suponiendo que queremos encontrar a todos los clientes del banquero Juan, así como las ciudades en las que viven. El esquema de relaciones para r es:
(servicio.nombre_cliente, servicio.nombre_banquero, cliente.nombre_cliente, cliente.calle, cliente.ciudad_cliente)

Es decir, simplemente listamos todos los atributos de las dos relaciones, y adjuntamos el nombre de la relación de la que procede el atributo. Necesitamos adjuntar el nombre de la relación para distinguir entre servicio.nombre_cliente y cliente.nombre_cliente. Para aquellos atributos que solo aparecen en uno de los dos esquemas omitiremos el prefijo nombre_relación, lo que no producirá ninguna ambigüedad:

(servicio.nombre_cliente, nombre_banquero, cliente.nombre_cliente, calle, ciudad_cliente)

Ahora que ya conocemos el esquema de relaciones para r=servicio x cliente, ¿qué tuplas aparecen en r?. Construiremos una tupla de r por cada par posible de tuplas: una de la relación servicio y otra de cliente.

Suponiendo que tenemos n_1 tuplas en servicio y n_2 tuplas en cliente. Entonces existen n_1n_2 formas de elegir un par de tuplas, de forma que hay n_1n_2 tuplas en r. Obsérvese que puede darse el caso para algunas tuplas t en r que t[servicio.nombre_cliente] ≠ t[cliente.nombre_cliente].

En general, si tenemos las relaciones $r_1(R_1)$ y $r_2(R_2)$, entonces r_1xr_2 es una relación cuyo esquema es la concatenación de R_1 y R_2. La relación R contiene todas las tuplas t para las cuales existe una tupla t_1 en r_1 y t_2 en r_2 para las que $t[R_1]=t_1[R_1]$ y $t[R_2]=t2[R_2]$.

Volviendo a la pregunta, (clientes de Juan y sus ciudades), considerando r=servicioxcliente, tenemos:

$$\sigma_{\text{nombre\_banquero= “Juan”}} (\text{servicioxcliente})$$

Puesto que la operación producto cartesiano asocia todas las tuplas de cliente con todas las tuplas de servicio, sabemos que alguna tupla en servicioxcliente tiene la dirección del cliente del banquero. Esto ocurre en aquellos casos en los que sucede que servicio.nombre_cliente=cliente.nombre_cliente, así que escribimos:

$$\sigma_{\text{servicio.nombre\_cliente=cliente.nombre\_cliente}} (\sigma_{\text{nombre\_banquero= “Juan”}} (\text{servicioxcliente}))$$

Finalmente, puesto que solo queremos el nombre y la ciudad del cliente debemos realizar una proyección:

$$\pi_{\text{servicio.nombre\_cliente, ciudad\_cliente}} (\sigma_{\text{servicio.nombre\_cliente=cliente.nombre\_cliente}} (\sigma_{\text{nombre\_banquero= “Juan”}} (\text{servicioxcliente})))$$

d) La operación renombrar.

Introdujimos el convenio de nombrar atributos nombre_relación.nombre_atributo para eliminar ambigüedades. Otra forma de ambigüedad potencial surge cuando la misma relación aparece más de una vez en una pregunta.

Considérese encontrar los nombres de todos los clientes que viven en la misma calle y ciudad que Sánchez. Podemos encontrar la calle y la ciudad de Sánchez escribiendo:

$$\pi_{\text{calle, ciudad\_cliente}} (\sigma_{\text{nombre\_cliente=”Sánchez”}} (\text{cliente}))$$

Sin embargo, para encontrar otros clientes con esa calle y esa ciudad debemos hacer referencia una segunda vez a la relación cliente

$$\sigma_{P(\text{cliente})} \text{ X } \pi_{\text{calle, ciudad\_cliente}} (\sigma_{\text{nombre\_cliente=”Sánchez”}} (\text{cliente}))$$

Donde P es un predicado de selección que requiere que los valores de calle y ciudad_cliente sean iguales. Para especificar a qué valor de calle nos referimos, no podemos usar cliente.calle, ya que ambos valores de calle se toman de la misma relación cliente, y lo mismo pasa con ciudad_cliente. Este problema se resuelve usando el operando renombrar, representado por **ρ**.

La expresión $\rho_x(r)$ devuelve la relación r con el nombre x. Usaremos ésta para renombrar una referencia a la relación cliente, y así hacer referencia a la relación dos veces sin ambigüedad.

$$\pi_{\text{cliente.nombre\_cliente}}(\sigma_{\text{cliente2.calle=cliente.calle} \wedge \text{ cliente2.ciudad\_cliente=cliente.ciudad\_cliente}} (\text{cliente x } (\pi_{\text{calle,ciudad\_cliente}} (\sigma_{\text{nombre\_cliente=”Sánchez”}}(\rho_{\text{cliente2}}(\text{cliente}))))))$$

e) La operación unión.

Consideremos ahora una consulta que podría plantearse el departamento de publicidad de un banco: “encontrar a todos los clientes de una sucursal”, es decir, los que tienen una cuenta, un préstamo o ambos.

Sabemos encontrar a todos los clientes con un préstamo y a los de una cuenta. Para contestar la consulta, necesitamos la unión de estos dos conjuntos. Esto se logra mediante la operación binaria unión (**U**), y será:

$\Pi_{nombre\_cliente}(\sigma_{nombre\_sucursal=”Nombre”}(préstamo) \cup \Pi_{nombre\_cliente}(\sigma_{nombre\_sucursal=”nombre”}(depósito)))$

En nuestro ejemplo tomamos la unión de dos conjuntos, ambos formados por valores de nombre_cliente.

En general debemos asegurarnos de que las uniones se toman entre relaciones compatibles. Por tanto, para que una operación de unión r**u**s sea válida exigimos que se cumplan dos condiciones:

1.- Las relaciones r y s deben tener el mismo número de atributos.

2.- Los dominios del atributo iésimo de r y del atributo iésimo de s deben ser los mismos.

f) La operación diferencia de conjuntos.

La operación diferencia de conjuntos (-) nos permite encontrar tuplas que estén en una relación pero no en otra.

La expresión r-s da como resultado una relación que contiene aquellas tuplas que están en r pero no en s.

Podemos encontrar a todos los clientes de una sucursal que tiene cuenta allí, pero no un préstamo:

$\Pi_{nombre\_cliente}(\sigma_{nombre\_sucursal=”Nombre”}(depósito) - \Pi_{nombre\_cliente}(\sigma_{nombre\_sucursal=”nombre”}(préstamo)))$

Considérese la consulta “encontrar el saldo de cuenta mayor en el banco”. Nuestra estrategia para hacer esto es calcular primero una estrategia temporal que conste de aquellos saldos que no son los más grandes, y entonces tomar la diferencia de conjuntos entre la relación depósito y la relación temporal que acaba de ser calculada para obtener el resultado. La relación temporal se calcula por medio de:

$$\Pi_{saldo.depósito}(\sigma_{saldo.depósito<d.saldo}(depósito \times \rho_{d}(depósito)))$$

El resultado contiene todos los saldos excepto el mayor de todos los saldos menos el mayor de todos. La consulta puede escribirse tomando la diferencia entre el conjunto de todos los saldos y la relación temporal:

$$\pi_{saldo}\ (\text{depósito}) - \pi_{saldo.depósito}\ (\sigma_{saldo.depósito<d.saldo}\ (\text{depósito} \times \rho_{d}\ (\text{depósito})))$$

Definición formal del álgebra relacional.

Una expresión básica en el álgebra relacional consta de cualquiera de las siguientes: una relación de la BD, y una relación constante.

Una expresión general en el álgebra relacional se construye a partir de subexpresiones. Sean E1 y E2 expresiones del álgebra relacional. Entonces las siguientes son todas expresiones del álgebra relacional:

$$E_1 \cup E_2,\ E_1 - E_2,\ E_1 \times E_2,\ \sigma_P(E_1),\ \pi_S(E_1),\ \rho_X(E_1)$$

Operaciones adicionales.

Las operaciones fundamentales del álgebra relacional son suficientes para expresar cualquier consulta en álgebra relacional. Sin embargo, si nos restringimos sólo a las operaciones fundamentales, algunas consultas comunes son largas de expresar. Por tanto definimos operaciones adicionales que no añaden ninguna potencia al álgebra, pero que simplifican consultas comunes.

a) La operación intersección de conjuntos.

La intersección de conjuntos ($\cap$) comunes a dos relaciones, por ejemplo encontrar los clientes que tienen una cuenta y un préstamo en una sucursal sería:

$$\pi_{nombre\_cliente}(\sigma_{nombre\_sucursal=\text{"Nombre"}}(\text{préstamo}) \cap \pi_{nombre\_cliente}(\sigma_{nombre\_sucursal=\text{"nombre"}}(\text{depósito})))$$

Cualquier expresión del álgebra relacional que use la intersección de conjuntos puede reescribirse sustituyendo la operación intersección por un par de operaciones diferencia de conjuntos: $r \cap s = r-(r-s)$.

Así, la intersección de conjuntos no es una operación fundamental y no añade potencia al álgebra.

b) La operación producto natural.

Típicamente, una consulta que implica un producto cartesiano incluye una operación de selección en el resultado del producto. Considérese la consulta “encontrar a todos los clientes que tiene un préstamo y las ciudades en las que viven”. Primero formamos el producto cartesiano de las relaciones préstamo y

cliente, después seleccionamos aquellas tuplas que presenten un único nombre_cliente.

$$\pi_{\text{préstamo.nombre\_cliente,ciudad\_cliente}}(\sigma_{\text{prestamo.nombre\_cliente=cliente.nombre\_cliente}}(\text{préstamo} \mathbf{x} \text{ cliente}))$$

El producto natural es una operación binaria que nos permite combinar ciertas selecciones y un producto cartesiano en una operación. Se representa por el símbolo |x|. La operación producto natural forma un producto cartesiano de sus dos argumentos, realiza una selección formando la igualdad en aquellos atributos que aparezcan en ambas relaciones y, finalmente quita las columnas duplicadas. Así, la consulta anterior sería:

$$\pi_{\text{nombre\_cliente, ciudad\_cliente}}(\text{préstamo } |x| \text{ cliente})$$

Puesto que muchos esquemas de préstamo y cliente tiene el atributo nombre_cliente en común, la operación producto natural considera sólo pares de tuplas que tienen el mismo valor de nombre_cliente. Combina cada uno de estos pares de tuplas en un único par en la unión de los dos esquemas.

Ahora estamos preparados para la definición formal del producto natural. Considérese dos esquemas de relaciones R y S, que son listas de nombres de atributos. Si consideramos los esquemas como conjuntos mejor que como listas, podemos representar aquellos atributos que están en R y S por R∩S, y representar aquellos valores que aparecen en R, en S o en ambas por R**u**S. Nótese que nos referimos a conjuntos de atributos, no de relaciones.

Considerando las dos relaciones r(R) y s(S). El producto natural de r y s, representado por r |x| s es una relación del esquema R**u**S. Es la proyección sobre R**u**S de una selección en r x s, donde el predicado de selección requiere que r.A=s.A para cada atributo A en R∩S. Formalmente:

$$r \,|x|\, s = \Pi_{R \cup S}(\sigma_{r.A1=s.A1 \wedge r.A2=s.A2 \wedge \ldots \wedge r.An=s.An}(r \times s))$$

Como propiedad, dígase que el producto natural es asociativo, y además, sean r(R) y s(S) relaciones sin ningún atributo en común, es decir R∩S= Φ. Entonces r |x| s = r x s.

Puesto que el producto natural es muy importante, damos varios ejemplos de su uso:

1.- Encontrar el activo y el nombre de todas las sucursales que tienen depositantes que viven en Saltillo.

$$\Pi_{\text{nombre\_sucursal, activo}}(\sigma_{\text{ciudad\_cliente=”Saltillo”}}(\text{cliente } |x| \text{ depósito } |x| \text{ sucursal}))$$

2.- Encontrar a todos los clientes que tienen una cuenta y un préstamo en la sucursal Puebla.

$\Pi_{nombre\_cliente}(\sigma_{nombre\_sucursal="Puebla"}(préstamo \mid x \mid depósito))$

c) La operación división.

La operación división (÷) se establece para aquellas consultas que incluyen la frase “para todos. Supóngase que queremos encontrar a todos los clientes que tienen una cuenta en todas las sucursales que están en Veracruz. Podemos obtener todas las sucursales en Veracruz mediante:

$r_1 = \Pi_{nombre\_sucursal}(\sigma_{ciudad\_sucursal="Veracruz"}(depósito))$

Podemos encontrar todos los pares nombre_cliente, nombre_sucursal para los cuales el cliente tiene una cuenta en una sucursal escribiendo:

$r_2 = \Pi_{nombre\_cliente,\ nombre\_sucursal}(depósito)$

Ahora necesitamos encontrar los clientes que aparecen en r_2 con cada nombre de sucursal en r_1.La operación que proporciona exactamente esos clientes es la operación de dividir. La consulta puede expresarse con:

$\Pi_{nombre\_cliente,nombre\_sucursal}(depósito) \div \Pi_{nombre\_sucursal}(\sigma_{ciudad\_sucursal="Veracruz"}(depósito))$

Formalmente, sean r(R) y s(S) relaciones, y $S \subseteq R$. La relación r÷s es una relación de esquema R-S. Una tupla t está en r÷s si para cada tupla t_s es s existe una tupla t_r en r que satisface estas dos condiciones:

$t_s[R] = t_r[S]$
$t_s[R-S] = t[R-S]$

Definida en términos de operaciones fundamentales, la operación dividir resulta:

$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - r)$

d) La operación asignación.

A veces es conveniente escribir una expresión del álgebra relacional por partes usando la asignación a una variable de relación temporal. La operación asignación (←), funciona de forma parecida a la asignación de los lenguajes de programación.

La evaluación de una asignación no da como resultado una relación que se presenta al usuario, más bien, el resultado de la expresión a la derecha de ← es asignado a la variable de relación de la parte izquierda. Esta variable de relación puede usarse en subsiguientes expresiones.

Con la operación asignación, una consulta puede escribirse como un programa secuencial que consta de una serie de asignaciones seguidas de una expresión

cuyo valor se presenta como el resultado de la consulta. En álgebra relacional, la asignación debe hacerse siempre a una variable de relación temporal. Es importante observar que la operación de asignación no proporciona potencia adicional al álgebra, es una forma conveniente de expresar consultas complejas de forma más sencilla.

1.3.3.3 El cálculo relacional de tuplas.

Cuando escribimos una expresión en álgebra relacional, damos una secuencia de procedimientos que genera la respuesta a nuestra consulta. El cálculo relacional de tuplas, en cambio, es un lenguaje de consultas no procedimental.

Describe la información deseada sin dar un procedimiento específico para obtener esa información.

Una consulta en el cálculo relacional de tuplas se expresa como: {t|P(t)}

Es decir, el conjunto de todas las tuplas t, tal que el predicado P es verdadero para t. Usamos t[A] para representar el valor de la tupla t en el atributo A, y usamos tЄr para indicar que la tupla t está en la relación r.

Consultas ejemplo.

Encontrar el nombre_sucursal, número_préstamo, nombre_cliente y cantidad para préstamos mayores de 1200 dólares:

{t | tЄpréstamo Λ t[cantidad]>1200}

Suponiendo que queremos únicamente el atributo nombre_cliente. En el cálculo relacional de tuplas necesitamos escribir una expresión para una relación sobre el esquema (nombre_cliente). Necesitamos aquellas tuplas en (nombre_cliente) tales que exista una tupla en préstamo correspondiente a ese nombre_cliente con el atributo cantidad>1200. Para expresar esto necesitamos la construcción existe (∃) de la lógica matemática ∃tЄr(Q(t)) que significa “existe una tupla t en la relación r tal que el predicado Q(t) es verdadero”.

Usando esta notación podemos escribir la consulta “encontrar a todos los clientes con un préstamo superior a 1200” como:

{t| ∃s Є préstamo(t[nombre_cliente] = s[nombre_cliente] Λ s[cantidad]>1200}

La expresión anterior se lee “el conjunto de todas las tuplas t tal que existe una tupla s en la relación préstamo para la cual los valores de t y s para el atributo nombre_cliente son iguales y el valor de s para el atributo cantidad es mayor de 1200 dólares”.

La variable de tupla t se define sólo en el atributo nombre_cliente, puesto que éste es el único atributo para el que se especifica una condición para t. Así, el resultado es una relación sobre (nombre_cliente).

Considérese la consulta “encontrar a todos los clientes que tienen un préstamo de la sucursal Puebla y las ciudades en las que viven”. Esta consulta implica dos relaciones, por lo que requiere que tengamos dos clausulas conectadas por un y(Λ).

{t| ∃s Є préstamo(t[nombre_cliente]=s[nombre_cliente]Λs[nombre_sucursal]= “Puebla”Λ∃ u Є cliente(u[nombre_cliente] =s[nombre_cliente] Λ t[ciudad_cliente] = u[ciudad_cliente]))}

Este es “el conjunto de todas las tuplas (nombre_cliente, ciudad_cliente) para las cuales nombre_cliente es un receptor de préstamo en la sucursal Puebla y ciudad_cliente es la ciudad de nombre_cliente”. La variable de tupla s asegura que el cliente tiene un préstamo en la sucursal Puebla, y u tiene la restricción que debe pertenecer al mismo cliente que s, y u asegura que ciudad_cliente es la ciudad del cliente.

Ahora considérese la consulta “encontrar a todos los clientes que tienen una cuenta en Puebla pero no un préstamo en ella”.

{t| ∃u Є depósito (t[nombre_cliente] = u[nombre_cliente] Λ u[nombre_sucursal] = “Puebla”Λ¬∃s Є préstamo(t[nombre_cliente] = s[nombre_cliente] Λ s[nombre_sucursal] = “Puebla”)}

Esta expresión usa la cláusula ∃u Є depósito (…) para exigir que el cliente tenga una cuenta en Puebla y la cláusula ¬∃s Є préstamo (…) para eliminar los que tengan también un préstamo.

La consulta que vamos a considerar a continuación usa la implicación (⇒). La fórmula P⇒Q significa “P implica Q”; es decir, si P es verdadera, Q debe ser verdadera, y es equivalente lógicamente a ¬PVQ. El uso de la implicación a menudo sugiere una interpretación más intuitiva de una consulta.

Considérese la consulta “encontrar a todos los clientes que tienen una cuenta en todas las sucursales de Veracruz”. Para esta consulta introducimos la construcción “para todos”, representada por ∀. La notación ∀t Є r(Q(t)) significa “Q es verdadera para todas las tuplas t en la relación r”. La consulta sería:
{t| ∀ u Є sucursal(u[ciudad_sucursal]=”Veracruz”⇒ ∃s Є depósito(t[nombre_cliente] = s[nombre_cliente]Λu[nombre_sucursal]=s[nombre_sucursal]))}

Interpretamos la expresión como el conjunto de todos los clientes (t[nombre_cliente]) tal que para todas las tuplas u en la relación sucursal, si el

valor de u en el atributo ciudad_sucursal es Veracruz entonces el cliente tiene una cuenta en la sucursal cuyo nombre aparece en el atributo nombre_sucursal de u.

Definición formal.

Ahora estamos preparados para la definición formal. Una expresión del cálculo relacional de tuplas es de la forma: {t|P(t)}

donde P es una fórmula. En una fórmula pueden aparecer varias variables de tuplas. Se dice que una variable de tupla es una variable libre a menos que este cuantificada por un $\exists$ o por un $\forall$, que entonces se dice variable límite.

Una fórmula en el cálculo relacional de tuplas se compone de átomos. Un átomo tiene una de las siguientes formas:

1.- s$\in$r, donde s es una variable de tupla y r es una relación. No se permite la operación $\notin$.

2.- s[x]Θu[y], donde s y u son variables de tuplas, x e y son atributos sobre los que están definidos s y u respectivamente, y Θ es un operador de comparación (<, $\leq$, =, $\geq$, >). Se requiere que los atributos x e y tengan dominios cuyos miembros puedan compararse por medio de Θ.

3.- s[x] Θc, donde s es una variable de tupla, x es una atributo sobre el que s esta definida, Θ es un operador de comparación, y c es una constante en el dominio del atributo x.

Las fórmulas se construyen a partir de átomos usando las siguientes reglas:
1.- Un átomo es una fórmula.
2.- Si P_1 es una fórmula, entonces también lo son $\neg P_1$ y (P_1).
3.- Si P_1 y P_2 son fórmulas, entonces también lo son $P_1 \wedge P_2$, $P_1 \vee P_2$ y $P_1 \Rightarrow P_2$.
4.- Si $P_1(s)$ es una fórmula que contiene una variable de tupla libre, entonces $\exists s \in r(P_1(s))$ y $\forall s \in r(P_1(s))$ también lo son.

Como en el álgebra relacional, es posible escribir expresiones equivalentes que en apariencia no son idénticas, como estas tres:

1.- $P_1 \wedge P_2$ es equivalente a $\neg(\neg P_1 \vee \neg P_2)$.
2.- $\forall t \in r(P_1(t))$ es equivalente a $\neg \exists t \in r(\neg P_1(t))$.
3.- $P_1 \Rightarrow P_2$ es equivalente a $\neg P_1 \vee P_2$

Seguridad de expresiones.

Una relación en el álgebra relacional de tuplas puede generar una relación infinita. Supóngase:

{t|¬(t$\in$préstamo)}

Existe un número infinito de tuplas que no están en préstamo. La mayoría contienen valores que ni siquiera están en la BD, por lo que no queremos permitir estas expresiones.

Para ayudarnos a definir una restricción introducimos el concepto de dominio de una fórmula relacional de tuplas, P. El dominio de P, representado dom(P) es el conjunto de todos los valores referenciados por P. Así, como el conjunto de todos los valores que aparecen en una o más relaciones cuyos nombres aparecen en P. Por ejemplo, dom(tЄpréstamo Λ t[cantidad]>1200) es el conjunto de todos los valores que aparecen en préstamo mayores que 1200, y dom(¬(tЄpréstamo)) es el conjunto de todos los valores que no aparecen en préstamo.

Decimos que una expresión {t|P(t)} es segura si todos los valores que aparecen en el resultado son valores de dom(P).

Poder expresivo de los lenguajes.

El cálculo relacional de tuplas restringido es equivalente en poder expresivo al álgebra relacional. Esto quiere decir que para cada expresión en el álgebra relacional existe una relación equivalente en el álgebra relacional de tuplas.

1.3.3.4 El cálculo relacional de dominios.

Existe una segunda forma de cálculo relacional llamada cálculo relacional de dominios. Esta forma usa variables de dominio que toman valores del dominio de un atributo más que valores de una tupla completa, y esta íntimamente relacionado con el cálculo relacional de tuplas.

Definición formal.

Una expresión en el cálculo relacional de dominios es de la forma $\{<x_1, x_2,\ldots,x_n>|P(x_1,x_2,\ldots,x_n)\}$, donde $x_1,x_2,\ldots,x_n$ representan variables de dominio. P representa una fórmula compuesta por átomos, siendo un átomo una expresión con una de estas formas:

1.- $<x_1,x_2,\ldots,x_n>$Єr, donde r es una relación de n atributos, y $x_1,x_2,\ldots,x_n$ son variables de dominio o constantes de dominio.

2.- xΘy, donde x e y son variables de dominio, y Θ es un operador de comparación (<, ≤, =, ≥, >). Es requisito de los atributos x e y tengan dominios que puedan compararse.

3.- xΘc, donde x es una variable de dominio, Θ es un operador de comparación, y c es una constante en el dominio del atributo para el cual x es una variable de dominio.

Las fórmulas se construyen a partir de átomos usando las siguientes reglas:

1.- Un átomo es una fórmula.
2.- Si P_1 es una fórmula, entonces también lo son $\neg P_1$ y (P_1).
3.- Si P_1 y P_2 son fórmulas, entonces también lo son $P_1 \wedge P_2$, $P_1 \vee P_2$ y $P1 \Rightarrow P2$.
4.- Si $P_1(x)$ es una fórmula en x, donde x es una variable de dominio, entonces $\exists x(P_1(x))$ y $\forall x(P_1(x))$ también son fórmulas.

Para abreviar la notación escribimos $\exists a,b,c(P(a,b,c))$ en lugar de $\exists a(\exists b(\exists c(P(a,b,c))))$

Consultas ejemplo.

Ahora damos consultas en el cálculo relacional de dominios para los ejemplos ya considerados. Obsérvese el parecido con las correspondientes del cálculo relacional de tuplas.

1.- Encontrar nombre_sucursal, número_préstamo, nombre_cliente y cantidad de préstamos > 1200.

{<b,l,c,a>|<b,l,c,a>Єpréstamo Λ a>1200}

2.- Encontrar a todos los clientes que tienen un préstamo en Puebla y las ciudades en que viven.

{<c,x>|∃b,l,a(<b,l,c,a>Єpréstamo∧ b=”Puebla”∧∃y(<c,y,x>Єcliente))}

3.- Encontrar a todos los clientes que tienen una cuenta en todas las sucursales de Veracruz.

{<c>|∀ x,y,z(¬<x,y,z>Єsucursal V□z≠”Veracruz” V□(□∃a,n(<x,a,c,n>Єdepósito)))}

Interpretamos esta expresión como el conjunto de todas las tuplas nombre_cliente <c> tal que para todas las tuplas (nombre_sucursal, activo, ciudad_sucursal) <x,y,z>, al menos una de las siguientes declaraciones es verdadera.

<x,y,z> no es una tupla de la relación sucursal, y por tanto no corresponde a una sucursal de Veracruz.
z no tiene el valor Veracruz.

El cliente c tiene una cuenta (con número de cuenta a y saldo n) en la sucursal x.

Seguridad de las expresiones.

Una expresión como {<b,c,l,a>|¬(<b,c,l,a>Єpréstamo)} no es segura porque permite valores en el resultado que no están en el dominio de la expresión.

Para el cálculo relacional de dominios debemos preocuparnos también por la forma de las fórmulas dentro de las cláusulas existentes y para todos.

En el cálculo relacional de tuplas restringimos cualquier variable cuantificada existencialmente a moverse sobre una relación específica. Ahora añadimos reglas a la definición de seguridad para tratar con casos como el ejemplo anterior.

Decimos que una expresión $\{<x_1,x_2,\ldots,x_n>|P(x_1,x_2,\ldots,x_n)\}$ es segura si se cumplen todas las condiciones siguientes:

1.- Todos los valores que aparecen en tuplas de la expresión son valores de dom(P).

2.- Para cada subfórmula "existe" de la forma $\exists x(P_1(x))$, la subfórmula es verdadera si, y sólo si, existe un valor x de $dom(P_1)$ tal que $P_1(x)$ es verdadero.

3.- Para cada subfórmula "para todos" de la forma $\forall x(P_1(x))$, la subfórmula es verdadera si, y sólo si, $P_1(x)$ es verdadero para todos los valores de x de $dom(P_1)$.

El propósito de estas reglas adicionales es asegurar que podemos probar las subfórmulas "para todos" y "existe" sin tener que probar infinitas posibilidades.

Poder expresivo de los lenguajes.

Como el cálculo relacional de dominios se restringe a expresiones seguras, es equivalente al cálculo relacional de tuplas restringido a expresiones seguras, por lo que es equivalente al álgebra relacional.

1.3.3.5 Modificación de la base de datos.

Las modificaciones de la BD se expresan usando el operador asignación. Las asignaciones se hacen a relaciones ya existentes en la BD.

Eliminación.

Una solicitud de eliminación se expresa muy parecida a una consulta. Sin embargo, en vez de presentar tuplas al usuario, quitamos las tuplas seleccionadas de la BD. Sólo podemos eliminar tuplas completas, no únicamente valores de determinados atributos. En álgebra relacional, una eliminación se expresa:

$$R \leftarrow r - E$$

donde r es una relación y E es una consulta del álgebra relacional. Ejemplo, eliminar todas las cuentas de Sánchez:

depósito←depósito - $\boldsymbol{\sigma}_{nombre\_cliente}$="Sánchez" (depósito)

Inserción.

Para insertar datos en una relación, bien especificamos la tupla que se va a insertar o escribimos una consulta cuyo resultado sea un conjunto de tuplas que se va a insertar. En este último caso, los valores de los atributos para las tuplas

insertadas deben ser miembros del dominio de los atributos, y las tuplas insertadas deben ser del mismo orden.

En álgebra relacional una inserción se representa: $r \leftarrow r \cup E$

La inserción de una sola tupla se expresa haciendo que E sea una relación constante que contiene una tupla. Supóngase que queremos insertar el hecho de que Sánchez tiene 1200 dólares en la cuenta 9732 de Puebla.

depósito ← depósito **∪** {(“Perryde”, 9732, “Sánchez”, 1200)}

Podríamos querer insertar tuplas basadas en el resultado de una consulta. Supóngase que queremos proporcionar a todos los clientes con préstamos en Puebla una cuenta de ahorros de 200 dólares. El número de préstamo servirá como número de cuenta.

$r_1 \leftarrow (\sigma_{nombre\_sucursal}$=”Puebla” (préstamo))

$r_2 \leftarrow \Pi_{nombre_sucursal,número_préstamo,\ nombre_cliente}\ (r_1)$depósito ←depósito **∪** (r_2 x {(200)})

Actualización.

En ciertas ocasiones podemos querer cambiar un valor en una tupla sin cambiar todos los valores de la tupla.

Para ello usamos el operador actualización **δ** que tiene la forma:

$$\delta_{A \leftarrow E}\ (r)$$

donde r es una relación con atributo A al cual se le asigna el valor de la expresión E. La expresión E es cualquier expresión aritmética que implica constantes y atributos de planificación de relación r. Supóngase que estamos pagando el 5% de interés a todas las cuentas:

$$\delta_{\ saldo \leftarrow saldo * 1,05}\ (\text{depósito})$$

La sentencia anterior se aplica una vez a cada tulpa de depósito. Pero ahora supongamos que queremos dar el 6% a cuentas de más de 10,000 dólares y el 5% al resto:

$$\delta_{\ saldo \leftarrow saldo * 1,06}\ (\sigma_{saldo>10,000}\ (\text{depósito}))$$
$$\delta_{\ saldo \leftarrow saldo * 1,05}\ (\sigma_{saldo>10,000}\ (\text{depósito}))$$

Es importante el orden en que aplicamos las expresiones de actualizar. Si cambiamos el orden, una cuenta cuyo saldo estuviera justo por debajo de 10,000 recibiría el 11,3% de interés.

1.3.3.6 Vistas.

En los ejemplos expuestos hemos operado en el nivel del modelo conceptual, es decir, hemos supuesto que la colección de relaciones que se nos dan son las relaciones reales almacenadas en la BD.

No es conveniente que todos los usuarios vean el modelo conceptual completo. Las consideraciones de seguridad pueden requerir que se escondan ciertos datos al usuario.

Cualquier relación que no es parte del modelo conceptual, pero se hace visible al usuario como una relación virtual se llama vista.

Puesto que las relaciones reales en el modelo conceptual pueden modificarse, generalmente no es posible almacenar una relación correspondiente a una vista. Una vista debe volverse a calcular para cada consulta que se refiere a ella.

Cuando se define una vista, el DBMS debe almacenar la definición de la vista. Así, la definición de vista no es una expresión del álgebra relacional. Más bien, hace que se guarde una expresión que se va a sustituir en consultas utilizando la vista.

Definición de vista.

Una vista se define usando la sentencia create view de la forma create view v as <expresión de consulta> donde <expresión de consulta> es cualquier expresión legal de consulta en álgebra relacional. El nombre de la vista se representa por v.

Considérese la vista que consta de sucursales y sus clientes:

create view todos_clientes as $\Pi_{\text{nombre\_sucursal, nombre\_cliente}}$(depósito) $\cup$ $\Pi_{\text{nombre\_sucursal, nombre\_cliente}}$(préstamo)

Una vez que se ha definido la vista, el nombre de la vista puede usarse para referirse a la relación virtual que genera la vista. Los nombres de vistas pueden aparecer en cualquier lugar que pueda aparecer el nombre de una relación.

Actualización por medio de vistas y valores nulos.

Aunque las vistas son una herramienta útil para las consultas, presentan valores importantes si las actualizaciones, inserciones o eliminaciones se expresan usando vistas. La dificultad es que una modificación de la BD expresada en términos de vistas debe traducirse en una modificación de las relaciones reales en el modelo conceptual de la BD.

Para ilustrar el problema considérese la vista info_préstamo, definida create view info_préstamo as $\Pi_{\text{nombre\_sucursal, número\_préstamo, nombre\_cliente}}$ (préstamo)

Puesto que permitimos que un nombre de vista aparezca en cualquier lugar, podríamos escribir:

info_préstamo ← info_préstamo **U** {(“Puebla”, 3, “Ruth”)}

Esta inserción debe estar representada por una inserción en la relación préstamo, ya que préstamo es la relación real. Sin embargo, para insertar una tupla en préstamo, debemos tener algún valor para cantidad. Existen dos enfoques:

1.- Rechazar la inserción y devolver un mensaje de error al usuario.
2.- Insertar una tupla (“Puebla”, 3, “Ruth”, null) en la relación préstamo.

El símbolo null representa un valor_nulo o valor_en_lugar_de. Significa que el valor es desconocido o no existe. Todas las comparaciones que implican null son falsas por definición.

El problema general de la modificación de BD por medio de vistas ha sido el tema de investigaciones sustanciales.

Otra área de interés relacionado con vistas es el modelo de relación universal. En este modelo se da al usuario una vista que consta de una relación. Esta relación es el producto natural de todas las relaciones en la BD relacional real.

La principal ventaja de este modelo es que los usuarios no necesitan preocuparse de recordar que atributos están en la relación. Así, la mayor parte de las consultas son más fáciles de formular en un DBMS de relación universal que en uno de relación estándar.

Quedan cuestiones sin resolver referentes a modificaciones de BD de relación universal y todavía no se ha llegado a un consenso acerca de cuál es la mejor definición del significado de ciertos tipos complejos de consultas de relación universal.

1.3.4 Normalización.

“La normalización es un proceso que pone las cosas en su sitio, haciéndolas normales. En una base de datos, el término tiene un significado matemático específico, realizando una separación de datos (tales como nombres, direcciones u oficios) en grupos afines y definiendo las relaciones normales o (correctas entre ellos.)[12]”

Frecuentemente se reorganiza la información a partir de la información que obtiene el analista o programador a través de un análisis de requerimientos. El principio Básico de la normalización es optimizar dicha información a partir de

[12] Pérez López, César. ”MySQL para Windows y Linux”. 2008. 2a. Edición. Ed. Alfa Omega

reagrupamientos sucesivos, eliminando completamente la redundancia e inconsistencia de la información; Además de simplificar la estructura de los datos, por lo tanto, el proceso de Normalización identifica los datos redundantes que pueden existir en una estructura lógica, determina claves únicas necesarias para el acceso de los elementos de datos y ayuda a establecer las relaciones necesarias entre los elementos de datos, generalmente se tienen 3 etapas de Normalización que transforman las relaciones no normales en normalizadas llamadas FORMAS NORMALES (1FN[13], 2 FN[14], 3FN[15]).

1.3.4.1 Primera Forma Normal

La Primera Forma Normal consiste en agrupar los datos relacionados entre si de una manera tal que ninguna estructura en lo posible tenga datos repetidos.

Por lo que el primer paso en la normalización es poner los datos en la primera forma normal. Esto se hace situando los datos en tablas separadas, de manera que los datos de cada tabla sean de un tipo similar y dando a cada tabla una clave primaria y un identificador o etiqueta única. Esto elimina los grupos repetidos de datos.

Como ejemplo podemos mencionar la tabla EMPLEADOS la cual tiene los atributos Nombre, Dirección, Alojamiento, Dueño_Alojamiento, Oficio1, Oficio2, Oficio3.

Los usuarios tendrían problemas a la hora de almacenar los oficios en caso de que fueran más de 3 oficios. Por lo que se opta por generar otra tabla llamada OFICIOS la cual contenga una fila Nombre y otra fila llamada Oficio, otra más llamada Descripción y por último una llamada Calificación. Esta solución es mejor para no limitar la cantidad de oficios.

Ahora tenemos que definir un clave primaria para cada tabla para identificar cada registro, suponiendo que los nombres de las personas son únicos diremos que el nombre de la persona es la clave primaria de la tabla EMPLEADOS. Como una persona puede tener mas de un oficio tenemos que la clave primaria de la tabla OFICIOS será Nombre y Oficio.

1.3.4.2 Segunda Forma Normal

La Segunda Forma Normal se debe reorganizar las relaciones de manera que ningún dato que no sea clave quede completamente dependiente.

El segundo paso, que es la segunda forma normal, se centra en aislar los datos que solo dependen de un aparte de la clave. Para sacar la segunda forma normal

[13] Primera Forma Normal
[14] Segunda Forma Normal
[15] Tercera Forma Normal

se debe sacar Oficio y descripción a una tercera tabla, la clave primaria será Oficio. Esto es por que las descripciones se repiten por cada oficio que sea igual en diferente Nombre. También por que al no tener al menos un trabajador con un tipo de oficio este desaparece.

Así pues aplicando la segunda normal tenemos que de la tabla OFICIO obtenida con la primera forma normal se obtienen en dos tablas una llamada OFICIO-EMPLEADOS con atributos Nombre, Oficio y Calificación y la otra llamada OFICIO con los atributos Oficio y Descripción.

Es requisito indispensable que las tablas que están en la segunda forma normal, lo estén también en la primera forma normal.

1.3.4.3 Tercera Forma Normal

La Tercera Forma Normal no se puede realizar si todas las condiciones de la 2FN no son satisfechas y consiste en eliminar aquellos datos que no sean claves y que puedan derivarse de una combinación de otros datos y tampoco son claves en ninguna otra relación.

Entonces podemos decir que la tercera forma normal nos eliminará cualquier cosa de las tablas que no dependa únicamente de la clave primaria. En el ejemplo la tabla de los EMPLEADOS tenemos que el Alojamiento depende de donde viva el empleado, si se traslada entonces se tendrá que corregir el registro con el nuevo alojamiento pero el dueño del alojamiento y la dirección son independientes de si el trabajador vive ahí o no. Por lo que necesitamos trasladar Alojamiento a otra tabla llamada VIVIENDA la cual contendrá los campos Alojamiento, Dueño_Alojamiento, Dirección.

Podemos comentar que hecha la tercera forma normal ya están implícitas la primera y segunda forma normal, por lo que haciendo únicamente la tercera forma normal sería suficiente, solo basta organizar lo datos para que las columnas de cada tabla, aparte de la clave primaria, dependan únicamente de toda la clave primaria.

Así pues terminando el ejemplo nos quedan 4 tablas como siguen:
OFICIO-EMPLEADO (clave primaria Nombre, clave primaria Oficio, Calificación), OFICIO (clave primaria Oficio, clave primaria Descripción), EMPLEADO (clave primaria Nombre, Edad, Alojamiento) y VIVIENDA (clave primaria Alojamiento, Dueño_Alojamiento, Dirección).

1.3.5 SQL (Structured Query Language)

SQL (lenguaje de consultas estructurado) es un lenguaje de acceso a bases de datos relacionales con el cual se pueden crear y manipular las mismas. SQL “es

un conjunto de objetos eficientemente almacenados". La información de las bases de datos se guarda en tablas.

Una sentencia SQL es una frase con la que decimos lo que queremos obtener y de que tablas deseamos obtenerla. Una sentencia tiene un formato de cómo realizarse y empieza con una palabra reservada que indica la acción que se quiere realizar, luego siguen lo que uno quiere hacer respecto a la acción definida. SQL permite realizar consultas mediante sentencias SQL con el fin de poder desplegar información importante que se quiera de alguna o varias tablas.

SQL posee muchos servicios, entre ellos: servicio de duplicación, servicio de notificación, servicio de integración, Etc. El servicio de duplicación es para mantener varias copias de la base de datos o de alguno de sus objetos. El servicio de notificación es para enviar distintas notificaciones a uno o más dispositivos. El servicio de integración sirve para la creación de paquetes con diferentes tipos de datos. "Cabe destacar que la mayoría de los servicios se han creado teniendo presente la estrategia de Business Intelligence de Microsoft. Así que el nuevo SQL Server, más que un servidor de bases de datos, debería considerarse una plataforma completa de Windows Intelligence."

El lenguaje de consulta estructurado *(SQL)* es un lenguaje de base de datos normalizado, utilizado por el motor de base de datos de Microsoft Jet. Por lo que existen diversas aplicaciones de Microsoft que lo utilizan como lo hace el Acces.

1.3.5.1 Componentes del SQL

El lenguaje *SQL* está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos de acuerdo a las necesidades de cada aplicación o usuario.

a) Comandos

Existen dos tipos de comandos *SQL*:

1.- Los *DLL* que permiten crear y definir nuevas bases de datos, campos e índices.
2.- Los *DML* que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

Comandos DLL:

CREATE: Utilizado para crear nuevas tablas, campos e índices.

DROP: Empleado para eliminar tablas e índices.

ALTER: Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Comandos DML:

SELECT: Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.

INSERT: Utilizado para cargar lotes de datos en la base de datos en una única operación.

UPDATE: Utilizado para modificar los valores de los campos y registros especificados.

DELETE: Utilizado para eliminar registros de una tabla de una base de datos.

b) Cláusulas

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Cláusulas:

FROM: Utilizada para especificar la tabla de la cual se van a seleccionar los registros.

WHERE: Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.

GROUP BY: Utilizada para separar los registros seleccionados en grupos específicos.

HAVING: Utilizada para expresar la condición que debe satisfacer cada grupo.

ORDER BY: Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.

c) Operadores Lógicos

AND: Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.

OR: Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.

NOT: Negación lógica, devuelve el valor contrario de la expresión.

d) Operadores de Comparación

<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores
LIKE	Utilizado en la comparación de un modelo
IN	Utilizado para especificar registros de una base de datos

Fig. 1.17 Esquema de Operadores de Comparación.

e) Funciones de Agregado

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

AVG: Utilizada para calcular el promedio de los valores de un campo determinado.

COUNT: Utilizada para devolver el número de registros de la selección.

SUM: Utilizada para devolver la suma de todos los valores de un campo determinado.

MAX: Utilizada para devolver el valor más alto de un campo especificado.

MIN: Utilizada para devolver el valor más bajo de un campo especificado.

1.3.5.2 Consultas de selección.

Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las bases de datos, esta información es devuelta en forma de conjunto de registros. Este conjunto de registros es modificable.

Consultas básicas

La sintaxis básica de una consulta de selección es la siguiente:

SELECT Campos FROM Tabla;

Consultas con Predicado

El predicado se incluye entre la cláusula y el primer nombre del campo a recuperar, los posibles predicados son:

ALL: Devuelve todos los campos de la tabla.
SELECT ALL FROM Empleados;
*SELECT * FROM Empleados;*

TOP: Devuelve un determinado número de registros de la tabla.
SELECT TOP 25 Nombre, Apellido FROM Estudiantes ORDER BY Nota DESC;
SELECT TOP 10 PERCENT Nombre, Apellido FROM Estudiantes ORDER BY Nota DESC;

El valor que va a continuación de TOP debe ser un Integer sin signo. TOP no afecta a la posible actualización de la consulta.

DISTINCT: Omite los registros cuyos campos seleccionados coincidan totalmente.

SELECT DISTINCT Apellido FROM Empleados;

DISTINCTROW: Omite los registros duplicados basándose en la totalidad del registro y no sólo en los campos seleccionados.

SELECT DISTINCTROW Apellido FROM Empleados;

1.3.5.3 Consultas de acción.

Las consultas de acción son aquellas que no devuelven ningún registro, son las encargadas de acciones como añadir y borrar y modificar registros.

DELETE

Crea una consulta de eliminación que elimina los registros de una o más de las tablas listadas en la cláusula FROM que satisfagan la cláusula WHERE. Esta consulta elimina los registros completos, no es posible eliminar el contenido de algún campo en concreto. Su sintaxis es:

DELETE Tabla. FROM Tabla WHERE criterio;*

DELETE es especialmente útil cuando se desea eliminar varios registros. En una instrucción DELETE con múltiples tablas, debe incluir el nombre de tabla (Tabla.*). Si especifica más de una tabla desde la que tenemos que eliminar registros, todas deben ser tablas de muchos a uno. Si desea eliminar todos los registros de una tabla, eliminar la propia tabla es más eficiente que ejecutar una consulta de borrado.

INSERT INTO

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Esta consulta puede ser de dos tipos: insertar un único registro ó insertar en una tabla los registros contenidos en otra tabla.

Para insertar un único registro:

En este caso la sintaxis es la siguiente:

INSERT INTO Tabla (campo1, campo2,..., campoN)VALUES (valor1, valor2,..., valorN);

Esta consulta graba en el campo1 el valor1, en el campo2 y valor2 y así sucesivamente. Hay que prestar especial atención a acotar entre comillas simples (') los valores literales (cadenas de caracteres) y las fechas indicarlas en formato mm-dd-aa y entre caracteres de almohadillas (#).

Para insertar Registros de otra Tabla, en este caso la sintaxis es:

INSERT INTO Tabla [IN base_externa] (campo1, campo2,..., campoN) SELECT TablaOrigen.campo1, TablaOrigen.campo2,..., TablaOrigen.campoN FROM TablaOrigen;

En este caso se seleccionarán los campos 1,2,..., n de la tabla origen y se grabarán en los campos 1,2,..., n de la Tabla. La condición SELECT puede incluir la cláusula WHERE para filtrar los registros a copiar. Si Tabla y TablaOrigen poseen la misma estructura podemos simplificar la sintaxis a:

INSERT INTO Tabla SELECT TablaOrigen. FROM TablaOrigen;*

UPDATE

Crea una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en un criterio específico. Su sintaxis es:

UPDATE Tabla SET Campo1=Valor1, Campo2=Valor2, ... CampoN=ValorN
WHERE Criterio;

UPDATE es especialmente útil cuando se desea cambiar un gran número de registros o cuando éstos se encuentran en múltiples tablas. Puede cambiar varios campos a la vez. El ejemplo siguiente incrementa los valores Cantidad pedidos en un 10 por ciento y los valores Transporte en un 3 por ciento para aquellos que se hayan enviado al Reino Unido:

*UPDATE Pedidos SET Pedido = Pedidos * 1.1, Transporte = Transporte * 1.03 WHERE PaisEnvío = 'ES';*

UPDATE no genera ningún resultado. Para saber qué registros se van a cambiar, hay que examinar primero el resultado de una consulta de selección que utilice el mismo criterio y después ejecutar la consulta de actualización.

UPDATE Empleados SET Grado = 5 WHERE Grado = 2;
*UPDATE Productos SET Precio = Precio * 1.1 WHERE Proveedor = 8 AND Familia = 3;*

1.3.5.4 Consultas con parámetros.

Las consultas con parámetros son aquellas cuyas condiciones de búsqueda se definen mediante parámetros. Si se ejecutan directamente desde la base de datos donde han sido definidas aparecerá un mensaje solicitando el valor de cada uno de los parámetros. Si deseamos ejecutarlas desde una aplicación hay que asignar primero el valor de los parámetros y después ejecutarlas.

1.3.6 Método ACID para realizar transacciones.

Una transacción es una unidad de la ejecución de un programa que accede y posiblemente actualiza varios elementos de datos. Una transacción se inicia por la ejecución de un programa de usuario escrito en un lenguaje de manipulación de datos de alto nivel o en un lenguaje de programación (por ejemplo SQL, COBOL, C, C++ o Java), y está delimitado por instrucciones (o llamadas a función) de la forma inicio transacción y fin transacción. La transacción consiste en todas las operaciones que se ejecutan entre inicio transacción y el fin transacción.

Para asegurar la integridad de los datos se necesita que el sistema de base de datos mantenga las siguientes propiedades de las transacciones:

- Atomicidad. Ó todas las operaciones de la transacción se realizan adecuadamente en la base de datos o ninguna de ellas.

- Consistencia. La ejecución aislada de la transacción (es decir, sin otra transacción que se ejecute concurrentemente) conserva la consistencia de la base de datos.

- Aislamiento. Aunque se ejecuten varias transacciones concurrentemente, el sistema garantiza que para cada par de transacciones Ti y Tj, se cumple que para los efectos de Ti, o bien Tj ha terminado su ejecución antes de que comience Ti, o bien que Tj ha comenzado su ejecución después de que Ti termine. De este modo, cada transacción ignora al resto de las transacciones que se ejecuten concurrentemente en el sistema.

- Durabilidad. Tras la finalización con éxito de una transacción, los cambios realizados en la base de datos permanecen, incluso si hay fallos en el sistema.

Estas propiedades a menudo reciben el nombre de propiedades ACID; el acrónimo se obtiene de la primera letra de cada una de las cuatro propiedades en inglés (Atomicity, Consistency, Isolation y Durability, respectivamente). Para comprender mejor las propiedades ACID y la necesidad de dichas propiedades, considérese un sistema bancario simplificado constituido por varias cuentas y un conjunto de transacciones que acceden y actualizan dichas cuentas.

Por ahora se asume que la base de datos reside permanentemente en disco, pero una porción de la misma reside temporalmente en la memoria principal.

El acceso a la base de datos se lleva a cabo mediante las dos operaciones siguientes:

- leer (*X*), que transfiere el dato *X* de la base de datos a una memoria intermedia local perteneciente a la transacción que ejecuta la operación leer.

- escribir (*X*), que transfiere el dato *X* desde la memoria intermedia local de la transacción que ejecuta la operación escribir a la base de datos.

En un sistema de base de datos real, la operación escribir no tiene por qué producir necesariamente una actualización de los datos en disco; la operación escribir puede almacenarse temporalmente en memoria y llevarse a disco más tarde. Sin embargo, por el momento se supondrá que la operación escribir actualiza inmediatamente la base de datos.

Sea *Ti* una transacción para transferir $50 de la cuenta *A* a la cuenta *B*. Se puede definir dicha transacción como

Ti: leer(A);
A := A – 50;
escribir(A);
leer(B);
B := B + 50;
escribir(B).

Considérense ahora cada uno de los requisitos ACID (para una presentación más cómoda, se consideran en distinto orden al indicado por A-C-I-D).

Consistencia: en este caso el requisito de consistencia es que la suma de *A* y *B* no sea alterada al ejecutar la transacción. Sin el requisito de consistencia, ¡la

transacción podría crear o destruir dinero! Se puede comprobar fácilmente que si una base de datos es consistente antes de ejecutar una transacción, sigue siéndolo después de ejecutar dicha transacción.

La responsabilidad de asegurar la consistencia de una transacción es del programador de la aplicación que codifica dicha transacción. La comprobación automática de las restricciones de integridad puede facilitar esta tarea.

Atomicidad: supóngase que justo antes de ejecutar la transacción *Ti* los valores de las cuentas *A* y *B* son de $1,000 y de $2,000, respectivamente.

Supóngase ahora que durante la ejecución de la transacción *Ti* se produce un fallo que impide que dicha transacción finalice con éxito su ejecución. Ejemplos de este tipo de fallos pueden ser los fallos en la alimentación, los fallos del hardware y los errores de software. Además, supóngase que el fallo tiene lugar después de ejecutarse la operación escribir(*A*), pero antes de ejecutarse la operación escribir(*B*). Es ese caso, los valores de las cuentas *A* y *B* que se ven reflejados en la base de datos son $950 y $2,000. Se han perdido $50 de la cuenta *A* como resultado de este fallo. En particular se puede ver que ya no se conserva la suma *A* + *B*.

Así, como resultado del fallo, el estado del sistema deja de reflejar el estado real del mundo que se supone que modela la base de datos. Un estado así se denomina estado inconsistente. Hay que asegurarse de que estas inconsistencias no sean visibles en un sistema de base de datos. Nótese, sin embargo, que un sistema puede en algún momento alcanzar un estado inconsistente. Incluso si la transacción *Ti* se ejecuta por completo, existe un punto en el que el valor de la cuenta *A* es de $950 y el de la cuenta *B* es de $2,000, lo cual constituye claramente un estado inconsistente. Este estado, sin embargo, se sustituye eventualmente por otro estado consistente en el que el valor de la cuenta *A* es de $950 y el de la cuenta *B* es de $2,050. De este modo, si la transacción no empieza nunca o se garantiza que se complete, un estado inconsistente así no será visible excepto durante la ejecución de la transacción. Ésta es la razón de que aparezca el requisito de atomicidad. Si se proporciona la propiedad de atomicidad, o todas las acciones de la transacción se ven reflejadas en la base de datos, o ninguna de ellas.

La idea básica que hay detrás de asegurar la atomicidad es la siguiente. El sistema de base de datos mantiene los valores antiguos (en disco) de aquellos datos sobre los que una transacción realiza una escritura y, si la transacción no completa su ejecución, los valores antiguos se recuperan para que parezca que la transacción no se ha ejecutado. La responsabilidad de asegurar la atomicidad es del sistema de base de datos; en concreto, lo maneja un componente llamado componente de gestión de transacciones.

Durabilidad: una vez que se completa con éxito la ejecución de una transacción, y después de comunicar al usuario que inició la transacción que se ha realizado la transferencia de fondos, no debe suceder que un fallo en el sistema produzca la pérdida de datos correspondientes a dicha transferencia. La propiedad de durabilidad asegura que, una vez que se completa con éxito una transacción, persisten todas las modificaciones realizadas en la base de datos, incluso si hay un fallo en el sistema después de completarse la ejecución de dicha transacción.
A partir de ahora se asume que un fallo en la computadora del sistema produce una pérdida de datos de la memoria principal, pero los datos almacenados en disco nunca se pierden. Se puede garantizar la durabilidad si se asegura que:

1. Las modificaciones realizadas por la transacción se guardan en disco antes de que finalice la transacción.

2. La información de las modificaciones realizadas por la transacción guardada en disco es suficiente para permitir a la base de datos reconstruir dichas modificaciones cuando el sistema se reinicie después del fallo.

La responsabilidad de asegurar la durabilidad es de un componente del sistema de base de datos llamado componente de gestión de recuperaciones. El componente de gestión de transacciones y el componente de gestión de recuperaciones están estrechamente relacionados.

Aislamiento: incluso si se aseguran las propiedades de consistencia y de atomicidad para cada transacción, si varias transacciones se ejecutan concurrentemente, se pueden entrelazar sus operaciones de un modo no deseado, produciendo un estado inconsistente.

Por ejemplo, como se ha visto antes, la base de datos es inconsistente temporalmente durante la ejecución de la transacción para transferir fondos de la cuenta *A* a la cuenta *B*, con el total deducido escrito ya en *A* y el total incrementado todavía sin escribir en *B*. Si una segunda transacción que se ejecuta concurrente lee *A* y *B* en este punto intermedio y calcula *A* + *B*, observará un valor inconsistente. Además, si esta segunda transacción realiza después modificaciones en *A* y *B* basándose en los valores leídos, la base de datos puede permanecer en un estado inconsistente aunque ambas transacciones terminen.

Una solución para el problema de ejecutar transacciones concurrentemente es ejecutarlas secuencialmente, es decir, una tras otra. Sin embargo, la ejecución concurrente de transacciones produce notables beneficios en el rendimiento. Se han desarrollado otras soluciones que permiten la ejecución concurrente de varias transacciones. La propiedad de aislamiento asegura que el resultado obtenido al ejecutar concurrentemente las transacciones es un estado del sistema equivalente a uno obtenido al ejecutar una tras otra en algún orden. La responsabilidad de

asegurar la propiedad de aislamiento es de un componente del sistema de base de datos llamado componente de control de concurrencia. [16]

1.4 Lenguajes de 4ª. Generación.

A continuación se muestran las características de los lenguajes de programación.

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

Aunque muchas veces se usan los términos lenguaje de programación y lenguaje informático como si fuesen sinónimos, no tiene por qué ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el HTML (lenguaje para el marcado de páginas WEB que no es propiamente un lenguaje de programación).

Un lenguaje de programación permite a uno o más programadores especificar de *manera precisa* sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico. Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa.

Los procesadores usados en las computadoras son capaces de entender y actuar según lo indican programas escritos en un lenguaje fijo llamado lenguaje de máquina. Todo programa escrito en otro lenguaje puede ser ejecutado de dos maneras:

Mediante un programa que va adaptando las instrucciones conforme son encontradas. A este proceso se lo llama *interpretar* y a los programas que lo hacen se los conoce como intérpretes.

Traduciendo este programa, al programa equivalente escrito en lenguaje de máquina. A ese proceso se lo llama *compilar* y al programa traductor se le denomina compilador.

[16] Silberschatz, Abraham. ”Fundamentos de Base de Datos”. 2002. Ed. McGrawHill - Interamericana

1.4.1 Historia de los lenguajes.

Lenguaje de Primera Generación: Se refiere al tipo de lenguaje más básico que maneja una computadora como el ensamblador, también llamado lenguaje máquina o assembler. Específico para cada microprocesador, uso de código binario.

Lenguaje de Segunda Generación: Tipo de lenguajes de programación que son ensamblados a través de un macro ensamblador. Se trata del lenguaje máquina combinado con una serie de macros poderosas que permiten declarar estructuras de datos y de control complejas. Uso de nemotécnicos que abstraen del lenguaje máquina.

Lenguajes de Tercera Generación: Tipo de lenguajes de programación que emplean los programadores para hacer aplicaciones llamados también lenguajes de alto nivel. Logran un alto rendimiento respecto de las generaciones anteriores de lenguajes. En este tipo de lenguajes, los programadores deben indicarle los procedimientos específicos que debe hacer la computadora para lograr un objetivo, son lenguajes estructurados con comandos cercanos al lenguaje común. Ejemplos de lenguajes 3GL son C, Fortran, Smalltalk, ADA, C++, C#, Cobol, Delphi.

Lenguaje de Cuarta Generación: Tipo de lenguajes de programación en los que se especifica qué resultados son los que se quieren obtener, a diferencia de los de 3GL, en donde se especifica cómo deben obtenerse esos resultados. Son programas orientados a problemas específicos o de propósito especial. Permiten crear prototipos de una aplicación rápidamente, estos ayudan a tener una idea del aspecto y funcionamiento de la aplicación antes que el código sea terminado. Esto implica que los que tienen que ver con el desarrollo de la aplicación pueden aportar retroalimentación en conceptos como estructura y diseño desde el inicio del proceso.

Muchos de estos lenguajes tienen capacidad para bases de datos, dejando hacer programas que enlacen a las mismas. El SQL es un 4GL.

Uno de los cambios sustanciales que han tenido los lenguajes de cuarta generación ha sido su relación con las base de datos cambiando el uso de técnicas y recursos en comparación con los 4GL más viejos. Además del mayor manejo de interfaces para el usuario.

También los lenguajes de cuarta generación se han visto influenciados por las tendencias de cambio que ha sufrido la computación con el paso de los años, como la tendencia a programar con orientación a objetos, la incorporación de la arquitectura cliente/servidor, la naciente ingeniería de software y la tendencia a trabajar en equipo.

La programación orientada a objetos y la disposición de los desarrolladores están logrando poner en el mercado librerías de objetos que complementan el funcionamiento de los 4GL. Entre las librerías más comunes se encuentran objetos para el acceso a múltiples bases de datos relacionales, objetos para el acceso a sistemas de mensajería electrónica, objetos para el acceso a sistemas de Workgroup, objetos para el acceso a bases de datos jerárquicas, objetos para el uso avanzado de GUIs, entre otras.

1.4.2 Lenguajes de alto nivel.

El lenguaje máquina y el lenguaje ensamblador son considerados como lenguajes de bajo nivel por su complejidad y por que se acercan al funcionamiento de una computadora, ya que al programar en ensamblador se trabajan con los registros de memoria de la computadora de forma directa.

Los lenguajes de alto nivel son un tipo de lenguajes de programación que permite al programador escribir programas (algoritmos) que son más o menos independientes de un tipo particular de computadora (del hardware). Estos lenguajes son considerados de alto nivel porque son más parecidos al lenguaje natural humano y más lejano al lenguaje de las máquinas.

La principal ventaja de los lenguajes de alto nivel sobre los de bajo nivel, es que son más fáciles de leer, escribir y mantener por humanos. Al final, los programas escritos en alto nivel deben ser traducidos en un lenguaje máquina específico empleando un compilador o un intérprete. De esta manera pueden ser ejecutados por una máquina específica. No permite ambigüedades es decir, debe ser muy claro y conciso.

Se basan en la construcción de sentencias orientadas a la estructura lógica de lo deseado; una sentencia de lenguaje de alto nivel representa varias de bajo; cabe la posibilidad que las sentencias de un lenguaje de alto nivel no cubran todas las instrucciones del lenguaje de bajo nivel, lo que limita el control sobre la máquina. Para que un lenguaje de alto nivel sea legible por el sistema, debe traducirse a lenguaje ensamblador y posteriormente a lenguaje de máquina.

Una desventaja de los lenguajes de alto nivel es que limitan con respecto al hardware que se tenga en la computadora, tienen alto tiempo de traducción, se incrementa la ocupación de la memoria interna.

Los primeros lenguajes de programación de alto nivel fueron diseñados en los 50. Actualmente existen cientos de lenguajes de este tipo como Ada, Algol, BASIC, COBOL, C, C++, FORTRAN, LISP, Pascal, Prolog, Etc.

Entre los lenguajes de alto nivel que se usan en la actualidad está, JAVA, VISUAL STUDIO.NET, Etc. Y muchos más de distribución comercial y libre, entre

estos existen los lenguajes orientados a objetos que se describen en el siguiente punto.

1.4.3 Lenguajes orientados a objetos.

La Programación Orientada a Objetos (POO u OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

La Programación Orientada a Objetos es el paradigma de programación más utilizado en la actualidad. Su consistente base teórica y la amplia gama de herramientas que permiten crear código a través de diseños orientados a objetos la convierten en la alternativa más adecuada para el desarrollo de aplicaciones.

Durante años, los programadores se han dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas. Para conseguir que los esfuerzos de los programadores puedan ser utilizados por otras personas se creó la POO. Que es una serie de normas de realizar las cosas de manera que otras personas puedan utilizarlas y adelantar su trabajo, de manera que consigamos que el código se pueda reutilizar.

La POO no es difícil, pero es una manera especial de pensar, a veces subjetiva de quien la programa, de manera que la forma de hacer las cosas puede ser diferente según el programador. Aunque podamos hacer los programas de formas distintas, no todas ellas son correctas, lo difícil no es programar orientado a objetos sino programar bien. Programar bien es importante porque así nos podemos aprovechar de todas las ventajas de la POO.

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase. En los ejemplos anteriores en realidad hablábamos de las clases coche o fracción porque sólo estuvimos definiendo, aunque por encima, sus formas.

Las propiedades en clases o atributos son las características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenamos datos relacionados con los objetos.

Los métodos en las clases son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son como funciones que están asociadas a un objeto.

Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar (que viene de una mala traducción de la palabra instace que en inglés significa ejemplar). Por ejemplo, un objeto de la clase fracción es por ejemplo 3/5. El concepto o definición de fracción sería la clase, pero cuando ya estamos hablando de una fracción en concreto 4/7, 8/1000 ó cualquier otra, la llamamos objeto.

Para crear un objeto se tiene que escribir una instrucción especial que puede ser distinta dependiendo el lenguaje de programación que se emplee, pero será algo parecido a esto: miCoche = new Coche()

Con la palabra new especificamos que se tiene que crear una instancia de la clase que sigue a continuación. Dentro de los paréntesis podríamos colocar parámetros con los que inicializar el objeto de la clase coche.

1.4.4 Visual Studio .NET

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite crear aplicaciones, sitios y aplicaciones Web, así como servicios Web en cualquier entorno que soporte la plataforma .NET (a partir de la versión NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas Web y dispositivos móviles.

Visual Studio 2005 se empezó a comercializar a través de Internet a partir del 4 de Octubre de 2005 y llegó a los comercios a finales del mes de Octubre en inglés. En castellano no salió hasta el 4 de Febrero de 2006. Microsoft eliminó *.NET*, pero eso no indica que se alejara de la plataforma .NET, de la cual se incluyó la versión 2.0.

La actualización más importante que recibieron los lenguajes de programación fue la inclusión de *tipos genéricos*, similares en muchos aspectos a las plantillas de C#. Con esto se consigue encontrar muchos más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible. C++ tiene una actualización similar con la adición de C++/CLI como sustituto de C# manejado.

Se incluye un diseñador de implantación, que permite que el diseño de la aplicación sea validado antes de su implantación. También se incluye un entorno para publicación Web y pruebas de carga para comprobar el rendimiento de los programas bajo varias condiciones de carga.

Visual Studio 2005 también añade soporte de 64-bit. Aunque el entorno de desarrollo sigue siendo una aplicación de 32 bits Visual C++ 2005 soporta compilación para x86-64 (AMD64 e Intel 64) e IA-64 (Itanium). El SDK incluye compiladores de 64 bits así como versiones de 64 bits de las librerías.

Visual Studio 2005 tiene varias ediciones radicalmente distintas entre sí: Express, Standard, Professional, Tools for Office, y 5 ediciones Visual Studio Team System. Estas últimas se proporcionaban conjuntamente con suscripciones a MSDN cubriendo los 4 principales roles de la programación: Architects, Software Developers, Testers, y Database Professionals. La funcionalidad combinada de las 4 ediciones Team System se ofrecía como la edición Team Suite.

Tools for the Microsoft Office System está diseñada para extender la funcionalidad a Microsoft Office.

Las ediciones Express se han diseñado para principiantes, aficionados y pequeños negocios, todas disponibles gratuitamente a través de la página de Microsoft se incluye una edición independiente para cada lenguaje: Visual Basic, Visual C++, Visual C#, Visual J# para programación .NET en Windows, y Visual Web Developer para la creación de sitios Web ASP.NET. Las ediciones express carecen de algunas herramientas avanzadas de programación así cómo de opciones de extensibilidad.

Se lanzó el service Pack 1 para Visual Studio 2005 el 14 de Diciembre de 2006.
La versión interna de Visual Studio 2005 es la 8.0, mientras que el formato del archivo es la 9.0.

1.4.5 Visual Basic .NET

Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET Su introducción resultó muy controvertida, ya que debido a cambios significativos en el lenguaje VB.NET no es compatible hacia atrás con Visual Basic, cosa que causó gran división en la comunidad de desarrolladores de Visual Basic.

La gran mayoría de programadores de VB.NET utilizan el entorno de programación Microsoft Visual Studio .Net en alguna de sus versiones (Visual Studio .NET, Visual Studio .NET 2003 ó Visual Studio .NET 2005), aunque existen otras alternativas, como SharpDevelop (que además es libre).

Como pasa con todos los lenguajes de programación basados en .NET, los programas escritos en VB.NET requieren el Framework .NET para ejecutarse.

La sintaxis básica es prácticamente la misma entre VB y VB.NET, con la excepción de los añadidos para soportar nuevas características como el control estructurado de excepciones, la programación orientada a objetos, o los Genéricos.

Las diferencias entre VB y VB.NET son profundas, sobre todo en cuanto a metodología de programación y bibliotecas, pero ambos lenguajes siguen manteniendo un gran parecido, cosa que facilita notablemente el paso de VB a VB.NET.

Publicado el 4 de Octubre de 2005, se basó en el framework .NET 2.0.

Añade soporte de 64-bit (x86-64: AMD64 e Intel 64, e IA-64: Itanium)

Ediciones: Express, Standard, Professional, Tools for Office, y 5 ediciones Visual Studio Team System (Architects, Software Developers, Testers, y Database Professionals)

La versión interna de Visual Studio 2005 es la 8.0, mientras que el formato del archivo es la 9.0.

La mejor manera de crear proyectos nuevos es utilizar las plantillas o los asistentes para aplicaciones de Visual Basic. Las plantillas de proyecto se utilizan junto con los marcos de trabajo de aplicación y las bibliotecas para crear programas de inicio.

1.5 Técnicas de investigación

La técnica es indispensable en el proceso de la investigación científica ya que integra la estructura por medio de la cual se organiza la investigación, la técnica pretende los siguientes objetivos:

- Ordenar las etapas de la investigación.
- Aportar instrumentos para manejar la información.
- Llevar un control de los datos.
- Orientar la obtención de conocimientos.

En cuanto a las técnicas de investigación, se estudian dos formas generales: técnica documental y técnica de campo.

La técnica documental permite la recopilación de información para enunciar las teorías que sustentan el estudio de los fenómenos y procesos. Incluye el uso de instrumentos definidos según la fuente documental a que hacen referencia.

La técnica de campo permite la observación en contacto directo con el objeto

de estudio, y el acopio de testimonios que permitan confrontar la teoría con la práctica en la búsqueda de la verdad objetiva.

La encuesta es una técnica de adquisición de información de interés sociológico, mediante un cuestionario previamente elaborado, a través del cual se puede conocer la opinión o valoración del sujeto seleccionado en una muestra sobre un asunto dado.

Existen diversas técnicas de investigación como lo son la observación, entrevista, fichaje, test.

No obstante es importante destacar que la encuesta puede utilizar diversos instrumentos e incluso diversas técnicas para lograr el fin que el investigador o el estudio exigen.

La Observación

Es una técnica que consiste en observar atentamente el fenómeno, hecho o caso, tomar información y registrarla para su posterior análisis.

Existen dos clases de observación: la observación no científica y la observación científica. La diferencia básica entre una y otra esta en la intencionalidad: observar científicamente significa observar con un objetivo claro, definido y preciso: el investigador sabe qué es lo que desea observar y para qué quiere hacerlo, lo cual implica que debe preparar cuidadosamente la observación. Observar no científicamente significa observar sin intención, sin objetivo definido y por tanto, sin preparación previa.

Recursos auxiliares de la observación:

- Fichas
- Récords anecdóticos
- Grabaciones
- Fotografías
- Listas de chequeo de datos
- Escalas, Etc.
- Modalidades que puede tener la observación científica
- La observación científica puede ser:
- Directa o Indirecta
- Participante o no participante
- Estructurada o no estructurada
- De campo o de laboratorio
- Individual o de equipo

Tipos de observación:
- Observación directa y la indirecta
- Observación participante y no participante
- Observación estructurada y no estructurada
- Observación de campo y de laboratorio
- Observación individual y de equipo

La Entrevista

Es una técnica para obtener datos que consisten en un diálogo entre dos personas: El entrevistador "investigador" y el entrevistado; se realiza con el fin de obtener información de parte de este, que es, por lo general, una persona entendida en la materia de la investigación.

La Encuesta

La encuesta es una técnica destinada a obtener datos de varias personas cuyas opiniones impersonales interesan al investigador. Para ello, a diferencia de la entrevista, se utiliza un listado de preguntas escritas que se entregan a los sujetos, a fin de que las contesten igualmente por escrito. Ese listado se denomina cuestionario.

El Fichaje

El fichaje es una técnica auxiliar de todas las demás técnicas empleada en investigación científica; consiste en registrar los datos que se van obteniendo en los instrumentos llamados fichas, las cuales, debidamente elaboradas y ordenadas contienen la mayor parte de la información que se recopila en una investigación por lo cual constituye un valioso auxiliar en esa tarea, al ahorra mucho tiempo, espacio y dinero.

El Test

Es una técnica derivada de la entrevista y la encuesta tiene como objeto lograr información sobre rasgos definidos de la personalidad, la conducta o determinados comportamientos y características individuales o colectivas de la persona (inteligencia, interés, actitudes, aptitudes, rendimiento, memoria, manipulación, Etc.). A través de preguntas, actividades, manipulaciones, Etc., que son observadas y evaluadas por el investigador.

- Debe ser válido
- Debe ser confiable
- Debe ser objetivo
- Debe ser sencillo y claro escrito en lenguaje de fácil compresión para los investigadores.
- Debe ser económico, tanto en tiempo como en dinero y esfuerzo.

- Debe ser interesante, para motivar el interés de los investigadores.

1.5.1 Encuesta

La encuesta es impersonal porque el cuestionario no lleve el nombre ni otra identificación de la persona que lo responde, ya que no interesan esos datos.

Es una técnica que se puede aplicar a sectores más amplios del universo, de manera mucho más económica que mediante entrevistas.

Varios autores llaman cuestionario a la técnica misma. Los mismos u otros, unen en un mismo concepto a la entrevista y al cuestionario, denominándolo encuesta, debido a que en los dos casos se trata de obtener datos de personas que tienen alguna relación con el problema que es materia de investigación.

1.5.1.1 Riesgos que conlleva la aplicación de cuestionarios

La falta de sinceridad en las respuestas (deseo de causar una buena impresión o de disfrazar la realidad).

La tendencia a decir "sí" a todo.

La sospecha de que la información puede revertirse en contra del encuestado, de alguna manera.

La falta de comprensión de las preguntas o de algunas palabras.

La influencia de la simpatía o la antipatía tanto con respecto al investigador como con respecto al asunto que se investiga.

El investigador debe seleccionar las preguntas más convenientes, de acuerdo con la naturaleza de la investigación y, sobre todo, considerando el nivel de educación de las personas que se van a responder el cuestionario.

1.5.1.2 Clasificación de acuerdo con su forma

- Preguntas abiertas
- Preguntas cerradas
- Preguntas dicotómicas
- Preguntas de selección múltiple
- En abanico
- De estimación
- Clasificación de acuerdo con el fondo:
- Preguntas de hecho
- Preguntas de acción
- Preguntas de intención

- Preguntas de opinión
- Preguntas índices o preguntas test

En la actualidad, existen al menos cuatro tipos de encuesta que permiten obtener información primaria, por lo que es imprescindible que investigadores de mercados y mercadólogos conozcan cuáles son y en qué consiste cada uno de ellos.

1.5.1.3 Tipos de Encuesta, según el medio

En la actualidad, existen cuatro tipos de encuesta que se dividen según el medio que se utiliza para realizar la encuesta:

Encuestas basadas en entrevistas cara a cara o de profundidad: Consisten en entrevistas directas o personales con cada encuestado. Tienen la ventaja de ser controladas y guiadas por el encuestador, además, se suele obtener más información que con otros medios (el teléfono y el correo). Sus principales desventajas son el tiempo que se tarda para la recolección de datos, su costo que es mas elevado que las encuestas telefónicas, por correo o internet (porque incluye viáticos, transporte, bonos y otros que se pagan a los encuestadores) y la posible limitación del sesgo del entrevistador (por ejemplo, su apariencia, estilo de hacer preguntas y el lenguaje corporal que utiliza, todo lo cual, puede influir en las respuestas del encuestado).

Encuestas telefónicas: Este tipo de encuesta consiste en una entrevista vía telefónica con cada encuestado. Sus principales ventajas son: 1) se puede abarcar un gran número de personas en menos tiempo que la entrevista personal, 2) sus costos suelen ser bajos y 3) es de fácil administración (hoy en día, existen softwares especializados para la gestión de encuestas telefónicas). Sin embargo, su principal desventaja es que el encuestador tiene un mínimo control sobre la entrevista, la cual, debe ser corta (para no molestar al encuestado).

Encuestas postales: Consiste en el envío de un "cuestionario" a los potenciales encuestados, pedirles que lo llenen y hacer que lo remitan [1] a la empresa o a una casilla de correo. Para el envío del cuestionario existen dos medios: 1) El correo tradicional y 2) el correo electrónico (que ha cobrado mayor vigencia en los últimos años). Las principales ventajas de este tipo de encuesta están relacionadas con la sinceridad con que suelen responder los encuestados (al no tener la presión directa que supone la presencia del encuestador), el bajo costo (en relación a la encuesta cara a cara y por teléfono) y la amplia cobertura a la que se puede llegar (siempre y cuando se disponga de una buena base de datos). Sus desventajas son: La baja tasa de respuesta y la falta de listas con información actualizada.

Encuestas por Internet: Este tipo de encuesta consiste en "colocar" un cuestionario en una página web o en enviarlo a los correos electrónicos de un

panel predefinido. Sus principales ventajas son: 1) la amplia cobertura a la que se puede llegar (incluso a miles de encuestados en varios países y al mismo tiempo), 2) el ahorro de tiempo (se pueden obtener miles de encuestas respondidas en cuestión de horas), los bajos costos (que son menores a las encuestas cara a cara, por teléfono y postales) y la utilización de medios audiovisuales durante la encuesta. Sus desventajas son: No siempre se puede verificar la identidad del encuestado y la interrogante que deja la muestra en cuanto a su representatividad del universo.

Otros tipos de encuesta: Encuesta en el punto de venta: Es aquella que es realizada en los pasillos de un establecimiento comercial y que consiste en interceptar a los compradores de ese momento para solicitarles que rellenen el cuestionario.

Encuesta ómnibus: Consiste en un cuestionario cerrado multitemático, compuesto por varios módulos que recogen información de una misma muestra sobre diferentes temas, para distintos clientes, que se abonan al servicio y se benefician de un ahorro de costes, dado que estos son compartidos por todos los suscriptores. El hecho de que se lleven a cabo con periodicidad semanal, mensual o trimestral las hace muy indicadas para estudios de seguimiento.

Encuesta por suscripción: Es una encuesta de carácter único que es vendida a varios clientes interesados en ella y con necesidades parecidas. No debe ser confundida con la encuesta ómnibus.

CAPÍTULO 2: PLANTEAMIENTO DEL PROBLEMA Y ELECCIÓN DE LA SOLUCIÓN

2.1 Panorama general.

Dada la experiencia obtenida por el correr de los años laborales en áreas de la investigación, tanto en el Instituto Nacional de Estadística, Geografía e Informática (INEGI) como en el Instituto Nacional de Salud Pública (INSP), hemos encontrado que las técnicas de investigación utilizadas para recabar información en estas áreas son realizadas sin la ayuda de un Sistema informático, sobre todo en el área del INSP. Ya que se encuentran realizando el levantamiento de la información por medio de cuestionarios en papel.

Estas instituciones una de entre todas sus funciones están la de la investigación, nos enfocaremos más con respecto al Instituto Nacional de Salud Pública ya que es en este lugar donde llevamos más tiempo laborando. Su función principal es la de la promoción de la salud de la población mediante la aplicación de políticas basadas en evidencias científicas, el objetivo principal que el INSP se ha planeado consiste en contribuir a mejorar las condiciones de salud de la población, lo que necesariamente implica en reducir las desigualdades en el acceso a los servicios de salud mediante intervenciones focalizadas en comunidades marginadas y grupos vulnerables.

La formación del Centro de Investigación en Evaluación y Encuestas (CIEE) constituye una respuesta del Instituto Nacional de Salud Pública a la necesidad de conformar un grupo sólido en la evaluación de impacto de programas de desarrollo y salud.

A partir de la década de los ochenta, la Secretaría de Salud inició la operación de un esquema institucional para la elaboración sistemática de encuestas con aplicación nacional para conocer, de la manera más objetiva posible, el comportamiento de diversas variables y determinantes de la salud. Desde entonces se han llevado a cabo más de 20 ejercicios probabilísticos respecto de las condiciones de salud y nutrición, o sobre diversos temas epidemiológicos. Estos ejercicios culminan en la integración de bases de datos organizados e información relevante que permite identificar tendencias, así como evaluar el impacto y los resultados de las intervenciones, programas y acciones de salud.

Así, un insumo básico para la toma de decisiones es la evidencia científica, y las encuestas constituyen un medio de gran importancia para obtenerla, en particular en el campo de la salud pública.

Las encuestas nacionales son una herramienta sustantiva para generar información basada en evidencia científica que contribuya a proponer y mejorar las políticas públicas de salud o de cualquier otra índole. Aplicadas de forma continua, estas encuestas constituyen sistemas de vigilancia. Los datos que arrojan

permiten identificar la utilidad y el impacto de estrategias públicas en salud ó en cualquier otro rubro y posibilitan compartir los resultados a nivel local, estatal y nacional, así como compararlos con los de otros países y regiones.

En México, el Instituto Nacional de Salud Pública (INSP), en colaboración con diversas instituciones académicas y gubernamentales, es un referente indispensable para caracterizar la frecuencia y prevalencia de diversos factores de exposición en todos los grupos poblacionales.

Por otro lado, si se considera que las políticas nacionales de salud deben implementarse en función de la evaluación de su impacto y se toma en cuenta la disponibilidad de recursos humanos, financieros y de infraestructura, surge como prioridad la evaluación, basada en evidencia científica, de las acciones llevadas a cabo. Este curso de acción permite el óptimo ejercicio de los recursos y una rendición de cuentas transparente por parte de los tomadores de decisiones.

Las intervenciones elegidas por los responsables de la salud pública deben orientarse a maximizar la salud de la población general, reducir inequidades en los grupos marginados o vulnerables y a dar respuesta a situaciones de emergencia, pero se necesita de instrumentos como las encuestas nacionales para contener la influencia de criterios no científicos en la toma de decisiones, como aquellos de índole histórica o política, o los que obedecen a la intuición.

La dirección de encuestas se encuentra conformada por los dos grandes grupos de encuestas que han recolectado datos para muchas de las encuestas nacionales de gran escala que ha desarrollado el INSP, el área de sistemas que elabora los programas de captura de los cuestionarios de las encuestas y el área administrativa encargada de realizar los trámites correspondientes por cada proyecto.

2.1.1 Operación actual del levantamiento de información por medio de encuestas en papel.

La manera de operar actualmente en el levantamiento de la información requerida (encuestas) por los investigadores, que desarrollan las dos grandes direcciones de encuestas del sector salud a nivel nacional es la siguiente:

Se realiza de manera centralizada, es decir, se hace la contratación de personal que fungirá como entrevistadores, supervisores de entrevistadores y sus respectivos coordinadores estatales en la ciudad de México, se capacitan y se distribuyen en el territorio Nacional.

Todo esto implica el crear el cuestionario que se aplicará al beneficiario/entrevistado de los programas sociales o al sujeto a estudiar en turno. Este cuestionario es implementado en papel, es decir, que se lleva un gasto

considerado en impresión ya que hablamos de miles de encuestas las que se necesitan.

Posteriormente ya capacitados y con el material necesario para la elaboración de la encuesta parten todos a sus respectivos estados. Al igual que los altos costos de la impresión de los cuestionarios, aquí hay un elevado costo por la transportación de todo el material para la realización de la encuesta hacia los estados. Puede ser esta transportación vía terrestre, marítima o aérea.

Ya instalados en sus entidades respectivas, se dan a la tarea de buscar a las personas a quienes se les aplicará la entrevista para concentrarlos en un punto en común. Esto con la finalidad de realizar las encuestas en una sola jornada y con el mayor número de personas posibles. Se hace de esta manera ya que suelen ser en comunidades rurales la mayoría de las encuestas. Aunque también hay encuestas que se realizan en zonas urbanas en menor proporción.

Ya que se realizaron las entrevistas en papel y se obtuvieron las pruebas necesarias para el caso en particular de la encuesta, se dan a la tarea de enviarlas a la ciudad de México para la captura de la información, proceso que se detallará más adelante.

A parte del levantamiento de la encuesta en papel se hace necesario el llenado de una serie de formatos de reporte que ayudan a llevar el control del levantamiento de la encuesta. Estos reportes deben ser generados por el encuestador en primera instancia para el control personal de las entrevistas que tiene que realizar, los supervisores deben realizar al igual su llenado de formatos de reportes los cuales son generados por el levantamiento de entrevistas de cada entrevistador a su cargo, y finalmente los coordinadores estatales también realizan el llenado de sus formatos de reporte los cuales corresponden a la información obtenida por supervisores y entrevistadores.

A su vez toda esta información generada es concentrada en la ciudad de México por parte del líder de proyecto en turno de la encuesta, para posteriormente generar reportes de avance del levantamiento, desde su nivel más básico que es por Localidad hasta por AGEB[17], por municipio y por entidad. De igual manera se generan reportes por tipo de resultado de entrevista, es decir si es completa o si no se realizó por no encontrar a nadie en el hogar ó porque el entrevistado estaba en estado inconveniente, Etc.

Toda esta información generada del reporte de avance del levantamiento de la encuesta tiene que ser distribuida por correo electrónico para varias figuras como son: el cliente que solicitó la encuesta, directores de áreas y demás personal involucrado en el proyecto.

[17] Área GeoEstadística Básica

Terminado el levantamiento de la encuesta se hace el regreso de todo el personal enviado a campo. También se confirma la llegada de todos los cuestionarios que se levantaron en papel y ahora se da a la tarea de pasar toda esa información a bases de datos que serán las que finalmente utilizan los investigadores.

Para el proceso de la captura de la información se inicia con la validación de los cuestionarios, es decir, se hace la contratación de personal que después de una capacitación hace una revisión exhaustiva de la información de cada uno de todos los cuestionarios obtenidos. Esto es una tarea que lleva mucho tiempo de revisión.

Según sea el caso de la encuesta a veces es necesario realizar una segunda validación de los cuestionarios para que lleven menos errores.

Al ser validados se pasan al área de captura para ser introducidos por más personal que es contratado para este fin. Así pues a la par que se van validando se empieza la captura para que no se alargue tanto el término de entrega.

Por otro lado aproximadamente a la mitad del operativo de levantamiento de la encuesta, al momento de cerciorarse que no habrá más cambios en el cuestionario se comienzan a realizar los programas de captura de los cuestionarios, los cuales se hacen en su mayoría por ser institutos de gobierno de salud en sistemas desarrollados en Clipper y se hace la captura en equipos independientes, teniendo la necesidad de respaldar la información *n* veces según *n* equipos que se tengan para captura diariamente y juntarlo a su vez en otro equipo para su procesamiento.

Este procesamiento se hace diariamente y se necesita hacer algunos otros programas que depuren la información como hacer que encuentre y elimine encuestas duplicadas u otras inconsistencias de captura.

Al igual que la validación se necesita realizar doble captura para poder ser comparada con la primera. Si se encuentran diferencias se pregunta que dato es el correcto para ser capturado por tercera vez y quedar como dato final en la base de datos.

Hay encuestas que son necesarias hasta 3 ó 4 capturas para que sea más confiable la información.

Así pues al concluir la captura de la información es hora de entregar los resultados a los investigadores o clientes, por lo que es necesario entregar la base de datos con un formato y estructura definidos por el cliente. Este formato por lo regular es en SPSS que es un formato de software estadístico, al igual se deben generar etiquetas con sus respectivos valores de respuestas de las variables de las tablas de la base de datos.

2.1.2 Problemáticas generadas por el uso actual de captura de información de las encuestas.

Con esta metodología existente para el levantamiento de la encuesta hay algunas desventajas en el proceso, las cuales representan costos monetarios y dificultades operativas las cuales describiremos a continuación:

Hay un alto costo monetario en la impresión de los cuestionarios, así como el alto costo de la transportación de los cuestionarios hacia las entidades.

De igual manera se dificulta operativamente hablando el andar cargando los cuestionarios en papel por parte del entrevistador, ya que genera una incomodidad y retardo de tiempos así como pérdida de tiempo al tener que regresar por más cuestionarios al punto de encuentro con el personal de la encuesta y regresar al punto de reunión de los beneficiarios/entrevistados, tomando en cuenta que las distancias de traslado son grandes.

Lo mismo sucederá para regresar los cuestionarios a la ciudad de México por el costo de envío.

Por otro lado está que el llenado del cuestionario no siempre es de la mejor manera por lo que hay muchos errores de datos, como no hay forma de validar la información obtenida en campo sistemáticamente se genera esta desviación. Por lo que es necesario de validadores de la información antes de la captura y varias veces se necesita que sea validado hasta por segunda ocasión, teniendo con esto un alto impacto en recursos financieros y de tiempo.

De igual manera la mayoría de las veces se tiene que realizar una segunda captura para reafirmar la información obtenida en la primera captura, haciendo una comparación de las dos capturas y al encontrar diferencias se realiza una tercera captura de la información donde los datos no concordaron. Esto genera un costo extra tanto en tiempo como en dinero.

Otra de las desventaja muy comunes es la congruencia de los datos al momento de aplicar el cuestionario ya que no son validados sistemáticamente, es decir, una persona que dijo estar soltera siguiendo el flujo de la entrevista puede después decir que está casada con alguna persona.

Con respecto al sistema de captura se tienen las siguientes desventajas:

- Problemas para conjuntar la información, ya que cada capturista trabaja con bases de datos independientes.

- Para realizar cualquier cambio en las funcionalidades del sistema se requiere programar por un experto y el tiempo puede ser extenso.

- Los usuarios no reciben alertas y alarmas a la hora de capturar datos erróneos o incongruentes.

- La utilización de los datos es difícil para las personas que la requieren, es decir que no tienen la información oportunamente los investigadores y/o clientes.

- No existe flexibilidad para definir y adaptar los perfiles o roles a los grupos de usuarios.

- Información fragmentada que no permite un análisis integral oportuno.

- No se cuenta con comunicación en red interna y externa.

- No se tiene un modelo de un sistema unificado de información.

2.1.3 Solución propuesta al problema y ventajas.

Por estas desventajas en la forma de hacer encuestas a nivel nacional en diferentes entidades gubernamentales, se requiere de un nuevo sistema de cómputo que ayude a centralizar la información en tiempo real para tomar decisiones de una manera más rápida. Asimismo para que la misma área de sistemas dé un servicio más eficiente a los clientes o investigadores al tener la información reunida en un mismo lugar, se requiere que el sistema pueda ser accesado a través de Internet.

Este sistema será accesado vía Internet para la aplicación de la encuesta, se van a tener dos formas de aplicación: una es la autoaplicación que significa que el beneficiario/entrevistado responderá el cuestionario sin ayuda de nadie y el aplicado por medio de entrevistador, es decir, que habrá una persona físicamente aplicando la entrevista desde el lugar de origen del entrevistado/beneficiario.

La parte medular de este proyecto radicará en la creación de un motor el cual generará las encuestas sin importar su tipo ni el tamaño y sobre todo que sea fácil de crearlas, es decir, que no se necesite un experto de sistemas para programarla. Con esto podemos dar el sistema a los investigadores para que ellos mismos capturen el cuestionario como lo deseen.

Se quiere llegar a esto de una manera tan sencilla posible, esto es únicamente indicando el texto de la pregunta, el tipo de respuesta a obtener, este tipo de respuesta podrá ser de respuesta cerrada única, repuesta cerrada múltiple o de respuesta abierta. También se indicará el nombre de la variable donde se desee guardar la(s) respuesta(s) así como el nombre de la tabla donde se alojará esta variable. Por lo que habrá un módulo en el sistema el cual permitirá introducir esta información.

Se indicarán hacia que pregunta se irá la encuesta dependiendo de la respuesta dada en la pregunta que estemos. Se indicará los textos de las respuestas así como también el número de respuestas a mostrar.

Dados estos parámetros el sistema generará la encuesta automáticamente, esto es que se vaya generando una pregunta por pantalla con sus respectivas respuestas utilizando únicamente una pantalla o forma así como únicamente un solo objeto de control para cada tipo de respuesta por encuesta.

Si la respuesta es de tipo única cerrada se hará uso de un objeto de botón. Así el programa mostrara 'n' botones de respuesta según requiera la pregunta. Al momento de responder pasará a la siguiente pregunta que se cargará por medio del motor. Para el tipo de respuesta múltiple cerrada usaremos el objeto de checkbox y utilizando la misma lógica se mostraran ´n´ checks con sus respectivos textos de respuestas. Para el tipo de respuestas de tipo abiertas nos presentará un objeto de caja de texto para poder introducir la respuesta deseada.

El motor generador de encuesta nos permitirá salir de la encuesta en el momento que lo deseemos y poder regresar en otro momento al punto donde habíamos quedado cuando salimos, habiendo guardado toda la información recabada, así como también permitirá regresar a la pregunta anterior inmediata respondida para poder corregirla o hacer alguna modificación en la respuesta.

Con respecto al módulo del llenado de la entrevista mencionamos que se podrán hacer de 2 maneras: autoaplicable o aplicada por medio de un entrevistador frente a frente con el beneficiario/entrevistado.

Cuando es autoaplicable el sistema enviará un correo electrónico a los beneficiarios/entrevistados con una breve explicación de la encuesta así como con la liga la cual enviará a la pantalla principal del sistema donde tendrá que autenticarse, a partir de ahí entrará a la encuesta automáticamente.

Cuando es aplicada la encuesta por medio de un entrevistador este se dirigirá a la comunidad deseada para reunir a los beneficiarios/entrevistados y aplicarles la encuesta, solamente dará la dirección URL para ingresar al sistema y entrará a la pantalla de autenticación.

Para el módulo que respecta a la consulta y creación de reportes de avance de las entidades únicamente podrán accesar a ellos los supervisores, coordinadores y el cliente o investigador del proyecto. Estos permisos se darán por medio de la autenticación. Estos reportes se harán por fecha y por tipo de respuesta de la entrevista, a su vez se harán por localidad, municipio y por entidad. Además se necesitarán reportes de porcentajes de avance con respecto al número de encuestas a realizar.

Finalmente para el módulo de entrega de la base de datos se hará en formato SPSS que es un formato de un software estadístico las cuales contienen las tablas con etiquetas que son los textos de las preguntas así como los valores de las respuestas de cada pregunta en caso de que no sean abiertas. A este módulo solamente tendrá acceso el investigador o cliente.

Cabe mencionar que habrá una figura como administrador del sistema el cual podrá ingresar a todos los módulos sin restricción alguna.

2.1.4 Necesidades y requerimientos.

Dado el acceso a la encuesta se presentará una pantalla de bienvenida al beneficiario/entrevistado la cual le dará una breve explicación de lo que trata la encuesta. Dependiendo si la encuesta a aplicar tiene una muestra a la cual se le realizará la encuesta o si será nuevo beneficiario/entrevistado al cual se le valorará si entra al programa social, se presentará una pantalla a la cual elegirán un folio que se le asignó automáticamente. Este folio igualmente será enviado en el correo en caso de existir, o en su defecto si no son pertenecientes al programa social se mostrará una opción donde les creará un nuevo folio.

Posteriormente deberá permitirle capturar en otra pantalla los datos de identificación geográfica del beneficiario/entrevistado, la cual obtendrá información necesaria para ubicar el lugar donde reside el beneficiario/entrevistado. Entonces ya podrá dar inicio a la encuesta electrónica.

Al término de la encuesta se emitirá una constancia de realización de la encuesta en un documento electrónico el cual será enviado al correo electrónico del beneficiario/entrevistado y saldrá del sistema automáticamente. El beneficiario/entrevistado tendrá la oportunidad de salir de la encuesta en el momento que desee aun no habiendo terminado esta, y podrá acceder a esta en el momento que desee regresándolo al mismo lugar de la encuesta donde se quedó anteriormente.

La segunda etapa de consulta y realización de reportes será mediante una página Web la cual podrán tener acceso únicamente supervisor e investigadores y el ingeniero de sistemas que le dará mantenimiento al sistema.

Esta página contendrá opciones que dirán que acción deseo realizar, consultas de los avances para lo cual se pedirá primero la contraseña y nombre de usuario. El investigador y/o supervisor que realice las consultas y/o reportes elegirá que programa social es el que necesitará para obtener información. Después de esto se desplegará una lista de posibles consultas a realizar. Estas consultas serán:

Consultas de avance por entidad, consultas de avance por municipio, consulta de avance por AGEB, consulta de avance por ruta, consulta de avance por encuestador. Así como también se desplegará una lista de opciones de reportes a

obtener y que serán imprimibles. Estos reportes serán básicamente del mismo estilo que las consultas.

Finalmente podrá salir de la creación de consultas y reportes dándole un desplegado de agradecimiento y saldrá del sistema.

La tercera etapa consta de la creación de cuadros estadísticos, cruces de variables, y demás análisis. Esto será dependiendo de las reglas que el investigador indique. Por medio de esos cuadros y el análisis de estos, el investigador podrá dictaminar teoremas o postulados que servirán para evaluar el programa social, así como el impacto que tiene en los beneficiarios/entrevistados.

Para acceder a esta parte será mediante una autentificación del investigador que será formado por un nombre de usuario y una contraseña, los cuales los investigadores ya tendrán conocimiento de ellas. Este análisis de los cuadros estadísticos será imprimible.

Como etapa extra, se encuentra el módulo donde se cambiarán los parámetros de la encuesta, es decir, el cambiar la encuesta para otro tipo de programa social. Esto se hará por medio del ingeniero de sistemas que será el encargado de introducir las nuevas preguntas y las nuevas respuestas, así como también las validaciones correspondientes a los reactivos. Esto será guiándose del cuestionario creado por las personas que hacen el programa social.

Con este sistema se podrán mejorar y corregir algunos puntos como son:

Desaparecer la técnica utilizada actualmente aplicando la encuesta por medio de cuestionarios en papel. Aminorar considerablemente los gastos tanto de impresiones de cuestionarios, envíos de paquetería, captura y validación de los cuestionarios. En el tema de impresión de cuestionarios también tendrá impacto a favor del medio ambiente, también será un menor desgaste físico por parte del personal ya que no tendrán que cargar paquetes en el transcurso de su labor de encuestar y así será mejor su desempeño laboral.

Pasando a la parte técnica mencionamos que por las características del desarrollo, en este caso el acceso a través de Internet se propone que sea un sistema en Web, con lenguaje de programación VisualBasic.NET y con procedimientos almacenados, con estos últimos separamos la lógica de negocio con la capa de presentación, asimismo hace más flexible el mantenimiento de la aplicación. Este es un lenguaje de última generación para la creación de sitios Web de alto desempeño, con acceso a bases de datos remotas, permitiendo hacer aplicaciones dinámicas con manipulación de datos, tiene una gran afinidad con los manejadores de bases de datos SQL, es una combinación de Visual Basic con ASP.NET, permitiendo un diseño eficiente para las interfaces gráficas y programar los eventos internos propios de una página Web.

Como Base de Datos se utilizará Microsoft SQL Server 2005, ya que es un Sistema Manejador de Bases de Datos relacionales con un amplio soporte técnico, este administrador de base de datos tiene gran capacidad de almacenamiento al estar limitada por el especio del servidor, la cual puede crecer al aumentar la capacidad del servidor, la capacidad de SQL también aumenta, aunado a esto la programación de respaldos locales y remotos de forma periódica garantiza que la pérdida de datos sea minima, además cuenta con la posibilidad de importar y exportar datos desde y hacia otras herramientas comerciales como son las hojas de cálculo, procesadores de palabras entre los mas comunes y por último la gran compatibilidad con el conjunto de herramientas de programación visuales como los lenguajes .NET, además de que es el software de manejador de bases de datos oficial en las áreas gubernamentales de salud y se cuenta con la licencia correspondiente para la utilización del software.

Como servidor de aplicación utilizaremos el IIS de Microsoft. Se seleccionó el IIS(Los servicios de Microsoft Internet Information) porque admite la creación, configuración y administración de sitios Web; incluye el Protocolo de transferencia de noticias a través de la red (NNTP), el Protocolo de transferencia de archivos (FTP) y el Protocolo simple de transferencia de correo (SMTP). Facilita la publicación de información en una intranet o en Internet; autenticación robusta y segura de los usuarios, así como comunicaciones seguras vía SSL; además de crear contenido dinámico utilizando los componentes y secuencias de comandos del servidor independiente del explorador mediante páginas Active Server (ASP).

Admite servidor PHP y la base de datos SQL, de modo que se puedan ejecutar páginas dinámicas PHP, así como enlazar tanto desde ASP como desde PHP con dicho gestor de base de datos; también permite construir espacios seguros con SSL, así como espacios privados.

2.2 Requisitos de Hardware

Monitor: Las herramientas gráficas de SQL Server requieren VGA o una resolución mayor: resolución mínima de 1.024 x 768 píxeles.

Dispositivo señalador: Se requiere Microsoft Mouse o un dispositivo señalador compatible.

Unidad de CD o DVD: Se requiere una unidad de CD o DVD, según proceda, para instalar desde un CD o DVD.

Requisitos de hardware de clúster: En las plataformas de 32 bits, se admiten las instalaciones de clústeres de ocho nodos (es decir, el número máximo de nodos que admite Microsoft Windows Server 2003).

SQL Server 2005 (32 bits)	Tipo de procesador	Velocidad de procesador	Memoria (RAM)
SQL Server 2005 Standard Edition	Se requiere un procesador compatible con Pentium III o superior	Mínimo: 600 MHz Recomendado: 1 GHz o superior	Mínimo: 512 MB Recomendado: 1 GB o superior Máximo: sistema operativo máximo

Fig. 2.1 Requisitos de Hardware para el MDBD SQL.

Los requisitos de espacio en disco duro reales para los componentes instalados dependen de la configuración del sistema y de las aplicaciones y características que haya decidido instalar. En la siguiente tabla se muestran los requisitos de espacio en disco de los componentes de SQL Server 2005.

Característica	Espacio en disco duro
Motor de base de datos y archivos de datos, Réplica y Búsqueda de texto	150 MB
Analysis Services y archivos de datos	35 MB
Reporting Services y Administrador de informes	40 MB
Componentes del motor de Notification Services, componentes de cliente y componentes de reglas	5 MB
Integration Services	9 MB
Componentes de cliente	12 MB
Herramientas de administración	70 MB
Herramientas de desarrollo	20 MB
Libros en pantalla de SQL Server y Libros en pantalla de SQL Server Mobile	15 MB
Ejemplos y bases de datos de ejemplo	390 MB

Fig. 2.2 Requisitos de HD.

2.3 Requisitos de Software

Requisitos de software de red: Windows 2003, Windows XP y Windows 2000 que tienen software de red integrado.

Requisitos de Internet: Los requisitos de Internet son para la versión de 32 bits de SQL Server 2005. La tabla siguiente indica los requisitos de Internet para SQL Server 2005.

Componente	Requisito
Software de Internet	Se requiere SP1 de Microsoft Internet Explorer 6.0 ó posterior para todas las instalaciones de SQL Server 2005, ya que se necesita para Microsoft Management Console (MMC) y la Ayuda en HTML. Es suficiente una instalación mínima de Internet Explorer, pero no es necesario que sea el explorador predeterminado. Sin embargo, si sólo instala componentes de cliente y no se conecta a un servidor que requiere cifrado, es suficiente Internet Explorer 4.01 con Service Pack 2.
Servicios de Internet Information Server (IIS)	Se necesita IIS 5.0 ó posterior para las instalaciones de Microsoft SQL Server 2005 Reporting Services (SSRS). Para obtener más información acerca de cómo instalar IIS, vea el tema "Cómo instalar los Servicios de Microsoft Internet Information Server (IIS)" en los Libros en pantalla de SQL Server 2005.
ASP.NET 2.0	Se requiere ASP.NET 2.0 para Reporting Services. Durante la instalación de Reporting Services, el programa de instalación de SQL Server habilitará ASP.NET en el caso de que no se encuentre habilitado.

Fig. 2.3 Requisitos de Software.

El programa de instalación de SQL Server requiere Microsoft Windows Installer 3.1 ó posterior, y SP1 de Microsoft Data Access Components (MDAC) 2.8 ó posterior.

Requisitos del sistema operativo (32 bits): Esta tabla muestra los sistemas operativos que ejecutan el software de servidor para la versión de 32 bits de SQL Server 2005.

Sistemas operativos	Standard Edition
Windows 2000	No
SP4 de Windows 2000 Professional Edition	Sí
SP4 de Windows 2000 Server	Sí
SP4 de Windows 2000 Advanced Server	Sí
SP4 de Windows 2000 Datacenter Edition	Sí
Windows XP Embedded	No
SP2 de Windows XP Home Edition	No
SP2 de Windows XP Professional Edition	Sí
SP2 de Windows XP Home Edition	Sí
SP2 de Windows XP Home Edition	Sí
SP1 de Windows 2003 Server	Sí
SP1 de Windows 2003 Enterprise Edition	Sí
SP1 de Windows 2003 Datacenter Edition	Sí
SP1 de Windows 2003 Web Edition	No
SP1 de Windows Small Business Server 2003 Standard Edition	Sí
SP1 de Windows Small Business Server 2003 Standard Edition	Sí
SP1 de Windows 2003 64 bits Itanium Datacenter Edition	No
SP1 de Windows 2003 64 bits Itanium Enterprise Edition	No

Fig. 2.4 Tipos de Sistema Operativo.

SQLSERVER2005

Se usará este administrador de base de datos por tener gran capacidad de almacenamiento al estar limitada por el especio del servidor, la cual puede crecer al aumentar la capacidad del servidor, la capacidad de SQL también aumenta, aunado a esto la programación de respaldos locales y remotos de forma periódica garantiza que la pérdida de datos sea mínima, además cuenta con la posibilidad de importar y exportar datos desde y hacia otras herramientas comerciales como son las hojas de cálculo, procesadores de palabras entre los más comunes y por último la gran compatibilidad con el conjunto de herramientas de programación visual como los lenguajes Net.

VISUAL WEB BASIC DEVELOPER

Es un lenguaje de última generación para la creación de sitios web de alto desempeño, con acceso a bases de datos remotas, permitiendo hacer aplicaciones dinámicas con manipulación de datos, tiene una gran afinidad con los manejadores de bases de datos SQL, es una combinación de Visual Basic con asp.net, permitiendo un diseño eficiente para las interfaces gráficas, y programar los eventos internos propios de una página web.

2.4 Análisis Costo-Beneficio.

Retomando el objetivo de la presente investigación la cual es analizar, diseñar y realizar un ejemplo de desarrollo de una herramienta la cual realizará encuestas electrónicas de vía Internet. Haremos el análisis de costo-beneficio empezando por hacer un análisis del presente, es decir de los costos que en este momento llevaría hacer estas encuestas en formato de papel y posteriormente capturar los datos.

SITUACIÓN Y COSTOS ACTUALES

Es difícil poder dar costo ya que depende del tamaño que tenga la encuesta, así como de la logística del levantamiento. Sin embargo trataremos de dar un estimado tomando un ejemplo de un proyecto. Este proyecto tendrá un cuestionario de 56 páginas el contiene un total 850 preguntas y se requerirán 6,000 hogares a visitar.

Descripción		Número de páginas	Total de cuestionarios	Factor que indica cuántos más	Costo x página	Costo Estimado
Impresión de cuestionarios		56	6000	6900	$0.20	$77,280.00
Material de Papelería (bolígrafos, lápices, gomas, sobres, cajas, libretas)						$10,000.00

Descripción			Cuestionarios por caja	Costo por caja de 5,000 hojas	Total de cajas	Costo Estimado
Costo de envío de paquetería de ida			89.29	$500.00	77.28	$38,640.00
Costo de envío de paquetería de vuelta			89.29	$500.00	77.28	$38,640.00

Descripción	Total de preguntas	Tiempo de programación por pregunta (minutos)	Total de horas de programación	Costo hora/programación		Costo Estimado
Sistema de captura forma local	850	15	212.5	$150.00		$31,875.00

Descripción	Total de preguntas	Costo por golpe de captura	Costo de captura por cuestionario	Costo por capturar 6,000 cuestionarios	Número de capturas	Costo Estimado
Capturistas	850	$0.10	$85.00	$510,000.00	2	$1,020,000.00

Descripción			Total de cuestionarios	Costo por cuestionario		Costo Estimado
Validadores de la información			6,000	$25.00		$150,000.00

Costo Estimado	$1,366,435.00

Este estimado es un poco subjetivo ya que depende también los costos de envío de paquetería del lugar donde se encuentren, de igual manera no siempre se capturan los 850 datos por cuestionarios, es decir que podrán ser menos dependiendo del flujo que tenga la entrevista. Así pues pasaremos al costo beneficio implementando el sistema generador de encuestas vía internet.

COSTO DEL SISTEMA QUE GENERA ENCUESTAS ELECTRÓNICAS VÍA INTERNET

Implementando el sistema que genera encuestas electrónicas vía Internet tenemos las siguientes cuantificaciones:

Descripción	Total	Total de preguntas	Tiempo de programación por pregunta por minuto	Tiempo de desarrollo (hrs.)	Costo * Hr.	Costo unitario	Costo Total
Programación		850	25	354.17	$150.00		$53,125.00
Software Visual Studio 2005							$7,646.00
Software SQL server 2005							$88,452.00
Equipo Nuevo (laptop)	60					$15,000.00	$900,000.00
Capacitación							$10,000.00

Costo Total	$1,059,223.00

BENEFICIOS TANGIBLES

Un sistema de cómputo puede producir una gama tan grande e indefinible de servicios y logros difíciles de cuantificar, que convierten en algo compleja, la tarea de estimar los beneficios tangibles que ofrecerá el sistema.

En el mejor de los casos se llega a enumerar un conjunto de beneficios intangibles tales como:

- Se reducirá el costo de personal.
- Se lograrán disminuir los tiempos de atención a los usuarios.
- Se reducirá el número de errores humanos en algún proceso.

Por ello hay que determinar beneficios tangibles, que puedan ser medidos en forma numérica, tanto cuando se les enuncia, como posteriormente cuando se evalúen los beneficios.

Así pues dentro de los beneficios tangibles podemos mencionar que prácticamente todos los gastos que se tienen a como se opera en el presente se evitarían, siendo así los siguientes conceptos:

Descripción	Costo Estimado
Impresión de cuestionarios	$77,280.00
Material de Papelería (bolígrafos, lápices, gomas, sobres, cajas, libretas)	$10,000.00

Descripción	Costo Estimado
Costo de envío de paquetería de ida	$38,640.00
Costo de envío de paquetería de vuelta	$38,640.00

Descripción	Costo Estimado
Sistema de captura forma local	$31,875.00

Descripción	Costo Estimado
Capturistas	$1,020,000.00

Descripción	Costo Estimado
Validadores de la información	$150,000.00

Costo Total Estimado	$1,366,435.00

Así también hay otro tipo de beneficios como los son:

Que todos los datos estarán centralizados en una sola base de datos.

Las mejoras a la programación se hacen en un solo lugar, evitando la instalación en cada computadora.

Se pueden hacer informes con los datos en tiempo real.

Se evita el gasto innecesario de papel al dejar de imprimir tantos cuestionarios, este rubro afecta también directamente a favor del medio ambiente.

El equipo de cómputo (laptops) así como el software comprado es reutilizable, es decir, que en proyectos subsecuentes se eliminarán en la lista de gastos (inversión) a realizar y es una suma muy considerable.

Se podrá tener información más veraz toda vez que se podrá validar la información al momento de ser capturado.

Y uno de los beneficios más importantes que se obtienen de esto es que se podrá tener una base de datos recién terminado el levantamiento de la información y ya no meses después como es que se opera actualmente.

BENEFICIOS INTANGIBLES

Hay una serie de usuarios y personas que serán beneficiados por este sistema, desde la parte de encuestadores los cuales ya no tendrán que preocuparse por ir siguiendo una serie de instrucciones las cuales le permitan tener un llenado perfecto de sus cuestionarios, ya que este sistema los estará guiando automáticamente así como irá validando la información conforme la vayan capturando.

Otra de las ventajas que se tendrán es que se reducirán muchísimos los costos al no tener que enviar personal a encuestar directamente a los lugares de origen de los encuestados cuando sea el caso de la encuesta autoaplicable, esto es que con una simple conexión de Internet que tenga a la mano el encuestado podrá ir a realizar su encuesta.

Una gran ventaja es que conforme avanza el operativo y al tener la información en tiempo real, se puede hacer análisis de cómo va funcionando el operativo, es decir que de ser necesario según los datos arrojados por la base de datos, se pueda cambiar la logística o realizar cambios al tipo de muestreo, Etc.

Otro de los beneficios es que este Sistema que genera encuestas electrónicas vía Internet, podrá ser utilizado no solo para realizar encuestas a programas sociales, sino que puedan servir en un futuro como precedente para que sea a todos niveles, es decir que pueda ser usado por encuestas de todo tipo y que toda institución ya sea gubernamental o privada pueda apoyarse de esta herramienta.

ANÁLISIS FINAL DEL COSTO-BENEFICIO

Finalmente podemos concluir que en principio de cuentas el tiempo utilizado para estos cálculos es por proyecto, es decir que esta será nuestra unidad de medida de tiempo.

El costo de la implementación de proyecto, será de **$1,059,223.00**

El costo a evitar de cómo opera actualmente por proyecto es de: **$1,366,435.00**

Diferencia de costos por proyecto con computadoras nuevas: $1,366,435.00 - $1,059,223.00 = **$307,212.00**

Total por proyecto de costos a favor:
$1,366,435.00+ $307,212.00 = **$1,673,647**

Tiempo de recuperación (Relación Costo-Beneficio): $1,059,223.00 / $1,673,647 = 0.6329 % de avanzado el proyecto.

Estos montos se podrán modificar mucho mas a favor, cuando en un proyecto ya no se tenga que comprar equipo de cómputo y únicamente se tenga que pagar el mantenimiento de este. Estamos hablando de ya no sumar $15,000.00 por laptop.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Diagramas de flujo de datos.

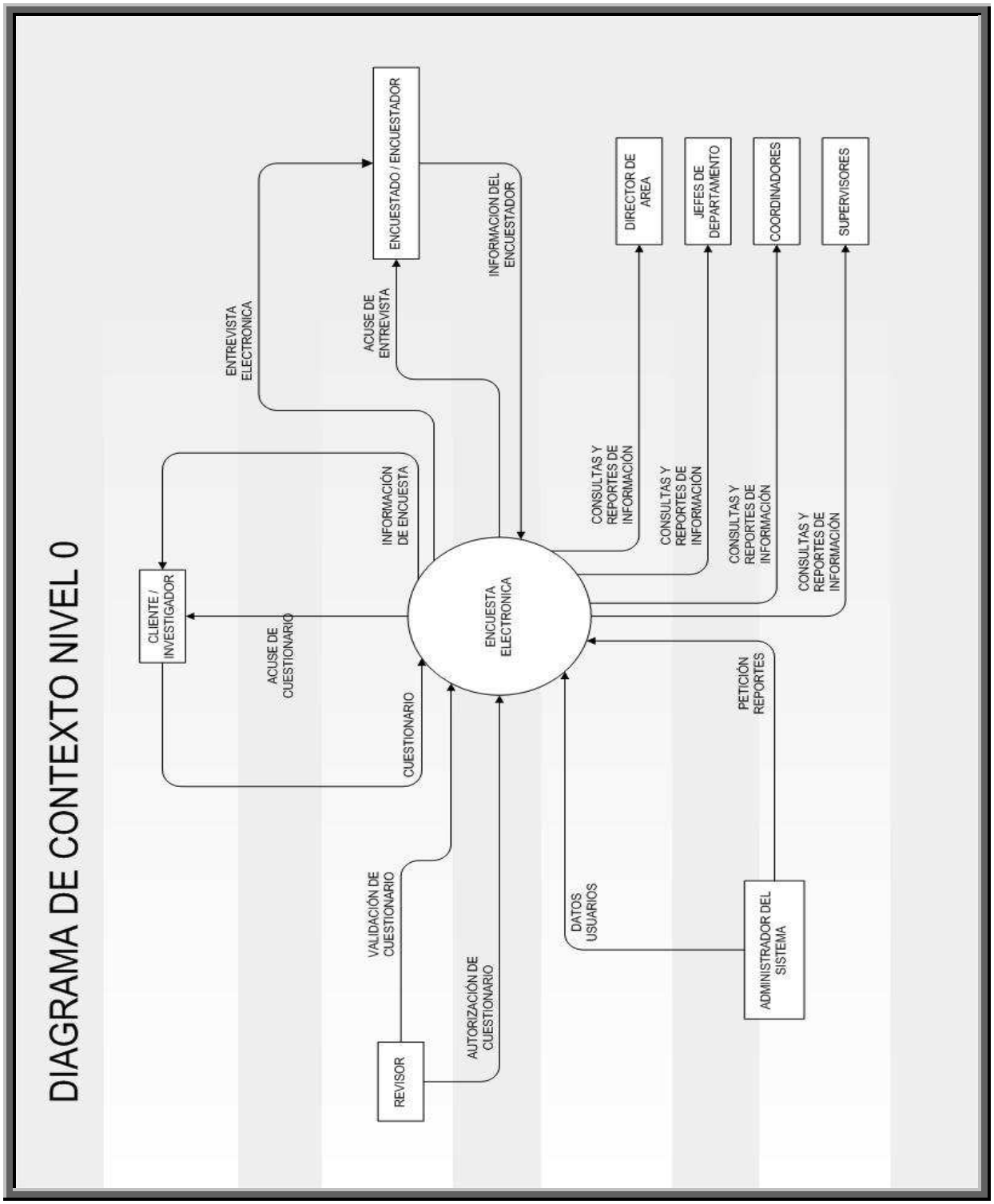


Fig. 3.1 Diagrama de contexto nivel 0.

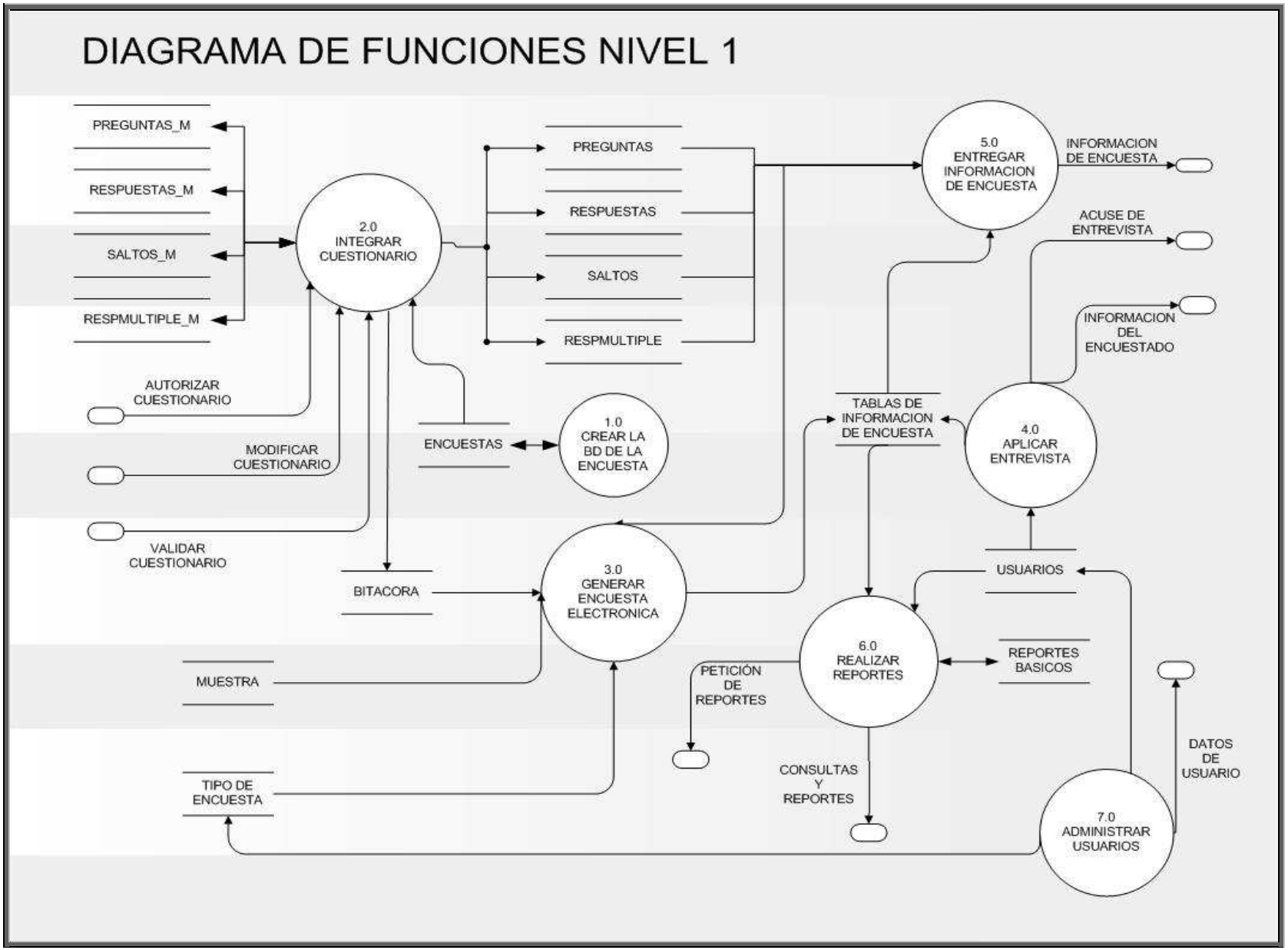


Fig. 3.2 Diagrama de funciones nivel 1.

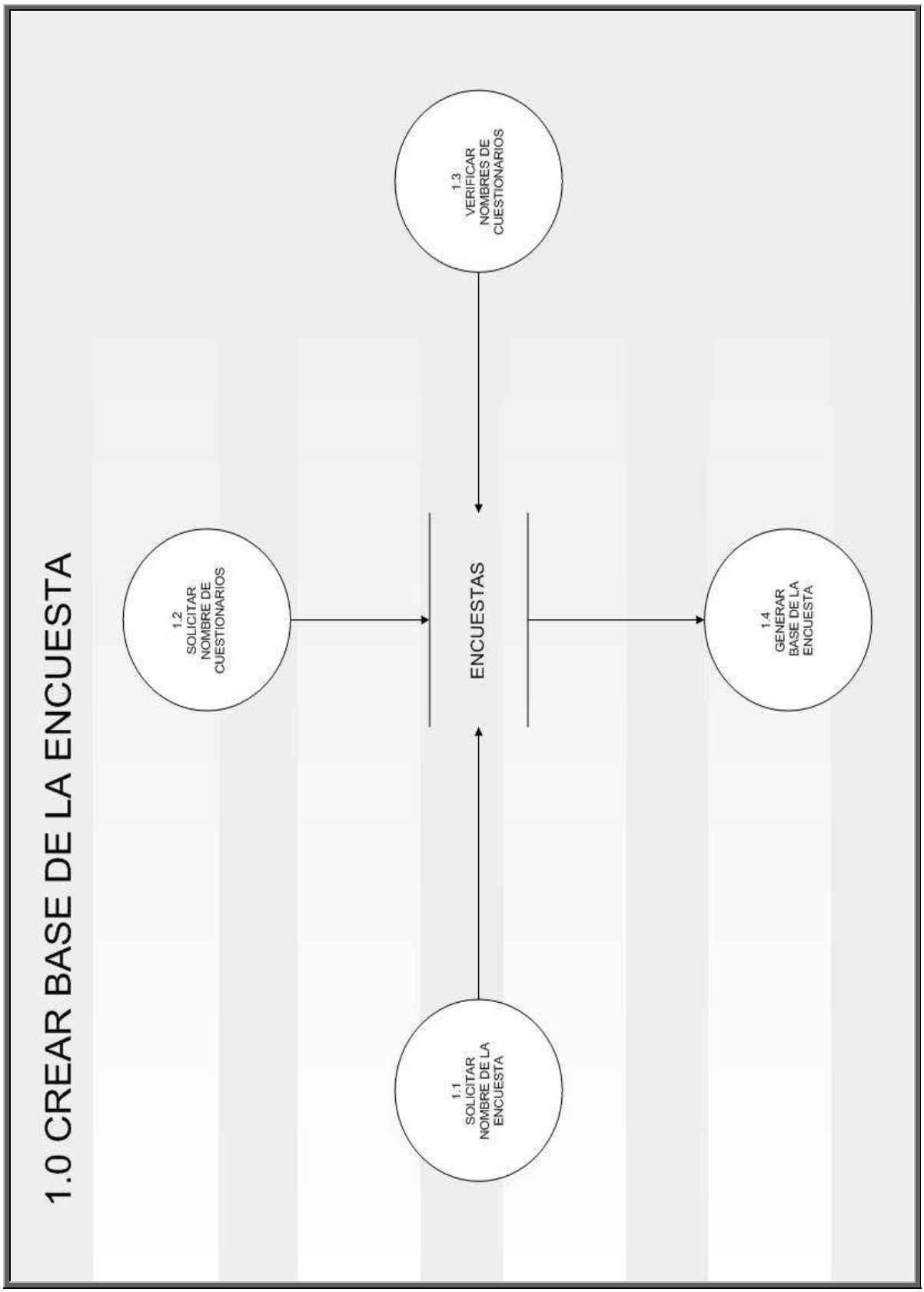


Fig. 3.3 Diagrama Flujo de datos nivel 2.

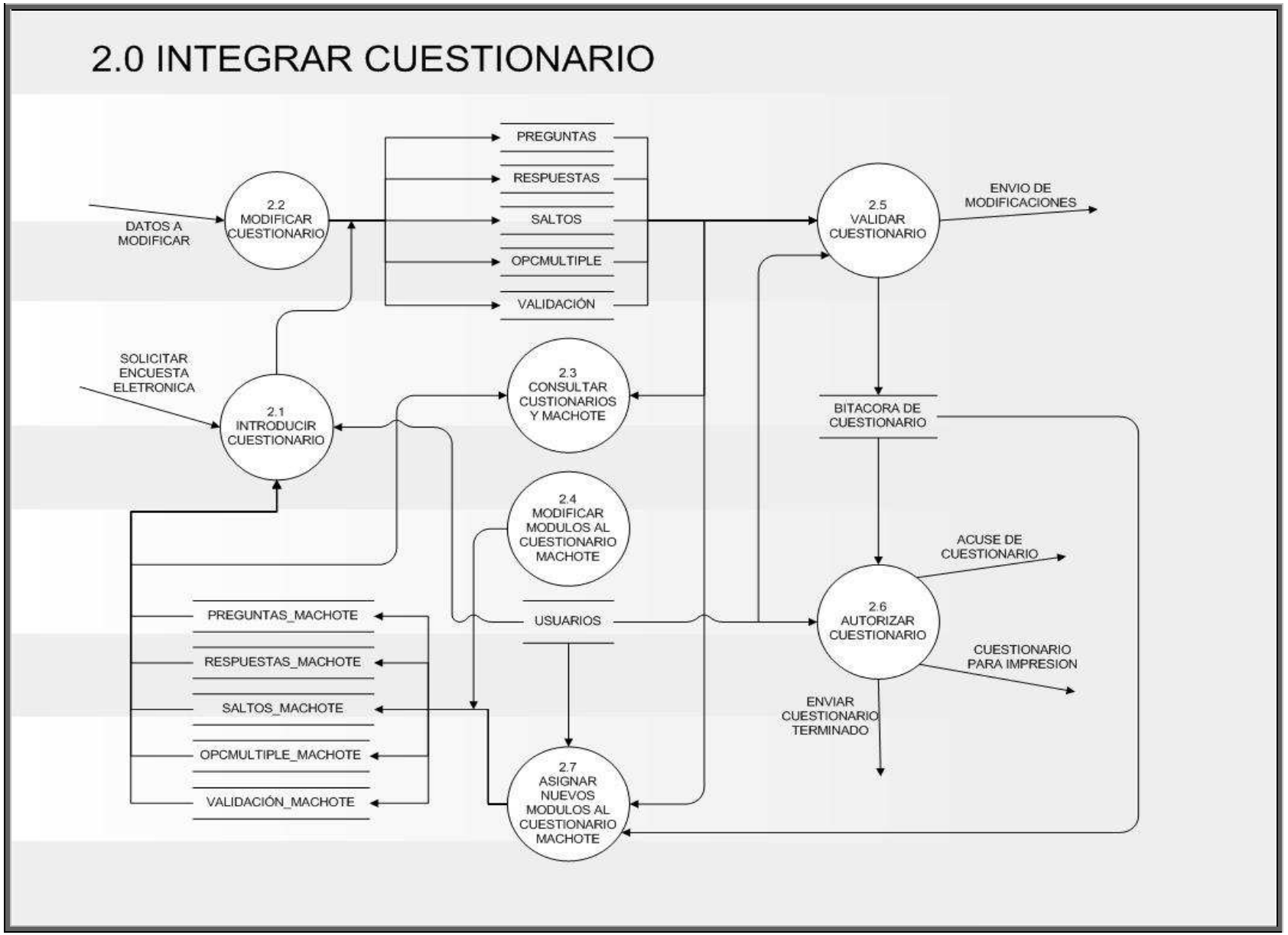


Fig. 3.4 Diagrama Flujo de datos nivel 2.

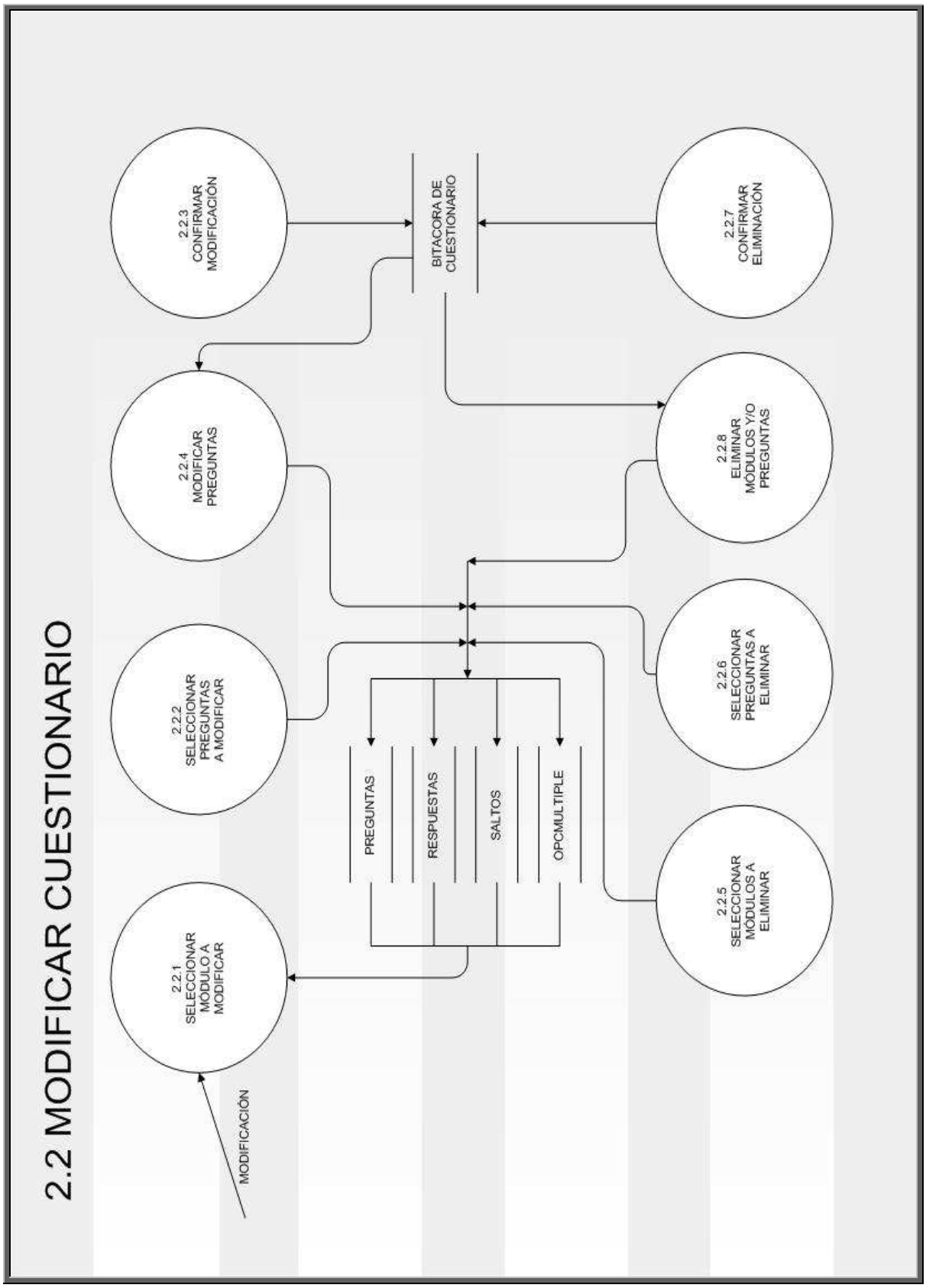


Fig. 3.5 Diagrama Flujo de datos nivel 3.

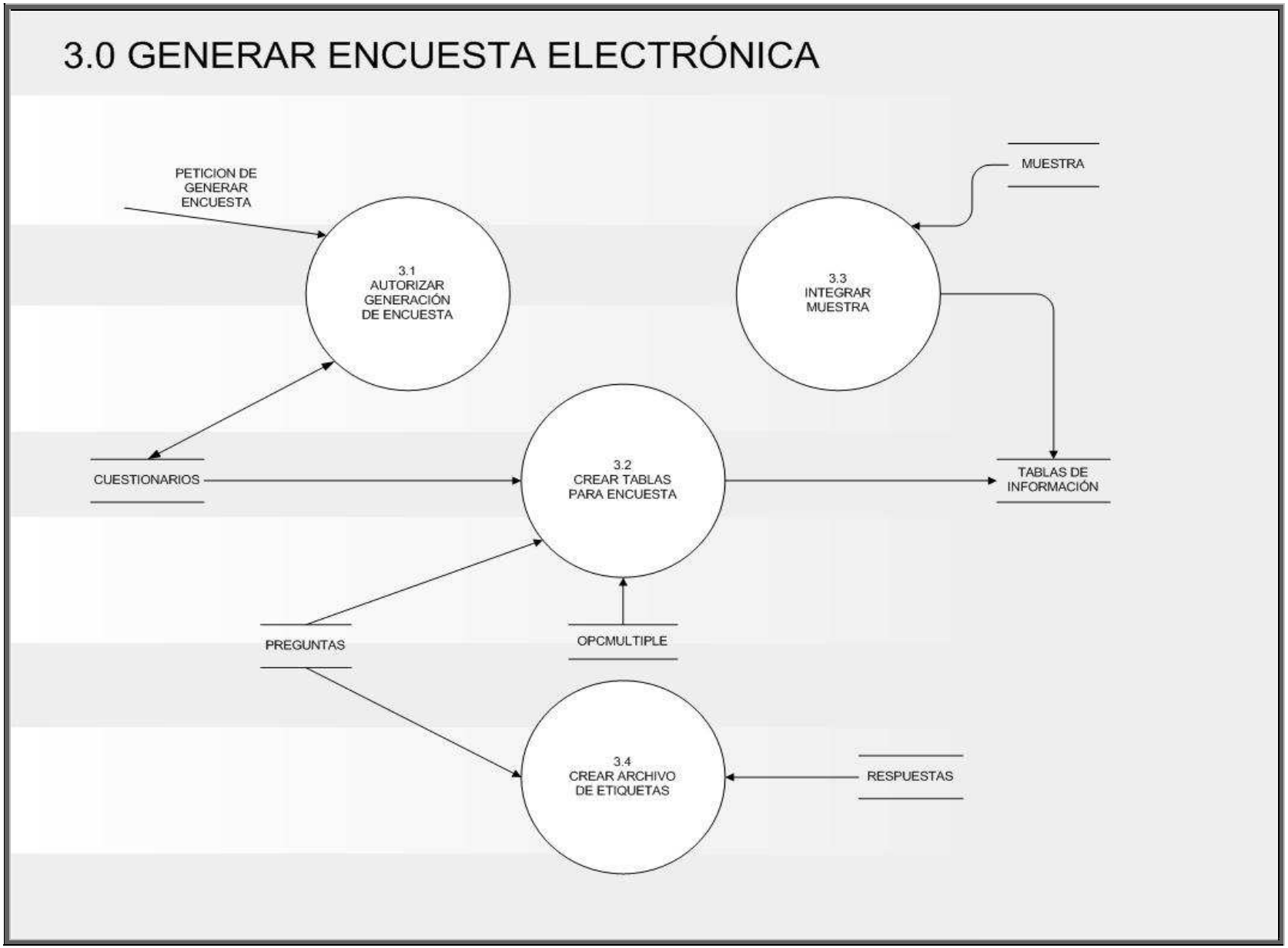


Fig. 3.6 Diagrama Flujo de datos nivel 2.

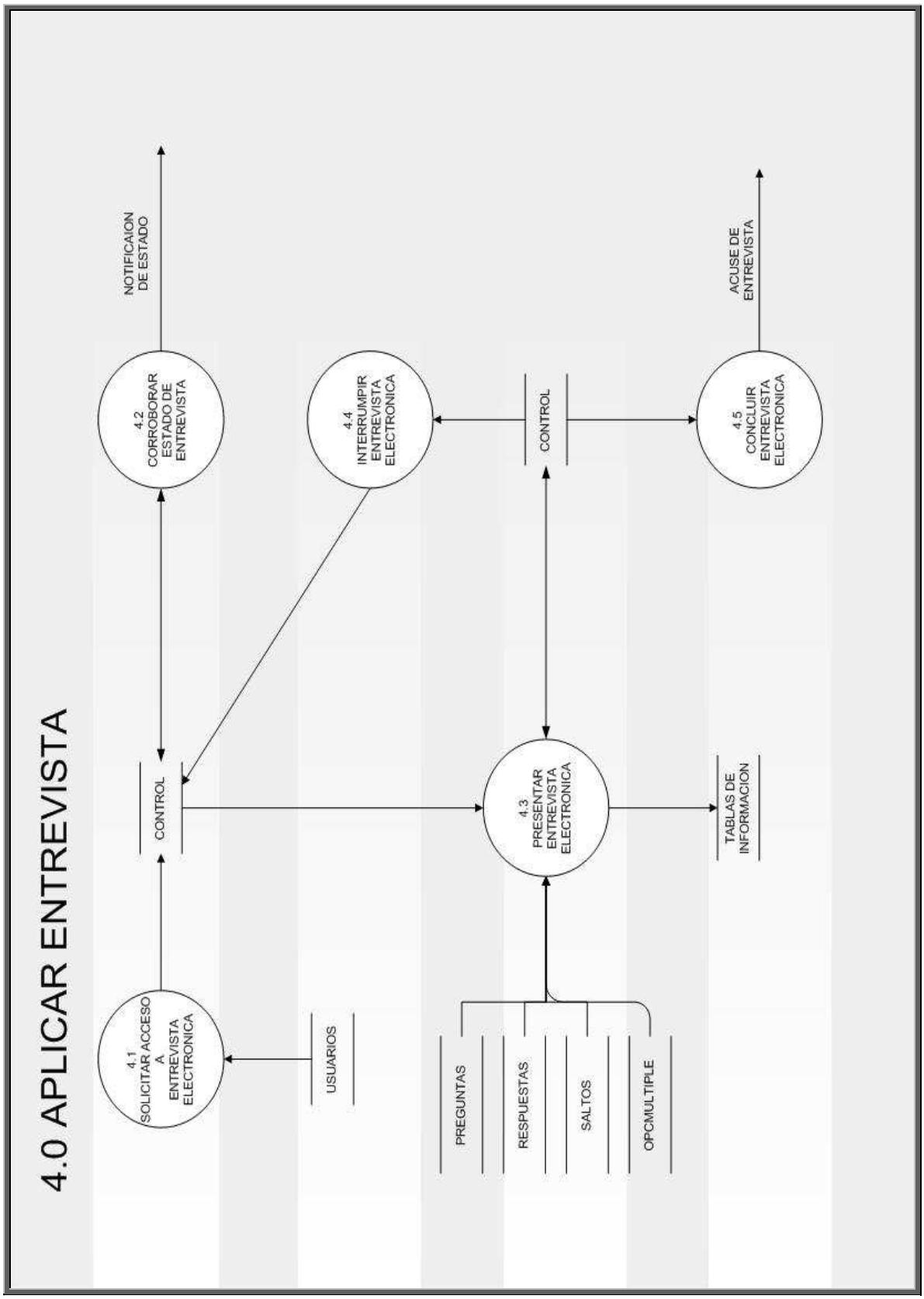


Fig. 3.7 Diagrama Flujo de datos nivel 2.

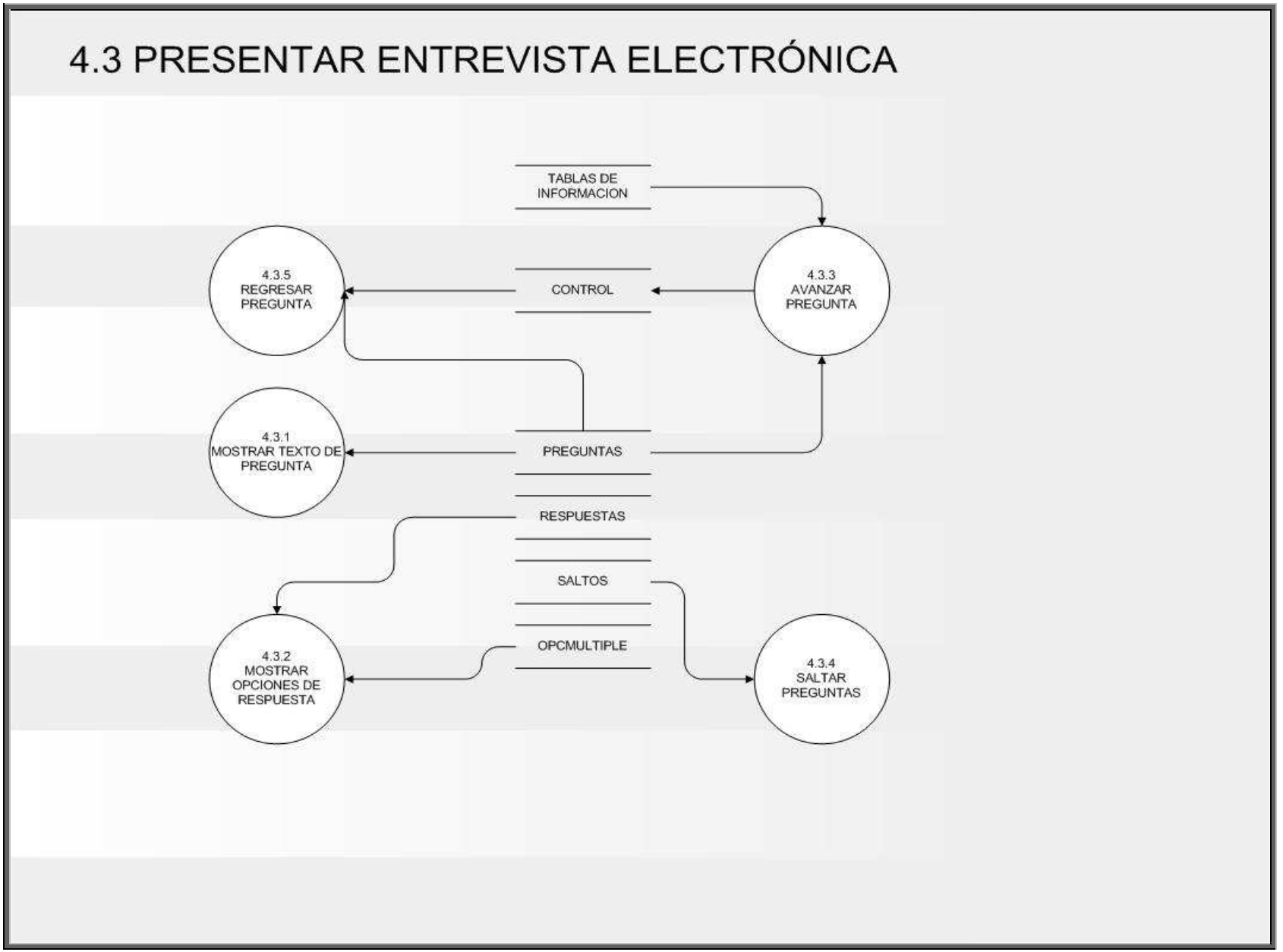


Fig. 3.8 Diagrama Flujo de datos nivel 3.

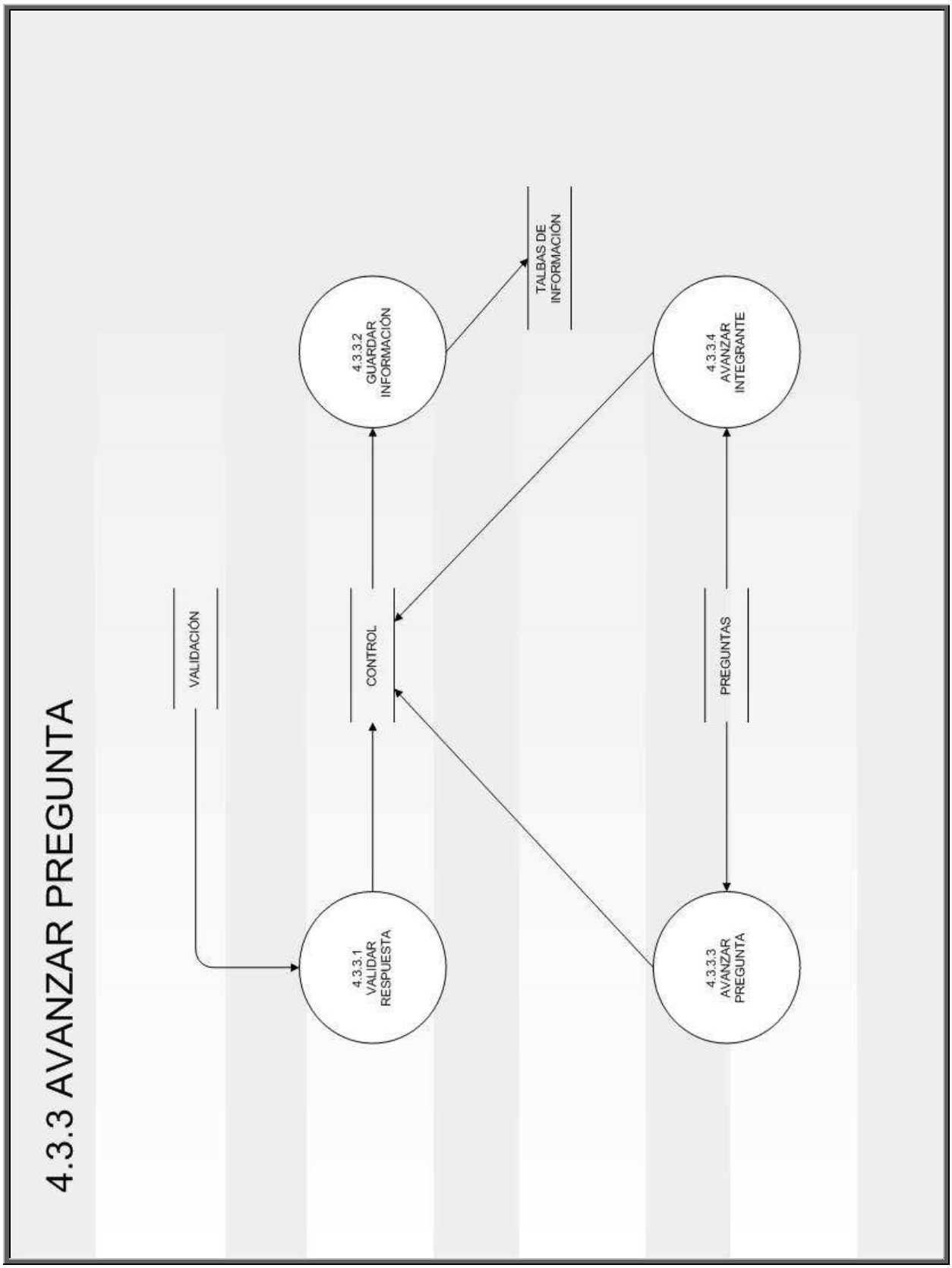


Fig. 3.9 Diagrama Flujo de datos nivel 4.

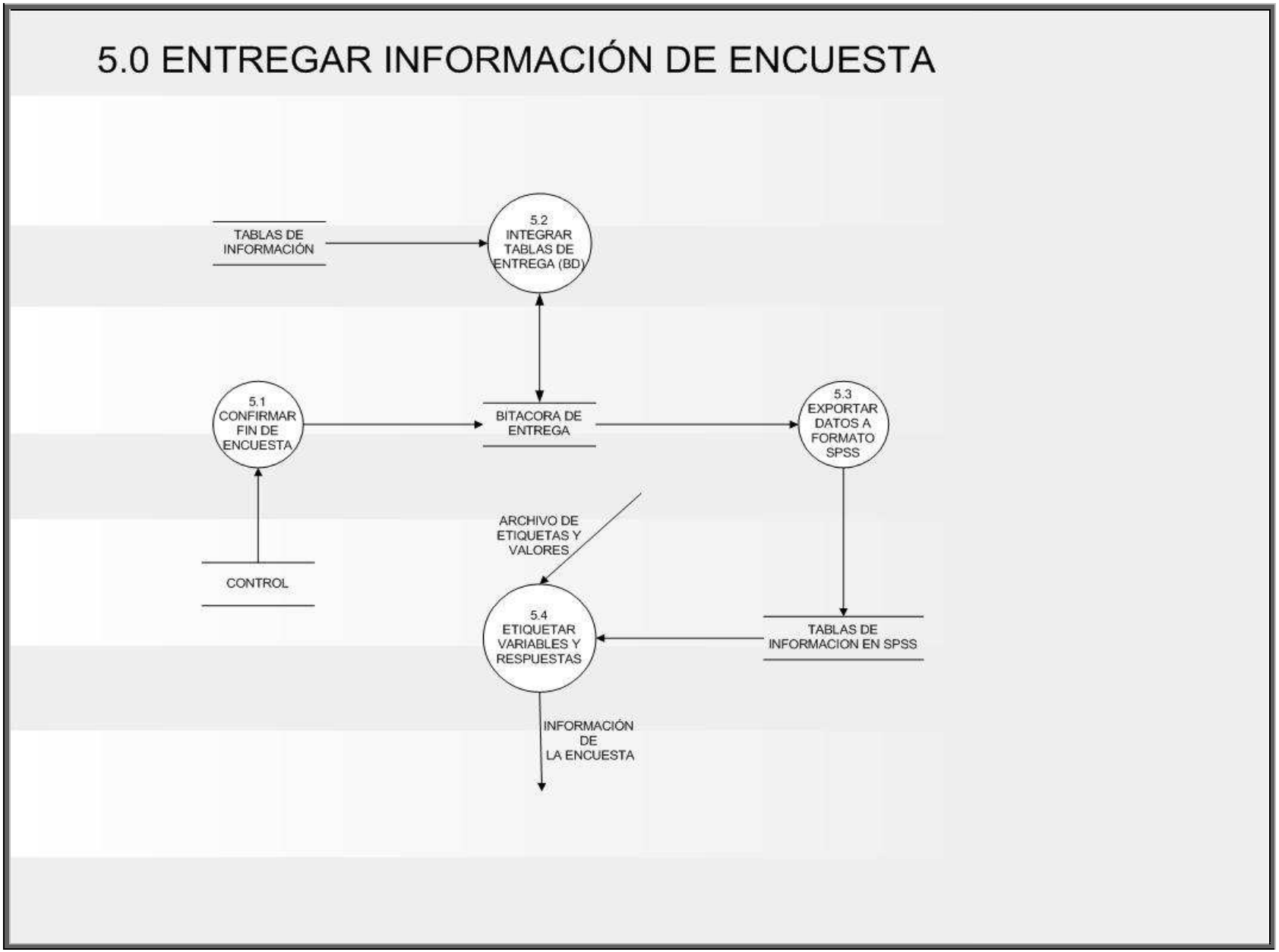


Fig. 3.10 Diagrama Flujo de datos nivel 2.

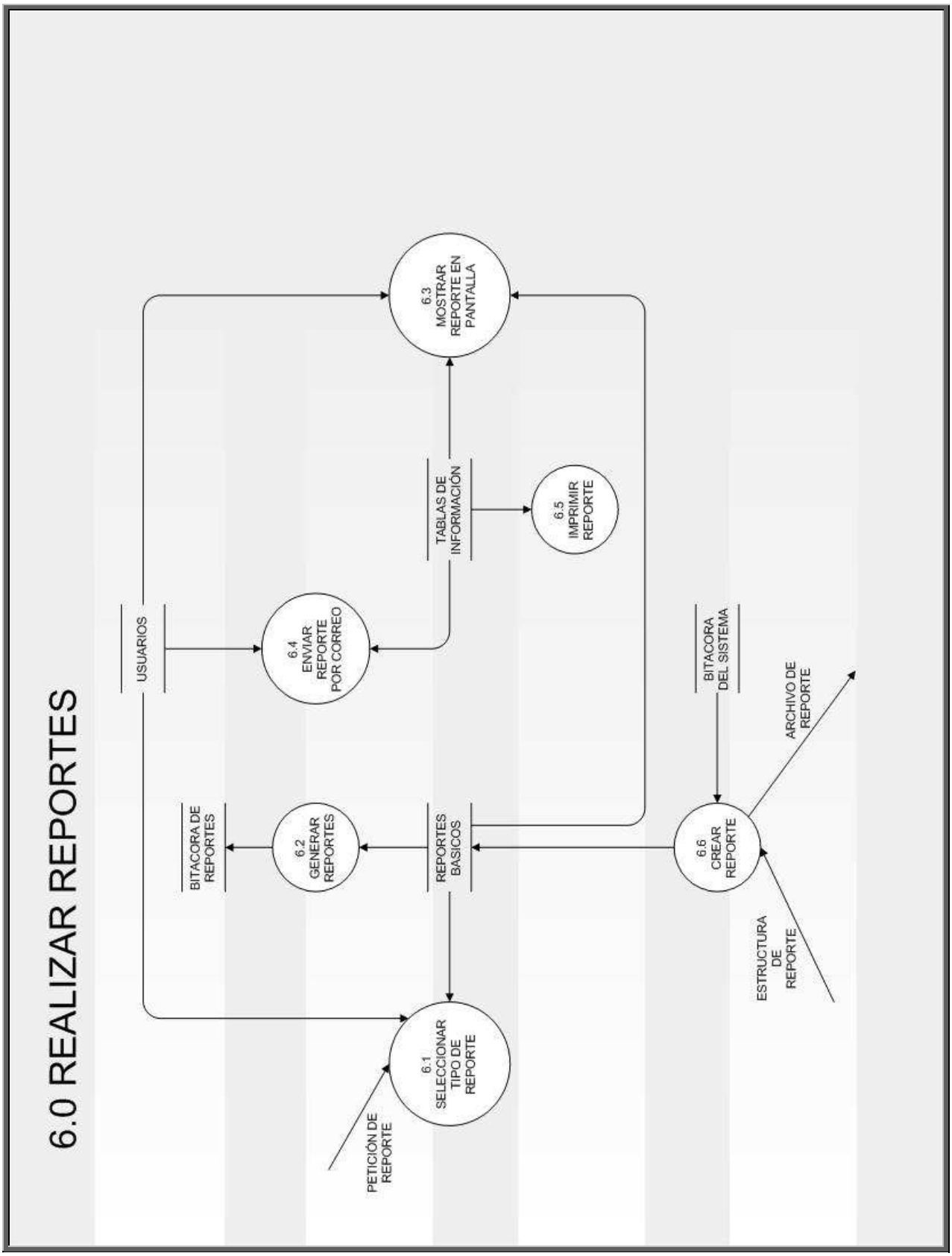


Fig. 3.11 Diagrama Flujo de datos nivel 2.

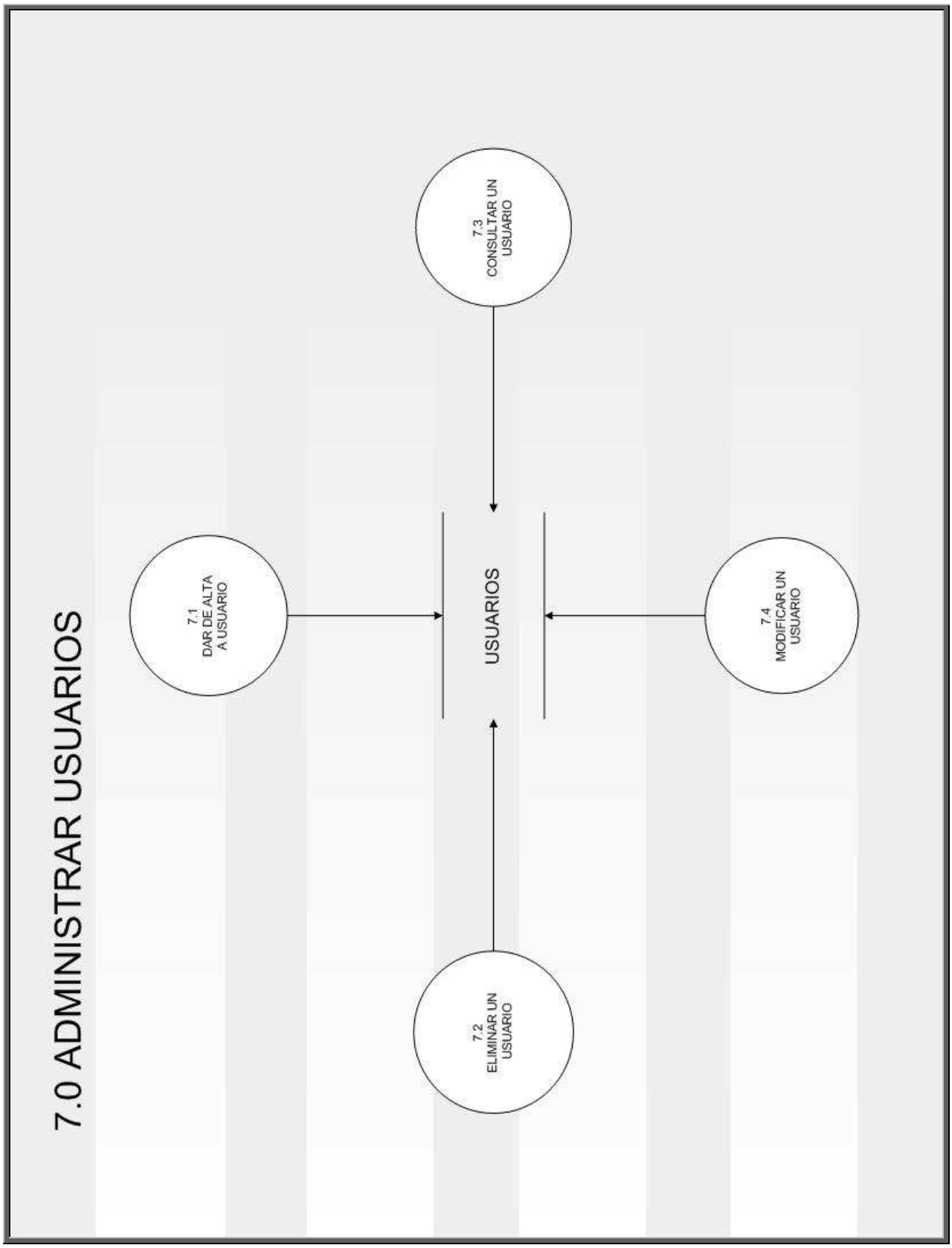


Fig. 3.12 Diagrama Flujo de datos nivel 2.

3.2 Diagrama Entidad-Relación.

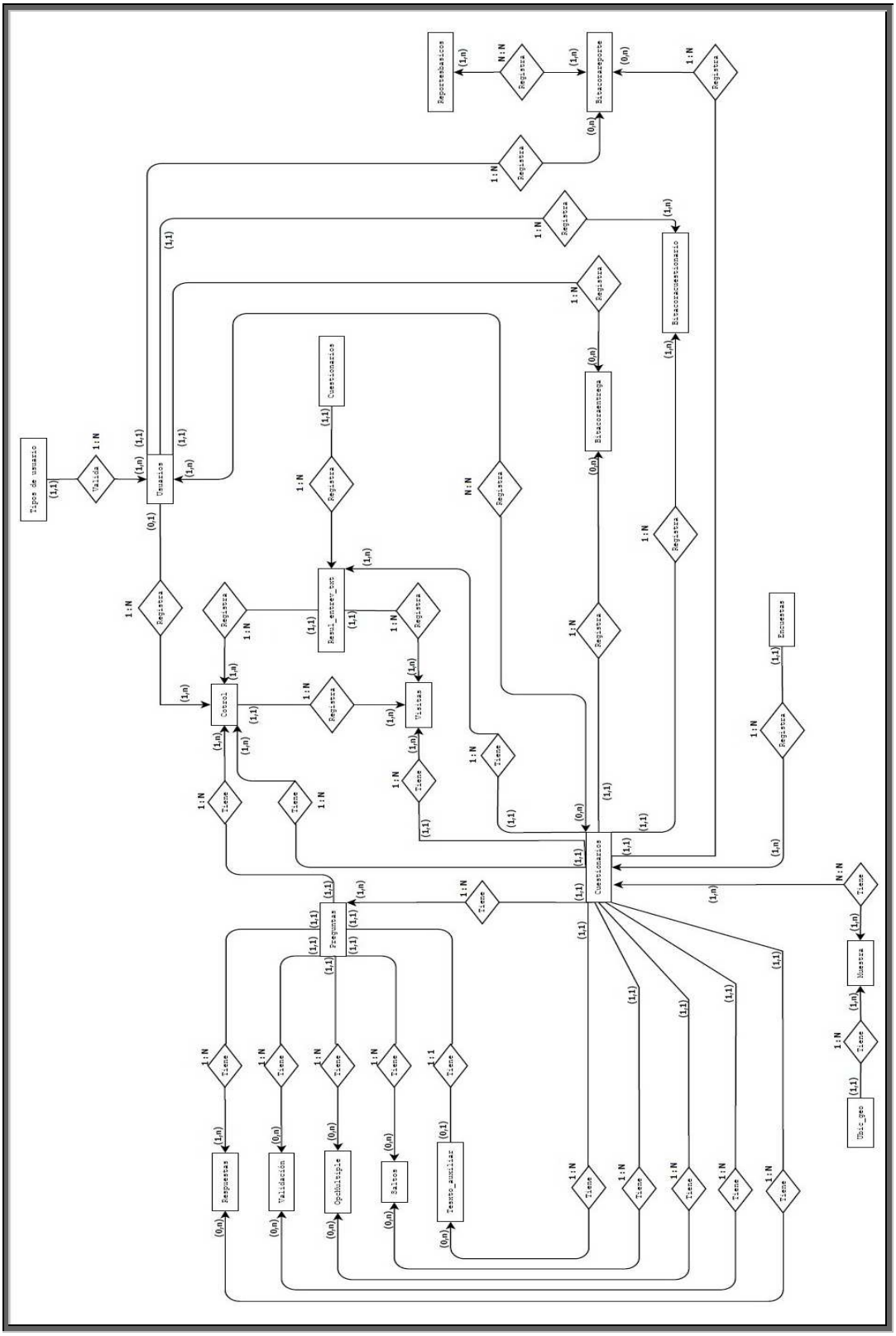


Fig. 3.13 Diagrama E-R.

3.3 Diagrama Relacional.

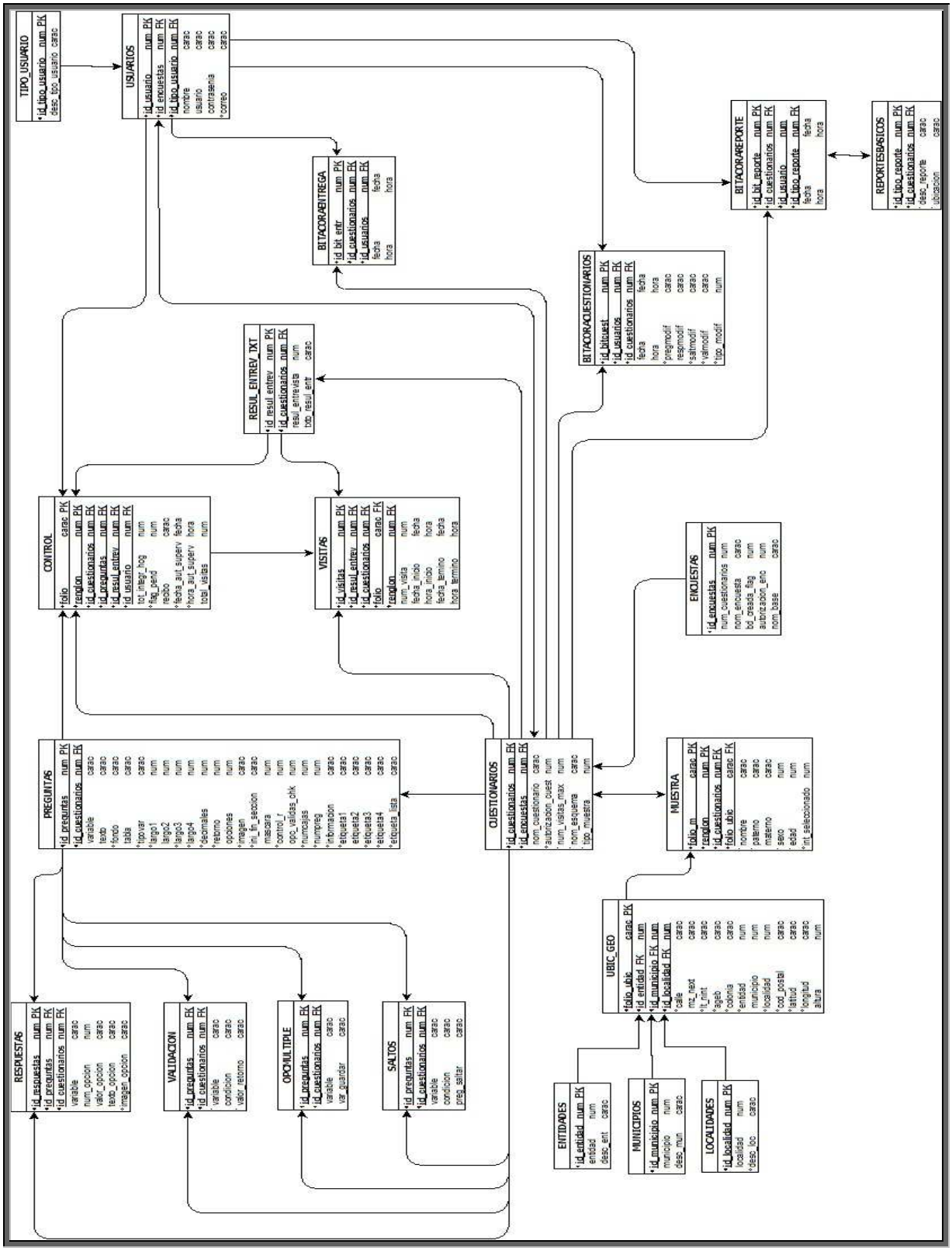
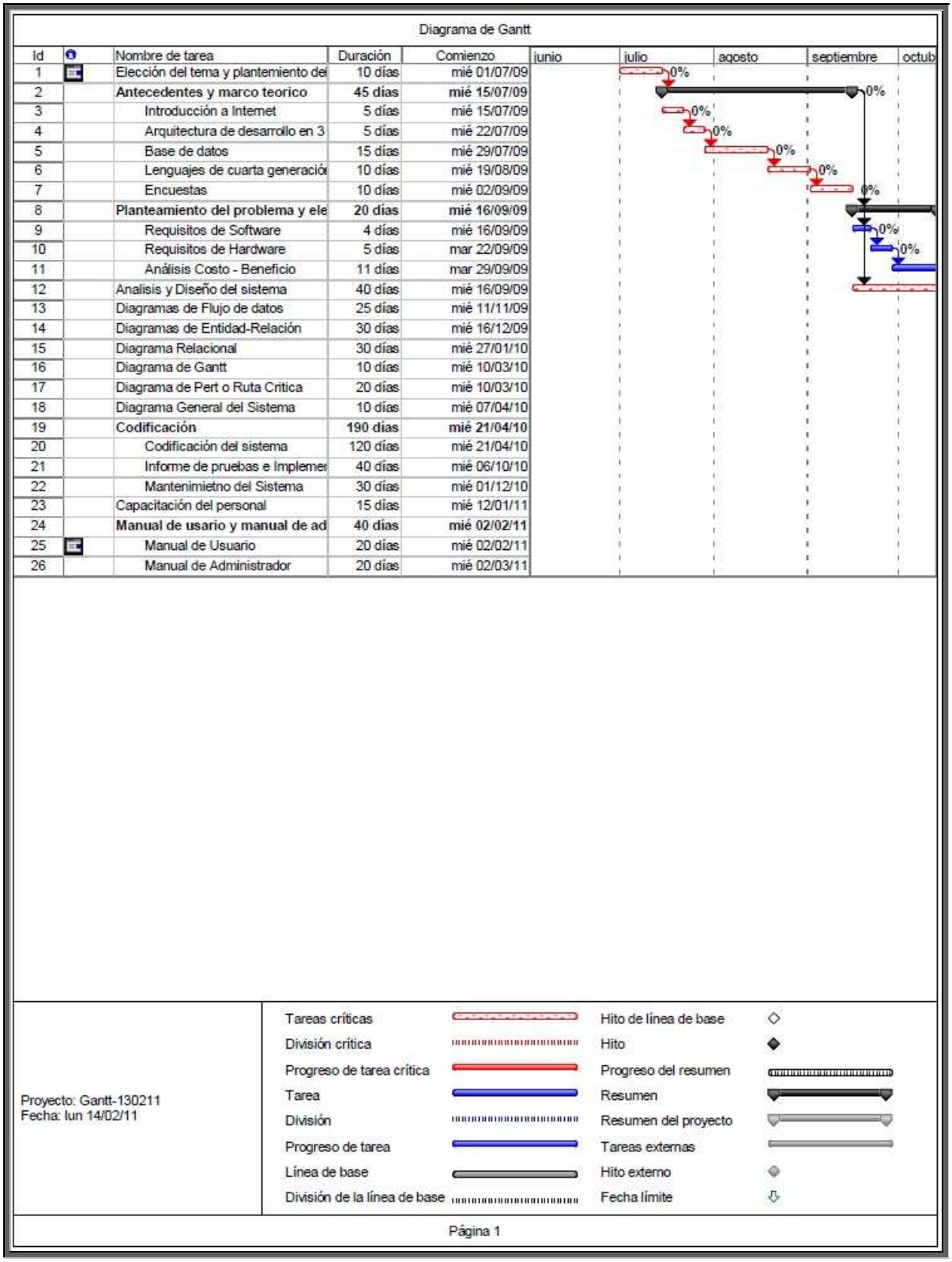


Fig. 3.14 Diagrama Relacional.

3.4 Diagrama de Gantt



Id	Nombre de tarea	Duración	Comienzo
1	Elección del tema y planteamiento del	10 días	mié 01/07/09
2	**Antecedentes y marco teorico**	**45 días**	**mié 15/07/09**
3	Introducción a Internet	5 días	mié 15/07/09
4	Arquitectura de desarrollo en 3	5 días	mié 22/07/09
5	Base de datos	15 días	mié 29/07/09
6	Lenguajes de cuarta generació	10 días	mié 19/08/09
7	Encuestas	10 días	mié 02/09/09
8	**Planteamiento del problema y ele**	**20 días**	**mié 16/09/09**
9	Requisitos de Software	4 días	mié 16/09/09
10	Requisitos de Hardware	5 días	mar 22/09/09
11	Análisis Costo - Beneficio	11 días	mar 29/09/09
12	Analisis y Diseño del sistema	40 días	mié 16/09/09
13	Diagramas de Flujo de datos	25 días	mié 11/11/09
14	Diagramas de Entidad-Relación	30 días	mié 16/12/09
15	Diagrama Relacional	30 días	mié 27/01/10
16	Diagrama de Gantt	10 días	mié 10/03/10
17	Diagrama de Pert o Ruta Critica	20 días	mié 10/03/10
18	Diagrama General del Sistema	10 días	mié 07/04/10
19	**Codificación**	**190 días**	**mié 21/04/10**
20	Codificación del sistema	120 días	mié 21/04/10
21	Informe de pruebas e Implemer	40 días	mié 06/10/10
22	Mantenimietno del Sistema	30 días	mié 01/12/10
23	Capacitación del personal	15 días	mié 12/01/11
24	**Manual de usario y manual de ad**	**40 días**	**mié 02/02/11**
25	Manual de Usuario	20 días	mié 02/02/11
26	Manual de Administrador	20 días	mié 02/03/11

Fig. 3.15 Diagrama de Gantt.

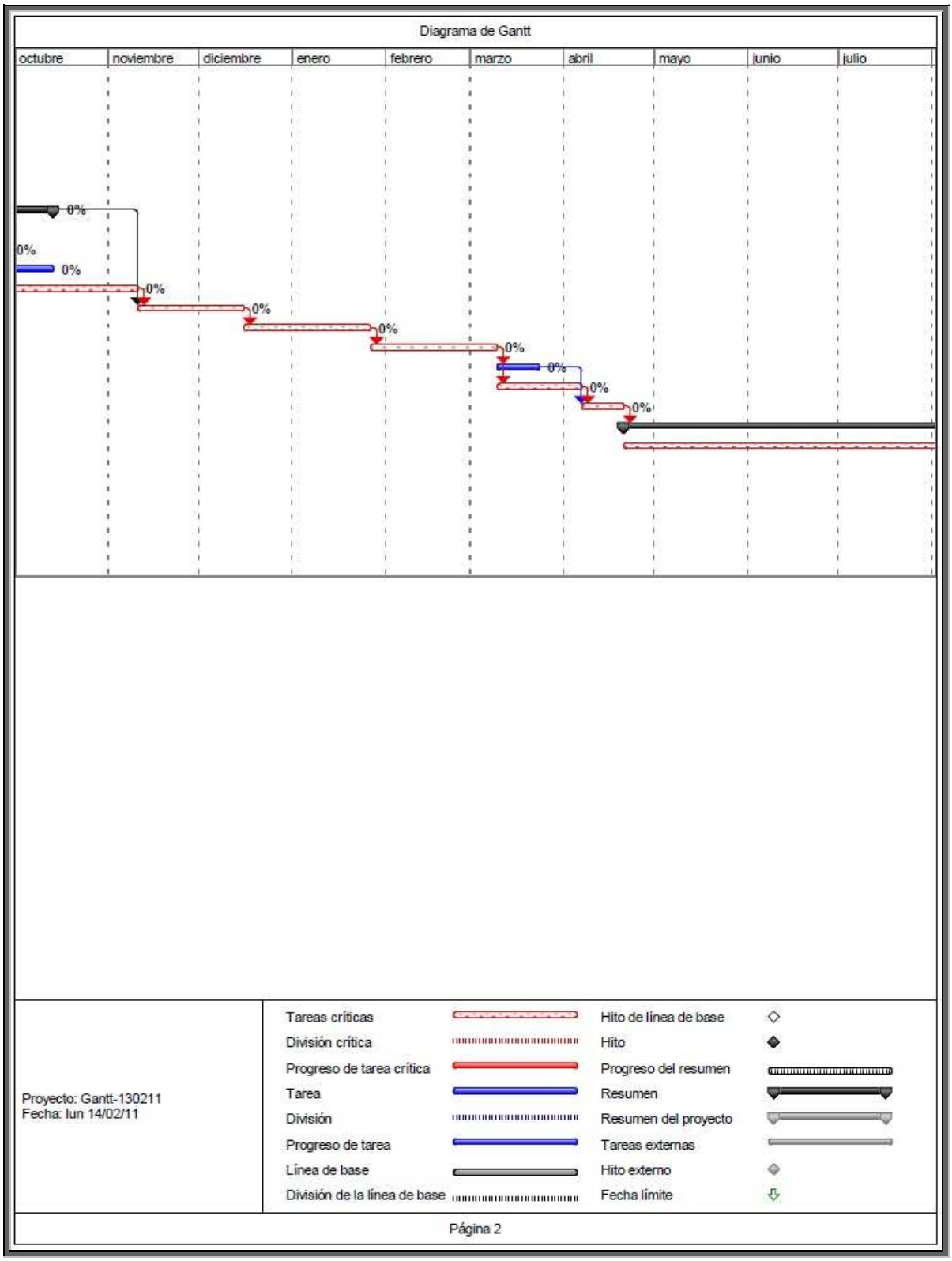


Fig. 3.16 Diagrama de Gantt.

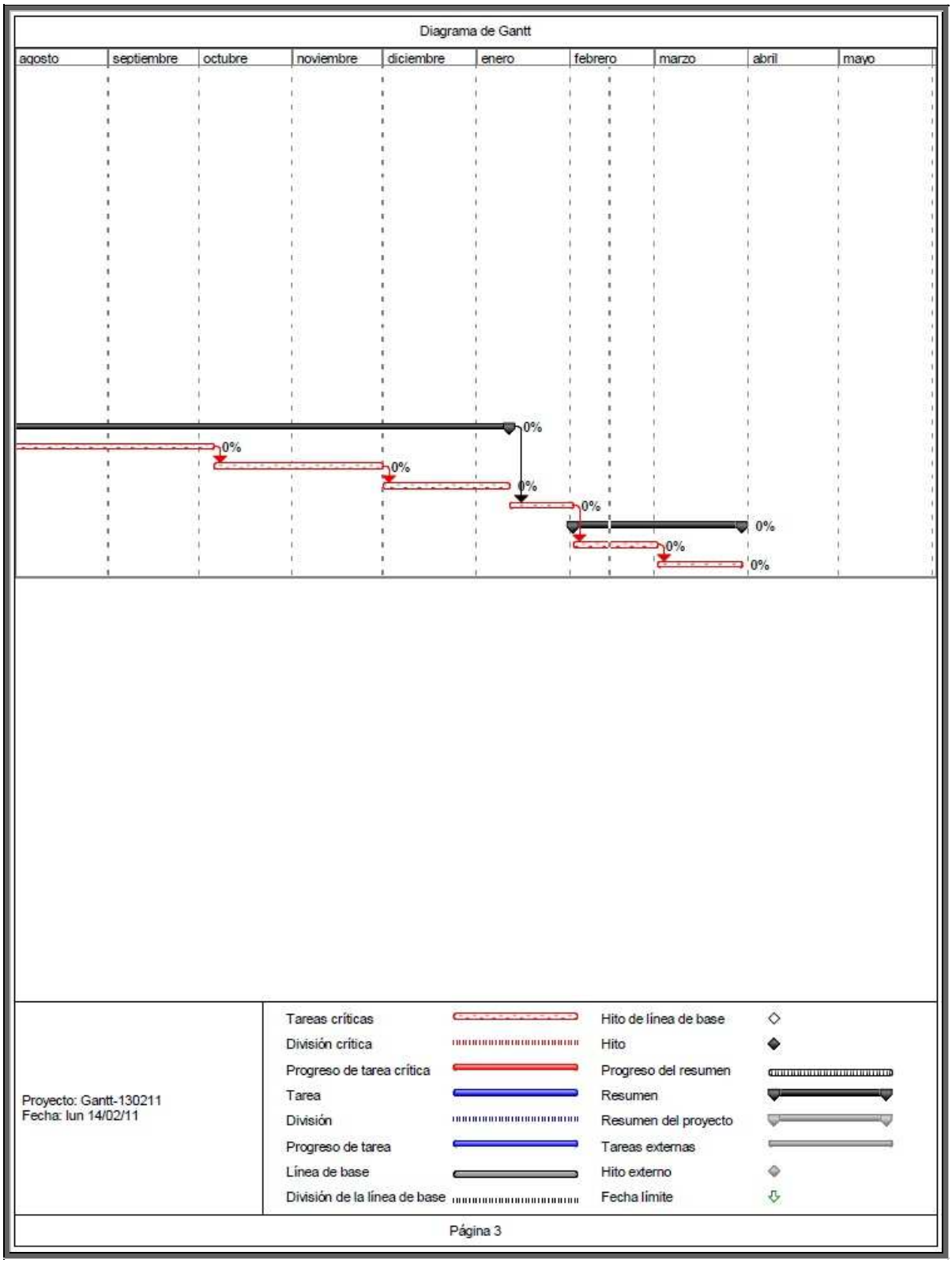


Fig. 3.17 Diagrama de Gantt.

3.5 Diagrama de Ruta Crítica.

Utilizaremos el método de la Ruta Crítica para poder planear, organizar y definir un tiempo aproximado en el desarrollo de esta investigación, así mismo poder determinar en qué punto o tiempo donde puede haber conflicto, es decir, si pueden llegar a estar fuera de tiempo. Así mismo nos podrá decir en qué punto tendemos algo de holgura para poder terminar a tiempo los procesos. Así pues iniciaremos con este análisis.

El primer paso para comenzar a desarrollar la ruta crítica es tener una relación de actividades que forman procesos interrelacionados en un proyecto. Es conveniente numerar estos procesos o etapas, también pueden denominarse en clave.

Lista de Actividades	
A	Antecedentes y marco teórico
B	Planteamiento del problema
C	Análisis y diseño del sistema
D-1	Diagramas de flujo de datos
D-2	Diagramas de Entidad - Relación
D-3	Diagrama Relacional
D-4	Diagrama de Gantt
D-5	Diagrama Ruta Crítica
E-1	Diagrama General del Sistema
E-2	Codificación
E-3	Capacitación del personal
F	Manual de Usuario

Fig. 3.18 Lista de Actividades.

A continuación crearemos la matriz de secuencia, esta se realiza para tener un listado de actividades a realizar e identificar qué actividad es la que sigue después de haber concluido otra.

MATRIZ DE SECUENCIAS	
Actividad	**Secuencias**
N	A
A	B, C
B	D-1
C	D-1
D-1	D2
D-2	D-3
D-3	D-4
D-4	D-5
D-5	E-1
E-1	E-2
E-2	E-3
E-3	F
F	-

Fig. 3.19 Matriz de secuencias.

También usaremos la matriz de tiempos, la cual se realiza con tiempos estimados por actividad, los cuales nos ayudarán a construir nuestra matriz de información.

MATRIZ DE TIEMPOS		
ACTIVIDAD		DURACIÓN (DÍAS)
A		45
B		20
C		40
D-1		25
D-2		30
D-3		30
D-4		10
D-5		20
E-1		10
E-2		190
E-3		15
F		40
		385

Fig. 3.20 Matriz de tiempos.

La matriz de información va a contener la matriz de secuencias y la matriz de tiempos con lo cual formaremos nuestros diagramas de la ruta crítica.

MATRIZ DE INFORMACIÓN				
Actividad		**Secuencias**		**Duración**
N		A		0
A		B, C		45
B		D-1		20
C		D-1		40
D-1		D-2		25
D-2		D-3		30
D-3		D-4 D-5		30
D-4		E-1		10
D-5		E-1		20
E-1		E-2		10
E-2		E-3		190
E-3		F		15
F		-		40

Fig. 3.21 Matriz de información.

El siguiente diagrama nos representará la forma gráfica de las matrices de tiempo y de actividades, es el arreglo lógico donde mostraremos la secuencia de actividades y en las cuales les asignamos un valor a los nodos que representan los eventos y las flechas que representan las actividades.

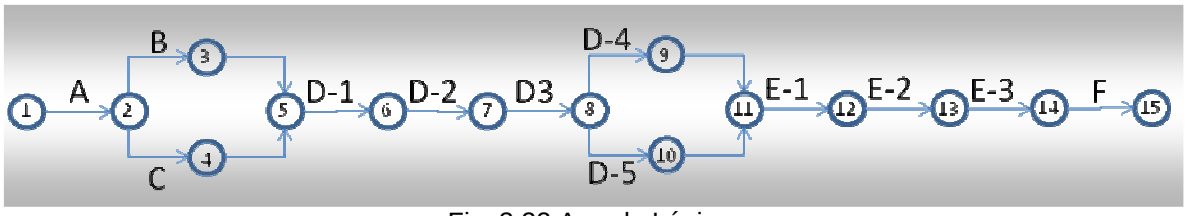


Fig. 3.22 Arreglo Lógico.

A continuación colocaremos los valores de tiempo de cada actividad para poder observar lo tiempos próximos de inicio (TPI).

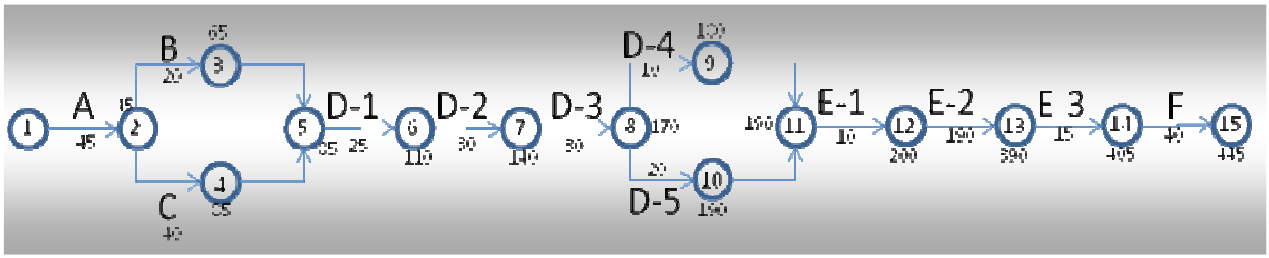


Fig. 3.23 Asignación de duraciones y cálculo de TPI.

Tanto la Matriz de Secuencias como la Matriz de Tiempos se reúnen en una sola llamada Matriz de información, que sirve para construir la Red Medida.

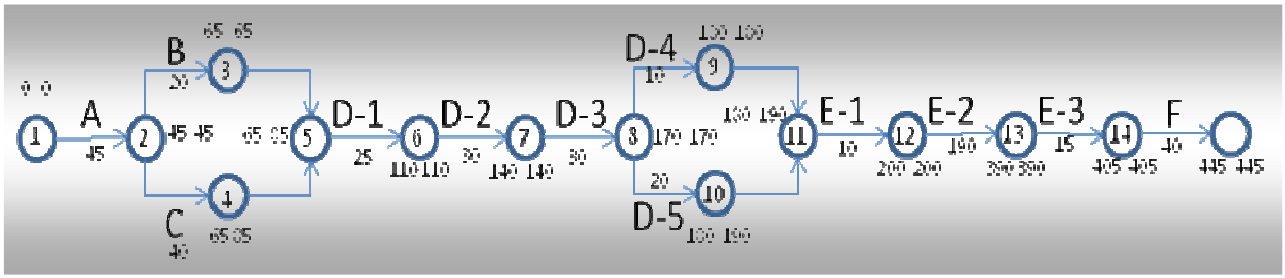


Fig. 3.24 Red Medida.

Con la información obtenida se calculan las holguras y se procede a formar la matriz de elasticidad.

Donde:

Holgura total (HT), esto es el exceso de tiempo disponible con respecto a la duración de una actividad.

Holgura libre (HL). Puede definirse como aquel en que podemos posponer la realización de una actividad sin afectar las fechas subsecuentes.

Holgura independiente (HI), es equivalente al tiempo que podemos posponer la ejecución de una actividad determinada, sin afectar las fechas próximas de las actividades subsecuentes, pero suponiendo que las anteriores se completaron en sus fechas remotas de realización.

MATRIZ DE ELASTICIDAD										
Actividad	**Duración**	**I**	**J**	**TPI**	**TRI**	**TPT**	**TRT**	**HT**	**HL**	**HI**
A	45	1	2	0	0	45	45	0	0	
B	20	2	3	45	45	65	65	0	-20	-20
C	40	2	4	45	25	85	65	-20	0	-20
D-1	25	5	6	85	85	110	110	0	0	20
D-2	30	6	7	110	110	140	140	0	0	0
D-3	30	7	8	140	140	170	170	0	0	0
D-4	10	8	9	170	170	180	180	0	-10	-10
D-5	20	8	10	170	160	190	180	-10	0	-10
E-1	10	11	12	190	190	200	200	0	0	10
E-2	190	12	13	200	200	390	390	0	-10	-10
E-3	15	13	14	380	390	395	405	10	0	-10
F	40	14	15	395	405	435	445	10	-435	-445

Fig. 3.25 Matriz de Elasticidad.

Finalmente con esta matriz de elasticidad podemos determinar qué actividades son críticas, las cuales cumplen con todas las condiciones siguientes:

1. TPI = TRI
2. TPI = TRT
3. TPT - TRI = Y
4. HT = HL = HI = 0

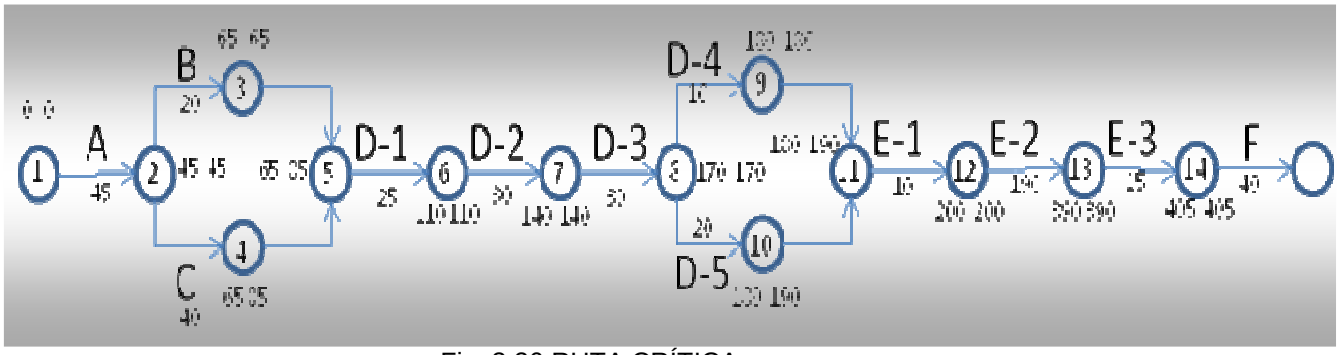


Fig. 3.26 RUTA CRÍTICA.

Donde:

I - Es el nodo de inicio de la actividad
J – Es el nodo de fin de la actividad
TPT - Tiempo Próximo de Terminación
TRI - Tiempo Remoto de Iniciación
TPI - Tiempo Próximo de Iniciación
TRT Tiempo Remoto de Terminación
HT Holgura Total
HL Holgura Libre
HI Holgura Independiente

CAPÍTULO 4: EJEMPLO DE DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

4.1 Diagrama General del Sistema

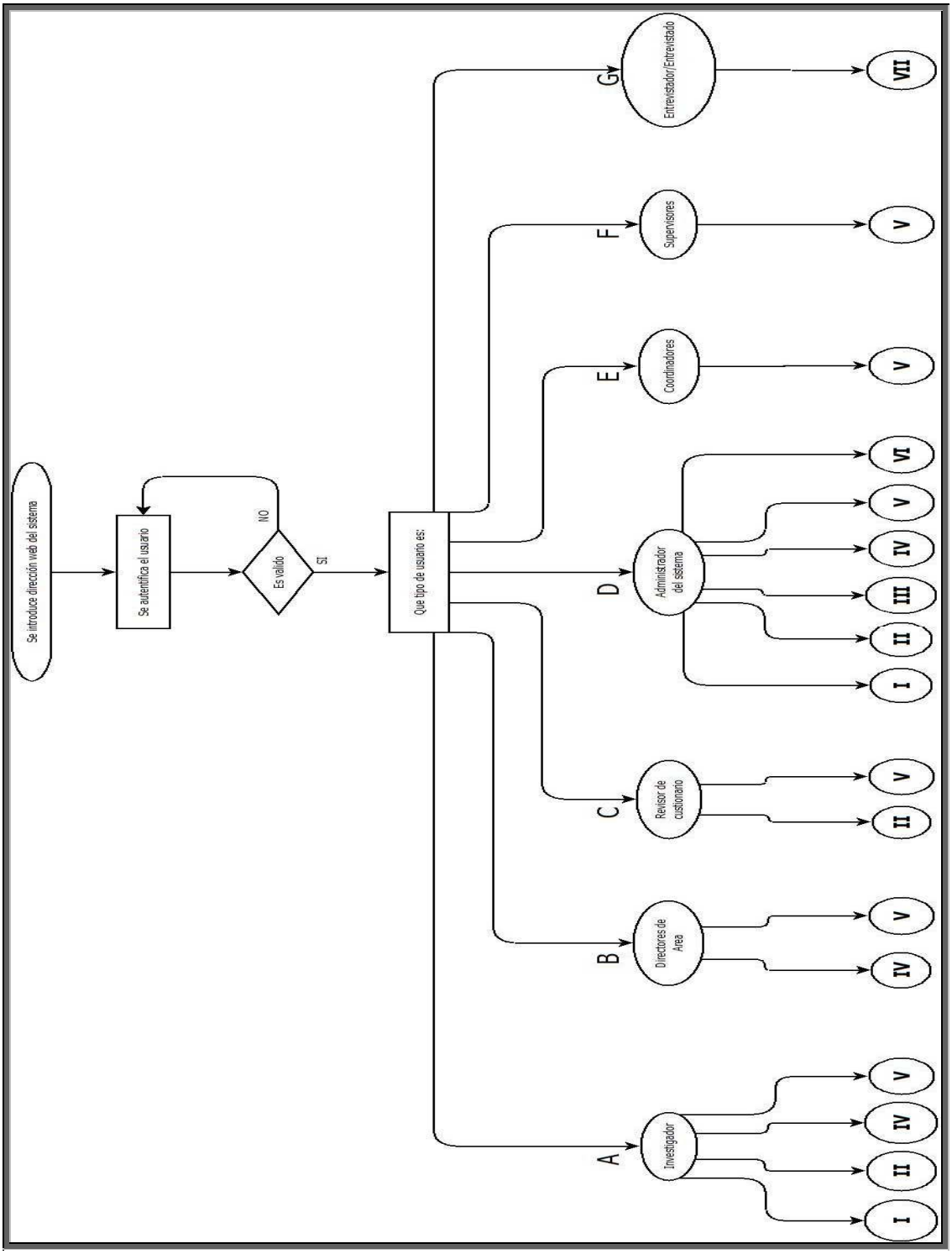


Fig. 4.1 Diagrama General del Sistema.

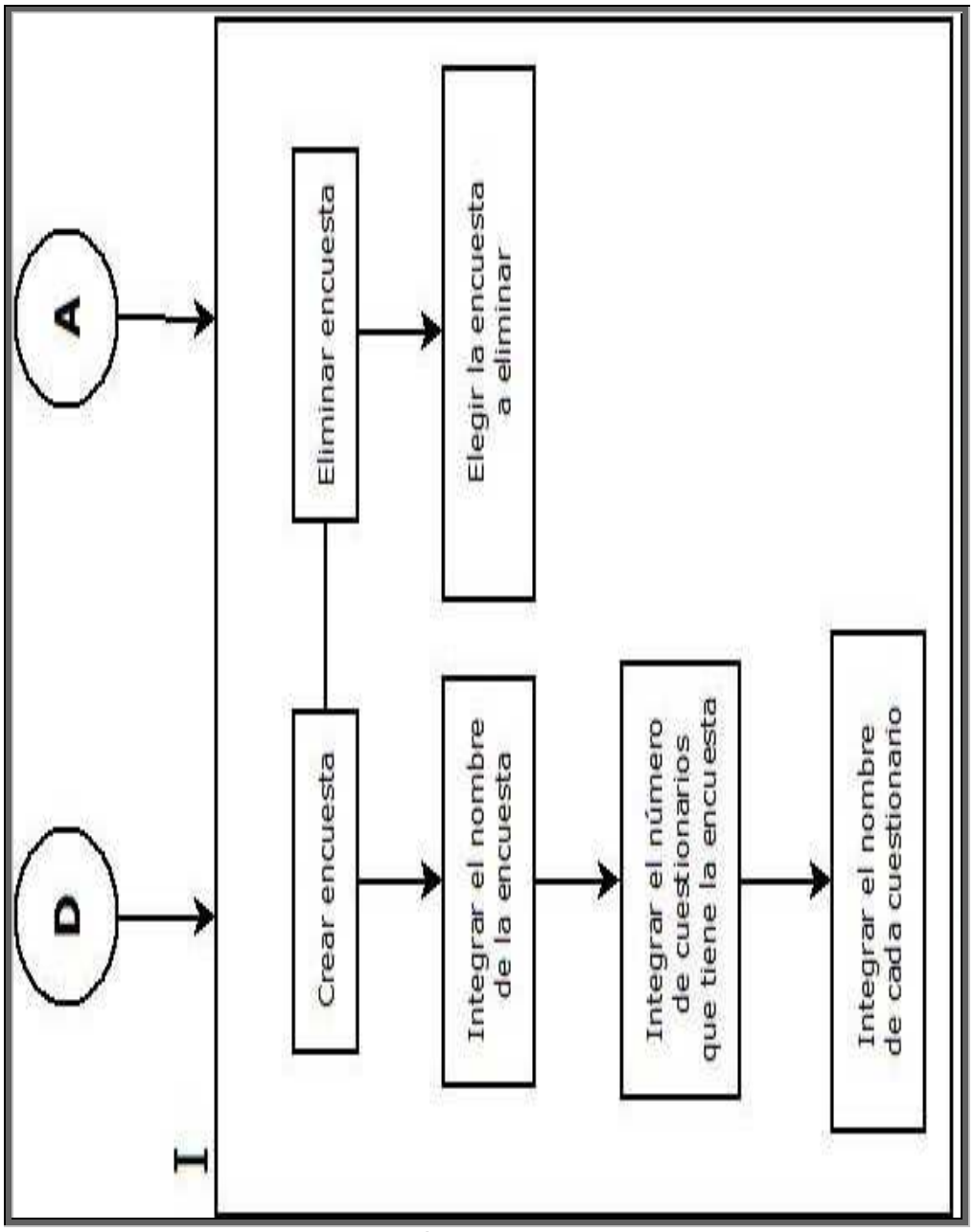


Fig. 4.2 Proceso de creación de base de datos para la encuesta.

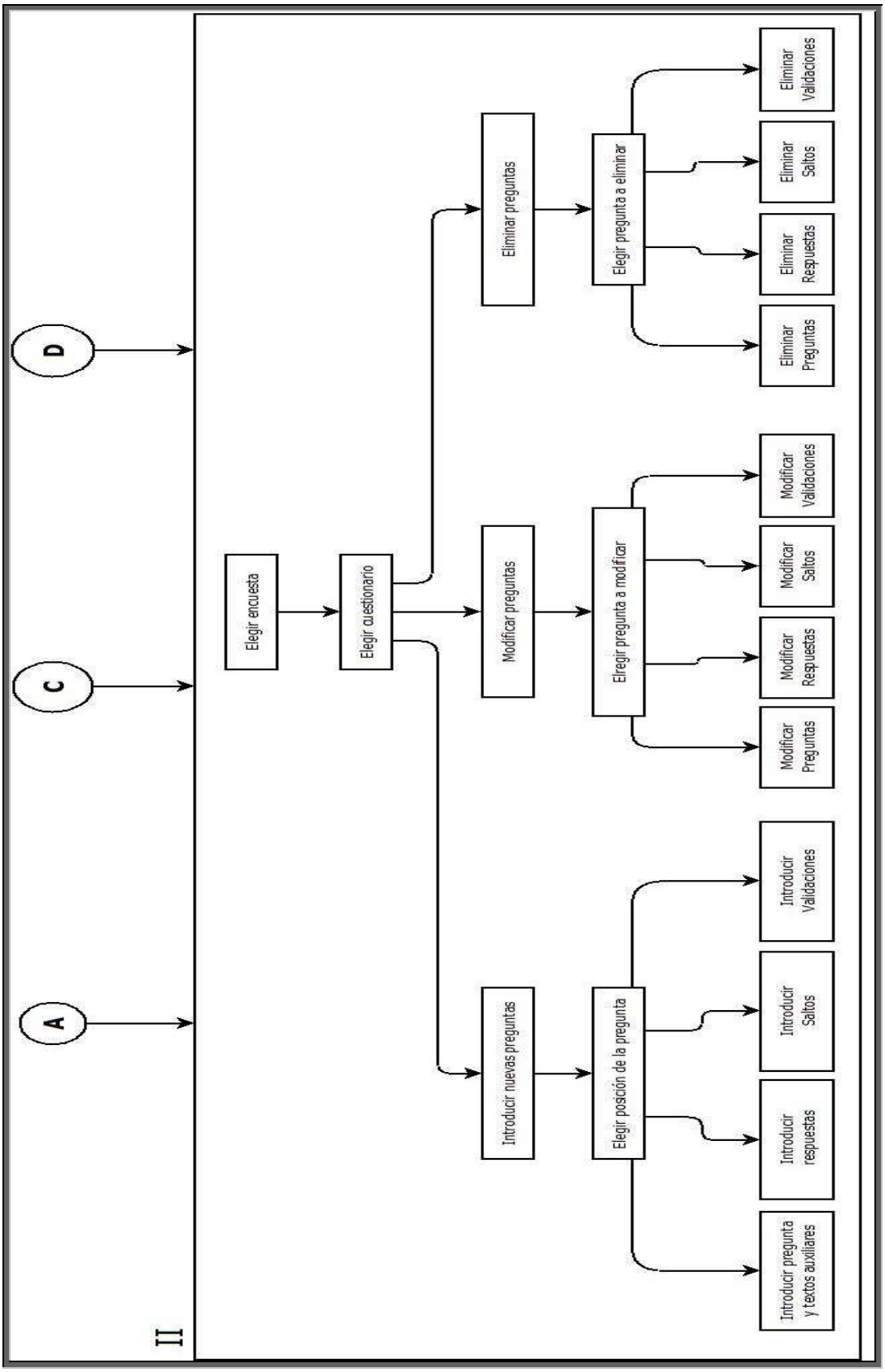


Fig. 4.3 Diagrama del proceso de integración de cuestionario.

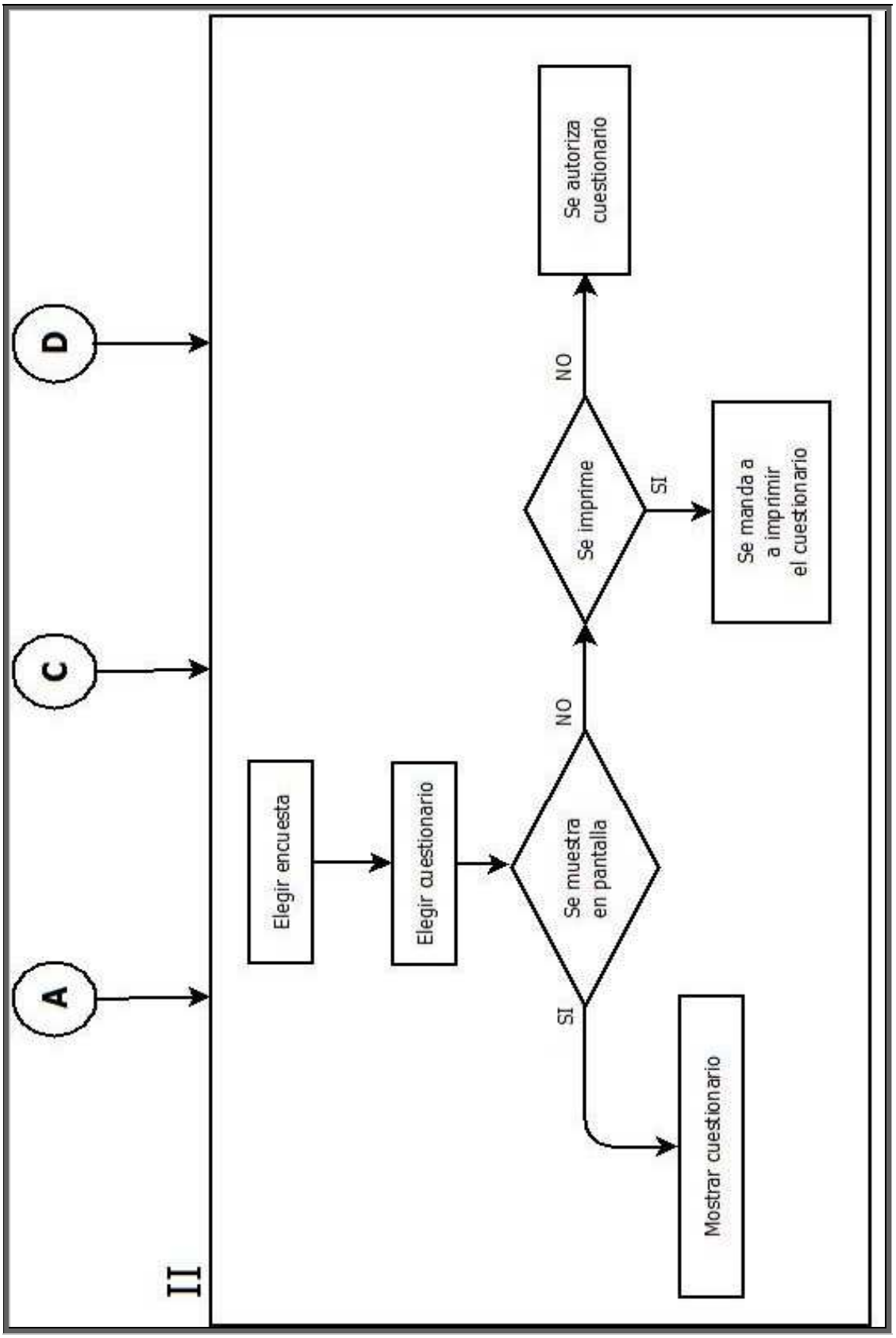


Fig. 4.4 Diagrama del proceso de revisión de cuestionario.

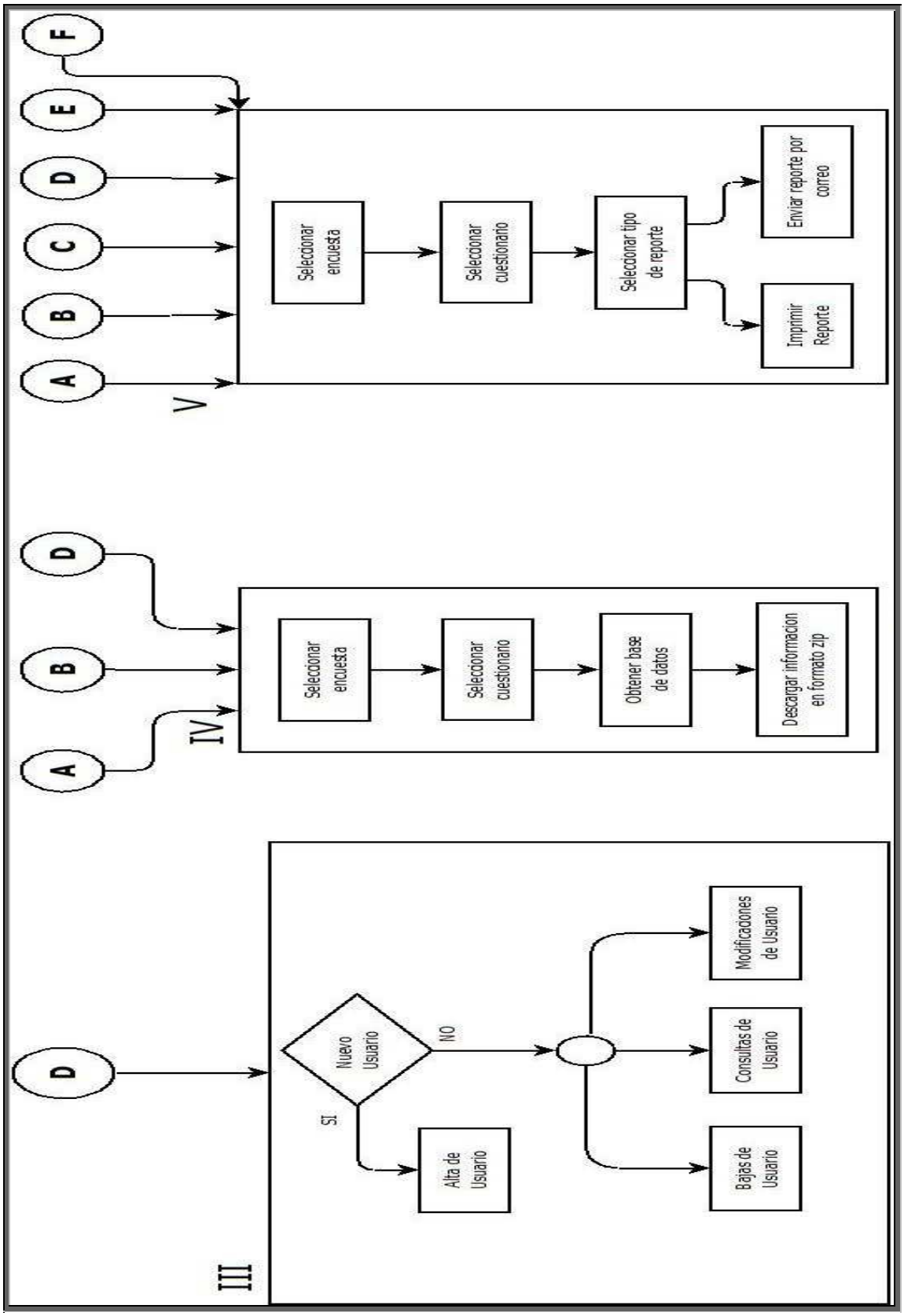


Fig. 4.5 Diagrama de administración de usuarios, entrega de información y realización de reportes.

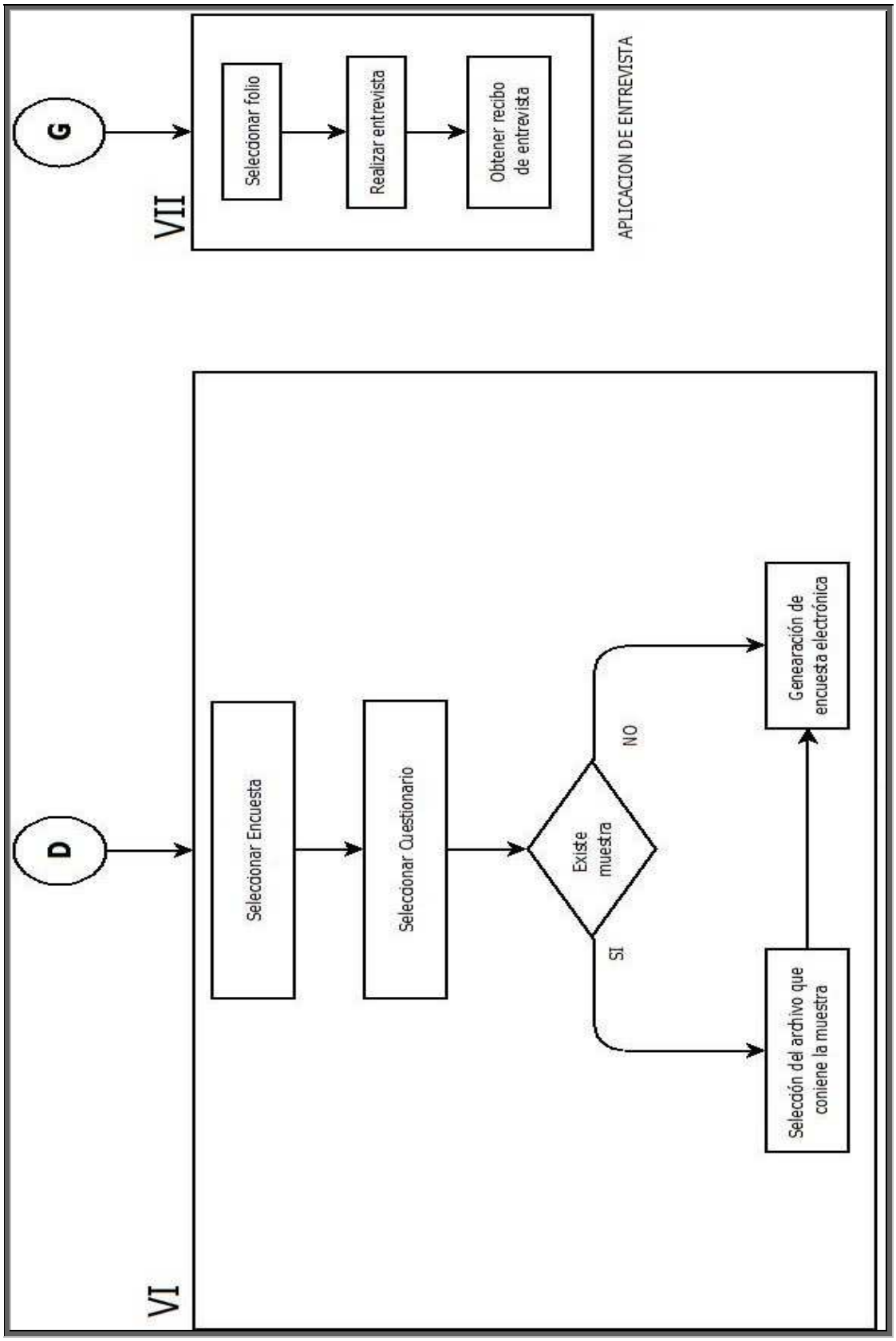


Fig. 4.6 Diagrama de la generación de encuesta electrónica y aplicación de entrevista.

4.2 Código

A continuación se describen algunos de los procesos mas importantes dentro del ejemplo de desarrollo, mostraremos parte del código al dar click al botón de **Autorizar cuestionariom**, éste da paso a la acción de generar la encuesta. Se crean DataSet para contener los datos que requerimos con los datos que se encuentran en la Base de Datos previa captura por parte de los que integrarón el cuestionario, se carga la encuesta y sus cuestionarios en el DataSet, entiendase por Individual para las preguntas que se hacen a cada uno de los miembros del Hogar y Hogar para las preguntas que se refieren a la casa.

```
Protected Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim n, i, j, k As Integer                    'declaraciones de contadores y variables auxiliares
    sql = "update cuestionarios set autorizacion_cuest=1 where id_cuestionarios=" &
          Me.LCuestionario.Text                  'cambio de bandera de cuestionarios
    x.Connection = cn                            'se asigna la conecion a utilizar
    x.CommandText = sql                          ' se agrega la instruccion sql que se construyo anteriormente
    Try                                           'atrapa excepciones
       x.ExecuteNonQuery()                       'ejecuta instruccion sql
    Catch ex As Exception                        'si hay una exepcion hacer:
       MsgBox(ex.Message)                        'mensaje de error que provoco la excepcion
    End Try                                      'fin atrapa excepciones
        sql = "Select tipo_muestra,id_muestra from cuestionarios where id_cuestionarios=" &
              Me.LCuestionario.Text                  'sql para datos del cuetionario
    Dim adapter As New SqlClient.SqlDataAdapter(sql, cn)        'crear el adapter con la instuccion sql y conexion
    adapter.Fill(tabla)                          'llenar el dataset con los resultados de la consulta
```

La siguiente parte del código hace que seleccionamos todas las variables que están catalogadas como no individual para generar la tabla, y llenamos un DataSet con estos datos que son la estructura de la tabla donde guardaremos la información de la encuesta y se crea la tabla para integrar posteriormente la variables a esta tabla.

```
 If tabla.Tables(0).Rows(0).Item(0) = 1 Then         'si individual =1 entonces
    'Seleccionar preguntas individuales
        sql = "Select variable, tipovar, num_caja,largo1,largo2,largo3,largo4, opc_validas_chk,control_r, id_preguntas
              from preguntas where " & _ " id_cuestionario=" & Me.LCuestionario.Text & " and  tipo=0"
        Dim adap As New SqlClient.SqlDataAdapter(sql, cn) 'Se crea el data adpater y se llena el data set con la
instruccion sql
        Dim dt As New DataSet               'declaracion del data set
        adap.Fill(dt)                       'llenar el dataset con los datos
```

A continuación se muestra como creamos la tabla para la parte Individual con los datos de encuesta, cuestionario y el data set creado anteriormente con los datos que se cargaron de preguntas.

```
sql = " create table datos_" & Me.LCuestionario.Text & "_I" & "(Folio nvarchar(8),"   'instruccion sql
n = dt.Tables(0).Rows.Count              'numero de registros encontrados
For i = 0 To n - 1                       'de cero hasta registros encontrados -1
  If dt.Tables(0).Rows(i).Item(1) = "N" Then      'si el tipo de variable es numerico
    sql = sql & dt.Tables(0).Rows(i).Item(0) & " nvarchar(" & dt.Tables(0).Rows(i).Item(3) & "),"   'concatena
  Else
  Select Case (dt.Tables(0).Rows(i).Item(8))  'casos tipo de control
     Case 1                              'caso de botones
       sql = sql & dt.Tables(0).Rows(i).Item(0) & " nvarchar(5),"  'concatenar
     Case 2                              'check box normal
       Dim r, m As Integer               'declaracion de registros y contador
       sql2 = "Select id_respuesta from respuestas where id_pregunta=" & dt.Tables(0).Rows(i).Item(9) 'sql
       Dim adap2 As New SqlClient.SqlDataAdapter(sql2, cn)  'Se crea el data adpater y se llena el data set con la
instruccion sql
       Dim dt2 As New DataSet            'segunda dataset para los datos
        adap2.Fill(dt2)                  'llenar el segundo dataset
        r = dt2.Tables(0).Rows.Count      'registros de respuestas
        For m = 1 To r                   'de i a registros encontrados
          sql = sql & dt.Tables(0).Rows(i).Item(0) & Chr(m + 96) & " nvarchar(5),"     'concatena
        Next                             'fin de hasta
    Case 3                               'caso caja de texto
      j = dt.Tables(0).Rows(i).Item(2)   'numero de cajas
      If j = 1 Then                      'si esuna caja
        sql = sql & dt.Tables(0).Rows(i).Item(0) & " nvarchar(" & dt.Tables(0).Rows(i).Item(3) & "),"  'conc
   Exit Select
   End If
   For k = 1 To j                        'asignacion de las letras cuando hay mas de 1 caja
     If k = 1 Then                       'si cajas =1
       sql = sql & dt.Tables(0).Rows(i).Item(0) & "a" & " nvarchar(" & dt.Tables(0).Rows(i).Item(3) & "),"  'concatena
     End If
     If k = 2 Then                       'si cajas =2
       sql = sql & dt.Tables(0).Rows(i).Item(0) & "b" & " nvarchar(" & dt.Tables(0).Rows(i).Item(4) & "),"  'concatena
     End If
     If k = 3 Then                       'si cajas=3
       sql = sql & dt.Tables(0).Rows(i).Item(0) & "c" & " nvarchar(" & dt.Tables(0).Rows(i).Item(5) & "),"   'concatena
     End If
     If k = 4 Then                       'si cajas=4
       sql = sql & dt.Tables(0).Rows(i).Item(0) & "d" & " nvarchar(" & dt.Tables(0).Rows(i).Item(6) & "),"   'concatena
     End If
   Next                                  'fin asinacion
   Case 4                                'caso check binario
     Dim x, y As Integer                 'declaracion de respuestas validas
      y = dt.Tables(0).Rows(i).Item(7)   'respuestas validas
      For x = 1 To y                     'de uno hasta respuestas validas
        sql = sql & dt.Tables(0).Rows(i).Item(0) & Chr(x + 96) & " nvarchar(5),"  'concatena
      Next
    End Select                           'fin casos
    End If
Next                                     'fin desde
sql = sql.Substring(1, (sql.Length - 2))  'quita espacios
sql = sql & ")"                          'termina la instruccion sql
x.Connection = cn                        'se asigna la conexión a utilizar
x.CommandText = sql                      'se agrega la instruccion sql que se construyo anteriormente
Try                                      'Atrapa excepciones
   x.ExecuteNonQuery()                   'Ejecuta  instruccion crea tabla
Catch ex As Exception                    'hay excepcion
   MsgBox(ex.Message)                    'mensaje
End Try                                  'fin atrapa excepciones
```

Se hace la selección de todas las preguntas de la encuesta y cuestionario que se han seleccionado, y se guardan en un DataSet para tener la información disponible para la parte Hogar.

```
 'Seleccionar preguntas todos
sql = "Select variable, tipovar, num_caja,largo1,largo2,largo3,largo4, opc_validas_chk,control_r, id_preguntas from
       preguntas where " & _" id_cuestionario=" & Me.LCuestionario.Text & " and  tipo=1"
Dim adap3 As New SqlClient.SqlDataAdapter(sql, cn)  'Se crea el data adpater y se llena el data set con la
instruccion sql
Dim dt3 As New DataSet                  ' data set
 adap3.Fill(dt3)                        'llenar tabla con datos
 sql = " create table datos_" & Me.LCuestionario.Text & "_T " & "( Folio nvarchar(8),"   'sql
 n = dt3.Tables(0).Rows.Count           'registros
 For i = 0 To n - 1                     'de 0 hasta registros-1
    If dt3.Tables(0).Rows(i).Item(1) = "N" Then     'si el tipo de variable es numerico
       sql = sql & dt3.Tables(0).Rows(i).Item(0) & " nvarchar(" & dt3.Tables(0).Rows(i).Item(3) & "),"   'concatena
    Else
      Select Case (dt3.Tables(0).Rows(i).Item(8))                   'casos tipo de control
    Case 1                                                          'caso botones
      sql = sql & dt3.Tables(0).Rows(i).Item(0) & " nvarchar(5),"     'concatena
    Case 2                                                          'caso check box normal
      Dim r, m As Integer                                           'declaracion de variables para contar
```

Se seleccionan las respuestas de acuerdo a la variable para hacer el DataSet y poder integrar la variables a la base.

```
sql2 = "Select id_respuesta from respuestas where id_pregunta=" & dt3.Tables(0).Rows(i).Item(9) 'de la tabla de
respuestas,  traer el numero de respuestas para generar los campos
Dim adap2 As New SqlClient.SqlDataAdapter(sql2, cn)   'Se crea el data adpater y se llena el data set con la
instruccion sql
Dim dt2 As New DataSet                      'se crea un segundo repositorio de datos
adap2.Fill(dt2)                             'se llena la segunda tabla con datos respuestas
r = dt2.Tables(0).Rows.Count                'cuantas respuestas
For m = 1 To r                              'de uno hasta respuestas
   sql = sql & dt3.Tables(0).Rows(i).Item(0) & Chr(m + 96) & " nvarchar(5),"   'concatena
Next
Case 3                                      'caso caja de texto
   j = dt3.Tables(0).Rows(i).Item(2)        'cuantas cajas
   For k = 1 To j                           'desde asignacion de las letras cuando hay mas de 1 caja
      If k = 1 Then                         'si es una caja
        sql = sql & dt3.Tables(0).Rows(i).Item(0) & "a" & " nvarchar(" & dt3.Tables(0).Rows(i).Item(3) & "),"   'concatena
      End If
      If k = 2 Then                         'si son dos cajas
        sql = sql & dt3.Tables(0).Rows(i).Item(0) & "b" & " nvarchar(" & dt3.Tables(0).Rows(i).Item(4) & "),"   'concatena
       End If
       If k = 3 Then                        'si son tres cajas
        sql = sql & dt3.Tables(0).Rows(i).Item(0) & "c" & " nvarchar(" & dt3.Tables(0).Rows(i).Item(5) & "),"   'concatena
       End If
       If k = 4 Then                        'si son cuatro cajas
        sql = sql & dt3.Tables(0).Rows(i).Item(0) & "d" & " nvarchar(" & dt3.Tables(0).Rows(i).Item(6) & "),"   'concatena
       End If
   Next                                     'fin desde
   Case 4                                   'caso check binario
      Dim x, y As Integer                   'declaracion de variables para contar
      y = dt3.Tables(0).Rows(i).Item(7)     'cuantas respuestas
      For x = 1 To y                        'desde 1 hasta respuestas
         sql = sql & dt3.Tables(0).Rows(i).Item(0) & Chr(x + 96) & " nvarchar(5),"   'concatena
       Next                                 'fin desde
   End Select                               'fin casos
End If
Next                                        'fin desde
sql = sql.Substring(1, (sql.Length - 2))    'quita espacios en blanco
sql = sql & ")"                             'termina la cadena sql
```

```
x.Connection = cn                                   'se asigna la conecion a utilizar
 x.CommandText = sql                                'se agrega la instruccion sql que se construyo anteriormente
Try                                                  'atrapa excepciones
   x.ExecuteNonQuery()                               'ejecuta la instruccion sql
Catch ex As Exception                                'excepcion atrapada
   MsgBox(ex.Message)                                'mensaje de error
End Try                                              'fin atrapa
```

Se actualiza la tabla de preguntas con las tablas donde finalmente quedaron las variables tanto para la parte de Indivudual como para la parte de Hogar.

```
 'actualizando el nombre de la tabla en preguntas
sql = "Update preguntas set tabla='Datos_" & Me.LCuestionario.Text & "_T' where tipo= 1 and id_cuestionario=" &
       Me.LCuestionario.Text
x.Connection = cn                                    'se asigna la conecion a utilizar
x.CommandText = sql                                  'se agrega la instruccion sql que se construyo anteriormente
sql = "Update preguntas set tabla='Datos_" & Me.LCuestionario.Text & "_I' where tipo= 0 and id_cuestionario=" &
       Me.LCuestionario.Text
x.Connection = cn                                    'se asigna la conecion a utilizar
x.CommandText = sql                                  'se agrega la instruccion sql que se construyo anteriormente
Else
'Seleccionar preguntas individuales
sql = "Select variable, tipovar, num_caja,largo1,largo2,largo3,largo4, opc_validas_chk,control_r, id_preguntas from
       preguntas where " & _ " id_cuestionario=" & Me.LCuestionario.Text & " and  tipo=0"
Dim adap As New SqlClient.SqlDataAdapter(sql, cn)  'Se crea el data adpater y se llena el data  set con la instruccion
sql
Dim dt As New DataSet                                'se crea data set
adap.Fill(dt)                                        'se llena la tabla con datos
'instruccion sql
sql = "create table datos_" & Me.LCuestionario.Text & "_I" & "(Folio nvarchar(8),calle nvarchar(40), mz_next
       nvarchar(5)," & _ "lt_nint nvarchar(5), ageb nvarchar(6),colonia nvarchar(30),entidad nvarchar(15),municipio
       nvarchar(30), localidad nvarchar(30),cod_postal nvarchar(5)," & _ "latitud nvarchar(10),longitud nvarchar(10),
       altura nvarchar(10), integrantes nvarchar(3),"
n = dt.Tables(0).Rows.Count                          'cuantas individuales
For i = 0 To n - 1                                   'desde 0 hasta individuales-1
  If dt.Tables(0).Rows(i).Item(1) = "N" Then         'si el campo es numeridco
    sql = sql & dt.Tables(0).Rows(i).Item(0) & " nvarchar(" & dt.Tables(0).Rows(i).Item(3) & "),"  'concatena
  Else
  Select Case (dt.Tables(0).Rows(i).Item(8))         'caso tipo de control
  Case 1                                             'caso botones
    sql = sql & dt.Tables(0).Rows(i).Item(0) & " nvarchar(5),"      'concatena
  Case 2                                             'caso check box normal
    Dim r, m As Integer                              'variables para contar y recorrer datos
    sql2 = "Select id_respuesta from respuestas where id_pregunta=" & dt.Tables(0).Rows(i).Item(9)  ''de la tabla de
respuestas,  traer el numero de respuestas para generar los campos
    Dim adap2 As New SqlClient.SqlDataAdapter(sql2, cn)  'Se crea el data adpater y se llena el data set con la
instruccion sql
     Dim dt2 As New DataSet                          'declaracion para datos internos
     adap2.Fill(dt2)                                 'tabla para datos internos
     r = dt2.Tables(0).Rows.Count                    'cuantos datos internos
     For m = 1 To r                                  'de 1 a n datos
       sql = sql & dt.Tables(0).Rows(i).Item(0) & Chr(m + 96) & " nvarchar(5),"    'concatena
     Next                                            'fin desde
  Case 3                                             'caso caja de texto
    j = dt.Tables(0).Rows(i).Item(2)                 'cuantas cajas
    If j = 1 Then                                    'si es una caja
      sql = sql & dt.Tables(0).Rows(i).Item(0) & " nvarchar(" & dt.Tables(0).Rows(i).Item(3) & "),"   'concatena
      Exit Select                                    'salir de casos
    End If
    For k = 1 To j                                   'desde 1 a n cajas asignacion de las letras cuando hay mas de 1 caja
      If k = 1 Then                                  'si es la primer caja
        sql = sql & dt.Tables(0).Rows(i).Item(0) & "a" & " nvarchar(" & dt.Tables(0).Rows(i).Item(3) & ")," 'concatena
      End If
      If k = 2 Then                                  'si es la segunda caja
```

```
      sql = sql & dt.Tables(0).Rows(i).Item(0) & "b" & " nvarchar(" & dt.Tables(0).Rows(i).Item(4) & ")," 'concatena
    End If
    If k = 3 Then                                   'si es la tercer caja
      sql = sql & dt.Tables(0).Rows(i).Item(0) & "c" & " nvarchar(" & dt.Tables(0).Rows(i).Item(5) & ")," 'concatena
    End If
    If k = 4 Then                                   'si es la cuata caja
      sql = sql & dt.Tables(0).Rows(i).Item(0) & "d" & " nvarchar(" & dt.Tables(0).Rows(i).Item(6) & ")," 'concatena
     End If
  Next                                              'fin desde
  Case 4                                            'caso check binario
    Dim x, y As Integer                             'declaracion de variables para conteo
    y = dt.Tables(0).Rows(i).Item(7)                'numero de respuestas
    For x = 1 To y                                  'de 1 hasta resouestas
      sql = sql & dt.Tables(0).Rows(i).Item(0) & Chr(x + 96) & " nvarchar(5)," 'concatena
    Next                                            'fin desde
  End Select                                        'fin casos
  End If                                            'fin si
Next                                                'fin desde
sql = sql.Substring(1, (sql.Length - 2))            'quitar espacios en blanco
sql = sql & ")"                                     'cierra instruccion sql
x.Connection = cn                                   'se asigna la conecion a utilizar
x.CommandText = sql                                 'se agrega la instruccion sql que se construyo anteriormente
Try                                                 'atrapa excepciones
  x.ExecuteNonQuery()                               'ejecuta instruccion sql
Catch ex As Exception                               'excepcion atrapada
  MsgBox(ex.Message)                                'mensaje
End Try                                             'fin atrapa
'Seleccionar preguntas todos
sql = "Select variable, tipovar, num_caja,largo1,largo2,largo3,largo4, opc_validas_chk,control_r, id_preguntas from
      preguntas where " & _" id_cuestionario=" & Me.LCuestionario.Text & " and  tipo=1"
Dim adap3 As New SqlClient.SqlDataAdapter(sql, cn) 'Se crea el data adpater y se llena el data set con la instruccion
sql
Dim dt3 As New DataSet                              'declaracion de la tabla3 para datos
adap3.Fill(dt3)
'inicia instruccion para crear la tabla
sql = "create table datos_" & Me.LCuestionario.Text & "_T " & "( Folio nvarchar(8),renglon nvarchar(2),nombre
      nvarchar(20), A_paterno nvarchar(15)," & _"A_materno nvarchar(15), sexo nvarchar(1),Edad
      nvarchar(3),Feh_nac nvarchar(15),"
n = dt3.Tables(0).Rows.Count                        'cuantas preguntas
For i = 0 To n - 1                                  'desde 0 a n preguntas -1
  If dt3.Tables(0).Rows(i).Item(1) = "N" Then       'si el control es numerico
    sql = sql & dt3.Tables(0).Rows(i).Item(0) & " nvarchar(" & dt3.Tables(0).Rows(i).Item(3) & ")," 'concatena
  Else
   Select Case (dt3.Tables(0).Rows(i).Item(8))      ' caso tipo de control
    Case 1                                          'caso botones
      sql = sql & dt3.Tables(0).Rows(i).Item(0) & " nvarchar(5),"     'concatena
    Case 2                                          'caso check box normal
      Dim r, m As Integer                           'declaracion de variables para contar y recorrer
      sql2 = "Select id_respuesta from respuestas where id_pregunta=" & dt3.Tables(0).Rows(i).Item(9) 'de la tabla
de respuestas,  traer el numero de respuestas para generar los campos
      Dim adap2 As New SqlClient.SqlDataAdapter(sql2, cn)  'Se crea el data adpater y se llena el data set con la
instruccion sql
      Dim dt2 As New DataSet                        'declaracion de la tabla2
      adap2.Fill(dt2)                               'llenar latabloa dos con datos
      r = dt2.Tables(0).Rows.Count                  'cuantas respuestas
      For m = 1 To r                                'de 1 hasta respuestas
        sql = sql & dt3.Tables(0).Rows(i).Item(0) & Chr(m + 96) & " nvarchar(5),"   'concatena
      Next                                          'fin desde
     Case 3                                         'caso caja de texto
      j = dt3.Tables(0).Rows(i).Item(2)             'numero de cajas
      For k = 1 To j                                'desde 1 asignacion de las letras cuando hay mas de 1 caja
        If k = 1 Then                               'si es la primera caja
      sql = sql & dt3.Tables(0).Rows(i).Item(0) & "a" & " nvarchar(" & dt3.Tables(0).Rows(i).Item(3) & ")," 'concatena
        End If
        If k = 2 Then                               'si es la segunda caja
```

```
        sql = sql & dt3.Tables(0).Rows(i).Item(0) & "b" & " nvarchar(" & dt3.Tables(0).Rows(i).Item(4) & "),"   'concatena
       End If
        If k = 3 Then                                   'si es la tecer caja
          sql = sql & dt3.Tables(0).Rows(i).Item(0) & "c" & " nvarchar(" & dt3.Tables(0).Rows(i).Item(5) & "),"   'concatena
        End If
        If k = 4 Then                                   'si es la cuarta caja
         sql = sql & dt3.Tables(0).Rows(i).Item(0) & "d" & " nvarchar(" & dt3.Tables(0).Rows(i).Item(6) & "),"   'concatena
        End If
      Next                                              'fin desde
      Case 4                                            'caso check binario
      Dim x, y As Integer                               'declaracin de contadores
      y = dt3.Tables(0).Rows(i).Item(7)                 'cuantas respuestas
      For x = 1 To y                                    'de 1 hasta respuestas
        sql = sql & dt3.Tables(0).Rows(i).Item(0) & Chr(x + 96) & " nvarchar(5),"   'concatena
      Next                                              'fin desde
      End Select                                        'fin casos
      End If                                            'fin si
Next                                                    'fin desde
sql = sql.Substring(1, (sql.Length - 2))                'quita espacios en blanco
sql = sql & ")"                                         'termina instruccion sql
x.Connection = cn                                       'se asigna la conecion a utilizar
x.CommandText = sql                                     ' se agrega la instruccion sql que se construyo anteriormente
Try                                                     'atrapa excepciones
   x.ExecuteNonQuery()                                  'ejecuta la sentencia sql
Catch ex As Exception                                   'excepcion atrapada
    MsgBox(ex.Message)                                  'mensaje
End Try                                                 'fin atrapa
```

Aquí llenamos la tabla de preguntas con la tabla en que finalmente quedaron la variables para que quede actualizada la tabla de preguntas con su referencia a la tabla donde se localiza la variable.

```
'actualizando el nombre de la tabla en preguntas
sql = "Update preguntas set tabla='Datos_" & Me.LCuestionario.Text & "_T' where tipo= 1 and id_cuestionario=" &
       Me.LCuestionario.Text
x.Connection = cn                                       'se asigna la conecion a utilizar
x.CommandText = sql                                     ' se agrega la instruccion sql que se construyo anteriormente
'actualizando el nombre de la tabla en preguntas
sql = "Update preguntas set tabla='Datos_" & Me.LCuestionario.Text & "_I' where tipo= 0 and id_cuestionario=" &
      Me.LCuestionario.Text
x.Connection = cn                                       'se asigna la conecion a utilizar
x.CommandText = sql                                     'se agrega la instruccion sql que se construyo anteriormente
End If                                                  'fin si
Me.Label1.Text = "El cuestionario ha sido autorizado"   'mensaje
End Sub                                                 'fin procedimiento
```

En las siguientes líneas de código se hace la rutina para mostrar los botones en pantalla del tipo de respuesta del control 1.

```
Sub crea_botones()
  trae_pregunta()
  respuestas()
  ncontrol = dresp.Tables(0).Rows.Count                 'numero de respuestas=numero de botones
  If ncontrol > 0 Then                                  'cuando es mas de un boton
    For i = 0 To ncontrol - 1                           'desde 0 a controles
      boton = New Button                                'crea control
      boton.ID = "btn" & i + 1
      boton.TabIndex = 10 + i
      boton.Text = dresp.Tables(0).Rows(i).Item(2)      'texto para el boton
      boton.CommandName = dresp.Tables(0).Rows(i).Item(1)  'valor para el boton
      AddHandler boton.Click, AddressOf Boton_Evento    'asignacion del evento a los botones creados
      Me.form1.Controls.Add(boton)
```

```
      LitControl.Text = Val(LitControl.Text) + 1
    Next
  End If
End Sub
```

Para poder mostrar los botones se necesitan los textos que iran en ellos, por lo cual se hace la siguiente consulta de la tabla de respuestas para obtenerlas.

```
Sub respuestas()
  'DATOS DE RESPUESTAS DE LA PREGUNTA SELECCIONADA
  sql = "Select num_opcion,valor_opcion,texto_opcion, imagen_opcion from respuestas where id_pregunta=" &
        Me.Request.QueryString(5)
  Dim adapres As New SqlClient.SqlDataAdapter(sql, cn)
  adapres.Fill(dresp)
End Sub
```

De la misma forma se manda traer el texto de la pregunta de la tabla de preguntas para mostrarse en pantalla.

```
Sub trae_pregunta()
  sql = "Select * from preguntas where id_preguntas=" & Me.Request.QueryString(5)
  Dim adap1 As New SqlClient.SqlDataAdapter(sql, cn)
  adap1.Fill(dt)
  Me.Litresp.Text = dt.Tables(0).Rows(0).Item(3)
End Sub
```

La siguiente rutina guardamos los datos obtenidos en la respuesta en caso de utilizar botones, es decir, control 1.

```
Sub guarda_dato()
  sql = "Select tipo from preguntas where id_preguntas=" & Me.Request.QueryString(5)
  Dim adap1 As New SqlClient.SqlDataAdapter(sql, cn)
  adap1.Fill(dt)
  If dt.Tables(0).Rows(0).Item(0).ToString = 1 Then
    sql="Update datos_" & Me.Request.QueryString(1) & "_T" & " Set " & Me.Request.QueryString(6) & "='" &
        Me.Litresp.Text.Trim & _ "' where folio=" & Me.Request.QueryString(4) & " and integrante='" &
        Me.Request.QueryString(2) & "'"
  Else
    sql="Update datos_" & Me.Request.QueryString(1) & "_I" & " Set " & Me.Request.QueryString(6) & "='" &
        Me.Litresp.Text.Trim & _ "' where folio='" & Me.Request.QueryString(4) & "'"
  End If
  If cn.State = ConnectionState.Closed Then
    A_conexion()
  End If
  x.Connection = cn   'se asigna la conexión a utilizar
  x.CommandText=sql 'se agrega la instruccion sql que se construyo anteriormente
  Try
    x.ExecuteNonQuery()        'se ejecuta la instruccion sql
    Catch ex As Exception
    MsgBox(ex.Message.ToString())
  End Try
End Sub
```

La siguiente rutina guardamos los datos obtenidos en la respuesta en caso de utilizar cajas de texto que es el control 3, así como también las respuestas de opción múltiple, es decir, control 2 y 4.

```
Sub guardar_dato(ByVal r As Integer)
  Dim tipopreg As String
  If tabla.Tables(0).Rows(Me.LitNp.Text).Item(13) = 1 Then  'Definimos el tipo de tabla en la que vamos a grabar si
    tipopreg = "T"                                          es en la de Hogar o en la de Individuos.
```

```
  tipopreg = "T"                ‘
Else
  tipopreg = "I"
End If
Select Case (tabla.Tables(0).Rows(Me.LitNp.Text).Item(9))
  Case 2, 4
    If tipopreg = "T" Then
      Sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & "='" &
            Me.CheckboxResp.SelectedValue & _"' where folio=" & Me.Request.QueryString(0) & " and  renglon="
            & Me.Request.QueryString(2)
    Else
      Sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & "='" &
            Me.CheckboxResp.SelectedValue & _"' where folio=" & Me.Request.QueryString(0)
    End If
    MsgBox(Me.CheckboxResp.SelectedValue)
  Case 3
    ncontrol = tabla.Tables(0).Rows(Me.LitNp.Text).Item(11)
    'si hay una caja de texto
    If ncontrol = 1 Then
      If tipopreg = "T" Then
        sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & "='" &
            Me.TextBox1.Text & _ "' where folio like '%" & Me.Request.QueryString(4) & "' and  renglon=" &
            Me.LitIntegrante.Text
      Else
        sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & "='" &
            Me.TextBox1.Text & _ "' where folio like '%" & Me.Request.QueryString(4) & """
      End If
    End If
      If ncontrol > 1 Then          'si hay mas de una caja de texto
        For i = 1 To ncontrol
          Select Case i
            Case 1                      ‘estas lineas se ejecutan cuando es solo una caja de texto
              If tipopreg = "T" Then
                sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & Chr(i +
                      96) & "='" & Me.TextBox1.Text & _ "' where folio like '%" & Me.Request.QueryString(4) & "' and
                      renglon=" & Me.LitIntegrante.Text
              Else
                sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & Chr(i +
                      96) & "='" & Me.TextBox1.Text & _ "' where folio like '%" & Me.Request.QueryString(4) & """
              End If
            Case 2                      ‘estas lineas se ejecutan cuando es son dos cajas de texto
              If tipopreg = "T" Then
                sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & Chr(i +
                      96) & "='" & Me.TextBox2.Text & _ "' where folio like '%" & Me.Request.QueryString(4) & "' and
                      renglon=" & Me.LitIntegrante.Text
              Else
                sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & Chr(i +
                      96) & "='" & Me.TextBox2.Text & _"' where folio like '%" & Me.Request.QueryString(4) & """
              End If
            Case 3                      ‘estas lineas se ejecutan cuando es son tres cajas de texto
              If tipopreg = "T" Then
                sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & Chr(i +
                      96) & "='" & Me.TextBox3.Text & _"' where folio like '%" & Me.Request.QueryString(4) & "' and
                      renglon=" & Me.LitIntegrante.Text
              Else
                sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & Chr(i +
                      96) & "='" & Me.TextBox3.Text & _ "' where folio like '%" & Me.Request.QueryString(4) & """
              End If
            Case 4                      ‘estas lineas se ejecutan cuando es son cuatro cajas de texto
              If tipopreg = "T" Then
                sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & Chr(i
                      + 96) & "='" & Me.TextBox4.Text & _"' where folio like '%" & Me.Request.QueryString(4) & "'
                      and  renglon=" & Me.LitIntegrante.Text
              Else
                sql = "Update datos_" & Me.cuest & "_" & tipopreg & " Set " & tabla.Tables(0).Rows(r).Item(2) & Chr(i
```

```
                    + 96) & "='" & Me.TextBox4.Text & _"' where folio like '%" & Me.Request.QueryString(4) & "'"
                End If
            End Select
            ejecuta_query()
        Next
    End If
End Sub
```

4.3 Como se implementó y se probó el ejemplo de desarrollo

Para poder implementar la versión que se utilizará de ejemplo del sistema se requiere como mínimo de un equipo que cuente con IIS como servidor web, de manera que se pueda hacer uso de los módulos de altas de encuestas, cuestionarios, y del mismo modo poder introducir las preguntas, repuestas, saltos y validaciones de cada una de las preguntas que formen el ejemplo, generar la encuesta con estos datos anteriormente introducidos, y finalmente utilizar la encuesta introduciendo datos hasta lograr concluir una entrevista y así tener los datos de las respuestas obtenidas en la base de datos.

La implementación se hará de manera local instalándola dentro de una intranet para poder hacer las pruebas necesarias requiriendo de 4 equipos más otro que funcionará como servidor.

Durante este proceso se planifica hacer pruebas utilizando todos los módulos del sistema, para lo cual se plantea la necesidad de requerir al menos cuatro personas involucradas, ya que una se ocupará de las funciones del investigador introduciendo preguntas, respuestas, saltos, validaciones. El segundo individuo realizará las función del revisor, supervisando el avance en la creación de la encuesta, hasta dar su visto bueno para así dar paso al Investigador; el cual dará la autorización con lo cual se podrá generar la encuesta; y finalmente el que realice la encuesta, el cual irá introduciendo los datos para ir llenando la base con la información que finalmente será el producto final del sistema.

4.4 Mantenimiento del Sistema

Cuando hablamos del Mantenimiento del Sistema nos referimos a la última fase del Ciclo de Vida a que todo Sistema debe someterse al desarrollarse, en este punto es donde se debe contemplar si el sistema tiene las características necesarias para comenzar nuevamente con el ciclo de vida de un sistema, ya que al necesitar cambios (reparaciones) o mejoras, nuevamente se someterá al ciclo de vida del desarrollo de un sistema.

Para poder comenzar con el mantenimiento del sistema de “ANÁLISIS Y DISEÑO DEL SISTEMA QUE GENERA ENCUESTAS ELECTRÓNICAS VÍA INTERNET” es necesario recabar los requerimientos de los usuarios que son los que al tener el contacto con el sistema poseen las necesidades más próximas, las cuales tendrán que ser evaluadas.

Posteriormente implementaremos estos nuevos requerimientos que han sido obtenidos a través de las necesidades de los usuarios del sistema, después se pasará al diseño de los cambios y así terminar con la implementación de estos, iniciando así nuevamente con el ciclo de vida del sistema.

Es importante mencionar que es de suma importancia que quien realice el mantenimiento al sistema, debe tener conocimiento de la finalidad que tiene el sistema, ya que tiene características muy particulares por el tipo de requerimientos con que fue concebido, por lo que se recomienda que sea el administrador del sistema el encargado de realizar estas modificaciones.

Las partes que más pueden sufrir modificaciones dada la naturaleza del sistema son la sección de reportes, ya que constantemente pueden estar solicitando nuevos reportes de acuerdo a las necesidades de la encuesta, los más básicos están ya implementados en el sistema, no así los que sean específicos para cada encuesta o algunos que requiera el cliente para poder revisar los datos y analizarlos de acuerdo a sus necesidades.

Otra sección que es susceptible de mantenimiento correctivo es la del revisor, ya que esta ha sido diseñada de acuerdo a los requerimientos generales que se han analizado y que en general cubren lo mínimo necesario para hacer una revisión del diseño del cuestionario, por lo que para encuestas particulares pudieran solicitar algo adicional a lo que el sistema por si solo ya brinda en esta sección.

Es importante contar con la información adecuada de las necesidades que se tienen adicionales a las bondades con que ya cuenta el sistema, para poder hacer la planeación correcta y hacer los cambios al sistema y obtener en menor tiempo y de mejor manera los nuevos cambios y requerimientos solicitados.

Así mismo se realizarán una lista con los requerimientos solicitados con el fin de poder decir cuáles son factibles y cuáles no podrán ser hechos, para poder informar a los solicitantes la respuesta, así como de los tiempos requeridos para realizar estos cambios en caso de que sean factibles.

4.5 Capacitación del personal

Es importante capacitar al personal en los diferentes niveles de usuario que maneja el sistema:

- Investigadores
- Directores de área
- Revisores de cuestionario
- Administrador de sistemas
- Coordinadores
- Supervisores

- Entrevistador
- Entrevistado

Cabe mencionar que la parte más importe a capacitar es a los investigadores, ya que ellos son quienes en primera instancia tendrán acceso al sistema para introducir la información básica para generar la encuesta. Tendremos que dar una explicación muy específica de como tendrán que indicar los saltos, así como las validaciones, ya que no será una manera tan sencilla, también se les dirá como capturar las respuestas para cada una de las preguntas, e indicar los parámetros básicos que tendrá cada uno de esos reactivos, por tanto es importante instruir correctamente a los investigadores para que puedan hacer el uso correcto y eficiente de sus requerimientos.

La capacitación para los Directores de área, revisores de cuestionario, Coordinadores y Supervisores, es más de conocimiento general de acceso y funcionamiento de la consulta de información que irá arrojando la aplicación, por tanto no es tan prioritaria como lo es la del investigador, pero no por ello deja de ser importante, ya que dichos usuarios interactúan con el sistema mientras está en desarrollo el cuestionario o bien cuando ya está en campo la aplicación y siendo su insumo principal de información para la toma de decisiones durante el levantamiento de la encuesta.

El entrevistador y/o el entrevistado deben tener capacitación previa a la aplicación de la encuesta electrónica, ya que debe estar familiarizado con el sistema para que sea más fácil el uso de esta. Para el caso de los entrevistadores se les dará la capacitación vía presencial en alguna instalación proporcionada por el centro de encuestas. Para el caso del entrevistado que responda la encuesta desde su lugar de origen se les creará un tutorial de cómo utilizar el programa, este tutorial será enviado vía correo electrónico.

El Administrador del Sistema, debe ser capacitado en todos los módulos del sistema, ya que el tendrá que resolver todas las dudas que surjan en torno al uso manejo y Administración del sistema, debe tener los conocimientos tanto del manejo de la aplicación, como en la parte técnica para poder describir en determinado momento la falla (nueva necesidad).

Como requerimientos para la capacitación de los Investigadores, Directores de área y Revisores con respecto al cuestionario, se solicita tengan conocimientos previos en el manejo de equipo de cómputo, conocimiento básico de Windows, conocimientos básicos y manejo de un navegador (Internet Explorer, FireFox, Chrome, Etc.), esto con la finalidad de poder hacer más dinámica y ágil dicha capacitación.

La capacitación para el personal que hará uso de la aplicación se calendarizará para darle el espacio necesario; para los Investigadores, Directores de área y

Revisores del cuestionario, se les integrarán los siguientes módulos a su capacitación:

MÓDULOS
Conocimiento de la aplicación
Autentificación
Entorno general del sistema
Integración del cuestionario
Reportes
Obtención de información final

Fig. 4.1 Cuadro de módulos de capacitación de Investigadores, directores de área y revisores.

A los Coordinadores y supervisores se les puede integrar a esta capacitación excluyéndolos de los módulos de Integración de cuestionario y Obtención de información final, que sólo atañe a los usuarios mencionados con anterioridad.

Como requerimientos para esta capacitación de los entrevistadores y los entrevistados se solicita tengan conocimientos previos en el manejo de equipo de cómputo, conocimiento básico de Windows, conocimientos básicos y manejo de un navegador (Internet Explores, FireFox, Chrome, Etc.), esto con la finalidad de poder hacer más dinámica y ágil la capacitación.

Para los entrevistadores y los entrevistados se les impartirán los siguientes módulos:

MÓDULOS
Conocimiento de la aplicación
Autentificación
Entorno general del sistema
Tipo de controles y su forma de respuesta
Uso de la entrevista

Fig. 4.2 Cuadro de módulos de capacitación de entrevistadores/entrevistados.

Durante la capacitación se resolverán las diversas dudas que llegaran a surgir, acerca del uso y funcionamiento del sistema.

La capacitación podría impartirse de manera local o bien de manera centralizada, es decir en la sede donde recabará la información sería la sede central que en el caso de tratarse de una encuesta a nivel Nacional, se requeriría el traslado del personal para poder impartir la capacitación; se requiere de un aula con la capacidad para la cantidad de personas a capacitar, contactos para conectar los equipos y hacer pruebas que sería mínimo el 50% del número de entrevistadores a capacitar, un proyector y micrófono como requerimientos mínimos. En el caso de ser a nivel Nacional pero con capacitación dentro de la

entidad, tendría que buscarse de manera similar: un lugar con la capacidad para el número de entrevistadores a capacitar y el número de contactos a utilizar; y, dependiendo del acuerdo con el cliente puede requerir del traslado del equipo de cómputo necesario para hacer las prácticas del sistema o bien el cliente proporcionará su equipo.

CONCLUSIONES

El presente trabajo de investigación “ANÁLISIS Y DISEÑO DEL SISTEMA QUE GENERA ENCUESTAS ELECTRÓNICAS VÍA INTERNET” nos ha permitido observar en primera instancia la magnitud que tiene la misma. Entre más nos introducíamos en el estudio, análisis y diseño, nos percatamos que podíamos hacerlo cada vez más específico y detallado según quisiéramos, encontrándonos en la situación de que era un tema muy complejo e interesante a la vez. Esto nos permitió tomar la decisión de delimitar más el ejemplo de desarrollo.

Aunado a esto nos percatamos de algunas limitaciones por parte del software utilizado, así como en el que se encuentra en el mercado actualmente. Estas limitantes hicieron tener que desarrollar mucho más el análisis y desarrollo para poder lograr nuestra meta que es realizar en esta etapa un ejemplo de desarrollo.

Hemos visto como resultado de esta investigación que se necesitará mucho más tiempo del contemplado inicialmente, esto para poder terminar la aplicación y poderla implementar finalmente en el hardware propuesto, así como en la parte de utilización en definitiva.

Al inicio fue un poco duro y complejo el poder realizar el análisis de la tesis, sin embargo con la ayuda de nuestra asesora de tesis pudimos llegar a los objetivos planeados.

Las ventajas que tiene el utilizar un manejador de bases relacionales es entre otras, el poder generar consultas y/u operaciones entre todos los elementos de la base de datos de manera rápida y más sencilla, ya sea utilizando los procedimientos almacenados o el algebra relacional para poder obtener la información deseada. De igual manera las ventajas que nos dio el utilizar el lenguaje .NET es el poder utilizar de mejor manera la creación de objetos o controles, los cuales se pudieron hacer de forma dinámica en conjunto con la elaboración de una página WEB.

En cuanto a la forma de programar dirigimos todo para tener el modelo de 3 capas (presentación, de negocios y de datos), ya que este modelo nos da ventajas como el poder realizar un mejor mantenimiento, según requerimientos nuevos solicitados por todos los usuarios del Sistema. Esto es debido a que sólo se harán modificaciones en la mayor parte en la capa de presentación.

Al inicio se complicó un poco, ya que no teníamos mucho conocimiento de este lenguaje de programación, así como del manejador de bases de datos utilizados en el presente proyecto, sin embargo guiándonos en los conceptos aprendidos durante nuestra estancia en la Universidad, la dificultad para comprender su funcionamiento fue menor.

Por lo anterior podemos concluir que el “ANÁLISIS Y DISEÑO DEL SISTEMA QUE GENERA ENCUESTAS ELECTRÓNICAS VÍA INTERNET”, cubre las necesidades mínimas requeridas para poder cambiar completamente la forma en cómo actualmente se hacen las encuestas, sea en el área de encuestas de salud o de cualquier otro tipo de encuestas. Ya que es poco el software que existe en este rubro y hay menos en la plataforma propuesta para esta tesis.

Así mismo pudimos observar que se contribuye a mejoras tanto económicas, sociales y de medio ambiente. Y que dado la importancia de esta investigación por la solución propuesta a los procesos actuales de trabajo con respecto a la aplicación de encuestas, se mejoran considerablemente los tiempos de entrega de información que es el producto final y que a su vez permitirán hacer un análisis en periodos de tiempos reales en un momento actual, y ya no hacer estos análisis meses después o incluso en algunos casos en periodos mayores a un año.

Esto ayudará también para que se puedan tomar decisiones oportunas sobre cualquier ámbito, ya sea de salud nacional o de algún programa social o incluso de aspectos económicos, trasladando esto a otras áreas de investigación que competen al desarrollo del país.

Al final de esta investigación consideramos que queda de manifiesto la formación que nos dio la Universidad, en donde pudimos poner en práctica los conocimientos adquiridos durante nuestra estancia en ella. Y donde gracias a ella hemos podido llegar a la solución de problemas tanto de instituciones públicas como privadas, en pro de nuestro desarrollo profesional.

GLOSARIO

Base de datos*: Colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada.*

Tabla: *Es un conjunto de registros almacenados en forma secuencial con atributos, tipos y longitudes de cada campo.*

Modelo de datos: *Es una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia.*

Instancia: *La colección de información almacenada en un determinado momento en el tiempo.*

Esquema: *Es el diseño global de la BD se llama esquema de la BD.*

Lenguaje de definición de datos*: La estructura de almacenamiento y los métodos de acceso se especifican por medio de un conjunto de definiciones en un tipo especial de DDL.*

Lenguaje de manipulación de datos: *es un lenguaje que capacita a los usuarios a acceder o manipular los datos.*

Gestor de bases de datos: *es un módulo de programa que proporciona el interfaz entre los datos de bajo nivel almacenados y los programas de aplicación y consultas.*

Administrador de bases de datos: *Es la persona que tiene el control central de los datos y de los programas que acceden a esos datos.*

Modelo entidad-relación: *Se basa en una percepción de un mundo real que consiste en una colección de objetos básicos llamados entidades, y relaciones entre estos objetos.*

Entidad: *Es un objeto distinguible de otros por medio de un conjunto de atributos.*

Relación: E*s una asociación entre varias entidades*.

Atributos: *Es una función que asigna un conjunto de entidades a un dominio.*

Cardinalidad de asignación: *Expresa el número de entidades con las que puede asociarse otra entidad mediante un conjunto de relaciones.*

Clave: *Es un conjunto de uno o más atributos que, considerados conjuntamente, nos permiten identificar de forma única a una entidad dentro del conjunto de entidades.*
Modelo relacional: *En él se representan los datos y relaciones entre los datos mediante una colección de tablas, cuyas columnas tienen nombres únicos.*

Base de datos relacional: *Es una colección de tablas, a cada una de las cuales se les asigna un nombre único.*

Relación: *Esta definida por una asociación entre varias entidades.*

Tupla: *Es la representación de una fila en una de las tablas que se está almacenando datos.*

IIS: *Es la abreviación de Internet Information Services y es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows.*

Ruta crítica: *Es un proceso administrativo (planeación, organización, dirección y control) de todas y cada una de las actividades componentes de un proyecto que debe desarrollarse durante un tiempo crítico y al costo óptimo.*

Red de actividades: *Es la ilustración gráfica del conjunto de operaciones de un proyecto y de sus interrelaciones. La red esta formada por flechas que representan actividades y nudos o uniones que simbolizan eventos.*

Renderización: *Es la interpretación de la computadora de las escenas en 3 dimensiones y plasmarlas en una imagen bidimensional.*

BIBLIOGRAFÍA

1.- Baena, Guillermina. [1995]. ***Instrumentos de Investigación***. [1ª Edición]. México: Ed. Mexicanos Unidos.

2.- Castaño, Adoración de Miguel. [2005]. ***Fundamentos y modelos de bases de datos***. [2ª. Edición]. México: Ed. Alfa Omega.

3.- Castaño, Adoración de Miguel. [2005]. ***Diseño de Base de Datos Relacionales***. [2ª. Edición]. México: Ed. Alfa Omega -Rama

4.- Garza Mercado, Ario. [1981]. ***Manual de técnicas de investigación***. [3ª Edición]. México: Ed. Colegio de México.

5.- Kroenke, David M. [1996]. ***Procesamiento de Bases de Datos***. [5ª. Edición]. México: Ed. Prentice Hall.

6.- Pérez, César López. [2008]. ***MySQL para Windows y Linux***. [2ª. Edición]. México: Ed. AlfaOmega.

7.- Silberschatz, Abraham. [2002]. ***Fundamentos de bases de datos***. [4ª. Edición]. España: Ed. MCGRAW-HILL INTERAMERICANA.

8.- Sommerville, Ian. [2005]. ***Ingeniería del Software***. [7ª Edición]. España. Ed. Pearson-Addison Wesley.

CIBERGRAFÍA

1.- Abraham Silberschatz, et al. [2002]. *Fundamentos de bases de datos 4ª. Edición* [Libro en línea]. McGraw-Hill. España. Disponible: http://www.scribd.com/doc/3987911/Fundamentos-de-bases-de-datos [Consulta: 2010, Mayo].

2.- Departamento de sistemas y computación Programa de elaboración de tutoriales, Instituto Tecnológico de la Paz. [1999]. [página Web]. Tutorial de Bases de Datos. Disponible: http://sistemas.itlp.edu.mx/tutoriales/basedat1/ [Consulta: 2010, Julio].

3.- *"¿Cuál es la razón de ser de una Base de Datos?"* Disponible: http://www.osmosislatina.com/aplicaciones/bases_de_datos.htm
[Consulta: 2010, Marzo 12].

4.- Promonegocios.net "*Tipos de Encuesta"*
Disponible: http://www.promonegocios.net/mercadotecnia/encuestas-tipos.html
[Consulta: 2010, Enero].

5.- Portal de relaciones públicas. *"TÉCNICAS DE INVESTIGACIÓN"*
Disponible: http://www.rrppnet.com.ar/tecnicasdeinvestigacion.htm
[Consulta: 2010, Septiembre]

6.- Apuntes de ficheros y bases de datos. Mercedes Marqués Febrero 2001.
Disponible: http://www3.uji.es/~mmarques/f47/apun/apun.pdf
[Consulta: 2009, Noviembre].

7.- NeuroAprendizaje. [página web]. *Zona de Descarga IUGT : Metodología de la Investigación . Disponible:* http://www.cartografiamental.com/iugt2
[2009, Diciembre 14].

8.- Guzmán, M. de. (1993) . *Enseñanza de las Ciencias y la Matemática: Matemática* [Libro en línea] Organización de Estados Iberoamericanos para la Educación, la Ciencia y la Cultura: Editorial Popular. Disponible: http://www.oei.org.co/oeivirt/edumat.htm [Consulta : 2010, 18 Enero].