Universidad Nacional Autónoma de México

Facultad de Ingeniería

SISTEMA PARA DEPURACION INTEGRAL DEL PADRON ELECTORAL Y CREDENCIAL PARA VOTAR CON FOTOGRAFIA

T E S I S

Que para obtener el Título de
INGENIERO EN COMPUTACION

p r e s e n t a:

JOSE ALFREDO ZARZA RAMIREZ

DIRECTOR DE TESIS:
ING. EDUARDO BADILLO GUTIERREZ
ING. ALBERTO TEMPLOS CARBAJAL

MEXICO, D. F. DICIEMBRE, 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

TEMA DE TESIS

ALUMNO: ZARZA RAMIREZ JOSE ALFREDO

TEMA: Sistema de cómputo para la revisión y actualización de la Base de Datos del Padrón Electoral Nacional y la generación de información para la impresión de la credencial para votar con fotografía en computadoras IBM-RISC/6000 con plataforma AIX y Oracle.

A mi tíos Román Ramírez y Alfonso Ramírez,
por su gran apoyo.
A mi Esposa,
por su comprensión y paciencia
A mi hijo,
por la fuerza que me motiva.
A mi madre,
por la vida que me entregó.
A mis hermanos,
por su alegría y compañía.
A mis tíos
por su gran motivación hacia mi.
A mis directores,
por su valiosa ayuda.
A mis maestros,
por sus enseñanzas
A mis amigos,
por su gran amistad.

TEMARIO DE TESIS

INSTITUTO FEDERAL ELECTORAL

CAPITULO I

INTRODUCCION

I. 1 INTRODUCCION

Dentro de nuestro entorno social existen varias formas de ejercer nuestros derechos constitucionales y uno de todos ellos es: el derecho de ejercer el voto.

Nuestro voto es universal, libre, secreto, directo, personal e intransferible. [1]

Votar en las elecciones constituye un derecho y una obligación del ciudadano que se ejerce para integrar los órganos de elección popular. [2]

Votar significa participar de manera directa en la elección de nuestros representantes que conforman los distintos órganos del estado.

De esta forma, la institución que tiene como función principal la organización y desarrollo del proceso electoral es el Instituto Federal Electoral.

Una de las funciones del Instituto Federal Electoral es, integrar el Registro Federal de Electores que tiene entre otros fines los siguientes:

- Formar el Catálogo General de Electores.
- Formar el Padrón Electoral.
- Expedir la Credencial para votar con fotografía.
- Revisar y actualizar el Padrón Electoral.
- Mantener actualizada la Cartografía Electoral del País.

Entonces el Programa de Depuración Integral del Padrón Electoral surge a partir de la necesidad de ofrecer un padrón real.

El sistema de cómputo para realizar el Programa de Depuración Integral del Padrón (PDI) contempla los siguientes puntos:

1. TIPIFICACION CARTOGRAFICA.
2. CARTOGRAFIA INVALIDA.
3. HOMONIMIAS.
4. FOLIOS NACIONALES.
5. VERIFICACION DOCUMENTAL.

Estas actividades tienen como fin dar un Padrón Electoral actualizado.

---

1 Art 4o. inciso 2. Código Federal de Instituciones y Procedimientos Electorales.
2 Art. 4o. inciso 1. Código Federal de Instituciones y Procedimientos Electorales.

También se desarrollo el sistema de cómputo para la Renumeración de Secciones, con el fin de obtener un mejor control sobre la Cacografía Electoral del país.

El Código Federal de Instituciones y Procedimientos Electorales es la ley que regula al Instituto Federal Electoral y en su artículo 144 dice:

"Los ciudadanos tienen la obligación de acudir a las oficinas o módulos que determine el Instituto Federal Electoral a fin de obtener su credencial para votar con fotografía."

Por lo anterior el Registro Federal de Electores como responsable directo, realizo un proyecto para generar dichas credenciales para votar con fotografía, y por lo tanto fue necesario un sistema de cómputo para realizar dicha tarea.

Todos los sistemas se construyeron en infraestructura computacional del Registro Federal de Electores a nivel nacional.
Las computadoras son IBM-RISC/6000 con sistema operativo AIX ver 3.2 y ORACLE como el manejador de la Base de Datos. Las computadoras se encuentran distribuidas en 17 estados de la República Mexicana conectadas a través de líneas privadas de Telmex formando una topología estrella, donde el nodo central es el Centro Nacional de Cómputo el cual se encuentra en el Distrito Federal y los nodos conectados a este son los Centros Regionales de Cómputo distribuidos en 17 estados de la República Mexicana.

La red utiliza el protocolo X.25 para la comunicación entre el Centro Nacional y los Centros Regionales.

Los sistemas desarrollados actúan directamente sobre la base de datos en ORACLE, así que su desarrollo se realizó utilizando herramientas de ORACLE como :

- SQL*FORMS
- SQL*REPORT
- SQL*PLUS
- PRO*C
- SQL*MENU

Los programas se encuentran en cada uno de los Centros Regionales para su operación local.

En los siguientes capítulos se describe a detalle cada uno de los aspectos aquí mencionados.

INSTITUTO FEDERAL ELECTORAL

CAPITULO II

EL INSTITUTO FEDERAL ELECTORAL Y REGISTRO FEDERAL DE ELECTORES

- II.1 INTRODUCCION
- II.2 BASES HISTORICO - JURIDICAS
- II.3 INSTITUTO FEDERAL ELECTORAL
- II.4 REGISTRO FEDERAL DE ELECTORES

II.1 INTRODUCCION.

Todo el entorno electoral gira alrededor del Padrón Electoral, como materia prima de trabajo, por ejemplo se utiliza para :

- generar la credencial para votar con fotografía.
- para generar las listas nominales.
- para realizar la insaculación, etc.

El Registro Federal de Electores como un órgano dependiente del Instituto Federal Electoral tiene como función dar mantenimiento al Padrón Electoral Nacional y generar todos los productos derivados de este.

Este capítulo tiene como fin describir de manera general al Instituto Federal Electoral, su historia, sus normas, sus funciones, sus límites, sus productos electorales, etc. Lo mismo se describirá del Registro Federal de Electores.

Lo anterior con el fin de comprender desde sus raíces el objetivo del Programa de Depuración Integral de Padrón y la generación de la Credencial Para Votar con fotografía.

II.2 BASES JURIDICAS

Se puede tomar como base que desde la Revolución Mexicana hasta nuestros días, el país ha evolucionado en los aspectos electorales con mayor constancia.

La ley electoral del **19 de diciembre de 1911** y su reforma en **1912** aportaron características fundamentales tales como:

- El otorgamiento de personalidad jurídica a los partidos políticos.
- La organización del registro de electores.
- La creación de colegios municipales sufragáneos (encargados de organizar la elección).
- La división del territorio en distritos electorales renovables.
- La entrega de la boleta electoral al sufragante en la casilla.

Esta ley fue el marco normativo electoral durante las elecciones en que Francisco I. Madero fue elegido presidente de la República.

También en la reforma de **Mayo de 1912** se establece la elección directa para los miembros del Congreso de la Unión.

La ley para las elecciones de los poderes federales del **2 de Julio de 1918** integra nuevos elementos como:

- La garantía del secreto de voto.
- Da el carácter de permanente al Padrón Electoral.
- La creación de consejos distritales y municipales.

Las reformas del **25 de Mayo y la del 7 de julio de 1920** , insistían en que los partidos políticos tenían que elaborar sus propias boletas y entregárselas a los presidentes municipales, además de que las credenciales tendrían que ser de los expedientes electorales y darle constancia al elector de haber votado.

La reforma del **21 de febrero de 1949** dictaminó de entre sus agregados los siguientes:

- La obligación de los partidos políticos para tener un comité directivo en cada entidad federativa donde cuenten con mas de mil afiliados.
- La prohibición de que los miembros de las comisiones federal electoral y local no figuren como candidatos a cargos de elección popular, a menos de que se separen de sus puestos seis meses antes de la elección.

Una reforma importante surgió en **1954** la cual **OTORGA EL VOTO A LA MUJER** y la de incluir a los diputados de partido como complemento del sistema de mayoría y para establecer la credencial permanente de elector.

El **26 de Diciembre de 1969** se reforma la constitución para reconocer el derecho del voto a los 18 años y la reforma del **14 de febrero de 1972** se establece en 21 y 30 años las edades mínimas para ser diputado y senador respectivamente, así como se incrementa a 250,000 el número de habitantes por distrito electoral federal.

El **30 de Diciembre de 1977** se publica en el *Diario Oficial de la Federación* la llamada reforma política, mediante la promulgación de la ***Ley Federal de Organizaciones Políticas y Procesos Electorales*** **(LOPPE)** que introduce conceptos como:

- El registro de los partidos políticos condicionado al resultado de las elecciones.
- El establecimiento de prerrogativas para los partidos políticos.
- El reconocimiento a las asociaciones políticas.

En **1987** se derogó al LOPPE con la promulgación del Código Federal Electoral que atribuyó al Gobierno Federal la responsabilidad de la organización de los comicios federales, así mismo permitir a los partidos políticos y a los ciudadanos participar en organismos electorales. También se creó el Tribunal Federal Electoral como órgano jurisdiccional en materia electoral.

El **6 de Abril de 1990** se publicaron en el *Diario Oficial de la Federación* reformas, las cuales fueron las siguientes:

- La reiteración de la obligación en el desempeño de los cargos de elección popular.
- La creación del Registro Nacional de Ciudadanos.
- La definición del Proceso Electoral.
- La creación del servicio profesional electoral.
- Reglas de integración de la Asamblea de Representantes del Distrito Federal.
- Plantea el proceso de insaculación para formar los representantes de casilla.

Estas funciones se realizan a través de un organismo público dotado de personalidad jurídica y patrimonio propios, que es precisamente el **INSTITUTO FEDERAL ELECTORAL.**

El **15 de Agosto de 1990** se publicó en el *Diario Oficial de la Federación* el **Código Federal de Instituciones y Procedimientos Electorales (COFIPE).** Al que se le realizaron adiciones complementarias que fueron publicadas en el *Diario Oficial de la Federación* el **3 de Enero de 1991** y el **14 de Julio de 1992** ; esta modificación incorpora **la generación de la nueva credencial para votar con fotografía así como para depurar y actualizar integralmente el Padrón Electoral para las elecciones federales de Agosto de 1994.** El cumplimiento de esta reforma es el tema principal de esta tesis.

El *Código Federal de Instituciones y Procedimientos Electorales* (COFIPE) esta integrado por ocho libros, estos a su vez divididos en varios títulos y capítulos.
Las materias principales que regula el COFIPE son:

1. La integración de los poderes legislativo y ejecutivo de la Unión.
2. Los derechos, prerrogativas y obligaciones de los partidos políticos.
3. La integración y atribuciones del Instituto Federal Electoral.
4. Los procedimientos especiales de las direcciones ejecutivas del Instituto.
5. La organización y desarrollo del proceso electoral.
6. La integración y atribuciones del Tribunal Federal Electoral.
7. El procedimiento para las nulidades, el sistema de medios de impugnación y las sanciones administrativas.
8. Los procedimientos para la elección e integración de la Asamblea de Representantes del Distrito Federal.

Los aspectos más importantes para el objetivo de esta tesis son los libros 3 (La integración y atribuciones del Instituto Federal Electoral) y el libro 4 (Los procedimientos especiales en las direcciones ejecutivas del Instituto). Los cuales serán tema principal de los siguientes incisos de este capítulo.

II.3 EL INSTITUTO FEDERAL ELECTORAL.

Apartir de la reforma del **15 de Agosto de 1990**, la cual dicta la creación del **Instituto Federal Electoral.**

La constitución Política señala en su artículo 41 lo siguiente:

> *" La organización de las elecciones federales es una función estatal que se ejerce por los Poderes Legislativo y Ejecutivo de la Unión, con la participación de los partidos políticos nacionales y de los ciudadanos según lo disponga la ley. Esta función se realizará a través de un organismo público dotado de personalidad jurídica y patrimonio propios. La certeza, legalidad, imparcialidad, objetividad y profesionalismo serán principios rectores en el ejercicio de esta función estatal."*

El Instituto Federal Electoral (IFE) es el organismo público encargado de la organización de las elecciones, además de autónomo en la materia, profesional en su desempeño y autónomo en sus decisiones.

Los fines que persigue el IFE son:

A. Contribuir al desarrollo de la vida democrática.
B. Preservar el fortalecimiento del régimen de partidos políticos.
C. **Integrar el Registro Federal de Electores.**
D. Asegurar a los ciudadanos el ejercicio de los derechos político-electoral y vigilar el cumplimiento de sus obligaciones.
E. Garantizar la celebración periódica y pacífica de la elecciones para renovar a los integrantes de los Poderes Legislativo y Ejecutivo de la Unión.
F. Velar par la autenticidad y efectividad del sufragio.
G. Coadyuvar en la promoción y difusión de la cultura política.

El Instituto Federal Electoral realiza sus funciones que la ley le confiere a través de organismos centrales tal como lo indica la
figura II.1.

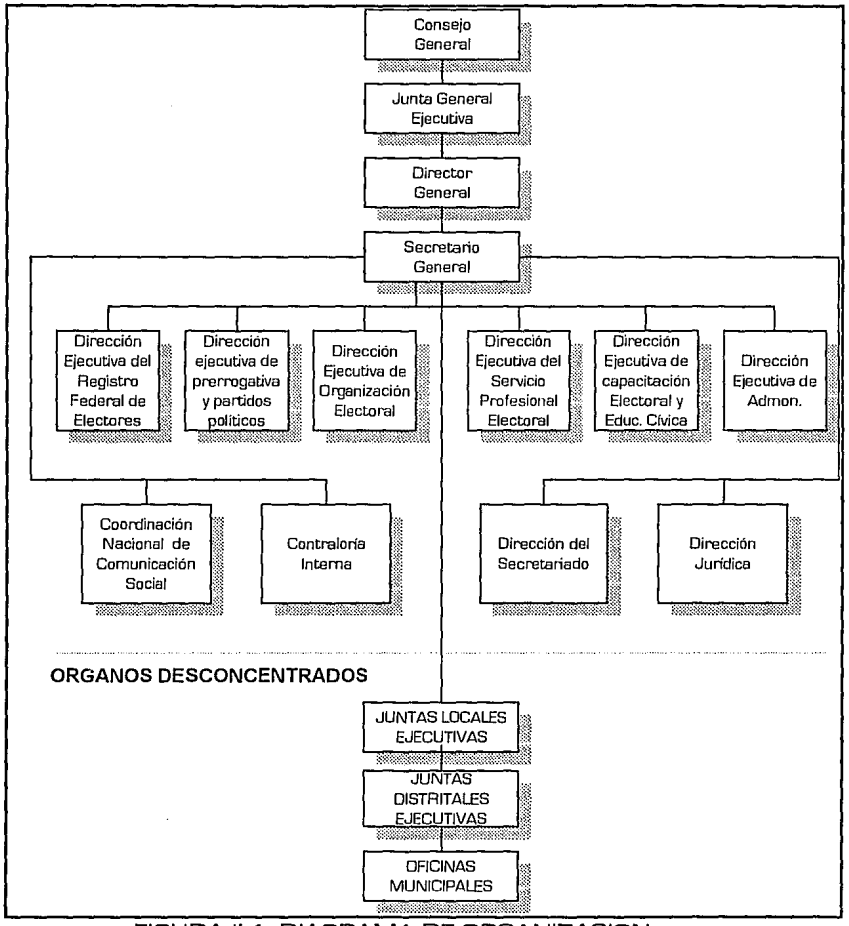


FIGURA II.1. DIAGRAMA DE ORGANIZACION

1. CONSEJO GENERAL.

Es el órgano superior de dirección de todo el Instituto, dicta las resoluciones y lineamientos que éste habrá de seguir.

Esta integrado por:

- Un consejero del Poder Ejecutivo como presidente (Secretario de Gobernación).
- Cuatro consejeros del Poder Legislativo.
- Seis consejeros magistrados.
- Representantes de los partidos políticos nacionales.
- Un secretario (Secretario General del IFE).
- El Director General del IFE.

2. JUNTA GENERAL EJECUTIVA.

Es el órgano ejecutivo de mayor jerarquía del Instituto, encargado de llevar a cabo las resoluciones dictadas por el Consejo General, a la vez que fija las políticas generales, los programas y los procedimientos que lo rigen.

Esta integrada por:

- El Director General.
- El Secretario General.
- El Director Ejecutivo del Registro Federal de Electores.
- El Director Ejecutivo de Prerrogativas.
- Partidos Políticos.

3. DIRECTOR GENERAL.

Preside y coordina la Junta General Ejecutiva, conduce la administración y supervisa el desarrollo adecuado de las actividades de los órganos ejecutivos y técnicos del Instituto.

4. EL SECRETARIO GENERAL.

Suple en sus funciones las ausencias temporales del Director General, actúa como secretario del Consejo General y de la Junta General Ejecutiva, recibe los informes de los órganos desconcentrados del Instituto, expide las certificaciones que requieran.

5. LAS DIRECCIONES EJECUTIVAS.

Son los órganos centrales de ejecución parcial de las atribuciones del Instituto de acuerdo al área de su competencia, estos cargos los ocupan funcionarios nombrados por el Director General. Estas direcciones son:

A. La dirección ejecutiva del Registro Federal de Electores.
- Integrar el Catálogo General de Electores.
- Formar, revisar y actualizar el Padrón Electoral.
- Expedir la credencial para votar con fotografía. Depurar y actualizar el Padrón.
- Proporcionar a los órganos del Instituto y a los partidos políticos nacionales las listas nominales de electores.
- Proponer la división territorial nacional en los trescientos distritos uninominales y de las cinco circunscripciones plurinominales.
- Mantener actualizada y clasificada la cartografía del país.
- Asegurar la integración y el funcionamiento de las comisiones de vigilancia, registrando la asistencia de los partidos políticos, solicitándoles los estudios y las consultas de su competencia.

B. Dirección ejecutiva de prerrogativas y partidos políticos.
- Conocer las notificaciones que formulen las organizaciones que pretenden constituirse en partidos políticos.
- Recibir las solicitudes de las organizaciones que cumplen con los requisitos de ley.
- Inscribir en el libro correspondiente el registro de los partidos y los convenios de fusión, frentes y coaliciones.
- Administrar a los partidos políticos nacionales el financiamiento público legal.
- Tramitar las franquicias postales y telegráficas a las que tienen derecho los partidos políticos nacionales.

C. Dirección ejecutiva del servicio profesional electoral. Esta dirección es responsable de:
- Formular el anteproyecto de Estatuto del Servicio Profesional Electoral.
- Llevar a cabo los programas de reclutamiento, selección, formación y desarrollo del personal profesional.

D. Dirección ejecutiva de capacitación electoral y educación cívica. Esta dirección tiene a su cargo:
- Elaborar y proponer los programas de educación cívica y capacitación electoral que desarrollarán las juntas, a la vez que vigila y coordina su cumplimiento.
- Preparar el material didáctico y los instructivos electorales.
- Orientar a los ciudadanos para el ejercicio de sus derechos y el cumplimiento de sus obligaciones político-electoral.

E. Dirección ejecutiva de administración. Esta dirección tiene a su cargo:
- Aplicar las políticas, normas y procedimientos para la administración de los recursos financieros y materiales del Instituto, así como organizar, dirigir y controlar la prestación de los servicios generales.
- Establecer y operar los sistemas administrativos para el ejercicio y control presupuestarios.
- Elaborar manual de organización y el catálogo de cargos y puestos del Instituto.

F. Coordinación nacional de comunicación social. Su función es la de coordinar las acciones de los órganos del Instituto en materia de comunicación social interna y externa.

G. Contraloría interna. Tiene a su cargo vigilar que el ejercicio de la función pública en el Instituto se apegue irrestrictamente a los lineamientos normativos establecidos.

H. Dirección Jurídica. Su función sustantiva radica en el manejo de los asuntos jurídicos en torno a las funciones y al que hacer del Instituto.

I. Dirección del Secretariado. Es la dirección encargada de apoyar los procesos documentales y de archivo.

6. JUNTAS LOCALES EJECUTIVAS.

Son órganos permanentes de ejecución de las acciones del Instituto en cada entidad federativa. Se integra por:

- Un Vocal Ejecutivo, quien preside.
- Un Vocal Secretario.
- Un Vocal del Registro Federal de Electores.
- Un Vocal de Organización Electoral.
- Un Vocal de Capacitación Electoral y Educación Cívica.

7. JUNTAS DISTRITALES EJECUTIVAS.

Son órganos permanentes de ejecución de las acciones del Instituto en cada distrito electoral federal. Esta integrado por:

- Un Vocal Ejecutivo, quien preside.
- Un Vocal Secretario.
- Un Vocal del Registro Federal de Electores.
- Un Vocal de Organización Electoral.
- Un Vocal de Capacitación Electoral y Educación Cívica.

8. OFICINAS MUNICIPALES.

El Instituto contará con oficinas municipales en aquellos lugares donde la Junta General Ejecutiva determine su instalación.

9. ORGANOS TEMPORALES.

- Consejos Locales. Son el órgano de vigilancia de las disposiciones electorales y los acuerdos y resoluciones de los órganos electorales superiores en las entidades federativas . Los consejos locales funcionan exclusivamente durante el proceso electoral federal.
- Consejos Distritales. Son los órganos de vigilancia de las disposiciones electorales y los acuerdos y resoluciones de los órganos electorales superiores en los distritos electorales federales. Los consejos distritales sólo funcionan durante el proceso electoral federal.
- Mesas Directivas de Casilla. Son por mandato constitucional, los órganos electorales formados por ciudadanos facultados para recibir la votación y hacer el escrutinio y el cómputo en cada una de las secciones electorales en las que se dividen los 300 distritos uninominales. Cada mesa se integra por un presidente, un secretario y dos escrutadores.

De todos los órganos que componen al Instituto Federal Electoral nos evocaremos sobre la Dirección Ejecutiva del Registro Federal de Electores, por tener la función de generar la credencial para votar con fotografía y la de depurar el padrón.

El Registro Federal de Electores será el tema principal de la siguiente sección.

II.4 EL REGISTRO FEDERAL DE ELECTORES.

El Registro Federal de Electores es un organismo público con carácter de permanente y tiene por objeto cumplir con lo previsto sobre el Padrón Electoral. A demás ésta compuesto por las siguientes secciones:

- Catálogo General de Electores. Se consigna la información básica de los varones y mujeres mexicanos mayores de 18 años, recabada a través de una técnica censal total.

- Padrón Electoral. Son los nombres de los ciudadanos que se encuentran en el Catálogo General de Electores y que además han presentado su solicitud de inscripción al Instituto Federal Electoral.

El Registro Federal de Electores tiene las siguientes atribuciones:

a. Formar el Catálogo General de electores.
b. Aplicar la técnica censal total en el territorio del país para formar el Catálogo General de Electores.
c. Formar el Padrón Electoral.
d. Expedir la Credencial para Votar con Fotografía.
e. Revisar y actualizar anualmente el Padrón Electoral.
f. Establecer con autoridades federales, estatales y municipales la coordinación necesaria, a fin de obtener la información sobre fallecimientos de los ciudadanos, o sobre pérdida, suspensión u obtención de la nacionalidad.
g. Proporcionar a los órganos competentes del Instituto Federal Electoral y a los partidos políticos nacionales, las listas nominales de electores.
h. Formular, con base en los estudios que realice, el proyecto de división del territorio nacional en 300 distritos electorales uninominales, así como el de las cinco circunscripciones plurinominales.
i. Asegurar que las comisiones de vigilancia nacional, estatales y distritales se integren, sesionen y funcionen en los términos previstos por este Código.
j. Llevar los libros de registro y asistencia de los representantes de los partidos políticos a las comisiones de vigilancia.
k. Solicitar a las comisiones de vigilancia los estudios y el desahogo de las consultas sobre los asuntos que estime conveniente dentro de la esfera de su competencia.

Para la incorporación al Padrón Electoral se requiere solicitud individual en que consten firma, huella digital y fotografía del ciudadano.
Tal como lo muestra la figura II.2.

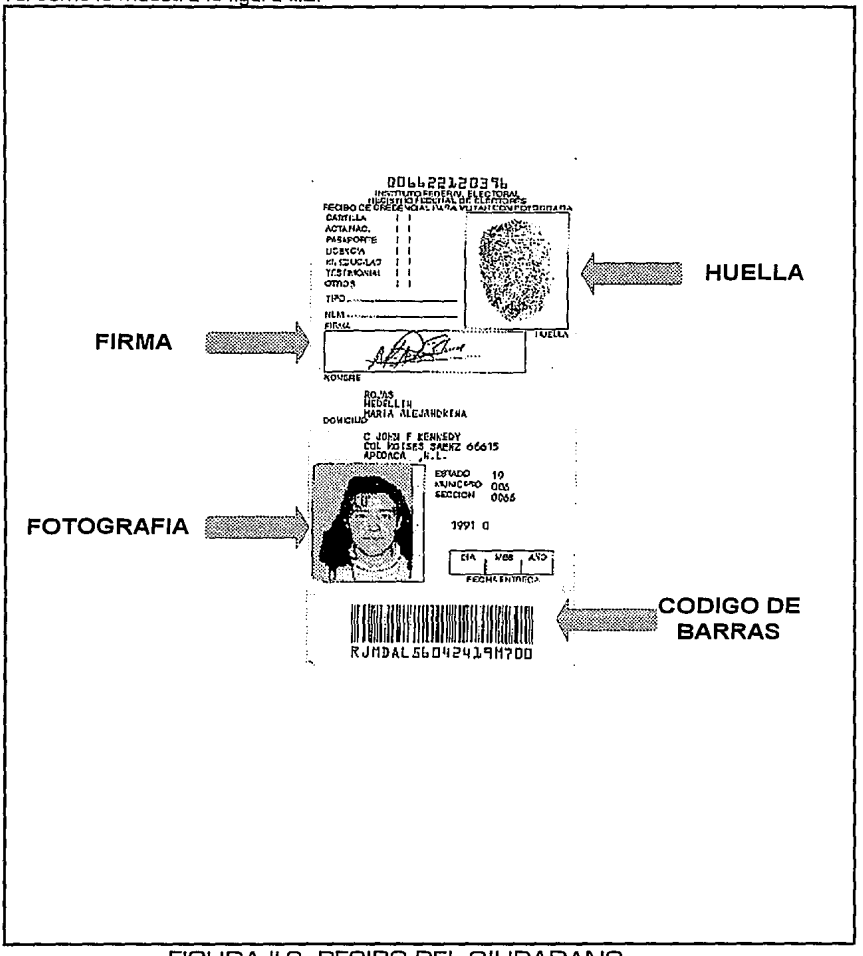


FIGURA II.2. RECIBO DEL CIUDADANO.

Con base en la solicitud del ciudadano, el Registro Federal de Electores expedirá la correspondiente Credencial para Votar con fotografía, como la que se muestra en la figura II.3.

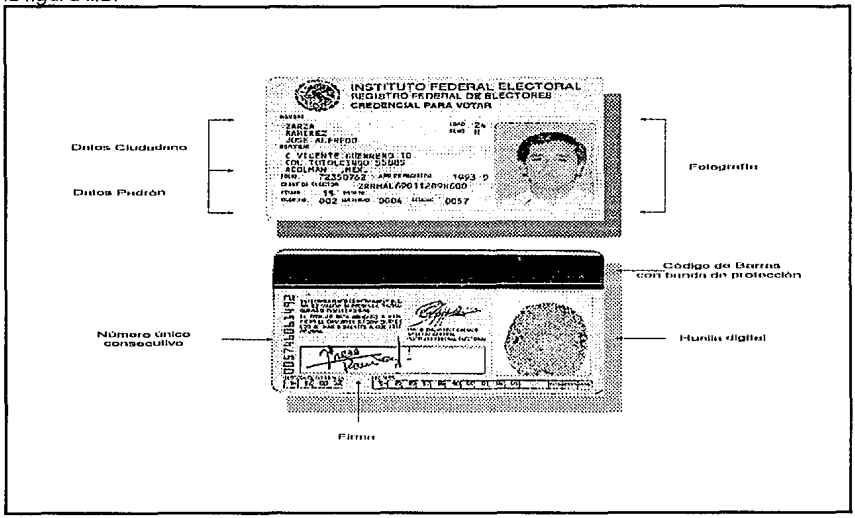


FIGURA II.3 CREDENCIAL PARA VOTAR CON FOTOGRAFIA.

Los ciudadanos tienen la obligación de acudir a las oficinas o módulos que determine el Instituto Federal Electoral, a fin de obtener su Credencial para Votar con fotografía.

El artículo 164 del Código Federal de Instituciones y Procedimientos Electorales dice:

" La credencial para votar deberá contener, cuando menos los siguientes datos de elector.

a. Entidad Federativa, municipio y localidad que corresponden al domicilio

b. Distrito Electoral Uninominal y sección electoral en donde deberá votar.

c. Apellido paterno, apellido materno y nombre completo.

d. Domicilio.

e. Sexo.

f. Edad y año de registro.

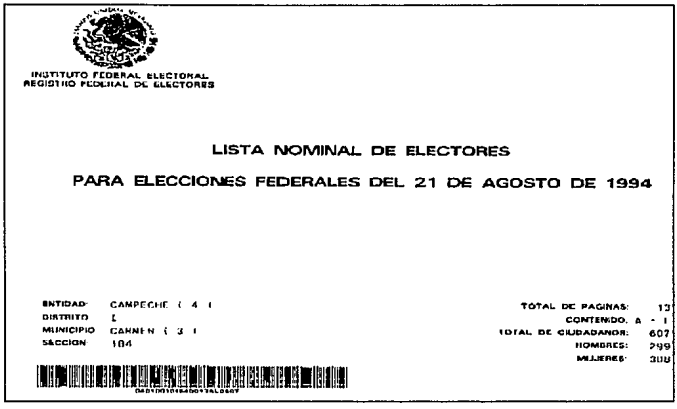
A demás tendrá :

a. Lugar para asentar la firma, huella digital y fotografía de elector.

b. Espacios necesarios para marcar año y elección de que se trate.

c. Firma impresa del Director General del Instituto Federal Electoral."

Una función del Registro Federal de Electores es la de generar las listas nominales de electores del Padrón Electoral con los nombres de aquellos a los que se les haya entregado su Credencial para Votar con Fotografía. Estos listados se formularán por distritos y por secciones. Tal como lo muestra la siguiente figura:



INSTITUTO FEDERAL ELECTORAL
REGISTRO FEDERAL DE ELECTORES

LISTA NOMINAL DE ELECTORES

PARA ELECCIONES FEDERALES DEL 21 DE AGOSTO DE 1994

ENTIDAD: CAMPECHE (4)
DISTRITO I
MUNICIPIO CARMEN (3)
SECCION 184

TOTAL DE PAGINAS: 13
CONTENIDO: 1 - 1
TOTAL DE CIUDADANOS: 607
HOMBRES: 299
MUJERES: 308

INSTITUTO FEDERAL ELECTORAL
Registro Federal de Electores

EJEMPLAR DE MUESTRA
21/AGO/94

Lista Nominal de Electores
para Elecciones Federales del 21 de Agosto de 1994
Entidad: CAMPECHE

FIGURA II.4. LISTADO NOMINAL

Posteriormente se pondrán a disposición de los partidos políticos para su revisión y, en su caso, para que formulen las observaciones que estimen pertinentes. El Registro Federal de Electores proveerá lo necesario para que las listas nominales se pongan en conocimiento de la ciudadanía en cada distrito.

INSTITUTO FEDERAL ELECTORAL

CAPITULO III

INFRAESTRUCTURA COMPUTACIONAL DEL REGISTRO FEDERAL DE ELECTORES

- III.1 INTRODUCCION
- III.2 MODELOS OPERATIVOS DEL REGISTRO FEDERAL DE ELECTORES
- III.3 INFRAESTRUCTURA DE COMPUTO DEL REGISTRO FEDERAL DE ELECTORES A NIVEL NACIONAL
- III.4 DESCRIPCION DEL SISTEMA DE ACTUALIZACION PERMANENTE DEL REGISTRO FEDERAL DE ELECTORES

III. 1 INTRODUCCION.

El Registro Federal de Electores cuenta con oficinas en todo el país dando servicio y apoyo a la ciudadanía para los procesos electorales. Para los fines de esta tesis únicamente tocaremos los temas referentes al Padrón Electoral y sus productos.

Debido a la magnitud del proyecto de ley encomendado al Instituto Federal Electoral y este a su vez al Registro Federal de Electores, de formar el Padrón Electoral con aquellos hombres y mujeres mayores de 18 años y en condiciones de votar. Se conformó una infraestructura computacional distribuida por toda la República Mexicana. Por lo cual se requirió de equipo de cómputo capaz de almacenar gran cantidad de información y un manejador de bases de datos capaz de controlar y administrar dicha información.
Una vez suministrados los insumos de hardware y Software, se requirió de un sistema de cómputo para el levantamiento de la información y otro, para la actualización de dicha información de manera permanente.

En las secciones siguientes entraremos en detalle en cada uno de los puntos aquí mencionados.

III.2 MODELOS OPERATIVOS DEL REGISTRO FEDERAL DE ELECTORES.

La actualización se inicia una vez creado en forma original el Padrón Electoral y se efectúa por:

- Campaña anual.
- Servicio permanente a través de módulos.
- Técnica censal parcial.
- Procesamiento de información de otras autoridades federales y estatales.

Así mismo puede afectar el Padrón alguna de las siguientes causas:

A. El registro Civil, los jueces y la Secretaría de Relaciones Exteriores a través del envío de avisos de defunciones, inhabilitaciones de derechos políticos y adquisiciones/pérdidas de nacionalidad, respectivamente.
B. La dirección de cartografía electoral del RFE cuando envía los cambios a la Cartografía Electoral Federal (reseccionamientos y redistritamientos).
C. El ciudadano se da de alta o actualiza su información a través de los módulos del Registro Federal de Electores.
D. Los partidos políticos y los ciudadanos en el caso de que existan observaciones a los Listados Nominales Preliminares.

Para todas las tareas del RFE existe un flujo de información que corre a partir de las oficinas locales del RFE (módulos) las cuales son el contacto entre los ciudadanos y el RFE. Estos módulos tienen la obligación de entregar y recibir apoyo logístico de las delegaciones estatales para la consolidación de la información. Estas a su vez entregan y reciben información de los centros de cómputo correspondiente. Los centros de cómputo son los únicos órganos responsables de emitir las credenciales y la lista nominal. Lo anterior se ilustra en la figura III.1:

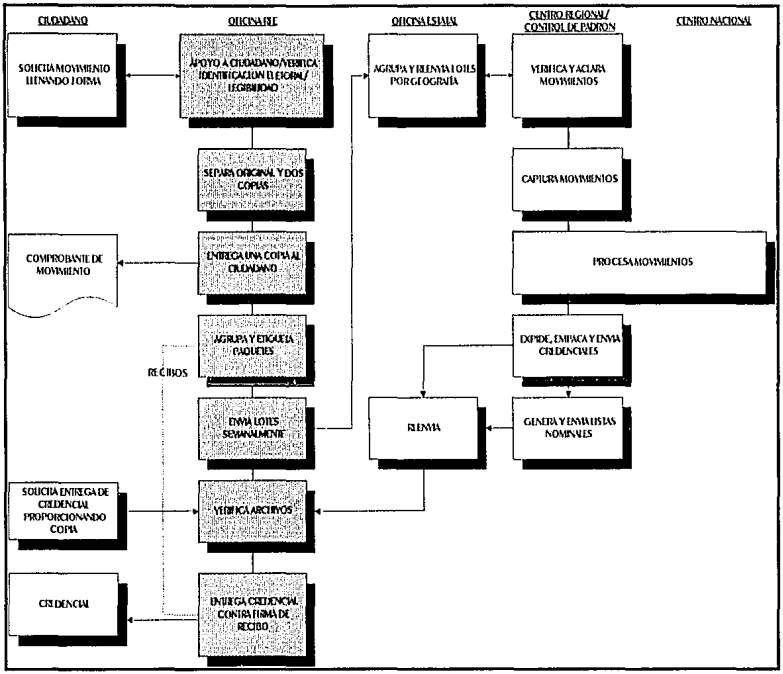


FIGURA III.1 MANTENIMIENTO DEL PADRON.

Como podemos observar en los módulos se da apoyo al ciudadano y la información se maneja manualmente, las funciones de esta oficina son:

- Proporcionar al ciudadano los formatos requeridos y asistirlo en su llenado.
- Verificar la identificación electoral del ciudadano (Estado, Distrito, Municipio y Sección.)
- Agrupar formatos en paquetes por tipo de movimiento.
- Enviar los paquetes semanalmente a la oficina estatal.
- Resguardar y entregar credenciales de elector.
- Resguardar copia de los documentos fuente.
- Recibir y destruir credenciales cuando así se requiera.

Los tipos de movimiento que el módulo maneja son:

- Alta al padrón.
- Baja al padrón.
- Reposición de Credencial.
- Cambio de domicilio.
- Corrección de datos y folio nacional.
- Alta a lista nominal.
- Baja a lista nominal

Las oficinas distritales a través del Vocal del RFE son los responsables de la operación de los módulos de su distrito electoral.

La delegación estatal a través de su Vocal del RFE son el contacto entre las oficinas distritales y los centros de cómputo. Tiene la obligación de facilitar los flujos de información, a través del control de paquetes. A demás es responsable de consolidar y agrupar geográficamente los paquetes de formatos recibidos de las oficinas locales y acumular cifras estadísticas de los movimientos a nivel estado.
Finalmente el flujo de la información será de las delegaciones estatales hacia los centros de cómputo.

El tiempo total del ciclo de producción de transacciones depende de las frecuencia en el envío de paquetes de movimientos. La figura III.2 muestra un ciclo de producción típico.

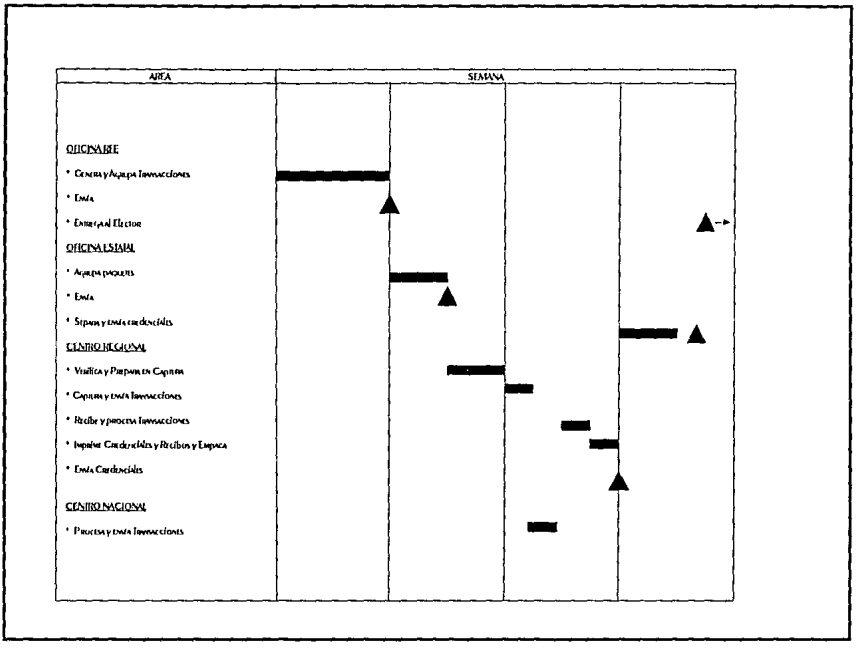


FIGURA III.2 CICLO DE PRODUCCION TIPICO.

Las oficinas locales acumulan y agrupan lotes de transacciones por tipo de movimiento y los envían a la delegación estatal.
La delegación estatal agrupa por geografía y los paquetes los envía al Centro Regional de Cómputo.
El Centro Regional y el Centro Nacional generan un ciclo de captura/actualización, por ciclos iniciados con transacciones de ciudadanos, que incluye la transmisión y recepción de transacciones vía la red de telecomunicaciones.

III.3 INFRAESTRUCTURA DE COMPUTO DEL REGISTRO FEDERAL DE ELECTORES.

El Registro Federal de Electores para cumplir con sus tareas encomendadas cuenta con una red de comunicaciones en todo el territorio nacional, y se encuentra interconectada a un Centro Nacional con 18 Centros Regionales distribuidos en la República Mexicana.

Los Centros Regionales de procesamiento de datos están conectados con el Centro Nacional, a través de líneas de telecomunicaciones dedicadas y conmutadas. La figura III.3 muestra la arquitectura de la red de comunicaciones.

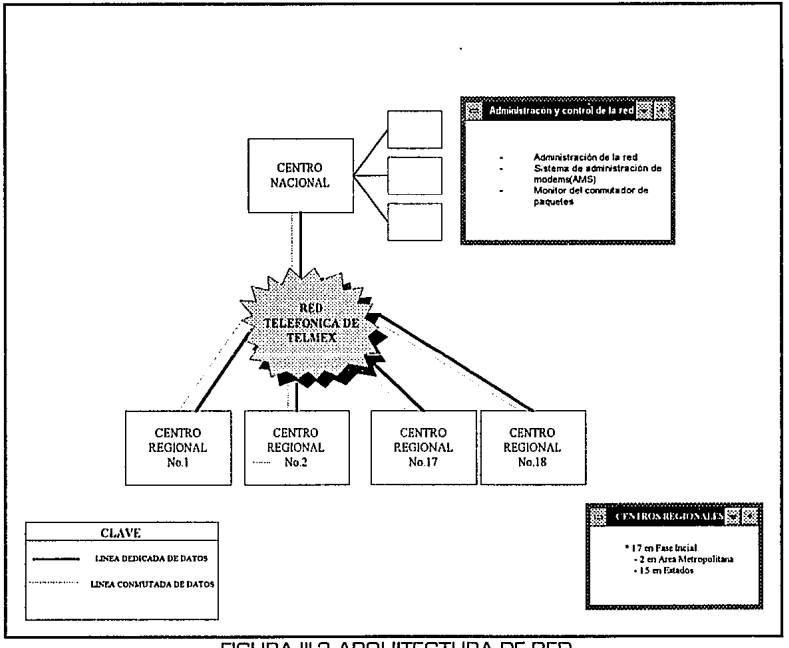


FIGURA III.3 ARQUITECTURA DE RED.

El Centro Nacional cuenta con 6 computadoras RISC/6000 que se encuentran bajo protocolos TCP/IP a 64 Kbps. Y los Centros Regionales cuentan con 1 o 2 computadoras RISC/6000. Las computadoras en el Centro Nacional se encuentran conectadas en Token-ring que a su vez se conecta a los Centros Regionales a través de un conmutador de paquetes X.25 y la conexión esta configurada como una topología estrella, y el conmutador actúa como un concentrador de datos para la computadora del Centro Nacional.

Cada Centro Regional cuenta con un modem, operando a 9.6 kbps, para proporcionar la conectividad con el Centro Nacional, a través de un enlace telefónico dedicado. El modem tiene la capacidad de conectarse por línea conmutada para el acceso al Centro Nacional en caso de falla de la línea dedicada.

La protección en las comunicaciones se proporciona mediante el uso de los Racal Milgo 64P Datacryptors. El Datacryptor 64P es un dispositivo para la encripción del enlace, diseñado para ofrecer comunicaciones seguras para información delicada. Encripta datos utilizando un algoritmo propio de Racal Milgo, que altera completamente los datos transmitidos y los hace totalmente ilegibles, excepto para el receptor indicado.

Cada Centro Regional cuenta con un Datacryptor 64P que se puede cambiar manualmente entre las dos opciones de enlace. EL Centro Nacional tiene un encriptor para cada uno de los Centros Regionales. El encriptor esta parchado manualmente entre las línea dedicada y la conmutada. Además, el Centro Nacional cuenta con cuatro encriptores adicionales como equipo de respaldo de emergencia.

Cada uno de los enlaces de los 18 Centros Regionales está configurado para la operación síncrona de punto a punto, en duplex total. En esta configuración los encriptores del Centro Nacional actuarán como unidad maestra y cada uno de los encriptores regionales lo hará como unidad remota.

Se utilizan claves de verificación separadas para cada una de las redes criptográficas de los 17 Centros Regionales. Este concepto acepta el cambio a modalidad conmutada sin ninguna operación de administración de claves en Centro Regional.

Cada una de las máquinas que se encuentran en el Centro Nacional de Cómputo tienen la siguiente configuración.

- IBM RISC/6000 Mod 590.
- AIX (UNIX) ver 3.2
- 2 Discos Duros de 2 Gb
- 21 Discos de 670 Mb.
- 2 Disco de 857 Mb
- 2 Tarjetas de Memoria de 64 Mb.
- 2 Tarjetas de Memoria de 32 Mb.
- 2 Tarjetas de Memoria de 16 Mb.
- 1 Unidad de cinta de 8 mm. de 2.3 Gb.
- 1 Unidad de cinta de 8 mm. de 5 Gb.
- 1 Unidad de CD-ROM.
- 1 Unidad de diskette de 3.5" de 1.4 Mb.
- 1 Tarjeta X.25
- 1 Tarjeta Token-Ring.
- 1 Impresora IBM 4224 Printer, Modelo 301.
- 2 Impresoras IBM 4234 dot band printer, Modelo 013.
- 1 Impresora IBM 4201 Modelo 3 Proprinter III
- 40 Terminales IBM-3151
- 1 Tarjeta controladora de 64 puertos
- 3 Concentradores de 16 puertos.

Para los Centros Regionales se cuenta con máquinas de las siguientes características.

- IBM RISC/6000 Mod 560.
- AIX (UNIX) ver. 3.2
- 2 Discos Duros de 2 Gb
- 2 Discos de 670 Mb.
- 1 Disco de 857 Mb
- 6 Tarjetas de Memoria de 64 Mb.
- 1 Unidad de cinta de 8 mm. de 2.3 Gb.
- 1 Unidad de cinta de 8 mm. de 5 Gb.
- 1 Unidad de CD-ROM.
- 1 Unidad de diskette de 3.5" de 1.4 Mb.
- 1 Tarjeta X.25
- 1 Tarjeta Token-Ring.
- 1 Impresora IBM 4224 Printer, Modelo 301.
- 2 Impresoras IBM 4234 dot band printer, Modelo 013.
- 40 Terminales IBM-3151
- 1 Tarjeta controladora de 64 puertos
- 3 Concentradores de 16 puertos.

Todas las computadoras RISC/6000 que contienen la base de datos tienen a ORACLE ver 6.0.37 como administrador de la base de datos, por lo tanto, el RFE tiene al Padrón Electoral almacenado en una base de datos relacional "distribuida" en el país.
La lista de los Centros Regionales de Cómputo es la siguiente:

NUM	Centro Regional de Cómputo	Entidad
1	Aguascalientes	Aguascalientes San Luis Potosí Zacatecas
2	Chihuahua	Chihuahua Durango
3	Conurbado	Hidalgo Estado de México
4	Cuernavaca	Guerrero Morelos
5	Culiacan	Baja California Sur Sinaloa
6	DF	DF
7	Guadalajara	Colima Jalisco Nayarit
8	Hermosillo	Baja California Norte Sonora
9	Jalapa	Veracruz
10	Merida	Campeche Quintana Roo Yucatan
11	Monterrey	Coahuila Nuevo León Tamaulipas
12	Morelia	Morelia
13	Oaxaca	Oaxaca
14	Puebla	Puebla
15	Queretaro	Guanajuato Queretaro
16	Toluca	Estado de México
17	Villa Hermosa	Chiapas Tabasco

Todos los programas realizados para el **Programa de Depuración Integral del Padrón y el proyecto de la Credencial para Votar con Fotografía** utilizaron los recursos antes mencionados.

Estructura de los sistemas AIX

La organización lógica de los sistemas de archivos está ligada a la administración que realiza AIX, tal como lo muestra la figura III.4.

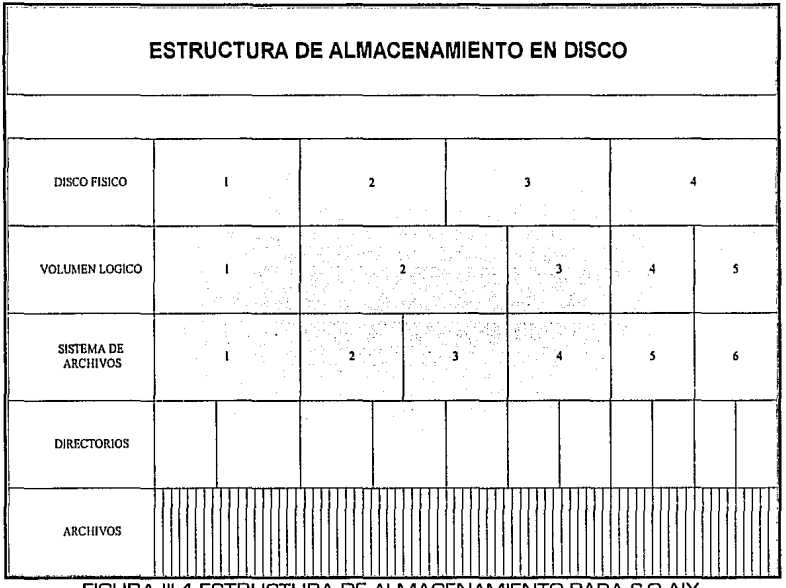


FIGURA III.4 ESTRUCTURA DE ALMACENAMIENTO PARA S.O AIX.

Los **Volúmenes Físicos (PHYSICAL VOLUMES)** no son más que los discos físicos, el cual contiene cierta información.

Así mismo el **grupo de volúmenes lógicos (LOGICAL VOLUME GROUP)**es una identificación para el acceso a uno o algunos discos físicos. Por omisión, al instalar en un equipo, el Sistema Operativo AIX, este se aloja en un GV llamado **rootvg**.

Para el objetivo del RFE cuenta con 2 GV, el standard **rootvg** ya mencionado y otro definido por el administrador del sistema con el nombre de **unovg**. El GV *rootvg* se definió en el disco 0 y alberga al Sistema Operativo (AIX) y a Oracle, por otro lado, *unovg* comprende a los archivos que conforman la base de datos llamada REG o NAL según sea el caso. Las ventajas de esta distribución de información se beneficia cuando existe una corrupción en el SO u Oracle, con lo cual no dañara la información.

Tal como lo muestra la figura III.5.

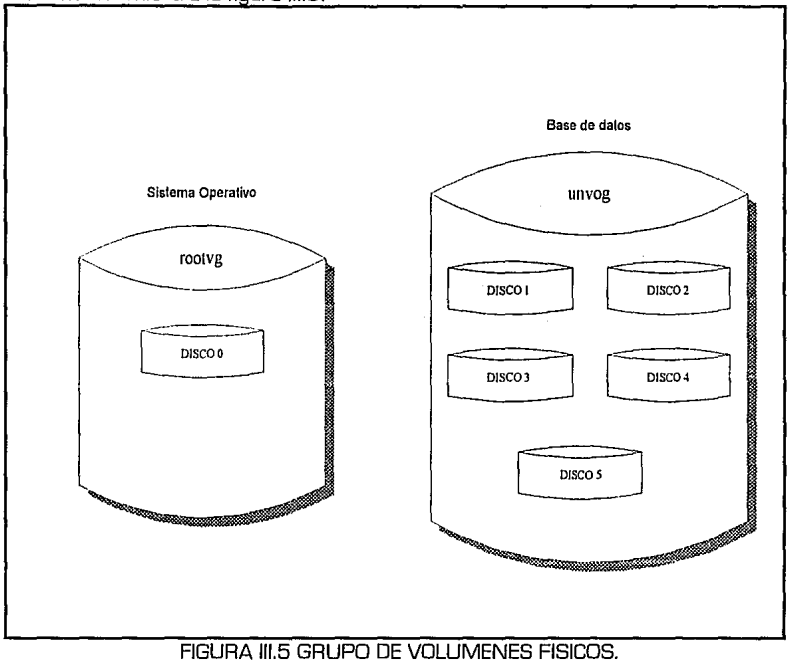
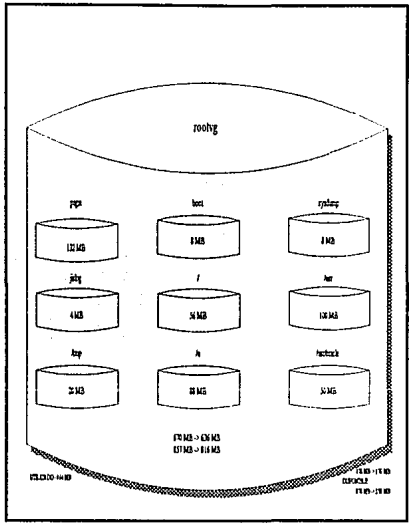


FIGURA III.5 GRUPO DE VOLUMENES FISICOS.

El **volumen lógico (LOGICAL VOLUME)** es la partición de un Volumen de Grupo (GV) montado sobre éste último y nos permite contar con unidades de menor tamaño y ordenadas para el almacenamiento de la información.

Para el RFE están definidos en VG *rootvg* los Volúmenes Lógicos (LV) hd00, hd01, ... ,hd08 que son utilizados por el Sistema Operativo y también se encuentra en este VG el VL lv00 para almacenar al Oracle. Tal como lo muestra la figura III.6.



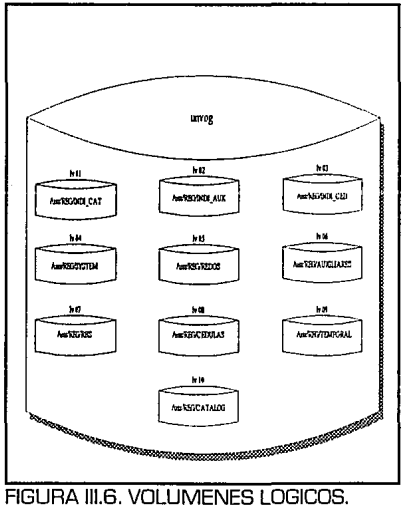


FIGURA III.6. VOLUMENES LOGICOS.

En *unovg* están definidos los VL lv01,lv02,...,lv10 para almacenar los archivos de la base de datos REG.
Durante la definición de los VL de *unovg* se determinó el disco físico que debe soportarlo, así como el lugar (al centro, parte media o parte externa del disco.) donde debe iniciarse la definición mientras su tamaño lo permita a fin de utilizar las áreas más rápidas de los discos para los archivos de mayor demanda y nivelar el acceso de las cabezas repartiendo en diferentes discos físicos los datos e índices. El tamaño de cada VL lo determina el tamaño del TABLESPACE de Oracle que lo soporta.

El **sistema de archivos (FILE SYSTEM)** es la caracterización de un directorio para convertirse en una unidad montable y desmontable que soporta a un grupo de información. Para cada uno de los Sistemas de Archivos está definido sobre el VL que le corresponde. Todos los SA relacionados con la base de datos REG están montados en directorios del tipo

/usr/REG/XXXXXXXXX

donde XXXXXXXXX = nombre del tablespace de Oracle que soporta.
Con esto se logra un estandar de nomenclatura para la creación que facilita las labores de mantenimiento y respaldo.
Un **SA** nunca debe estar utilizando un 100 % de su capacidad, debe existir cuando menos 4K libres para poder generar una ampliación al tamaño.

Estructura y Configuración de Oracle.

Debido a la amplia aceptación de las bases de datos relacionales por ofrecer beneficios tales como:

- Fácil acceso a todos los datos.
- Flexibilidad en el modelado de los datos.
- Reduce el almacenamiento de datos y redundancia.
- Independencia de almacenamiento físico y diseño lógico de datos.
- Un lenguaje de manipulación de datos de alto nivel (SQL).

El Registro Federal de Electores eligió a ORACLE como la herramienta para la administración de la base de datos. Por ofrecer portabilidad, compatibilidad, y conectividad, lo que la hace una herramienta poderosa.
El corazón de ORACLE RDBMS (Relational Data Base Management System) es SQL (Structure Query Lenguaje) un lenguaje parecido al idioma Inglés que es usado por la mayoría de las base de datos activas.SQL es bastante simple para que los usuarios principiantes puedan accesar fácil y rápidamente los datos de la base.

Las sentencias de SQL están divididas en cuatro categorías:

- Consultas
- Sentencias de manipulación de datos. (DML)
- Sentencias de definición de datos. (DDL)
- Sentencias de control de datos. (DCL)

Además de SQL, ORACLE RDBMS incluye diversas utilerías en las que se encuentran:

- SQL*DBA.
- SQL*FORMS.
- SQL*REPORT.
- SQL*MENU.
- Pro*C, Pro*Cobol, Pro*Fortran.
- SQL*Loader.
- SQL*Net
- CRT.
- Import / Export.

Un sistema de base de datos en Oracle puede ser configurada para proveer una amplia gama de servicios. En todas las configuraciones, un usuario puede accesar la base de datos (los archivos de base de datos DATABASE FILES y los archivos redo log) vía una instancia de Oracle (El software que manipula la base de datos). La siguiente figura muestra un sistema de base de datos multi-usuario.

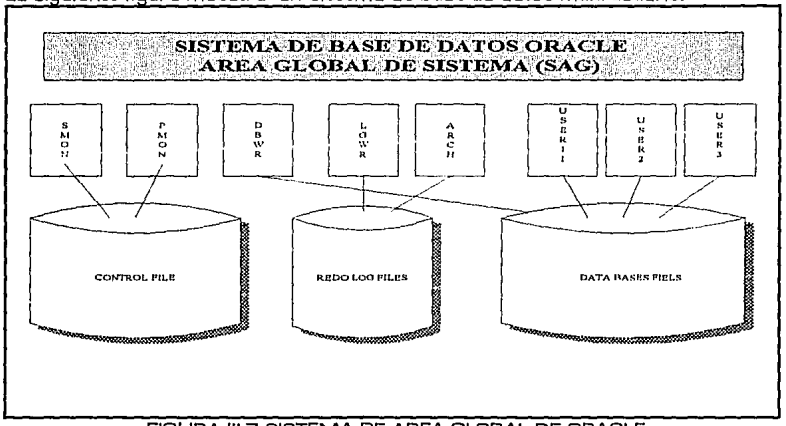


FIGURA III.7 SISTEMA DE AREA GLOBAL DE ORACLE.

Oracle almacena todo tipo de información en estructuras físicas, tales como archivos del sistema operativo en un medio de hardware (por ejemplo cinta magnética, disco flexible o disco duro). Una base de datos requiere para su operación diversos archivos, en los que se encuentra:

DATABASE FILE (Archivos de la base de datos). Una base de datos en Oracle requiere de varios *database files*. Estos archivos contienen todos los datos de la base. Las características de los *database files* son:

- Un archivo esta asociado con una y solo una base de datos.
- Uno o más archivos físicos forman una unidad lógica de almacenamiento de la base de datos (TABLESPACE).
- Todos los archivos de la base de datos son accesibles cuando se activa una instancia.
- El rendimiento de la base de datos es mejor si cada archivo, esta localizado en unidades contiguas del espacio físico del disco.
- Una vez creado un archivo de la base de datos no debe cambiar en tamaño.

CONTROL FILE (Archivos de Control). Siempre que una base de datos es abierta, uno o más archivos de control son accesados. Un archivo de control es un archivo binario y es asociado solo con una base de datos.
Un archivo de control contiene información acerca de la base de datos que se requiere para que una instancia pueda accesar la información. Los archivos de control contienen la siguiente información:

- Nombre de los archivos físicos de la base de datos y de los redo log.
- Fecha de la creación de la base de datos.
- Nombre de la base de datos.

REDO LOG FILES (Archivos de bitácora). Los redo logs es un conjunto de archivos del sistema operativo, externos a la base de datos, que almacenan cambios realizados durante las transacciones.

Cuando la base de datos se abre para el acceso, también entran en operación una serie de programas en background, que se encargan de administrar la base de datos, como el control de acceso, control de la información, configuración, operación de la bitácora, etc. Los programas encargados de estas tareas son los siguientes:

DBWR (Database Writer). El propósito de DBWR es escribir bloques modificados del cache buffer a la base de datos. Bloques que son escritos en un orden para mantener la integridad de la base de datos. El algoritmo de escritura asegura la integridad y la certeza de que siempre habrá buffers para escritura.

LGWR (Log Writer). Escribe las bitácoras de las transacciones (redo log) al disco. Las bitácoras son generadas en un buffer de la memoria global del sistema (System Global Area).

SMON (System Monitor). El propósito es ejecutar el proceso de recuperación de la instancia. Esto es usado para levantar la instancia y realizar la recuperación adecuada, para una falla del CPU en sistema de disco compartido. SMON es también responsable para limpiar segmentos temporales que no han sido usados por largo tiempo y para recuperar transacciones muertas durante las caídas del sistema.

PMON (Process Monitor). El propósito es ejecutar el proceso de recuperación cuando algún proceso de usuario falla. PMON es responsable de limpiar el buffer-cache y liberar recursos que el proceso estaba usando. La función es la misma que SMON solo que este se aplica a nivel proceso de usuario.

ARCH (Archiver). Es el proceso encargado de realizar copias en línea de los redo logs hacia cinta cuando estos se encuentran llenos. ARCH se encuentra activo, solo cuando los redo logs se encuentran en modo ARCHIVELOG y los archivos se encuentran disponibles.

Oracle utiliza una área de memoria física de la máquina para la operación de los procesos de background y los procesos de usuario, la **SGA** (System Global Area). Es la estructura de memoria que utilizan los programas en background para su propia operación.

La estructura bajo la cual almacena la información Oracle se muestra en la siguiente figura:

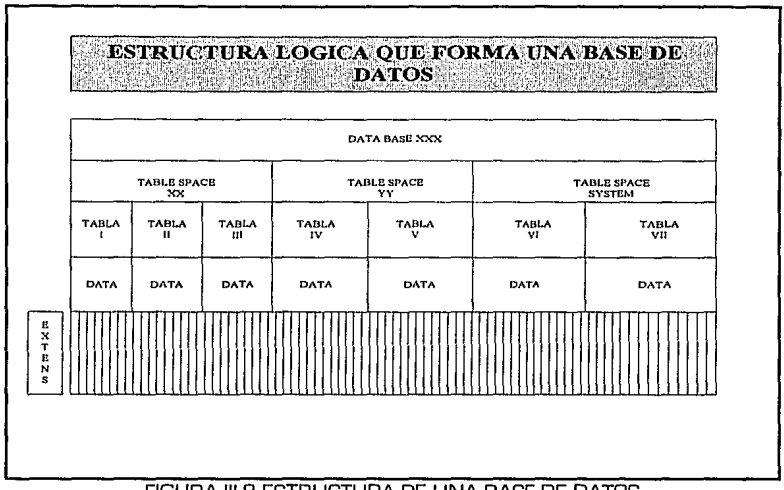


FIGURA III.8 ESTRUCTURA DE UNA BASE DE DATOS.

Un **tablespace** es un espacio formateado para soportar información de tablas de Oracle. Está conformado por DATAFILES que corresponden a archivos de AIX. Ningún DATAFILE es mayor de 500 Mb, en el caso de que el TABLESPACE sea mayor a esta cantidad se divide con más de un DATAFILE .

El nombre de los archivos de un TABLESPACE está definido con letras minúsculas y con la siguiente estructura:

<nombre>X.dbf

Donde:

<nombre>	corresponde al mismo del TABLESPACE que define.
X	Consecutivo
.dbf	extensión que debe ser constante (data base file).

Los valores de omisión durante la definición de algún TABLESPACE (DEFAULT STORAGE) son debidamente definidos, dado que los mismos son asignados a cada

tabla definida por omisión, En el caso de que un TABLESPACE soporte más de una tabla los valores son bajos, por ejemplo:

INITIAL 1M NEXT 1M PCTINCREASE 0
Para impedir que la definición de tablas con valores de omisión genere tamaños desproporcionados y no deseados.

Una **tabla** es la definición física y estructurada del espacio de disco que soportará cierta información. Cuando se crea una tabla, en el caso de que no se especifiquen los valores, estos son asignados por omisión y además si no se asigna un TABLESPACE la tabla se genera en el TABLESPACE SYSTEM o TEMPORAL.

Un Registro es la unidad lógica de información de una tabla. Un registro contiene la unidad de información que conforma la funcionalidad de una tabla.
Un campo es la unidad lógica bajo la cual se forman los registros. Los campos en Oracle pueden ser definidos con los siguientes tipos:

- CHAR. Este tipo de datos son usados para almacenar caracteres alfanuméricos. Los datos son almacenados en ASCII o EBCDIC, dependiendo de su computadora.
- NUMBER. Es usado para almacenar números (fijos y de punto flotante). Números de precisión de 38 dígitos. Números tan largos como 9.99×10^{124} pueden ser almacenados.
- DATE. Funciona para almacenar datos tipo fecha en una longitud fija de 7 bytes.
- LONG. Es usado para almacenar cadenas de caracteres que contienen no más de 65535 caracteres.
- RAW. Este tipo de datos es usado para almacenar información orientada al byte. Este tipo de datos guardan inoformación binaria.

La creación de una base de datos de Oracle típica, es como se muestra a continuación.

```
REM ==========================================
REM     Creación de la base de datos UNAM
REM             con 5 archivos en Redos
REM             con 128 MAXDATAFILES
REM             con ARCHIVELOG desactivado
REM     crea un rollback segment temporal llamado rbs_t
REM ==========================================
spool /tmp/reorden/crea_reg
startup nomount pfile=/usr/oracle/dbs/init.crea
connect internal
create database UNAM
logfile   '/usr/IFENAL/UNAM/log1NAL.rdo' size 1M,
          '/usr/UNAM/REDOS/log2.NAL.rdo' size 1M,
          '/usr/UNAM/REDOS/log3.NAL.rdo' size 1M,
          '/usr/UNAM/REDOS/log4.NAL.rdo' size 1M,
datafile  '/usr/UNAM/SYSTEM/system.dbf' size 1M,
maxdatafiles 128
noarchivelog
/
alter database mount
/
create rollback segment rbs_t
/
disconnect
shutdown
startup pfile=/usr/oracle/dbs/initNAL.rbs
connect internal
@/usr/oracle/rdbms/admin/catalog
@/usr/oracle/rdbms/admin/expvew
/
disconnect
shutdown
```

Los parámetros normales bajo los cuales opera una base de datos son los siguientes:

```
#
# $Header: /rcs/6.0/rdbms/olb/RCS/init.ora, v 614.1
# 1991/09/25 13:39:21
#
#
#
db_name = UNAM
db_file_multiblock_read_count = 8
db_block_buffers = 200
db_block_write_batch = 5
db_block_size = 2048
ddl_locks = 100
dml_locks = 100
log_allocation = 14000
log_checkpoint_interval = 10000
processes = 20
db_files = 62
log_files = 32
language=American_America.US7ASCII
nls_sort=False
rollback_segments=rbs_A, rbs_B
rollback_segments=rbs_t
transactions=36
transactions_per_rollback_segment=12
dc_rollback_segments=64
```

El software de administración de base de datos es el corazón con el cual opera el Padrón Electoral y todos los procesos electorales giran alrededor del mismo.

El Padrón electoral se encuentra almacenado en tablas de Oracle distribuidas en los Centros Regionales de Cómputo, con la siguiente descripción.

ALFA_CLAVE_ELECTORAL	NOT NULL	CHAR(6)
FECHA_NACIMIENTO	NOT NULL	NUMBER(6)
LUGAR_NACIMIENTO	NOT NULL	NUMBER(2)
SEXO	NOT NULL	CHAR(1)
DIGIT_VERIFICADOR	NOT NULL	CHAR(1)
CLAVE_HOMONIMIA	NOT NULL	NUMBER(2)
NUMERO_CREDENCIAL	NOT NULL	NUMBER(1)
ESTADO	NOT NULL	NUMBER(2)
DISTRITO	NOT NULL	NUMBER(2)
MUNICIPIO	NOT NULL	NUMBER(3)
SECCION	NOT NULL	NUMBER(3)
LOCALIDAD	NOT NULL	NUMBER(4)
MANZANA	NOT NULL	NUMBER(2)
CALLE	NOT NULL	CHAR(32)
NUM_EXTERIOR	NOT NULL	CHAR(8)
NUM_INTERIOR	NOT NULL	CHAR(8)
NOMBRE	NOT NULL	CHAR(32)
APELLIDO_PATERNO		CHAR(32)
APELLIDO_MATERNO		CHAR(32)
COLONIA		CHAR(32)
CODIGO_POSTAL		NUMBER(5)
INSACULADO		CHAR(1)
EN_LISTA_NOMINAL		CHAR(1)
NUM_CREDENCIAL		NUMBER(1)

La clave única bajo la cual se almacenan los ciudadanos en la base de datos es, la clave electoral, la cual se encuentra formada por los primeros 6 campos de la tabla del Padrón Electoral. Esta clave se encuentra impresa en la credencial para votar con fotografía.

En la actualidad se cuenta con 47.5 millones de ciudadanos empadronados en la República Mexicana.
Una vez creado el Padrón por vez primera, es necesario un sistema para mantenerlo en constante actualización, en el tema siguiente abordaremos el funcionamiento del Sistema de Actualización Permanente (SAP) del Padrón Electoral.

III.4 DESCRIPCION DEL SISTEMA DE ACTUALIZACION PERMANENTE DEL REGISTRO FEDERAL DE ELECTORES.

El sistema del Registro Federal de Electores esta dividido en 3 partes:

- **Creación del catalogo y Padrón Electoral.** Crear una base de datos de electores (Padrón) a través de la captura de solicitudes de empadronamiento de ciudadanos recabada durante el censo total 1991.

- **Actualización Permanente.** El objetivo principal de este módulo es la de imprimir credenciales de elector y listas nominales de electores y actualizar el Padrón Electoral y las listas nominales con los movimientos de altas, cambios de domicilio, corrección de datos, bajas, reposición de credenciales y observaciones a la lista nominal.

- **Apoyo a procesos electorales.** Programas encargados de apoyo a los procesos electorales como la insaculación de ciudadanos para integrar las mesas directivas de casilla y de integración de información de resultados de las elecciones federales.

El módulo principal bajo el cual opera el Registro Federal de Electores es el Sistema de Actualización Permanente.

El Sistema de Actualización Permanente (SAP) tiene como objetivo mantener los productos electorales e imprimir credenciales y listas nominales.
Los movimientos de actualización se capturan en los centros regionales para mantener al día el padrón electoral y los listados nominales.

Los tipos de movimiento que el SAP acepta son los siguientes:

- ALTAS AL PADRON. Es el movimiento para ingresar un ciudadano al Padrón Electoral, lo cual posteriormente requiere de la impresión de una credencial.

- SOLICITUD DE BAJA DEL PADRON. Es el proceso de dar de baja un registro del Padrón Electoral por causas justificadas por la ley.

- SOLICITUD DE REPOSICION DE CREDENCIAL. Este tipo de movimiento surge cada vez que un ciudadano pide la reposición de la credencial en los módulos del RFE con alguna causa justificada.

- CAMBIO DE DOMICILIO/CARTOGRAFIA. Es el movimiento que permite realizar la corrección al domicilio y a la cartografía del ciudadano. Requiere una reimpresión de credencial.

- CORRECCION DE DATOS. Este tipo de proceso permite realizar correcciones a la información general del ciudadano como el nombre, la fecha de nacimiento, el sexo, etc. Lo cual implica una reimpresión de credencial.

- ALTAS A LISTA NOMINAL. Este movimiento implica el actualizar la información para que aparezca en el listado nominal.

- BAJAS A LISTA NOMINAL. Se actualiza la información del ciudadano para que no aparezca en el listado nominal.

- ALTA FORZADA. Si resulta que se trata de dar de alta a un ciudadano homónimo, el SAP rechaza dicho movimiento, entonces para forzar al sistema a darlo de alta, se procesa como una alta forzada y se incrementa la clave homonimia que conforma la clave electoral del ciudadano.

Los procesos de captura de movimientos incluyen la captura, verificación, confirmación y supervisión, cada uno con un usuario.
El proceso en Centro Regional se muestra en la siguiente figura:

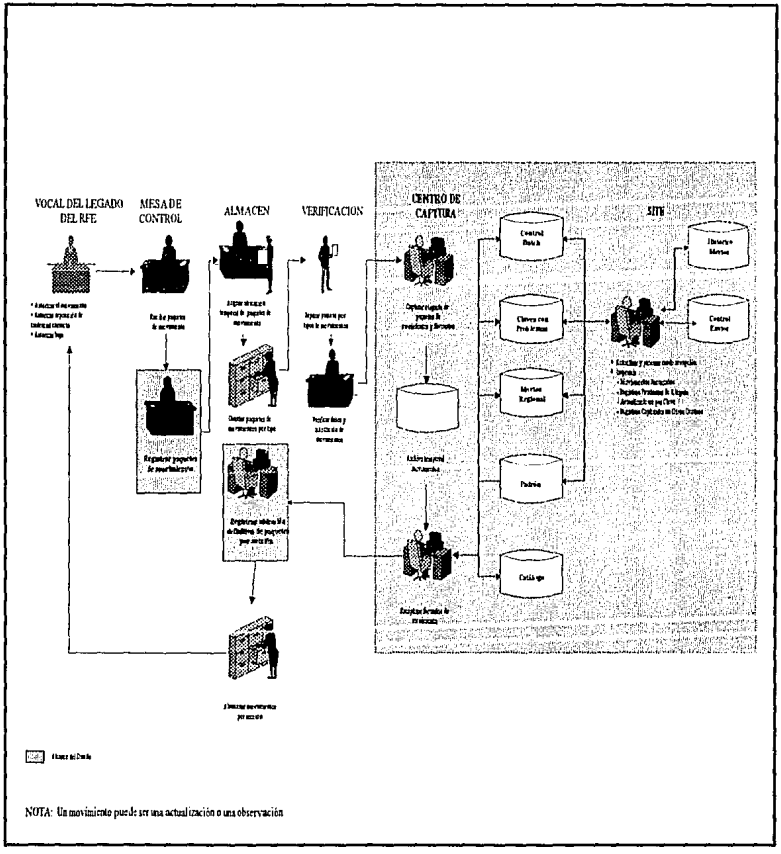


FIGURA III.9 MODELO OPERATIVO DE UN CENTRO REGIONAL

CAPTURA. El capturista captura la información de los formatos determinados por tipo de movimiento.
VERIFICACION. Realiza la segunda captura de los formatos. El verificador valida la información digitada por el capturista por medio de una recaptura o verificación.
CONFIRMACION. El confirmador realiza un cierre de operaciones de movimientos que así lo requieran.
SUPERVISION. Autoriza electrónicamente los formatos descartados en la captura, verificación y actualización al Padrón.

Mediante una variedad de procesos, el sistema de actualización a nivel regional le permite al operador ejecutar el envío y recepción de archivos del Centro Nacional, la actualización del Padrón, la generación de reportes y el mantenimiento de la base de datos.

- INICIO DE OPERACIONES DIARIAS (IOD): Este proceso se debe correr al comienzo de las operaciones diarias para actualizar la fecha y hora del sistema y para generar el reporte de Resultado de respaldos para garantizar la efectividad de los respaldos generados el día anterior.
- CIERRE DE OPERACIONES DIARIAS (COD): Este proceso se debe correr al final de las operaciones diarias para ejecutar los siguientes procesos:
    1. Recepción de archivos enviados por Nacional.
    2. Reporte de registros pendientes de llegada.
    3. Actualización del Padrón y listas nominales.
    4. Reporte de movimientos pendientes de proceso.
    5. Envío de archivos al Nacional.
    6. Reporte de movimientos rechazados.
    7. Reporte de candidatos de baja del Padrón.
    8. Reporte de candidatos de reposición de credencial.
    9. Reporte de duplicados en alta al Padrón.
    10. Reporte de duplicados en cambios de datos.
    11. Reporte de registros capturados en otros centros.
- CIERRE DE OPERACIONES SEMANALES (COS): Para ser ejecutado al final de la semana de trabajo, tiene las siguientes funciones:
    1. Genera el reporte de avance de paquetes, que monitorea el progreso de los paquetes desde que se registran en AVC, mientras se capturan y se envían al Centro Nacional, hasta que se regresan de Nacional y se actualiza el Padrón.
    2. Analiza la consistencia en la cartografía a nivel sección entre el Centro Regional y el Centro Nacional y presenta los resultados de este análisis en el Reporte de Secciones Inconsistentes.

- CIERRE DE OPERACIONES MENSUAL (COM): Produce lo siguiente:
  1. Genera tres reportes (Actualizaciones por clave, bajas por antigüedad y estadístico de movimientos).
  2. Depura el Padrón y los archivos de trabajo basado en su caducidad (120 años para los ciudadanos en el Padrón y un parámetro designado por el centro para los archivos de trabajo).

La actualización a nivel nacional se realiza mediante una variedad de procesos, el sistema de actualización a nivel nacional permite al operador ejecutar el envío y recepción de archivos de los centros regionales, la actualización del padrón, la generación de reportes y el mantenimiento de la base de datos.

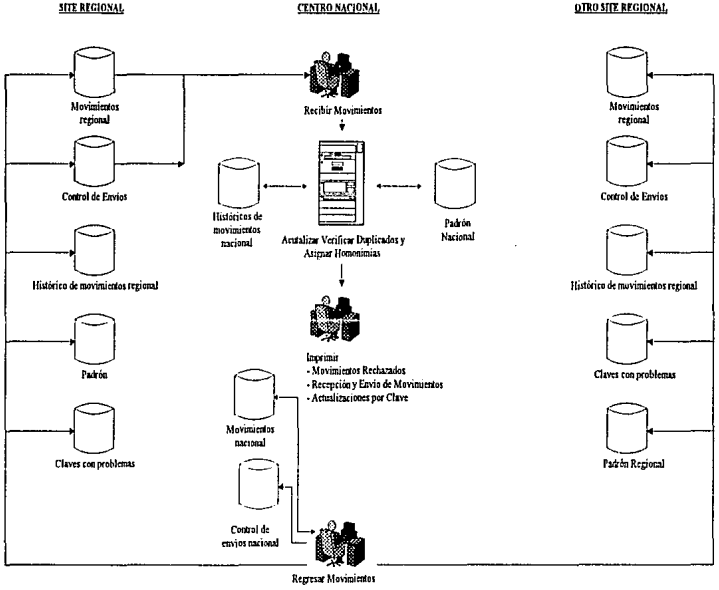


FIGURA III.10 MODELO OPERATIVO DEL CENTRO NACIONAL.

El operador ejecuta las siguientes tareas:

- INICIO DE OPERACIONES DIARIAS (IOD): Este proceso se debe correr al comienzo de las operaciones diarias para:
    1. Actualizar la fecha y hora del sistema.
    2. Recibir archivos de movimientos de los centros regionales.
    3. Actualizar el Padrón.
    4. Enviar registros procesados a los centros regionales.
    5. Imprimir el reporte de recepción y envío de movimientos.
- CONTINGENCIA DEL INICIO DE OPERACIONES DIARIAS: Permite ejecutar varios procesos de IOD de manera opcional para casos de emergencia.

- CIERRE DE OPERACIONES DIARIAS (COD): Este proceso se debe correr al final de las operaciones diarias para la generación de tres reportes:
    1. Movimientos rechazados durante los procesos de actualización.
    2. Actualizaciones por clave.
    3. Estadístico de movimientos.

- CIERRE DE OPERACIONES SEMANAL (COS): Tiene las siguientes funciones primordiales.
    1. Depurara registros de la base de datos que han cubierto su antigüedad establecida, lo cual hace innecesario su almacenamiento.
    2. Analiza la consistencia en la cartografía a nivel sección entre el Centro Regional y el Centro Nacional, presenta los resultados de este análisis en el reporte de secciones inconsistentes.
- CIERRE DE OPERACIONES MENSUAL (COM): Principalmente realiza los siguientes respaldos:
    1. Respaldo de archivos auxiliares.
    2. Respaldo de archivos de trabajo.
    3. Respaldo de padrones.

INSTITUTO FEDERAL ELECTORAL

CAPITULO IV

HERRAMIENTAS Y METODOLOGIA DEL CICLO DE DESARROLLO DE LOS SISTEMAS DEL REGISTRO FEDERAL DE ELECTORES.

- IV.1 INTRODUCCION
- IV.2 HERRAMIENTAS
- IV.3 METODOLOGIA

IV.1 INTRODUCCION.

El programa de Depuración Integral del Padrón Electoral y la Nueva Credencial para Votar con fotografía, que se implementó a partir del acuerdo del Consejo General del Instituto Federal Electoral (IFE), y publicado en el diario oficial de la federación el día 14 de Julio de 1992, es una respuesta a la demanda ciudadana y de los partidos políticos por perfeccionar los instrumentos electorales de nuestro país.

Realizando un gran esfuerzo por parte de las instituciones electorales y también por la participación ciudadana se logró consolidar los objetivos planteados.

Todos los sistemas se desarrollaron dentro de la infraestructura computacional del Registro Federal de Electores utilizando los recursos del mismo.

El Registro Federal de Electores, como se describió en capítulos anteriores, actualmente cuenta con equipo IBM RISC/6000 , AIX ver. 3.2 y ORACLE ver 6.0.36 como equipo de cómputo, sistema operativo y manejador de base de datos respectivamente. La infraestructura esta organizada en un Centro Nacional de Cómputo y 17 Centros Regionales de Cómputo, distribuidos en la República, todos conectados al Centro Nacional formando una topología estrella.

En este capítulo tocaremos los temas de herramientas y metodología utilizada para el desarrollo de las aplicaciones y la puesta en marcha de los mismos.

Debido a que el manejador de base de datos es ORACLE resulta consecuente que las aplicaciones se desarrollaron utilizando las herramientas que este nos proporciona.

IV.2 HERRAMIENTAS

Las herramientas utilizadas para los proyectos desarrollados por el Registro Federal de Electores, son los productos que acompañan a ORACLE, estos son:

- SQL*FORMS
- SQL*REPORT
- SQL*PLUS
- SQL*MENU
- PRO*C

IV.2.1 ORACLE

Oracle es un Sistema de Administración de Base de Datos Relacionales (RDBMS), Los sistemas relaciónales ofrecen beneficios tales como:

- Fácil acceso a todos los datos.
- Flexibilidad en modelado de datos.
- Reduce el espacio de datos y la redundancia.
- Proporciona independencia del almacenamiento físico y diseño lógico.
- Ofrece un lenguaje de alto nivel para la manipulación de datos (SQL).

El corazón de ORACLE RDBMS es el lenguaje SQL. Es un lenguaje parecido al idioma inglés y es usado para la mayor parte de las actividades de la base de datos.
SQL fue desarrollado y definido por IBM, posteriormente refinado por el American National Standard Institute (ANSI) como el lenguaje standard para sistemas de administración de base de datos relacional. Sentencias de SQL pueden ser usadas en algunos productos de ORACLE (SQL*FORMS , SQL*REPORT, SQL*PLUS, etc.) en utilerías como SQL*DBA y programas escritos por usuario (PRO*C,PRO*COBOL o PRO*FORTRAN)
Para mayor información referirse a Capítulo III inciso 3.

IV.2.2 SQL*PLUS

Es la herramienta que nos permite realizar y ejecutar sentencias de SQL y se encuentra dividido dentro de 4 categorías.

- Consultas.
- Sentencias de manipulación de datos (DML)
- Sentencias de definición de datos (DDL).
- Sentencias de control de datos (DCL).

Consultas (QUERIES): Son sentencias para recuperar datos, en cualquier combinación, expresión u orden. Los Queries usualmente inician con la palabra reservada SELECT, seguida por el dato deseado, y las tablas o vistas conteniendo la fuente del dato, como en:

```
SELECT ALFA_CLAVE_ELECTORAL,FECHA_NACIMIENTO
FROM EMP;
```

Sentencias de Manipulación de Datos (DML): Estas sentencias son usadas para cambiar datos en uno de los tres caminos siguientes:

- INSERT. Inserta nuevos renglones a la tabla.
- UPDATE. Actualiza valores de columnas en renglones existentes.
- DELETE. Elimina renglones de las tablas.

Algunos ejemplos de DML son los siguientes:

```
INSERT INTO PADRON ('ZARZA', 'RAMIREZ', 'JOSE
ALFREDO', 'MORELOS #23', 'SAN MIGUEL TOTOLCINGO',
55885 ) ;

DELETE FROM PADRON WHERE
      NOMBRE = 'JOSE ALFREDO' AND
      APELLIDO_PATERNO = 'ZARZA' AND
      APELLIDO_MATERNO = 'RAMIREZ';
```

Sentencias de Definición de Datos (DDL). Son utilizados para dar mantenimiento a los objetos en la base de datos. Este incluye todos los comandos CREATE (CREATE TABLE, CREATE VIEW) y su correspondiente ALTER y DROP. Por ejemplo:

```
CREATE TABLE PADRON (
        NOMBRE                  CHAR(15),
        APELLIDO_PATERNO        CHAR(20),
        APELLIDO_MATERNO        CHAR(20),
        CALLE                   CHAR(20),
        COLONIA                 CHAR(30),
        CP                      NUMBER(5)
);

DROP TABLE PADRON;
```

Sentencias de Control de Datos (DCL). Son usadas para controlar dos tipos de acceso.

- Acceso a la base de datos por ejemplo GRANT y CONNECT.
- Acceso a datos de la base de datos usando GRANT SELECT o REVOKE DELETE.

sentencias DCL conceden a un usuario dar permiso de ver, cambiar y usar datos de sus tablas a otros usuarios. (GRANT SELECT .. WITH GRANT OPTION). El control de datos también incluye el comando AUDIT.

IV.2.3 SQL*FORMS

Es una herramienta de propósito general para desarrollar y ejecutar aplicaciones interactivas basadas en formas (pantallas de captura). El componente de diseño de esta herramienta es especialmente diseñado para programadores y desarrolladores, y esta permite ejecutar las siguientes tareas:

- Define transacciones que combinan datos de múltiples tablas en la única forma.
- Creación rápida de formas usando un conjunto rico de comandos.
- Optimiza todos los aspectos de definición de aplicaciones.

Aún la mas simple aplicación en SQL*FORMS con cierta definición por default, contiene la habilidad de validar tipos de datos, ejecutar la navegación a través de la forma, y accesar la base de datos.
Una forma esta realizada utilizando objetos adicionales. Esos objetos ligan la forma a elementos de la base de datos, tal como columnas y tablas y provee control sobre el flujo de la información.

Bloques. Compone una sección o una sub-sección de la forma, y funciona como la base de interacción de la base de datos.

Campos. Representan columnas o áreas completas de datos y describe como deberían desplegarse los datos validados y como interactua el operador con la base de datos.

Páginas. La colección de información de despliegue, tal como texto y gráficas. Todos los campos son desplegados en una sola página.

Triggers. Un conjunto de comandos asociados con ciertos puntos, como cuando ocurre que una tecla fue presionada por el operador.

Procedimientos. Un conjunto de comandos que pueden ser invocados a través de un trigger o procedimiento a nivel de forma.

Cualquier forma contiene un bloque, una página, y uno o más campos. Cada objeto en una forma tiene un conjunto de atributos o características, las cuales proveen información acerca de los objetos. Los campos tienen nombre, así ellos pueden ser referenciados en cálculos y en la manipulación de la base de datos.
Una pantalla típica de SQL*FORMS es como se muestra a continuación.

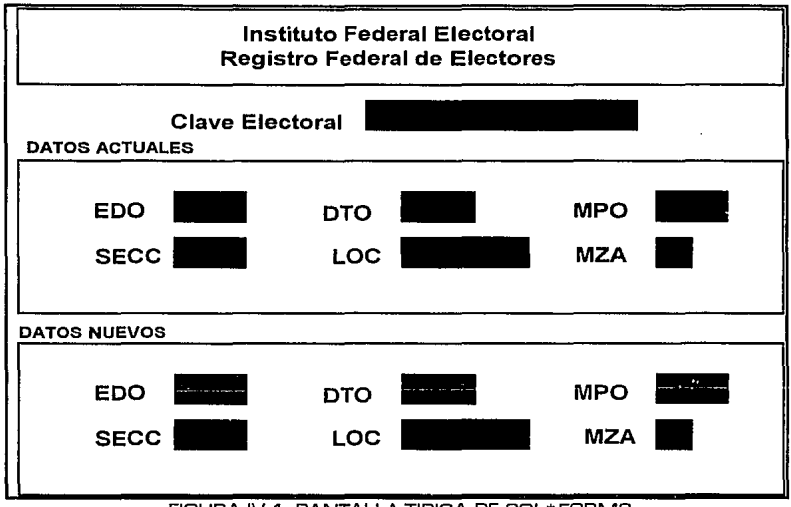


FIGURA IV.1. PANTALLA TIPICA DE SQL*FORMS

IV.2.4 PRO*C

Es un precompilador que permite incluir sentencias de SQL en programas fuente de alto nivel en lenguaje C, da poder y flexibilidad de SQL en programas de aplicación general.

Las características especiales de PRO*C son las siguientes:

- Sigue los estándares de ANSI para incluir sentencias SQL.
- Toma ventajas de SQL dinámico y ventajas de técnicas de programación para aceptar o construir cualquier SQL válido al tiempo de corrida.
- Diseño y desarrollo de aplicaciones de alto nivel.
- Conversión automática entre tipos de datos internos de ORACLE y tipos de datos en C.
- Acceso concurrente a bases de datos ORACLE en múltiples nodos utilizando SQL*NET usa arreglos como entradas y variables de salida.
- Secciones condicionales de código en el programa, de esta forma permite correr en diferentes plataformas.
- Acceso directo con SQL*Forms vía un " user exit " escrito en lenguaje C .

Como por ejemplo:

```
/*************************************************
    Programa: ChecaTablaPadron
    Objetivo: Checar la existencia de la tabla Padron
    Entrada: Nada
    Salida:      Imprime mensaje de existencia o de no existencia de
                 tabla Padron.
    Realizado por: Jose Alfredo Zarza Ramirez.
*************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR pwd[10], usr[10]
EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;

main (){
    pwd.len = sprintf (pwd.arr,"PASSWORD");
    usr.len = sprintf (usr.arr,"USER");

    /* ____ Conexion a la base de datos.
    EXEC SQL CONNECT :usr IDENTIFIED BY :pwd;
    if (SQL_CODE != SQL_OK){
        printf ("Error en la conexion %s\n", SQL_ERROR);
        exit (-1);
    }

    EXEC SQL SELECT *
    FROM PADRON;
    if (SQL_CODE == SQL_NOT_FOUND){
        printf ("LA TABLA PADRON NO EXISTE FAVOR DE VERIFICAR \n");
        exit (0);
    }

    printf ("LA TABLA PADRON EXISTE FAVOR DE CONTINUAR\n");
    exit(1);

}/*Fin del programa */
```

IV.2.5 SQL*REPORT

Es un lenguaje procedural para la creación de reportes accesando bases de datos relacionales a través de ORACLE. SQL*REPORT permite:

- Ejecutar múltiples sentencias de SQL usando construcciones procedurales.
- Mezclar texto e información de la base de datos en el mismo reporte.
- Utilizar variables para retener valores.
- Acentar valores en cualquier lugar del reporte.
- Utilizar definiciones de macros.
- Usar operaciones aritméticas.
- Usar operaciones lógicas.

SQL*REPORT esta compuesto de dos programas.

1. Generador de reportes (RPT).
   Selecciona o manipula información de bases de datos usando sentencias de SQL.
   Procesa directivas procedurales, texto, y extrae datos para producir una salida que pueda ser formateada por un formateador de texto.

2. Formateado de reporte (RPF)
   Lee un archivo que contiene texto e información de comandos de formato para producir el reporte final. Asegura la portabilidad de los reportes.

Secuencia de procesamiento:

- RPT lee el archivo de control del reporte, buscando sentencias del generador.
- RPT se conecta a la base de datos apropiada para obtener información del reporte.
- Consultas de SQL en el programa del reporte (.RPT) obtienen los datos deseados.
- El archivo intermedio (.RPF) contiene:
  - -Texto suministrado para el control del archivo.
  - - Información obtenida de la base de datos.
  - - Comandos de formato vía el archivo de control.
- RPF procesa el archivo intermedio (.RPF)

Como por ejemplo:

```
.REM  Ejemplo de un programa que genera una lista de empleados.
#DT 1 5 75 #
#DT 2  5 15  20 25  30 40  45 54  59 68 #
#DT 3  50 68 #
.rem Declaración de variables usadas en el reporte.
.DECLARE NAME                 A9
.DECLARE ENUMBER    9999
.DECLARE POSITION   A10
.DECLARE HIRE                 A9
.rem
.DECLARE SALARY     $99,999.99
.DECLARE TOTSAL     $99,999.99
.rem definición de la sentencia del select
.DEFINE SELEMP
        SELECT ename, empno, job, hiredate, sal
        INTO name, enumber, position, hire, salary
        FROM emp
        ORDER BY ename
.rem define la cabecera.
.DEFINE  REPHEAD
        #T 1
        #CEN  LISTA DE EMPLEADOS
        #N
        #CEN COORDINACION DE INFORMATICA
        #TE
        .rem Determinación de nombres y justificaciones de columnas.
        #S 3
        #T 2
        #UL NAME #              #NC
        #UL NUMBER #            #NC
        #UL HIRED #             #NC
        #R      #UL SALARY #    #NC
        .rem inicializa el acumulador.
        .SET TOTSAL 0
        .rem llama a el cuerpo del reporte
        .REPBODY
.DEFINE REPBODY
        .PRINT NAME
        #NC
        .PRINT ENUMBER
        #NC
        .PRINT POSITION
        #NC
        .PRINT HIRE
        #NC
        .PRINT SALARY
        #NC
        .ADD TOTSAL TOTSAL SALARY
```

```
.DEFINE REPFOOT
        #TE
        #T 3
        #S 3
        #R #UL TOTAL SALARY # #NC
        .PRINT TOTSAL
        #NC
        #TE
.REPORT SELEMP REPBODY REPHEAD REPFOOT
.rem fin del reporte.
```

IV.2.6 SQL*MENU

Es una herramienta productiva para proveer una interface a través de menús para correr múltiples herramientas de procesamiento de datos. SQL*MENU tiene tres categorías de usuarios.

- *Operadores* , usan SQL*MENU para correr aplicaciones de software.
- *Diseñadores*, usan SQL*MENU para correr aplicaciones utilizando menús.
- *Administradores de SQL*MENU*, dan mantenimiento a SQL*MENU y permisos a operadores y diseñadores.

SQL*MENU pueden ser utilizados con software del ORACLE o con otros productos de software que corran el sistema operativo de la computadora. Cualquier comando válido del sistema operativo puede ser incluido en la aplicación. Las características de SQL*MENU son las siguientes:

- Despliega solo las opciones disponibles a un operador.
- Reduce la cantidad de conocimiento técnico a sus operadores.
- Da estructura a una aplicación, haciéndolo fácil de aprender, usar y mantener.
- Provee seguridad por restringir el número de comandos que un operador puede dar.
- Reduce costos de soporte para una organización.

SQL*MENU puede generar menús dinámicos los cuales contienen cambios dependiendo de los privilegios de acceso de un operador.
Por ejemplo, un operador puede tener acceso solo a tres opciones del menú, mientras que otro operador puede disponer de todas las opciones en el mismo menú.
Las autorizaciones para usar las opciones de menú son dadas por varias categorías de privilegios, llamadas *roles*. Un role accesa a cierto conjunto de opciones del menú. Cada usuario puede ser miembro de uno o varios roles.

La siguiente figura muestra una pantalla típica de un menú.

Fig IV.2. Pantalla común de un menú.

En conclusión; las aplicaciones desarrolladas para el proyecto del Registro Federal de Electores utilizaron las herramientas antes mencionadas para:

1. Pantallas de Captura. (SQL*FORMS y PRO*C).
2. Reportes. (SQL*REPORT).
3. Menús. (SQL*MENU y Lenguaje C).
4. Procesamiento de información. (PRO*C).

y como todas estas herramientas tienen como base SQL, por consecuencia también se utilizó SQL*PLUS.

IV.3 METODOLOGIA DE DESARROLLO DE LOS SISTEMAS.

La siguiente metodología fue aplicada para el desarrollo de todos los sistemas del Registro Federal de Electores.
Todos los programas se desarrollaron a través del ciclo de vida clásico para la ingeniería del software. El cual se ilustra en la siguiente figura:

CICLO DE VIDA CLASICO DE DISEÑO DE SOFTWARE

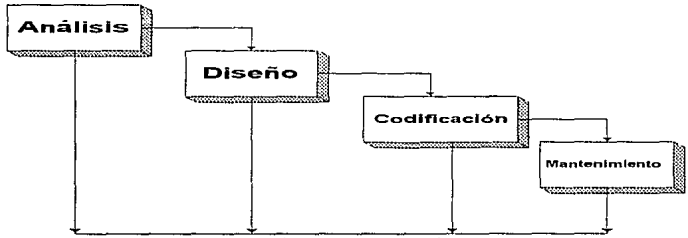


Figura IV.3 Ciclo de vida del software.

IV.3.1 ANALISIS DE LOS REQUERIMIENTOS DE SOFTWARE .

Las metodologías de análisis de requerimientos combinan procedimientos sistemáticos con una notación única para analizar los dominios de información y funcional de un problema de software, suministra un conjunto de heurísticas para subdividir el problema y define una forma de representación para las visiones lógicas y físicas. En esencia, los métodos de análisis de requerimientos del software, facilitan al ingeniero de software aplicar principios de análisis fundamentales, dentro del contexto de un método bien definido.
Para el Registro Federal de Electores los datos se dividen en tres atributos: flujo de datos, contenido de los datos y estructura de los datos.

Las metodologías de análisis de requerimientos facilitan al analista la aplicación de los principios fundamentales del análisis de una manera sistemática. La metodología utilizada es el método de **análisis orientado al flujo de datos.** En éste método, la información se transforma como un *flujo* a través de un sistema basado en computadora. El sistema acepta entrada de distintas formas; aplica un hardware, software y elementos humanos para transformar la entrada en salida; y produce una salida en distintas formas.
Una técnica para representar el flujo de información a través del sistema basado en computadora se ilustra en la siguiente figura

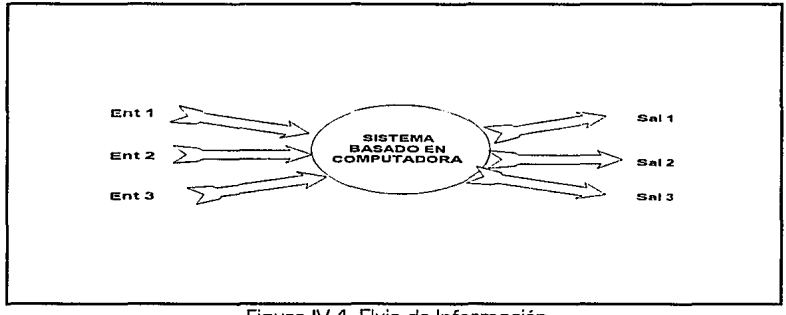


Figura IV.4. Flujo de Información.

La función global de sistema se representa como una transformación sencilla de la información, representada en la figura como una *burbuja*. Una o más entradas,

representadas como flechas con etiqueta, conducen la transformación para producir la información de salida. Puede observarse que el modelo puede aplicarse a todo el sistema o sólo a un elemento de software. La forma básica de un diagrama de flujo de datos (DFD) se ilustra en la siguiente gráfica.

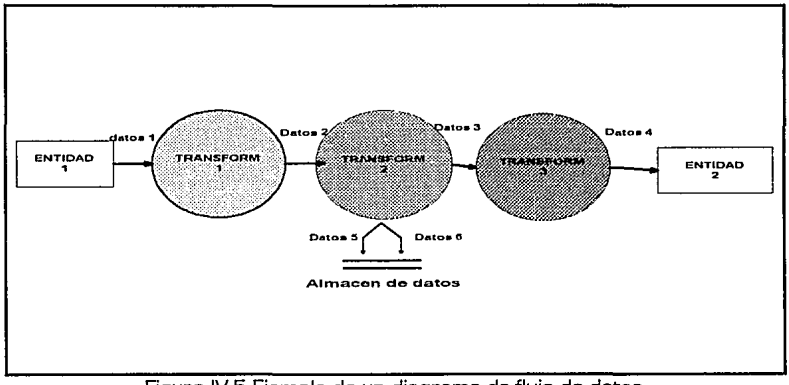


Figura IV.5 Ejemplo de un diagrama de flujo de datos.

El DFD puede usarse para representar un sistema o software a cualquier nivel de abstracción. De echo, los DFD pueden particionarse en niveles que representan flujo incremental de información y detalle funcional. La simbología de un DFD se ilustra en la siguiente figura.

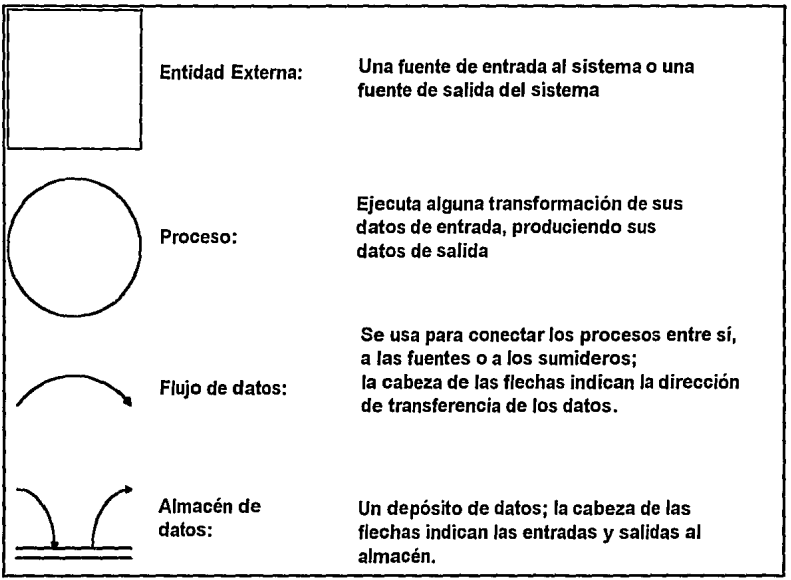


Figura IV.6 Simbología de los Diagramas de Flujo de Datos.

Se utiliza un rectángulo para representar una entidad externa, esto es un elemento del sistema (hardware, persona, etc) u otro sistema que produce información que ha de ser transformada por el software o que recibe información o que recibe información producida por el software. Un círculo representa un proceso o transformación que se aplica a los datos y que los cambia de alguna forma. Una flecha representa uno o más elementos de datos. Todas las flechas de un diagrama de flujo de datos deben estar etiquetadas. Una doble línea representa un almacenamiento de datos. La excepcional simplicidad de la simbología de los Diagramas de Flujo de Datos es una de las razones por las que las **técnicas de análisis orientada al flujo de datos** son ampliamente usadas.

Un análisis del dominio de la información puede ser incompleto si sólo se considera el flujo de datos. Cada flecha de un diagrama de flujo de datos representa uno o más elementos de información.
El diccionario de datos contiene las definiciones de todos los datos mencionados en el DFD, los datos compuestos se definen en términos de sus componentes; los datos elementales se definen en términos del significado de cada uno de los valores que puede asumir. Por tanto el diccionario de datos esta compuesto de definiciones del flujo de datos, archivos y datos usados en los procesos. La notación de un diccionario de datos, facilita la representación de datos compuestos en una de las tres formas fundamentales en que puede ser construído: 1) como una secuencia de elementos de datos, 2) como una selección entre un conjunto de elementos de datos, y 3) como una agrupación repetida de elementos de datos. Cada entrada de un elemento de datos que se representa como parte de una secuencia, selección o repetición, puede a su vez ser otro elemento de datos compuestos, el cual necesita un posterior refinamiento dentro del diccionario.

Para grandes sistemas basados en computadora, el diccionario de datos crece rápidamente en tamaño y complejidad. De hecho es extremadamente difícil mantener un diccionario de datos manualmente. Por esta razón están disponibles varios sistemas de diccionario de datos automatizados. Además, las nuevas estaciones de trabajo de ingeniería del software ayudado por computadora soportan la generación automática del DFD y el acoplamiento y gestión directa de los diccionarios de datos asociados.

IV.3.2 ANALISIS DE LOS REQUERIMIENTOS DE LA BASE DE DATOS.

El análisis de requerimientos para una base de datos incorpora las mismas tareas que el análisis de requerimientos del software. Es esencial la identificación de las funciones e interfaces, se requiere la estructura y asociatividad de la información y un documento formal de los requerimientos.
Se define una base de datos como: una colección de información organizada de forma que facilita el acceso, análisis y creación de informes. Una base de datos contiene entidades de información que están relacionadas vía organización y asociación. La arquitectura física de una base de datos depende de la configuración del hardware residente. Sin embargo tanto el esquema como la organización deben adecuarse para satisfacer los requerimientos funcionales y de comportamiento para el acceso al análisis y creación de informes.

IV.3.3 NORMALIZACION

En el dominio del análisis de información se requiere la definición del contenido de los datos. Cada elemento de información es listado y finalmente organizado en estructuras de archivo lógicas. Sin embargo, frecuentemente es posible simplificar la organización de tales archivos, facilitando así el diseño de una base de datos. Una técnica llamada normalización se utiliza para simplificar la estructura lógica de los datos.

El proceso de normalización identifica los datos redundantes que pueden existir en la estructura lógica, determina claves únicas necesarias para el acceso a los elementos de datos y ayuda a establecer las relaciones necesarias entre los elementos de datos. Pueden aplicase tres niveles de normalización, llamados formas normales. Para ilustrar el proceso de normalización, podemos observar la figura IV.7. Describe cada uno de los elementos de datos, que es parte del dominio de la información del sistema de procesamiento de pedidos.

	DATOS-PEDIDO
	No. pedido
No. pedido	**Fecha pedido**
Fecha pedido	**Nombre cliente**
Nombre cliente	**Dirección**
Dirección	**No. cargo**
No. cargo	
Lista pedido*	**LISTA-PEDIDO**
No. software	
Título	**No. pedido**
Autor	**No. software**
Cantidad	**Título**
Precio unitario	**Autor**
Precio Total	**Cantidad**
	Precio unitario
	Precio total
(a)	**(b)**
Estructura no normalizada	**Estructuras normalizadas**

Figura IV.7 Normalización

Para normalizar esta lista se separan todos los grupos de datos, de forma que ningún archivo tenga grupos repetidos (en este caso, la lista de pedidos de software, lista de pedidos puede ocurrir varias veces si se solicitan diferentes productos). Este nivel de simplificación se llama primera forma normal (1 NF). Representamos esta estructura de datos 1NF de la siguiente forma:

LISTA-PEDIDO (n.º- pedido, n.º-software, título, autor , cantidad,
precio-unidad, precio-total).
DATOS-PEDIDO (n.º-pedido, fecha-pedido, nombre-cliente,
dirección, nº-cargo).

Pueden efectuarse otras normalizaciones identificando los elementos de datos clave y los que no son clave. Un ejemplo clave se utiliza para identificar uno o más elementos que no son clave.

Por ejemplo, no-programa, es un dato clave, identifica únicamente a:

- título
- autor
- precio-unidad

En este ejemplo nº.programa y nº.pedido son datos clave. Para las relaciones anteriores cantidad es dependiente completa y funcionalmente debido a que puede obtenerse sólo si se conocen los elementos clave para la relación (nº. pedido y nº. programa). El elemento título que no es clave no es dependiente funcionalmente, debido a que necesitamos sólo conocer una clave, nº.programa, para acceder a él.

Para conocer la segunda forma normal (2NF), deben reorganizarse las relaciones de forma que ningún dato que no sea clave sea completa y funcionalmente dependiente. Los siguientes ejemplos son de relaciones 2NF:

DATOS-PEDIDO (nº.pedido, fecha-pedido, nombre-cliente,
dirección, nº.cargo)
LISTA-PEDIDO (nº·pedido, nº.software, cantidad, precio-total)
INFO-SOFTWARE (nº. software, título, autor, precio-unidad)

La simplificación de la tercera forma normal (3NF) puede realizarse, si todas las condiciones para la 2NF se cumplen y ningún elemento que no sea clave, puede derivarse de una combinación de otros elementos que no son clave en ninguna de las relaciones. Por ejemplo, precio-total puede calcularse como suma de los productos de precio-unidad y cantidad. Por tanto, no es necesario mantenerla en las relaciones. Ejemplos de relaciones en 3NF son:

DATOS-PEDIDO (nº.pedido, fecha-pedido, nombre-cliente,
dirección, nº.cargo)
LISTA-PEDIDO (nº.pedido, nº.software, cantidad)
INFO-SOFTWARE (nº.software, título, autor, precio-unidad)

El proceso de normalización simplifica las estructuras de datos y quita las redundancias y elementos de datos innecesarios de una base de datos.

IV.3.4 DIAGRAMA DE FLUJO DE DATOS.

La normalización de los archivos de datos y el análisis subsecuente de la capacidad de una base de datos deben acoplarse con una técnica efectiva para representar las relaciones entre los objetos de datos. El diagrama de relaciones de las entidades y subsecuentes extensiones se utilizan frecuentemente como herramienta gráfica para representar las relaciones entre los datos.

La notación básica para los diagramas de relaciones entre entidades se muestra en la figura IV.8. Cada caja representa un objeto de datos. Las líneas de conexión indican una asociación entre datos y objetos que se especifica mediante la simbología mostrada. La barra vertical que aparece sobre una línea de conexión puede verse como un "1". El círculo implica un "O".

Un diagrama de relaciones entre entidades sencillo se muestra en la figura IV.9. Se puede observar que el cliente esta asociado con un número, nombre y dirección de cliente y uno o más pedidos. Cada pedido puede estar asociado con un número de pedido, fecha y número del cargo, así como con uno o más números de programas. Los pedidos pueden también asociarse con números de clientes. Cada número de programa está asociado con un título, un autor y un precio por unidad.

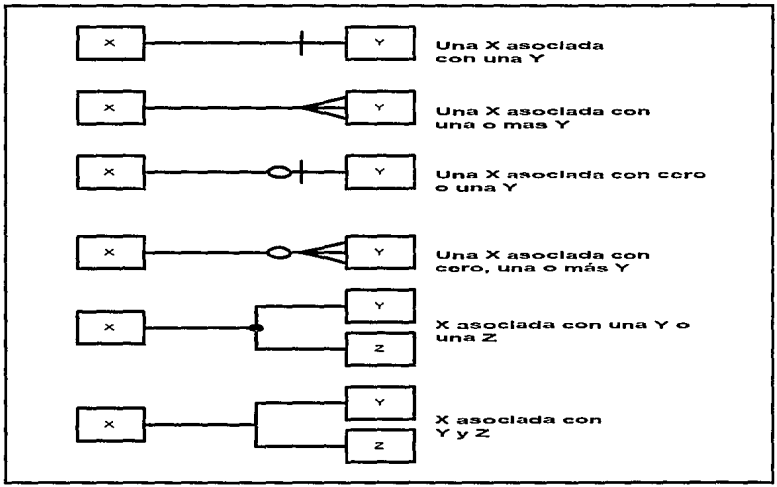


Figura IV.9 Notaciones para relaciones entre entidades.

IV.3.5 DISEÑO DEL SOFTWARE.

El diseño es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería. Puede ser definido como:

> ***El proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física.***

Una vez que se han establecido los requerimientos del software, la fase de desarrollo comprende tres pasos distintos, diseño, generación de código y prueba. Cada paso transforma la información de forma que finalmente se obtiene un software para computadoras validado.

El flujo de información durante la fase de desarrollo se muestra en la figura IV.10. Los requerimientos del programa manifestados mediante el dominio de la información, requerimientos funcionales y de comportamiento alimenta el paso del diseño.

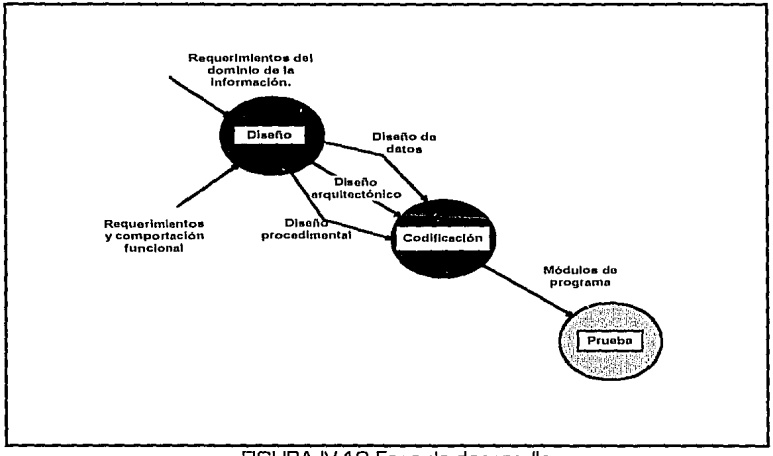
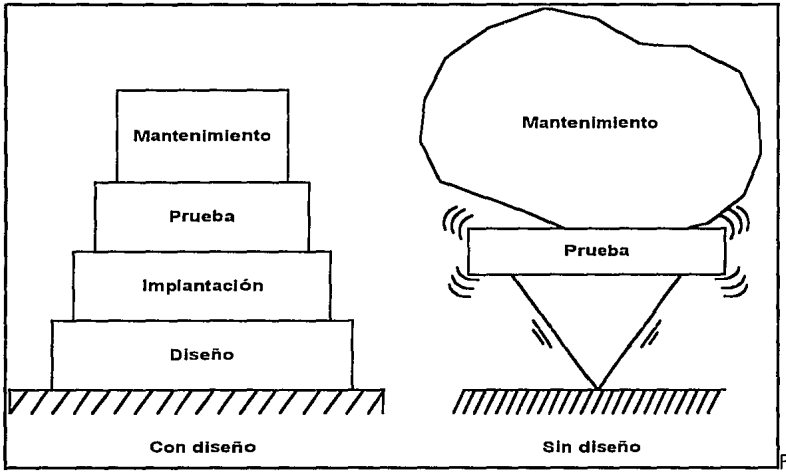


FIGURA IV.10 Fase de desarrollo.

El diseño del software es realmente un proceso multipaso que se enfoca sobre tres atributos distintos del programa:

- estructura de datos.
- arquitectura del software.
- detalle procedimental.

El proceso de diseño traduce los requerimientos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación. El diseño es la única forma mediante la que podemos traducir con precisión los requerimientos del cliente en un producto o sistema acabado. La figura IV.11 muestra una visión. El diseño de programas sirve como base para todos los pasos de desarrollo y fase de mantenimiento que siguen.



FI

GURA IV.11. Importancia del diseño.

IV.3.6 DISEÑO DE DATOS.

La estructura de datos es una representación de la relación lógica entre elementos individuales de datos. Debido a que la estructura de la información afectará invariablemente al diseño procedimental final, la estructura de datos es tan importante como la estructura de programas en la representación de la arquitectura del software.
El diseño de datos es la primera (y de alguna forma podríamos decir la más importante) de las tres actividades de diseño realizadas durante la ingeniería del software. El impacto de la estructura de datos sobre la estructura de programa y la complejidad procedimental, hace que el diseño de datos tenga una profunda influencia en la calidad del software.

El proceso de diseño de datos es resumido por Wasserman:

> La actividad primaria durante el diseño de datos es seleccionar las representaciones lógicas de los objetos de datos (estructuras de datos), identificadas durante las fases de definición y especificación de requerimientos. El proceso de selección puede implicar el análisis algorítmico de estructuras alternativas, en orden a determinar el diseño más eficiente o puede simplemente implicar el uso de un conjunto de módulos (un paquete), que suministra las operaciones deseadas sobre alguna representación de un objeto.
>
> Una actividad relativamente importante durante el diseño, es identificar los módulos de programa que deben operar directamente sobre las estructuras de datos lógicas. De esta forma, puede restringirse el ámbito del efecto de las decisiones del diseño de datos individuales.

El diseño toma el siguiente conjunto de principios para la especificación de los datos.

1.- *Los métodos de análisis sistemático aplicados al software deben también aplicarse a los datos.* También se debe desarrollar y revisar las representaciones de flujo y estructura de datos, considerarse organizaciones de datos alternativas y evaluarse el impacto del diseño de datos sobre el diseño del software.

2.- *Deben identificarse todas las estructuras de datos y operaciones que han de ejecutarse sobre cada una de ellas.* El diseño de una estructura de datos eficiente debe tener en cuenta las operaciones que han de ejecutarse sobre dicha estructura de datos.

3.- *Debe establecerse y usarse un diccionario de datos para definir el diseño de los datos y el software.* Un diccionario de datos identifica explícitamente las relaciones entre los datos y las ligaduras de los elementos de una estructura de datos. Los algoritmos que deben aprovecharse de las relaciones específicas, pueden definirse más fácilmente si existe una especificación de los datos, como la de un diccionario.

4.- *Las decisiones del diseño de los datos a bajo nivel deben retrasarse hasta las últimas etapas del proceso de diseño.* Puede usarse un proceso de refinamiento sucesivo para el diseño de los datos. Esto es, puede definirse una organización global de los datos durante el análisis de requerimientos, refinarse durante el diseño preliminar y especificarse en detalle durante el diseño detallado. El enfoque descendente en el diseño de los datos presenta unas ventajas análogas a las del enfoque descendente en el diseño de software.

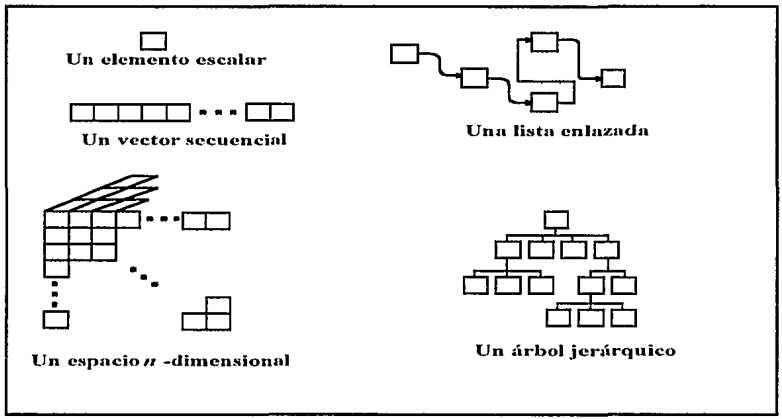
5. *La representación de una estructura de datos debe ser conocida sólo por módulos que hagan un uso directo de los datos contenidos dentro de la estructura.* El concepto de ocultación de la información y el concepto relacionado de acoplamiento proporciona un aspecto importante en la calidad del diseño de software.

6.- *Debe desarrollarse una biblioteca de estructuras de datos útiles y de las operaciones que pueden aplicarse a ellas.* Las estructuras de datos y las operaciones deben verse como un recurso para el diseño del software. Las estructuras de datos pueden diseñarse de forma que sean reusuables.

7.- *El diseño de software y el lenguaje de programación deben soportar la especificación y realización de tipos abstractos de datos.* La implementación de una estructura de datos sofisticada puede hacerse excesivamente difícil si no hay una forma de realizar una especificación directa de la estructura.

Los principios descritos anteriormente forman la base de un método de diseño de datos, que puede ser integrado en la fase de definición y desarrollo del proceso de ingeniería del software.

La estructura de datos dicta la organización, métodos de acceso, grado de asociatividad y alternativas de procesamiento para la información obtenida de la base de datos. La organización y complejidad de una estructura de datos está sólo limitada por la ingeniosidad del diseñador. Sin embargo, hay un número limitado de estructuras de datos clásicas que forman los bloques con los que se construyen estructuras más sofisticadas. Estas estructuras de datos clásicas se ilustran en la figura IV.12.



FFIGURA IV.12. Estructuras de datos clásicas.

Un elemento escalar es el más simple de todas las estructuras de datos. Como su nombre lo indica, un elemento escalar representa un elemento simple de información que puede ser direccionado mediante un identificador; esto es, el acceso puede realizarse especificando una dirección de memoria.

Cuando los elementos escalares se organizan como una lista o grupo contiguo, se forma un vector secuencial. Los vectores son la más común de todas las estructuras de datos y abre la puerta a la indexación variable de información.

Cuando un vector secuencial se extiende a dos, tres y finalmente, a un número arbitrario de dimensiones, se crea un espacio *n-dimensional*.

Los elementos, vectores y espacios pueden organizarse en distintos formatos. Una lista enlazada es una estructura de datos que está organizada en elementos escalares, vectores o espacios no contiguos de una manera que les posibilita ser procesados como una lista.

IV.3.6.1 DISEÑO ARQUITECTONICO.

El objetivo principal del diseño arquitectónico es desarrollar una estructura de programa modular y representar las relaciones de control entre los módulos. Además, el diseño arquitectónico mezcla la estructura de programas y la estructura de datos y define las interfaces que facilitan el flujo de los datos a lo largo de programa.

IV.3.6.2 DISEÑO PROCEDIMENTAL.

El diseño procedimental se realiza después de que se ha establecido la estructura del programa y de los datos. En un mundo ideal, la especificación procedimental requiere definir los detalles algorítmicos que deben establecerse en un lenguaje natural.
No cabe duda de que las herramientas gráficas, tales como los diagrama de flujo o de cajas, dan una excelente forma gráfica de describir fácilmente los detalles procedimentales. Sin embargo, si las herramientas gráficas se usan mal, un dibujo erróneo puede conducir a un software erróneo.

El diagrama de flujo de datos es la representación gráfica más ampliamente usada para el diseño procedimental. Desafortunadamente, es también el método del que se ha abusado más ampliamente.

Un diagrama de flujo es una gráfico muy sencillo. Una caja indica un paso del procesamiento, un rombo representa una condición lógica y las flechas muestran el flujo de control. La figura IV.12 muestra las tres construcciones de programación estructurada tratadas.

La secuencia se representa como dos cajas de procesamiento conectadas por una línea (flecha) de control. La condición, también llamada *if-then-else*, se dibuja como un rombo de decisión, el cual, sí es verdad, hace que se realice el procesamiento de la parte de decisión *then* y si es falso llama al procesamiento de la parte *else*. La repetición se representa usando dos formas algo diferentes. La forma *do-while* prueba una condición y ejecuta repetidamente una tarea mientras la condición se verifique. La forma *repeat-until* ejecuta primero el bucle, luego prueba la condición y repite la tarea hasta que falla la condición. La construcción selección (o select-case) mostrada en la figura es realmente una extensión del *if-then-else*. Se compara un parámetro mediante sucesivas decisiones hasta que una condición es verdadera y se ejecuta entonces un camino de procesamiento de una parte del *case*.

Los siguientes atributos de las notaciones de diseño se han establecido en el contexto de las características generales de programación del sistema.

Facilidad de edición. El diseño procedimental puede requerir modificaciones durante el paso de diseño, durante la prueba del software y finalmente durante la fase de mantenimiento del proceso de ingeniería del software. La facilidad con la que una representación de diseño pueda ser editada, puede ayudar a facilitar cada uno de estos pasos de la ingeriería del software.

Legible por la máquina. Una notación que pueda se introducida directamente en un sistema de desarrollo basado en computadora, ofrece unos enormes beneficios potenciales.

Mantenimiento. El mantenimiento del software es la fase más costosa del ciclo de vida del software. El mantenimiento de la configuración del software casi siempre significa mantenimiento de la representación del diseño procedimental.

Exigencia de estructura. Las ventajas de un método de diseño que utilice los conceptos de la programación estructurada han sido ya trasados. Una notación de diseño que refuerce el uso de únicamente construcciones estructuradas, promueve la práctica de un buen diseño.

Procesamiento automático. Un diseño de detalles contiene información que puede ser procesada para dar al diseñador nuevos o mejores conocimientos respecto a la corrección y calidad de un diseño. Tales conocimientos pueden ser aumentados con informes dados mediante un procesador automático.

Representación de los datos. La habilidad para representar datos locales y globales es un elemento esencial en el diseño detallado. Idealmente, una notación de diseño debe representar directamente tales datos.

Verificación lógica. La verificación automática de la lógica de un diseño es un objetivo supremo durante la prueba del software. Una notación que refuerce la habilidad para verificar la lógica, mejora grandemente la sufuciencia de la prueba.

Disposición para la codificación. El paso de ingeniería del software que sigue al diseño procedimental es la codificación. Una notación que se convierta fácilmente a código fuente reduce el trabajo y los errores.

IV.3.6.3 DOCUMENTACION DE DISEÑO.

El índice de documentación que sigue, es el usado para la especificación de los sistemas del Registro Federal de Electores.

1.0. Ambito.
- 1.1. Objetivos del sistema.
- 1.2. Hardware, software e interfaces humanas.
- 1.4. Base de datos definida.

2.0. Descripción del diseño.
- 2.1. Descripción de los datos.
  - 2.1.1. Revisión del flujo de datos.
  - 2.1.2. Revisión de la estructura de datos.
- 2.2. Interfaces.
  - 2.2.1. Menús.
  - 2.2.2. Pantallas.

3.0. Módulos.
- Para cada módulo.
- 3.1. Texto explicativo .
- 3.2. Organización de los datos.
- 3.3. Comentarios.

4.0. Instalación del sistema.
5.0. Notas especiales.

El índice de documentación presenta una descripción completa del diseño del software. Las secciones numeradas de las Especificación del diseño se completan conforme el diseñador refina su representación del software.

IV.3.7 CODIFICACIÓN.

Todos los pasos anteriormente descritos estan encaminados a obtener las representaciones del software a una forma que pueda ser comprendida por la computadora. Se ha llegado finalmente al paso de la codificación - un proceso que transforma el diseño a un lenguaje de programación, en este caso PRO*C. Para mayor información ver herramientas de diseño, de este capítulo.

IV.3.8 PRUEBAS.

La prueba de software es un elemento crítico para la garantía de calidad del software y representa un último repaso de las especificaciones, del diseño y de la codificación.
La prueba presenta una interesante anomalía para el ingeniero de software. Durante las fases anteriores de definición y de desarrollo, el ingeniero intenta construir el software partiendo de un control abstracto y llegando a una implementación tangible. A continuación llega la prueba. El ingeniero crea una serie de casos de prueba que intenta "demoler" el software que ha sido construido. De hecho, la prueba es uno de los pasos de la ingeniería de software que se puede ver como un destructivo en lugar de constructivo.
La prueba requiere que se descarten ideas preconcebidas sobre la "corrección" del software que se acaba de desarrollar y superar cualquier conflicto de intereses que aparezcan cuando se descubran errores.

El flujo de información para la prueba sigue el esquema descrito en la figura IV.13. Se proporcionan dos clases de entrada al proceso de prueba: 1) una configuración del software que incluye la Especificación de Requerimientos del software, la Especificación del Diseño y el código fuente; y 2) una configuración de prueba que incluye un Plan y Procedimientos de Prueba, casos de prueba y resultados esperados. En realidad la configuración de prueba es un subconjunto de la configuración del software cuando se considera el proceso de ingeniería del software completo.

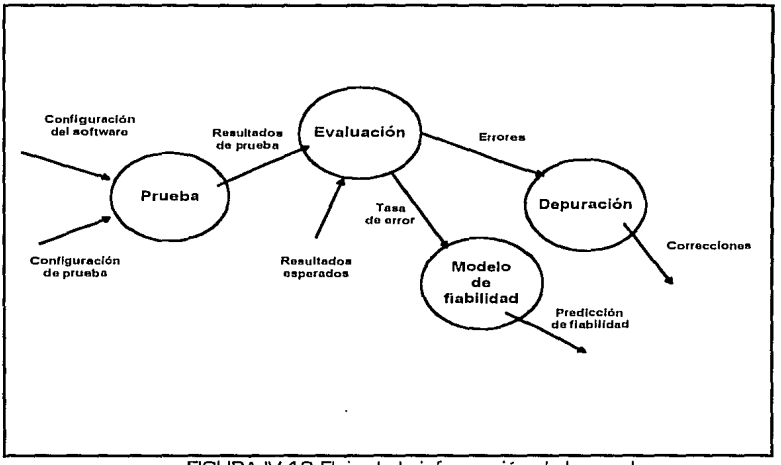


FIGURA IV.13 Flujo de la información de la prueba.

Se lleva a cabo la prueba y se evalúan los resultados. O sea, se comparan los resultados de la prueba con los esperados. Cuando se descubren datos erróneos, esto implica que hay un error y comienza la depuración. El proceso de depuración es una impredecible consecuencia de la prueba. Un "error" que indique una discrepancia de un 0.01 por ciento entre los resultados esperados y los reales puede llevar una hora, un día o un mes de diagnóstico y corrección. Es la inherente incertidumbre de la depuración lo que hace difícil planificar la prueba de forma eficaz.

A medida que se van recopilando y evaluando los resultados de la prueba, comienza a vislumbrarse una medida cualitativa de la calidad y fiabilidad del software quedan en entredicho, siendo necesarias posteriores pruebas. Si, por otro lado, el funcionamiento del software parece ser correcto y los errores que se encuentran son fácilmente corregibles, se puede sacar una de dos conclusiones: 1) la calidad y la fiabilidad del software son aceptables, o 2) las pruebas son inadecuadas para descubrir serios errores. Finalmente, si la prueba no descrube errores, quedará la sospecha de que no se ha pensado cuidadosamente la configuración de prueba y de que los errores están escondidos en el software. Estos defectos serán eventualmente descubiertos por el usuario y corregidos por el profesional.

Los resultados acumulados durante la prueba también pueden ser evaluados de un modo más formal. Los modelos de fiabilidad del software utilizan los coeficientes de error para predecir futuras ocurrencias de errores y, por tanto, la fiabilidad.

IV.3.9 MANTENIMIENTO.

El mantenimiento de software es un conjunto de actividades de ingeniería del software que se dan una vez que el software ha sido puesto en marcha.

La primer actividad de mantenimiento se da debido a que no es razonable asumir que la prueba del software haya decubierto todos los errores latentes de un gran sistema de software.

La segunda actividad que contribuye a la definición de mantenimiento se da debido al rápido cambio inherente a todo aspecto de la informática. La vida útil del software de aplicación puede fácilmente sobrepasar los diez años, haciéndose obsoleto para el entorno del sistema para el que fue originalmente desarrollado.

La tercera actividad que se puede aplicar a la definición del mantenimiento se da cuando un paquete de software tiene éxito. A medida que usa el software, se reciben de los usuarios recomendaciones sobre nuevas posibilidades, sobre modificaciones de funciones ya existentes y sobre mejoras en general. Para satisfacer estas peticiones, se lleva a cavo el mantenimiento perfectivo.

La cuarta actividad de mantenimiento se da cuando se cambia el software para mejorar una futura facilidad de mantenimiento o fiabilidad o para proporcionar una base mejor para futuras mejoras.

Los enfoques técnicos y de gestión para la fase de mantenimiento se pueden implementar con poco esfuerzo, sin embargo, la facilidad de mantenimiento viene dada en gran medida por las tareas de las anteriores fases del proceso de ingeniería del software, teniendo un gran impacto sobre el éxito del método de mantenimiento.

Para el desarrollo de los sistemas se utilizaron las herramientas que nos proporciona ORACLE RDBMS y, para el diseño se utilizaron las técnicas antes mencionadas.

En los siguientes capítulos se describirán los programas desarrollados con su respectiva documentación de diseño, programas fuente y manuales de operación.

INSTITUTO FEDERAL ELECTORAL

CAPITULO V

SISTEMA DE COMPUTO PARA REALIZAR LA DEPURACION DE LA BASE DE DATOS DEL PADRON ELECTORAL CON RESPECTO A LA DOCUMENTACION CIUDADANA Y AL MARCO CARTOGRAFICO ELECTORAL.

- V.1 INTRODUCCION
- V.2 ACTUALIZACION EN LA BASE DE DATOS EL MARCO CARTOGRAFICO ELECTORAL NACIONAL
- V.3 MENUS,REPORTES, PROGRAMAS Y PANTALLAS DE CAPTURA.

V.1. INTRODUCCION.

A partir del levantamiento masivo de información en 1991 llevada a cabo por el Instituto Federal Electoral, es de esperarse que una proporción de esa información sea errónea. El Programa de Depuración Integral del Padrón (PDI), trata de enmendar esta problemática, implementado varios operativos en las áreas correspondientes.

En forma simplificada, se puede decir que el objetivo del PDI fue: que a cada registro de un ciudadano en la base de datos, existiera su correspondiente registro documental (Expediente) y que este ciudadano estuviera bien ubicado cartográficamente.

En el área de Informática, la cual tiene a su cargo el Padrón Electoral en una base de datos, se diseñó, desarrolló e implementó cada uno de los sistemas de cómputo para solventar el proyecto.

El Programa de depuración Integral del Padrón, comprende la revisión total de la cartografía electoral del país, y la revisión documental de los empadronados.

Los sistemas que en este capítulo se tratan, serán descritos de la siguiente forma:

1.0. Ambito.
- 1.1. Objetivos del sistema.
- 1.2. Hardware, software de desarrollo.
- 1.4. Base de datos definida.

2.0. Descripción del diseño.
- 2.1. Interfaces.
  - 2.1.1. Menús.
  - 2.1.2. Pantallas.

3.0. Módulos.
- Para cada módulo.
- 3.1. Texto explicativo .
- 3.2. Organización de los datos.

4.0. Instalación del sistema.
5.0. Comentarios.

V.2 ACTUALIZACIÓN EN LA BASE DE DATOS DEL MARCO CARTOGRAFICO ELECTORAL.

V.2.1 AMBITO

V.2.1.1 OBJETIVO:

- Diseñar un sistema para actualizar el marco Cartográfico electoral en la base de datos del Padrón Electoral Nacional.

V.2.1.2 DIAGRAMA DE FLUJO DE DATOS DEL SISTEMA EN GENERAL.

Para mayor control sobre el sistema, se dividió en tres grandes actividades.
1. Identificación y marcado Cartográfico.
2. Nuevas Cartografías.
3. Cambio cartográfico a nivel ciudadano.

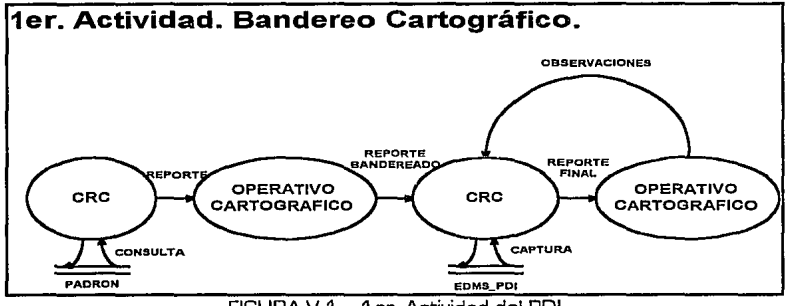


FIGURA V.1. 1er. Actividad del PDI.

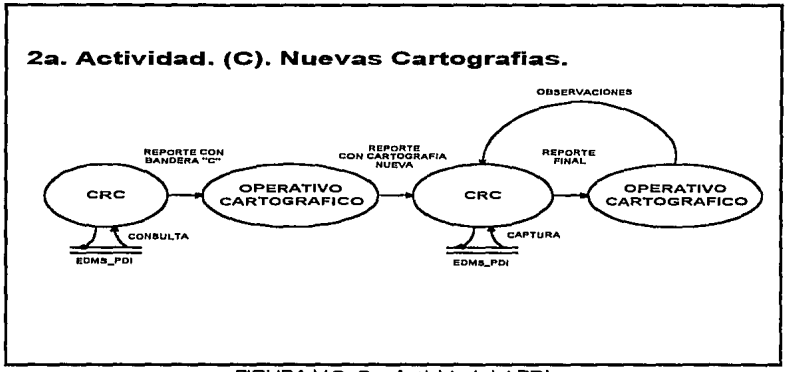


FIGURA V.2. 2a. Actividad del PDI.

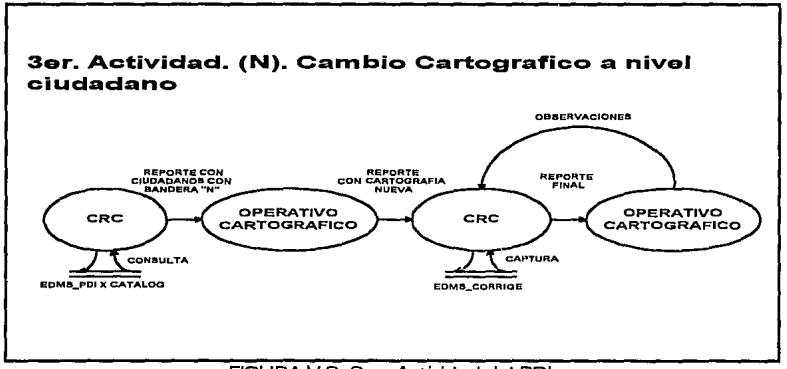


FIGURA V.3. 3er. Actividad del PDI.

De las figuras se deriva que las actividades particulares son :

- El Centro Regional de Cómputo(CRC) emite un primer reporte y lo envía al Operativo Cartográfico.
- El Operativo Cartográfico revisa y plasma sus revisiones.
- El CRC captura revisiones y emite un segundo reporte con información en la base de datos.
- El Operativo Cartográfico revisa y en su caso envía correcciones al CRC.

V.2.1.3 HARDWARE Y SOFTWARE:

Para el desarrollo y pruebas se utilizó una computadora IBM RISC/6000 con AIX 3.2 como sistema operativo y una capacidad de 96 MB en memoria y 2.8 GB en capacidad de almacenamiento en discos. Para mayor información ver CAPITULO III.

El software que se utilizó fue, PRO*C para algunos módulos de proceso complicado, SQL*FORMS para las pantallas de captura y SQL*REPORT para el desarrollo de reportes. Para mayor información de estas herramientas ver capitulo III y IV.

V.2.1.4 BASE DE DATOS DEFINIDA:

La base de datos esta definida en los Centros Regionales de la siguiente forma.

TABLESPACE_NAME	MAX(BYTES)	SUM(BYTES)
SYSTEM	48,969,696	57,909,191
RBS	100,000,000	100,000,000
TEMPORAL	54,212,567	64,232,526
INDI_CAR	71,099,425	81,189,568
CARTOGRA	123,121,568	134,123,456

Como podemos observar existen 5 espacios para tablas, el espacio para tablas SYSTEM es de uso especial, lo utiliza ORACLE para almacenar tablas propias de su operación. El tablespace RBS es utilizado para almacenar las últimas operaciones, datos y resultados de los querys de usuario, a este espacio se le llama espacio para ROLL-BACK. El espacio de tabla TEMPORAL es utilizado por ORACLE para almacenar temporalmente una serie de información que es generada por ciertas operaciones de los usuarios, tal como un ordenamiento, una búsqueda extensa, etc. El espacio de tabla INIDI_CAR es utilizado para almacenar todos los índices que se generen en las tablas de cartografía, las tablas se almacenan en CARTOGRA.

V.2.1.5 DESCRIPCION DE LA BASE DE DATOS DEL SISTEMA.

Las tablas e índices que se utilizaron en el diseño son las siguientes;

NOMBRE TABLA	DUEÑO	TABLESPACE	INITIAL (Mb)
SECCIONES	OPS$DBAREG	CARTOGRA	5
GEOGRAFIA_1_TABLE	OPS$DBAREG	CARTOGRA	2
EDMS_PDI	OPS$DBAREG	CARTOGRA	40
HOMONIMIAS	OPS$DBAREG	CARTOGRA	30
CATALOG	OPS$DBAREG	CARTOGRA	300
LDI_01_CAPTURA	OPS$DBAREG	CARTOGRA	30
LDI_FOLIOS_CORRECTOS	OPS$DBAREG	CARTOGRA	30
EDMS_CORRIGE	OPS$DBAREG	CARTOGRA	30

Las tablas que en este momento nos interesan son EDMS_PDI y EDMS_CORRIGE.

La tabla EDMS_PDI almacena un catálogo del estado actual cartográfico del Padrón Electoral, es decir: ESTADO, DISTRITO, MUNICIPIO, SECCION, LOCALIDAD, MANZANA, tal como lo indica la siguiente descripción.

ESTADO_ACT	NUMBER(2)
DISTRITO_ACT	NUMBER(2)
MUNICIPIO_ACT	NUMBER(3)
SECCION_ACT	NUMBER(3)
LOCALIDAD_ACT	NUMBER(4)
MANZANA_ACT	NUMBER(2)
ESTADO_NUE	NUMBER(2)
DISTRITO_NUE	NUMBER(2)
MUNICIPIO_NUE	NUMBER(3)
SECCION_NUE	NUMBER(3)
LOCALIDAD_NUE	NUMBER(4)
MANZANA_NUE	NUMBER(2)
NUM_PADRON	NUMBER(4)
NOM_LOCALIDAD	CHAR(30)
TIPO	CHAR(1),
VALIDAR	CHAR(1)
OBSERVACIONES	CHAR(30)
OTRA_VEZ	NUMBER(2)
FECHA_CAPTURA	DATE

Los campos ESTADO_ACT, DISTRITO_ACT, MUNICIPIO_ACT, SECCION_ACT, LOCALIDAD_ACT, MANZANA_ACT pertenecen a la cartografía actual, y los campos ESTADO_NUE, DISTRITO_NUE, MUNICIPIO_NUE, SECCION_NUE, LOCALIDAD_NUE, MANZANA_NUE pertenecen a la correspondiente cartografía nueva. El campo de NUM_PADRON, indica el número de ciudadanos que pertenecen a la cartografía actual. El campo NOM_LOCALIDAD contendrá el nombre de la localidad. TIPO es el campo en donde se almacenará el tipo de la sección (1. Urbana, 2 Rural, 3 Mixta). OBSERVACIONES, contendrá las observaciones que el cartógrafo considere pertinentes. OTRA_VEZ contiene el número de veces que un capturista selecciona la cartografía actual para su actualización. FECHA_CAPTURADA es la fecha más reciente de acceso a la cartografía actual. El campo más importante es el de VALIDAR, el cual puede tomar solo uno de los siguientes valores:

1. **V.** Significa que la cartografía es correcta, por lo tanto la cartografía nueva es la misma.
2. **C.** Significa que la cartografía no es correcta y se realizará un cambio a nivel cartográfico.
3. **N.** Significa que la cartografía no es correcta y se realizará una rehubicación cartográfica de los ciudadanos que pertenecen a esta cartografía. En este caso no necesariamente todos los ciudadanos cambian a una sola cartografía. Se realiza una reubicación ciudadano por ciudadano y posiblemente tomen cartografías distintas.

La tabla de EDMS_CORRIGE tiene la siguiente descripción.

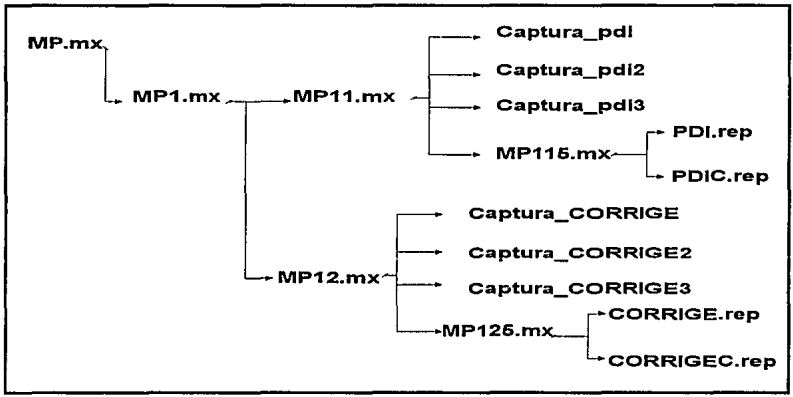
ALFA_CLAVE_ELECTORAL	CHAR(6),
FECHA_NACIMIENTO	NUMBER(6),
LUGAR_NACIMIENTO	NUMBER(2),
SEXO	CHAR(1),
DIGIT_VERIFICADOR	NUMBER(1),
CLAVE_HOMONIMIA	NUMBER(2),
ESTADO_ACT	NUMBER (2),
DISTRITO_ACT	NUMBER (2),
MUNICIPIO_ACT	NUMBER (3),
SECCION_ACT	NUMBER (3),
LOCALIDAD_ACT	NUMBER (4),
MANZANA_ACT	CHAR (2),
ESTADO_NVO	NUMBER (2),
DISTRITO_NVO	NUMBER (2),
MUNICIPIO_NVO	NUMBER (3),
SECCION_NVO	NUMBER (4),
LOCALIDAD_NVO	NUMBER (4),
MANZANA_NVO	NUMBER (2),
LOCALIDAD	CHAR (32),
MODIFICADO	CHAR (1),
FECHA	DATE

Como podemos observar la única diferencia con respecto a la tabla EDMS_PDI es que tenemos además de la cartografía nueva y actual, la clave electoral del ciudadano, la cual esta compuesta por ALFA_CLAVE_ELECTORAL, FECHA_NACIMIENTO, LUGAR_NACIMIENTO, SEXO, DIGIT_VERIFICADOR, CLAVE_HOMONIMIA. Lo que significa que los cambios cartográficos se realizarán a nivel ciudadano.
También tenemos el campo LOCALIDAD que contiene el nombre de la localidad a donde será ubicado. MODIFICADO indica si la cartografía fue capturada (N- no capturada , M- capturada). FECHA indica la fecha de captura.

V.2.2 DESCRIPCION DEL DISEÑO.

V.2.2.1 INTERFACES - MENUS.

La estructura de los menús y de las interfaces con el usuario son las siguientes:



Para mayor ilustración ver las figuras en la opción de MENUS al final de este capítulo.
La notación de los menús MP$x_1x_2x_3$.. significa:

$X_1X_2X_3$. Opciones seleccionadas desde el menú principal a menús subsecuentes..

V.2.3 MODULOS

V.2.3.1 REACION DE TABLAS PARA OPERACION DEL SISTEMA.

CREA EDMS_PDI.

```
create table EDMS_PDI (
        ESTADO_ACT              NUMBER(2),
        DISTRITO_ACT            NUMBER(2),
        MUNICIPIO_ACT           NUMBER(3),
        SECCION_ACT             NUMBER(3),
        LOCALIDAD_ACT           NUMBER(4)
        MANZANA_ACT             NUMBER(2),
        ESTADO_NUE              NUMBER(2),
        DISTRITO_NUE            NUMBER(2),
        MUNICIPIO_NUE           NUMBER(3),
        SECCION_NUE             NUMBER(3),
        LOCALIDAD_NUE           NUMBER(4),
        MANZANA_NUE             NUMBER(2),
        NUM_PADRON              NUMBER(4),
        NOM_LOCALIDAD             CHAR(30),
        TIPO                      CHAR(1),
        VALIDAR                   CHAR(1),
        OBSERVACIONES           CHAR(30),
        OTRA_VEZ                NUMBER(2),
        FECHA_CAPTURA               DATE
E
)
TABLESPACE CARTOGRA
STORAGE (INITIAL 80M NEXT 5M)
/
COMMIT
/
EXIT
```

CREA EDMS_CORRIGE

```
Create table EDMS_CORRIGE
(
        ALFA_CLAVE_ELECTORAL                    CHAR(6),
        FECHA_NACIMIENTO                      NUMBER(6),
        LUGAR_NACIMIENTO                      NUMBER(2),
        SEXO                                    CHAR(1),
        DIGIT_VERIFICADOR                     NUMBER(1),
        CLAVE_HOMONIMIA                       NUMBER(2),
        ESTADO_ACT                            NUMBER (2),
        DISTRITO_ACT                          NUMBER (2),
        MUNICIPIO_ACT                         NUMBER (3),
        SECCION_ACT                           NUMBER (3),
        LOCALIDAD_ACT                          NUMBER(4)
        MANZANA_ACT                             CHAR (2),
        ESTADO_NVO                            NUMBER (2),
        DISTRITO_NVO                          NUMBER (2),
        MUNICIPIO_NVO                         NUMBER (3),
        SECCION_NVO                           NUMBER (4),
        LOCALIDAD_NVO                         NUMBER(4),
        MANZANA_NVO                           NUMBER (2),
        LOCALIDAD                              CHAR (32),
        MODIFICADO                              CHAR (1),
        FECHA                                      DATE
)
TABLESPACE CARTOGRA
STORAGE (INITIAL 40M NEXT 5M)
/
COMMIT
/
EXIT
```

V.2.3.2 REPORTES Y PANTALLAS DE CAPTURA.

Los reportes fueron desarrollados en una herramienta que ORACLE nos proporciona, es SQL*REPORT.
Debido a que esta herramienta genera código que sólo es entendible por ella misma, solo se ilustrará a continuacion cada uno de los reportes generados.

1. **REPORTE CARTOGRAFICO (PDI.rep).** Este reporte contiene la cartografía actual del Padrón Electoral.Este reporte se entrega al Operativo Cartográfico para su bandereo y su corrección cartográfica. Tal como lo ilustra la figura V.4

Instituto Federal Electoral
Registro Federal de Electores
Coordinación de Informatica

Listado de Cartografía Actual

EstAct	DistAct	MuniAct	SeccAct	LocAct	ManAct	NumCiud	EstNue	DistNue	MuniNue	SeccNue	LocNue	ManNue	NomLoc	TipoSec	Bandera
9	8	16	17	1	4	2									
9	8	16	17	1	8	35									
9	8	16	17	1	9	114									
9	8	16	17	1	10	100									
9	8	16	17	1	11	1									
9	8	16	17	1	12	32									

Figura V.4. Reporte con cartografía Actual.

Una vez concluida la tarea del Operativo Cartográfico se regresa el reporte para su captura en Base de Datos.

2. **REPORTE CARTOGRAFICO CON INFORMACION CAPTURADA (PDIC.rep).**Este reporte contiene la Cartografía Actual junto con la bandera que nos indica el tipo de modificación a realizarse y la Cartografía Nueva en su caso. Tal como lo ilustra la figura V.5

Toda esta información se saca de la base de datos.

Instituto Federal Electoral
Registro Federal de Electores
Coordinación de Informatica

Listado de Cartografía Nueva

EstAct	DistAct	MuniAct	SeccAct	LocAct	ManAct	NumCiud	EstNue	DistNue	MuniNue	SeccNue	LocNue	ManNue	NomLoc	TipoSec	Bandera
9	8	16	17	1	4	2	9	8	16	17	1	8		1	N
9	8	16	17	1	8	35	9	8	16	17	1	8		1	V
9	8	16	17	1	9	114	9	8	16	17	1	9		1	V
9	8	16	17	1	10	100	9	8	16	17	1	10		1	V
9	8	16	17	1	11	1	9	8	16	17	1	10		1	C
9	8	16	17	1	12	32	9	8	16	17	1	12		1	V

Figura V.5 Reporte con cartografía y bandereo capturado.

3. **REPORTE DE CARTOGRAFIA INVALIDA (CORRIGE.rep)** .Este reporte contiene todos aquellos ciudadanos que no pudieron ser ubicados vía el conjunto cartográfico, si no que son ubicados uno a uno. Contiene la Clave Electoral y Cartografía Actual del ciudadano.
Tal como lo ilustra la figura V.6
Este reporte se envía al Operativo Cartográfico para su actualización.

Instituto Federal Electoral
Registro Federal de Electores
Coordinación de Informatica

Listado de los no validados

CLAVE ELECTORAL	EstAct	DistAct	MuniAct	SeccAct	LocAct	ManAct	EstNue	DistNue	MuniNue	SeccNue	LocNue	ManNue
HDMRMR67070209H500	9	8	16	17	1	4						
BRCRLS190040332H600	9	8	16	17	1	8						
CCIVLS57120626H900	9	8	16	17	1	9						
GMRNAN27080109H400	9	8	16	17	1	10						
VLSLMN72020820H100	9	8	16	17	1	11						
OSGRAR39092409M400	9	8	16	17	1	12						

FIGURA V.6 Reporte de cambios cartográficos a nivel ciudadano.

4. **REPORTE CON CARTOGRAFIA NUEVA CAPTURADA (CORRIGE.rep).** Este reporte contiene además de lo anterior la información referente a la cartografía nueva del ciudadano. Tal como lo ilustra la figura V.7

Instituto Federal Electoral
Registro Federal de Electores
Coordinación de Informatica

Listado de los no validados

CLAVE ELECTORAL	EstAct	DistAct	MuniAct	SeccAct	LocAct	ManAct	EstNue	DistNue	MuniNue	SeccNue	LocNue	ManNue
HDMRMR67070209H500	9	8	16	17	1	4	9	8	16	17	1	8
BRCRLS190040332H600	9	8	16	17	1	8	9	8	16	17	1	8
CCIVLS57120626H900	9	8	16	17	1	9	9	8	16	17	1	9
GMRNAN27080109H400	9	8	16	17	1	10	9	8	16	17	1	10
VLSLMN72020820H100	9	8	16	17	1	11	9	8	16	17	1	10
OSGRAR39092409M400	9	8	16	17	1	12	9	8	16	17	1	12

FIGURA V.7. Reporte con cartografía nueva capturada.

Las pantallas de captura se realizaron en PRO*C y debido a que el código es extenso se mostrará al final de este capítulo:

1. **CAPTURA CARTOGRAFICA(Captura_pdi.pc).** Pantalla que permite realizar la captura del bandereo cartográfico y la cartografía nueva correspondiente dada por el Operativo Cartográfico. Este programa actua sobre la tabla EDMS_PDI.

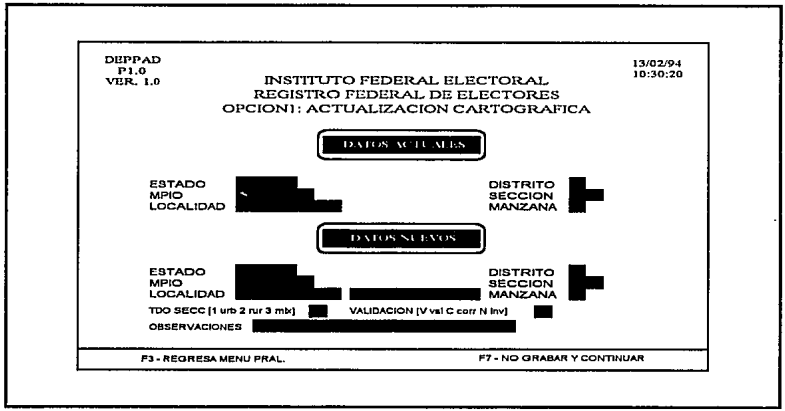


FIGURA V.8 Módulo de captura para el bandereo cartográfico

Esta pantalla permite capturar la cartografía nueva, la bandera, el nombre de la localidad y las observaciones a partir de la cartografía actual.

2. **CAMBIOS A LA CARTOGRAFÍA CAPTURADA (Captura_PDI2.pc).** Con este módulo se pueden realizar cambios a la captura realizada en el punto anterior.

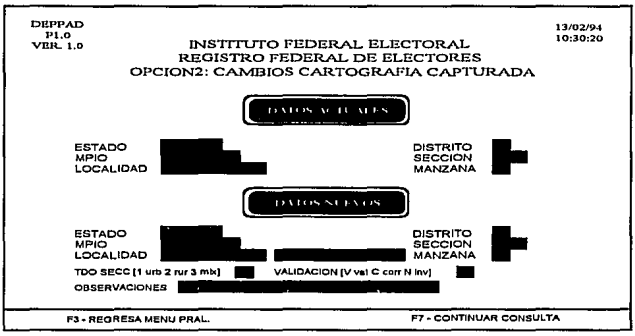


FIGURA V.9. Cambios a la cartografía nueva.

Esta pantalla permite realizar cambios a la cartografía nueva, una vez capturara ésta.

3. **CONSULTA A LA CARTOGRAFÍA CAPTURADA (Captura_PDI3.pc)**
Módulo de consulta a la captura realizada en el punto 1.

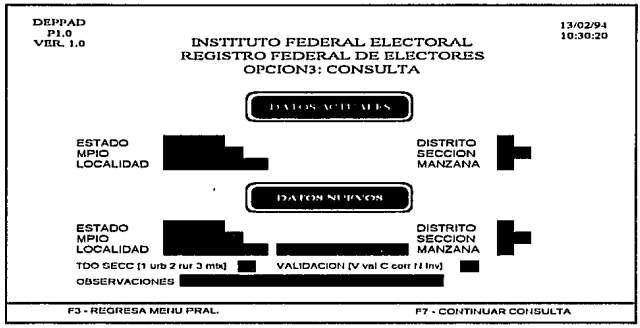


FIGURA V.10. Consuta a la cartografía capturada.

4. **CAPTURA DE CARTOGRAFÍA INVALIDA(Captura_CORRIGE.pc).** Realiza la actualización cartográfica de aquella cartografía que fue bandereada con "N" en la tabla EDMS_PDI. Insertando ahora los registros a nivel ciudadano en la tabla EDMS_CORRIGE. Para actualizar cartográficamente a nivel ciudadano.

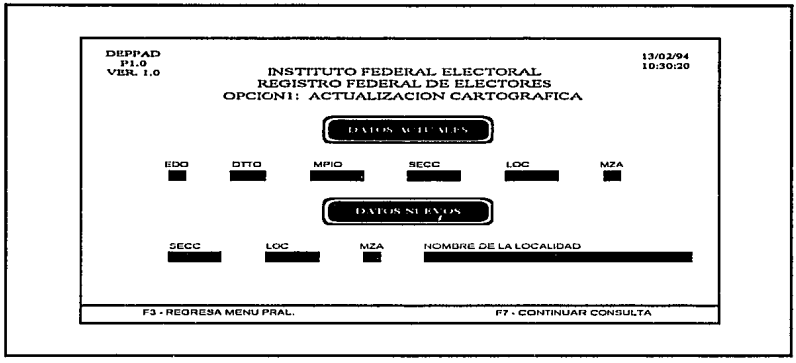


Figura V.11 Ciudadanos que se encuentran en cartografías no validadas

En esta pantalla, a partir de la cartografía actual se captura una parte de la cartografía nueva (sección, localidad, manzana) y nombre de la localidad, de esta forma los ciudadanos son insertados con cartografía actual y nueva en la tabla EDMS_CORRIGE. La pantalla de captura siguiente permite realizar cambios a la cartografía nueva de cada ciudadano.

5. **CAMBIOS A LA CARTOGRAFÍA DEL CIUDADANO (Captura_CORRIGE2.pc).** Permite realizar cambios a la cartografía del ciudadano. Lo anterior se realiza en la tabla EDMS_CORRIGE.

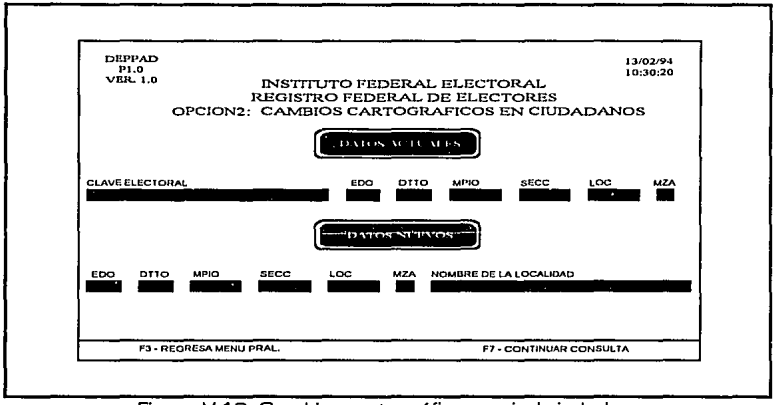


Figura V.12. Cambios cartográficos a nivel ciudadanos.

en esta pantalla, a partir de la clave electoral se realizan los cambios correspondientes a su cartografía.

6. **CONSULTA A LA CARTOGRAFÍA DEL CIUDADANO (Captura_CORRIGE3.pc).** Módulo que permite realizar una consulta a la cartografía del ciudadano de la tabla EDMS_CORRIGE.

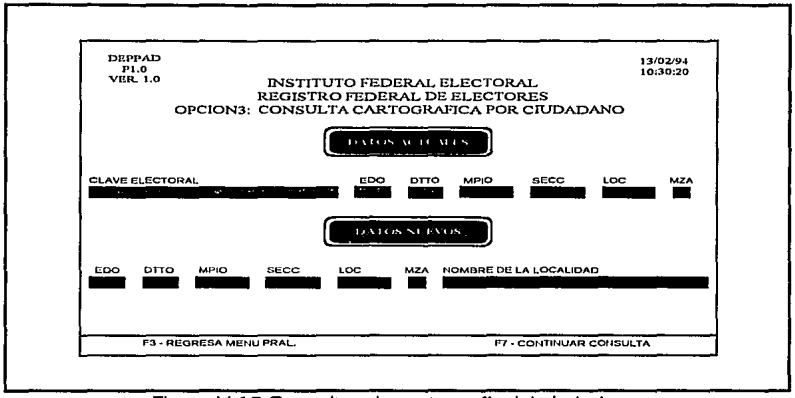


Figura V.13 Consulta a la cartografía del ciudadano.

Los listados de dichos programas se encuentran el final de este capítulo.

V.2.4 INSTALACION DEL SISTEMA

Una vez concluido el desarrollo y las pruebas del sistema, se procede a realizar el envio de los programas a los Centros Regionales a través de la red X.25 con la que cuenta el Registro Federal de Electores. Tomando en cuenta los siguientes pasos:

1. Estar como usuario root del host de desarrollo.
2. Localizarse en el directorio /usr2/zarza/carto/PDI
3. Teclear el siguiente comando.
   $ rcp CARTO.Z regionalx:/u/dbareg/carto/
4. Conectarse al regional y teclear lo siguiente.
   $ rlogin regionalx
   $ uncompress /u/dbareg/carto/CARTO.Z
   $ tar -xvpf /u/dbareg/carto/CARTO
5. Desconectarse del regional.
   $ exit

Donde *regionalx* es:
El nombre del host remoto.

V.2.5 COMENTARIOS.

El resultado de este trabajo conjunto, permitió ubicar correctamente el domicilio del ciudadano en sus respectivas demarcaciones electorales; que las manzanas y localidades tanto urbanas como rurales se encontraban dentro de los límites seccionales correspondientes, y que cada sección electoral se integra con el rango entre 50 a 1,500 electores. Para ello se fusionaron y se subdividieron las secciones que no entraban en este rango, las que presentaban gran dispersión territorial o las que conforme a la proyección de su crecimiento poblacional para 1994, se estimaba que rebasarían los 1,500 electores.
Se revisaron 62,900 secciones electorales en que se dividía el territorio nacional. Como resultado de estos trabajos se determinaron 63,589 secciones electorales (689 más que las de 1991), a las que se les asignó una nueva clave de identificación electoral de manera progresiva por entidad federativa.
En el siguiente cuadro se resumen los resultados de dicho trabajo.

SECCIONES							
Adecuadas	Divididas	Fusionadas	Altas	Bajas	Diferencia	Total	
						1991	1992
72	918	390	1,080	390	698	62,900	63,598

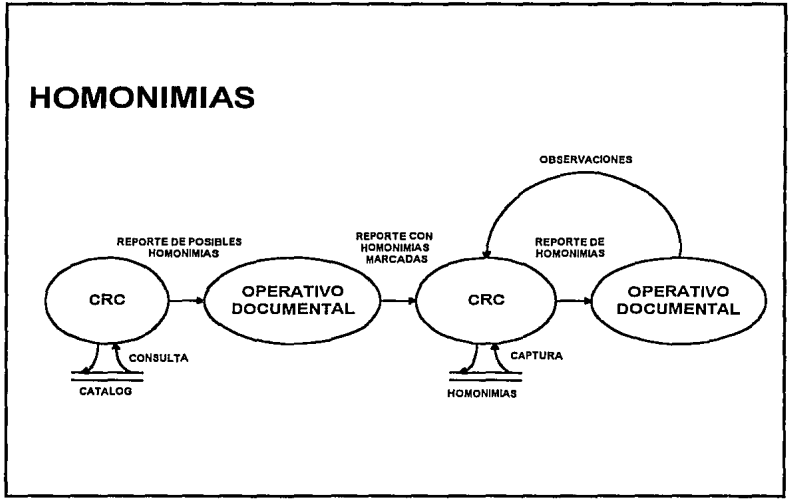
V.3 RASTREO EN LA BASE DE DATOS NACIONAL DE CIUDADANOS HOMONIMOS Y EN SU CASO ACTUALIZACION DE SU INFORMACION.

V.3.1 AMBITO

V.3.1.1 OBJETIVO:

Diseñar e Instalar un sistema para la corrección de ciudadanos homonimos en el Padrón Electoral.

V.3.1.2 DIAGRAMA DE FLUJO DE DATOS DEL SISTEMA:



Como lo ilustra la figura anterior el Centro Regional de Cómputo (CRC) se encarga de sacar un reporte con los posibles homonimos, para que el operativo documental lo revise y en su caso marque las posibles homonimias, regresando dicho reporte al Centro Regional de Cómputo para que se capturen las homonimias reales. Posteriormente estas homonimias se darán de baja del Padrón Electoral.

V.3.1.3 HARDWARE Y SOFTWARE.

El hardware que se utilizó para el desarrollo del sistema es una máquina IBM RISC/6000 con plataforma AIX 3.1 y una capacidad en disco de 2.8 GB y 96MB en memoria.
El software que se utilizó fué PRO*C, SQL*REPORT y SQL*PLUS.
Estas herramientas las proporciona ORACLE para la explotación de la base de datos.

V.3.1.4 BASE DE DATOS DEFINIDA.

La base de datos definida es, la misma que se utilizó para la consolidación del marco cartográfico, descrita en el inciso anterior.

V.3.1.5 BASE DE DATOS PARA EL FUNCIONAMIENTO DEL SISTEMA.

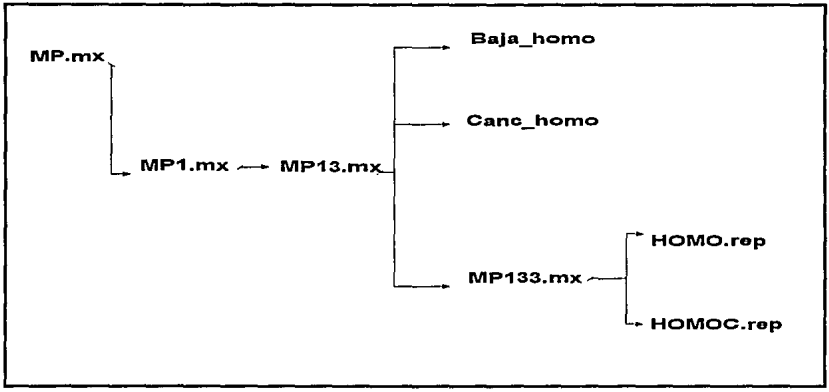
Las tabla que se utilizó para el funcionamiento del sistema fue la tabla de HOMONIMIAS con la siguiente descipción:

ALFA_CLAVE_ELECTORAL	CHAR(6)
FECHA_NACIMIENTO	NUMBER(6)
LUGAR_NACIMIENTO	NUMBER(2)
SEXO	CHAR(1)
DIGIT_VERIFICADOR	NUMBER(1)
CLAVE_HOMONIMIA	NUMBER(2)
ESTADO_ACT	NUMBER(2)
DISTRITO_ACT	NUMBER(2)
MUNICIPIO_ACT	NUMBER(3)
SECCION_ACT	NUMBER(3)
LOCALIDAD_ACT	NUMBER(4)
MANZANA_ACT	CHAR(2)
T_MOV	CHAR(1),

Como podemos observar la tabla contiene la clave electoral, la cual se encuentra compuesta por (ALFA_CLAVE_ELECTORAL, FECHA_NACIMIENTO, LUGAR_NACIMIENTO, SEXO, DIGIT_VERIFICADOR, CLAVE_HOMONIMIA), también tiene la cartografia actual (ESTADO_ACT,DISTRITO_ACT, MUNICIPIO_ACT, SECCION_ACT, LOCALIDAD_ACT, MANZANA_ACT) y por la bandera que indica si es homonimia "H".

V.3.2 DESCRIPCION DEL DISEÑO.
V.3.2.1 INTERFACES Y MENUS.

La estructura de los menús y de las interfaces con el usuario son las siguientes:



Para mayor ilustración ver las figuras en la opción de MENUS al final de este capítulo.
La notación de los menús MP$x_1X_2X_3$.. significa:

$X_1X_2X_3$. Opciones seleccionadas desde el menú principal a menús subsecuentes..

V.3.3 MODULOS

V.3.3.1 RELACION DE TABLAS PARA LA OPERACION DEL SISTEMA

CREA_HOMONIMIAS.

```
CREATE TABLE HOMONIMIAS (
        ALFA_CLAVE_ELECTORAL    CHAR(6)   NOT NULL ,
        FECHA_NACIMIENTO        NUMBER(6) NOT NULL ,
        LUGAR_NACIMIENTO        NUMBER(2) NOT NULL ,
        SEXO                    CHAR(1)   NOT NULL ,
        DIGIT_VERIFICADOR       NUMBER(1) NOT NULL ,
        CLAVE_HOMONIMIA         NUMBER(2) NOT NULL ,
        ESTADO_ACT              NUMBER(2) NOT NULL ,
        DISTRITO_ACT            NUMBER(2) NOT NULL ,
        MUNICIPIO_ACT           NUMBER(3) NOT NULL ,
        SECCION_ACT             NUMBER(3) NOT NULL ,
        LOCALIDAD_ACT           NUMBER(4) NOT NULL ,
        MANZANA_ACT             CHAR(2)   NOT NULL ,
        T_MOV                   CHAR(1),
)
TABLESPACE ZARZA
STORAGE (INITIAL 30M NEXT 1M MAXEXTENT 242)
```

V.3.3.2 REPORTES Y PANTALLAS DE CAPTURA.

Los reportes se desarrollaron en SQL*REPORT y son los siguientes:

1. **REPORTE DE POSIBLES HOMONIMIAS (HOMO.rep).** Este reporte contiene todos los ciudadanos que posiblemente sean homonimias. El reporte se ilustra a continuación.

INSTITUTO FEDERAL ELECTORAL
REGISTRO FEDERAL DE ELECTORES

REPORTE DE POSIBLES HOMONIMIAS

CLAVE ELECTORAL	NOMBRE	E	D	M	S	L	M	CALLE	T
RJANGR18071922H400	GERONIMA ROJO ANGELES	9	40	7	129	1	30	BENITO ZUÑIGA	
RJANGR18071922M400	GERONIMA ROJO ANGELES	9	40	7	129	1	30	BENITO ZUÑIGA	
CSGRGL74122115H900	GLISERIO CASTANEDA GARCIA	9	40	7	207	1	7	C DALIA MZ 10	
CSGRGL74122115H901	GLISERIO CASTANEDA GARCIA	9	40	7	207	1	7	C DALIA MZ 10	
ABMNAD61040512H00	ADRIAN ABORTO MENDOZA	9	40	7	207	1	26	AVE STIM	
ABMNAD61040512H01	ADRIAN ABURTO MENDOZA	9	40	7	207	1	26	AVE STIM	

2. **REPORTE DE HOMONIMIAS (HOMOC.rep).** Este reporte se confoma de aquellos registros que fueron confirmados como homonimas..

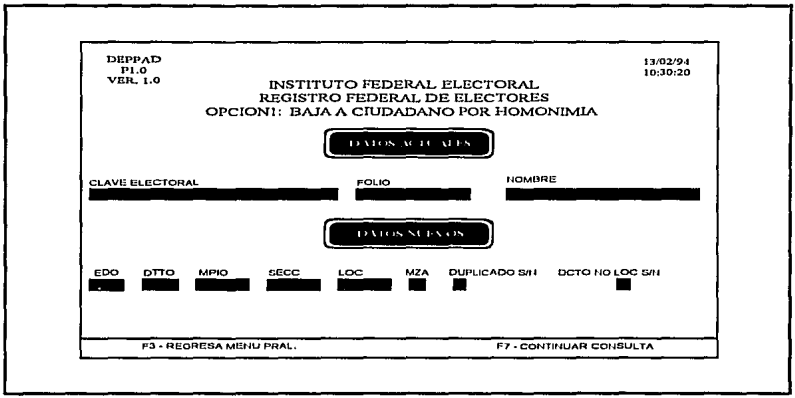
INSTITUTO FEDERAL ELECTORAL
REGISTRO FEDERAL DE ELECTORES

REPORTE DE POSIBLES HOMONIMIAS

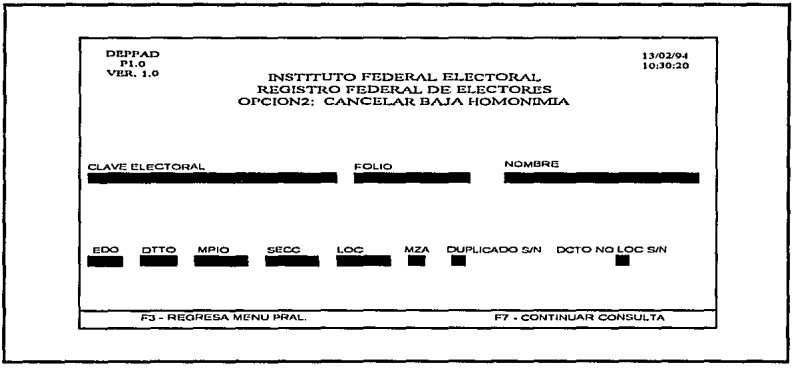
CLAVE ELECTORAL	NOMBRE	E	D	M	S	L	M	CALLE	T
RJANGR18071922H400	GERONIMA ROJO ANGELES	9	40	7	129	1	30	BENITO ZUÑIGA	N
RJANGR18071922M400	GERONIMA ROJO ANGELES	9	40	7	129	1	30	BENITO ZUÑIGA	N
CSGRGL74122115H900	GLISERIO CASTANEDA GARCIA	9	40	7	207	1	7	C DALIA MZ 10	H
CSGRGL74122115H901	GLISERIO CASTANEDA GARCIA	9	40	7	207	1	7	C DALIA MZ 10	H
ABMNAD61040512H00	ADRIAN ABORTO MENDOZA	9	40	7	207	1	26	AVE STIM	N
ABMNAD61040512H01	ADRIAN ABURTO MENDOZA	9	40	7	207	1	26	AVE STIM	N

Las pantallas de captura se desarrollarón en PRO*C. El listado de dichos programas se encuentra al final del capítulo.

1. **CAPTURA HOMONIMIA(baja_homo).** Este programa permite registrar a un registro como homonimia.



2. **CANCELA HOMONIMIA(canc_homo).** Este programa permite cancelar un registro que fue marcado como homonimia.



V.3.4 INSTALACION

Una vez concluido el desarrollo y las pruebas del sistema, se procede a realizar el envio de los programas a los Centros Regionales a través de la red X.25 con la que cuenta el Registro Federal de Electores. Tomando en cuenta los siguientes pasos:

1. Estar como usuario root del host de desarrollo.
2. Localizarse en el directorio /usr2/zarza/carto/ HOMO
3. Teclear el siguiente comando.
   $ rcp HOMO.Z regionalx:/u/dbareg/carto/
4. Conectarse al regional y teclear lo siguiente.
   $ rlogin regional
   $ uncompress /u/dbareg/carto/HOMO.Z
   $ tar -xvpf /u/dbareg/carto/HOMO
5. Desconectarse del regional.
   $ exit

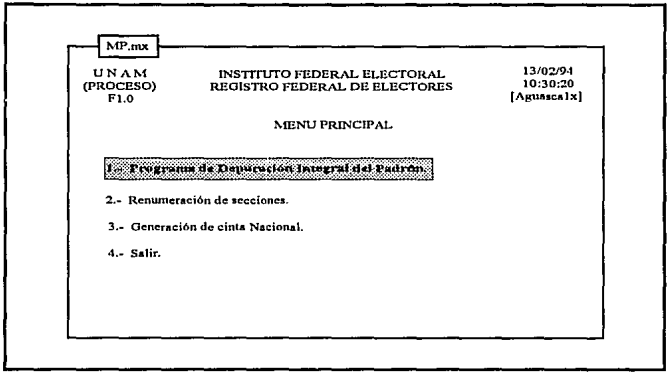
V.2.5 COMENTARIOS

Los resultados de este trabajo arrojaron 209,121 inconsistencias las cuales fueron dados de baja de la base de datos previa autorización de las Comisiones Locales de Vigilancia. Esta cifra representa el 0.5 % del Padrón Electoral.

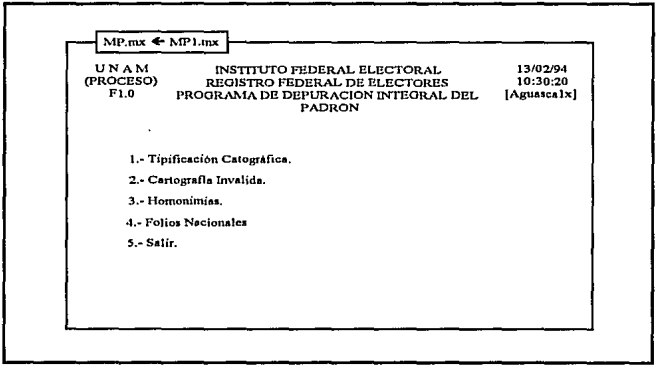
ANEXO

MENUS

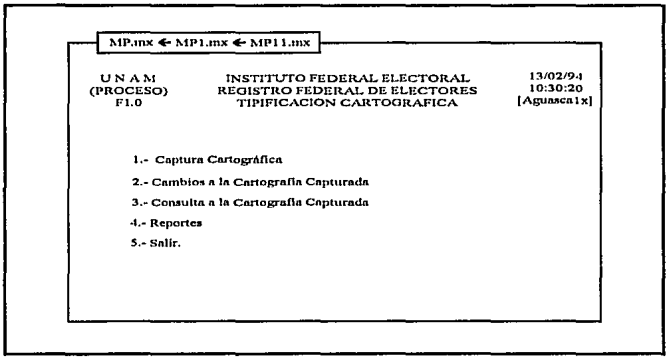
MP.mx



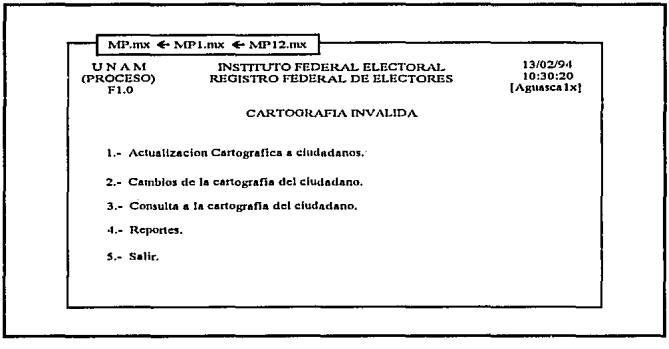
MP1.mx



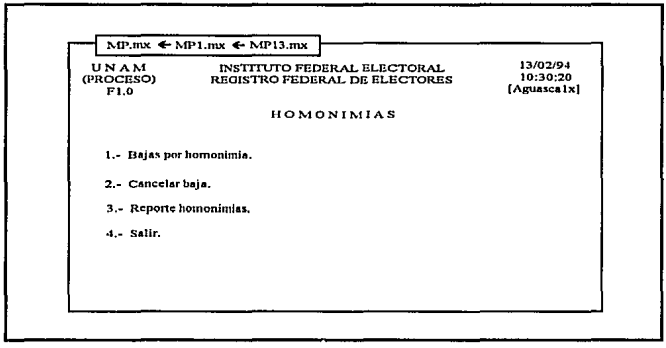
MP11.mx



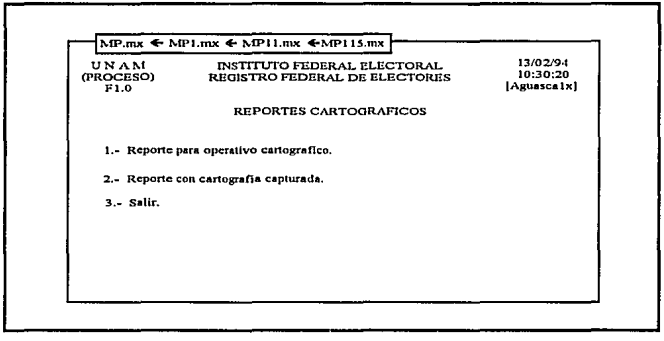
MP12.mx



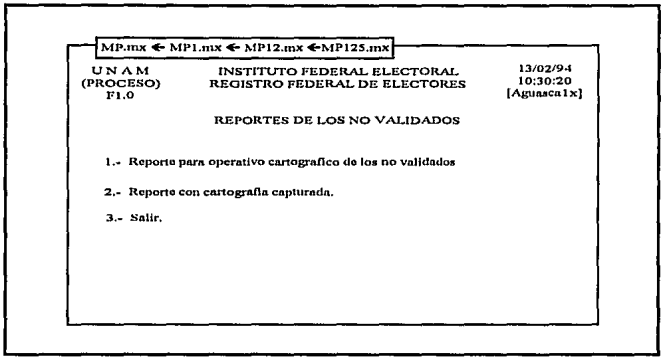
MP13.mx



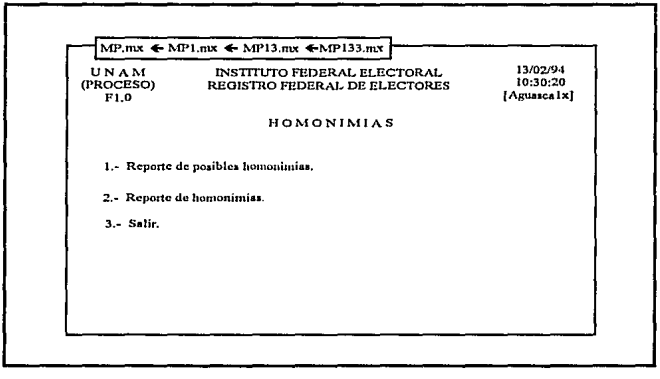
MP115.mx



MP125.mx



MP133.mx



PROGRAMAS

1.CAPTURA CARTOGRAFICA . Captura_PDI.pc

```
/*****************************************************************************************
*************************

  Programa : Permite Capturar la cartografía nueva, a partir de la cartografía
             actual.

  ENTRADAS : Cartografía Actual, Cartografía Nueva, Nombre localidad,
             Tipo de sección, Tipo de Modificacion y observaciones.

  SALIDAS  : Respuesta al proceso. OK o NOOK.

  TABLAS   : EDMS_PDI, GEOGRAFIA_1_TABLE.

  REALIZADO POR: JOSE ALFREDO ZARZA RAMIREZ

*****************************************************************************************
*******************************/


/*---
 INCLUDES
---*/

#include <stdio.h>
#include <string.h>
#include "/usr2/zarza/include/pantalla.h"
#include "/usr2/zarza/include/pan_io.h"
#include "/usr2/zarza/include/act_car.h"
#include "/usr2/zarza/proc/oracle.h"


/*___
 Declaracion de variables que son utilizadas por SQL
___*/
EXEC SQL BEGIN DECLARE SECTION;

    VARCHAR uid[20];
    VARCHAR pwd[20];
```

```
    /*___
          datos actuales
    ___*/
    int estado_act;  VARCHAR estado[30];
    int distrito_act;
    int municipio_act; VARCHAR municipio[30];
    int seccion_act;
    int localidad_act;
    VARCHAR manzana_act[2];
    int total;
    int num_padron;
    varchar validar2[3];
    short validar_nulo;

    /*___
          Datos nuevos
    ___*/
    int estado_nue;
    int distrito_nue;
    int municipio_nue;
    int seccion_nue;
    int localidad_nue;

    VARCHAR nom_localidad[30];
    int manzana_nue;
    VARCHAR tipo[2];
    VARCHAR validar[1];
    VARCHAR observaciones[30];
    int otra_vez;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

/*...
 COMENZAMOS
...*/

main()
{

          int fin, continua;
          int register cont;
          fin= 0;


          strcpy(uid.arr,"ops$dbareg");
          uid.len= strlen(uid.arr);
          strcpy(pwd.arr,"ops$dbareg");pwd.len=strlen(pwd.arr);
```

```
        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;



        for (cont= 0; cont < NUM_DAT_CAP; cont++)
                memset(menu[cont].valor,'\0',sizeof(menu[cont].valor));

        /*
        ** Inicializamos CURSES
        */
        inicio_curses();

        m_pon_pantalla("DEPPAD","P1.0","VER 1.0", "OPCION 1: ACTUALIZACION CARTOGRAFICA");

        /*___
         Inicializa la informacion del teclado
        ___*/
    init_kinf();

        /*___ Pone mensaje al final de la pantalla ___*/
        pon_mensaje();

        while(fin != IS_K3){
                /*___ Pone la pantalla inicial de captura ___*/
                pantalla_inicial();
                /*___ Realiza la captura de la informacion ___*/
                fin= captura_informacion();
        }
        /*___ finaliza los curses ___*/
    fin_curses();

        exit(0);

        /*___ En caso de error en oracle termina con el mensaje ___*/
        error:
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        printf ("error al tiempo de ejecucion");
        exit(1);
}

/*___ Módulo que realiza la captura de la información (estado, distrito,
    municipio, sección, localidad y manzana) tanto actuales como las
    nuevas ___*/
captura_informacion()
{
 int xx,kbresult,valor,i,valido;
```

```
 char cad[50];
 int register cont;

/*inicializa menu*/
for (cont= 0; cont < NUM_DAT_CAP; cont++)
        memset(menu[cont].valor,'\0',sizeof(menu[cont].valor));
m_pon_hora();
 for (pmenu= &menu[0],i= 0,pact= &act[0]; pact->x > 0;){
        memset(cad,'\0',sizeof(cad));
        xx= pact->x+strlen(pact->nombre)+1;
        move(pact->y,xx);
        switch(i){
                case 5:
                case 15:
                case 11:
                        valor= obten_cad_dig(i,pact->y,xx,pact->len,cad);
                        if (i == 5 && valor == OK){
                            strcpy(pmenu->valor,cad);
                                valido= validar_edms_pdi();
                                if (valido != OK){
                                        valor = IS_K7;
                                        pon_mensaje_actualizado();
                                        m_doble_beep();
                                        m_doble_beep();
                                }
                        }
                        /*grabar informacion*/
                        if (i == 15 && valor != IS_TAB){
                            strcpy(pmenu->valor,cad);
                                valor= grabar_informacion();
                                return(valor);
                        }
                        break;
                case 14:
                        valor= obten_cadena(i,pact->y,xx,pact->len,cad);
                        break;
                default:
                        valor= obten_digito(i,pact->y,xx,pact->len,cad);
        }
        switch(valor){
                case IS_K3:
                        return(IS_K3);
                case IS_K7:
                        return(IS_K7);
                case IS_K5:
                        if (i > = 15)
                                return(IS_K5);
```

```
                case IS_TAB:
                        if (i > 0){
                                i--;
                pmenu--;
                                pact--;
                        }
                        break;
                case OK:
                        strcpy(pmenu->valor,cad);
                        cad[0] = '\0';
                        valido = validar_campos(i);
                        if (valido == OK){
                                if(pact->tipo == 0)
                                 if(pact->len == 2)
                                   m_mvprintf(pact->y,xx,"%02d",
                        atoi(pmenu->valor));
                                   else if(pact->len == 3)
                                   m_mvprintf(pact->y,xx,"%03d",
                        atoi(pmenu->valor));
                                   else if(pact->len == 4)
                                   m_mvprintf(pact->y,xx,"%04d",
                        atoi(pmenu->valor));
                                   else if(pact->len == 5)
                                   m_mvprintf(pact->y,xx,"%05d",
                        atoi(pmenu->valor));

                                if (i == 8){
                                        valido= validar_adm_val1();
                                        if (valido != OK){
                                                m_doble_beep();
                                                m_doble_beep();
                                                return(IS_K7);
                                        }
                                }
                                i++;
                                pmenu++;
                                pact++;
                        }
                        else {
                                m_beep();
                                ilumina(i);
                        }


        }
}

}
```

```
/*___ ilumina para el campo i su longitud aceptable ___*/
ilumina (int i)
{
 int register j;

 struct campo_act *ppact;

 ppact= &act[i];
 atributo (PAN_INVERTIDO);
 for (j=0; j < ppact->len; j++)
 {
        m_mvprintf(ppact->y,ppact->x+strlen(ppact->nombre)+1+j," ");
 }
 atributo(PAN_NORMAL);
}

/*___ Valida los campos de estado y municipio tanto actuales como nuevos ___*/
validar_campos(int i)
{
int valido = 0;

 switch (i){
        case 0:
                valido= validar_estado_act();
                break;
        case 2:
                valido= validar_municipio_act();
                break;
        case 6:
                valido= validar_estado_nue();
                break;
        case 8:
                valido= validar_municipio_nue();
                break;
 }
 return(valido);
}


/*___
 Valida estado actual desplegando el nombre del estado
___*/

validar_estado_act ()
{
 char est[30];
```

```
estado_act= atoi(menu[0].valor);
memset(est,' ',sizeof(est));
est[sizeof(est)] = '\0';

m_mvprintf(9,22,"%s",est);

estado.arr[0]= '\0';
EXEC SQL SELECT NOMBRE_MUNICIPIO
        INTO :estado
        FROM GEOGRAFIA_1_TABLE
        WHERE ENTIDAD = :estado_act AND MUNICIPIO=0;

estado.arr[estado.len]= '\0';

if (strlen(estado.arr) > 0){
  m_mvprintf(9,26,"%s",estado.arr);
  return(OK);
}
else
        return(-1);
}


/*____
 Validar estado nuevo desplegando el nombre del estado
____*/

validar_estado_nue()
{
char est[30];

estado_nue= atoi(menu[6].valor);
memset(est,' ',sizeof(est));
est[sizeof(est)] = '\0';

m_mvprintf(15,22,"%s",est);

estado.arr[0]= '\0';
EXEC SQL SELECT NOMBRE_MUNICIPIO
        INTO :estado
        FROM GEOGRAFIA_1_TABLE
        WHERE ENTIDAD = :estado_nue AND MUNICIPIO=0;

estado.arr[estado.len]= '\0';

if (strlen(estado.arr) > 0){
  m_mvprintf(15,26,"%s",estado.arr);
  return(OK);
```

```
 }
 else
        return(-1);
}


/*___
 Validar municipio actual desplegando el nombre del municipio
___*/

validar_municipio_act ()
{
 char mun[30];

 municipio_act= atoi(menu[2].valor);
 memset(mun,' ',sizeof(mun));
 mun[sizeof(mun)] = '\0';
 memset(municipio.arr,'\0',sizeof(municipio.arr));

 m_mvprintf(10,22,"%s",mun);

 EXEC SQL SELECT NOMBRE_MUNICIPIO
        INTO :municipio
        FROM GEOGRAFIA_1_TABLE
        WHERE ENTIDAD = :estado_act AND MUNICIPIO= :municipio_act;

 municipio.arr[municipio.len]= '\0';

 if (strlen(municipio.arr) > 0){
  m_mvprintf(10,26,"%s",municipio.arr);
  return(OK);
 }
 else
        return(-1);
}


/*___
 Validar municipio nuevo desplegando el nombre del municipio
___*/

validar_municipio_nue ()
{
 char mun[30];

 estado_nue= atoi(menu[6].valor);
 municipio_nue= atoi(menu[8].valor);
```

```
memset(mun,' ',sizeof(mun));
mun[sizeof(mun)] = '\0';


municipio.arr[0]= '\0';
EXEC SQL SELECT NOMBRE_MUNICIPIO
        INTO :municipio
        FROM GEOGRAFIA_1_TABLE
        WHERE ENTIDAD = :estado_nue AND MUNICIPIO= :municipio_nue;

municipio.arr[municipio.len]= '\0';

if (strlen(municipio.arr) > 0){
  m_mvprintf(16,26,"%s",municipio.arr);
  return(OK);
}
else
        return(-1);
}


/*___ Validar la clave estado, distrio, municipio, sección, localidad,
    manzana en la tabla edms_pdi en caso de ser invalida retorna OK
    en caso contrario retorna -1 ___*/

validar_edms_pdi()
{

num_padron= -1;
estado_act= atoi(menu[0].valor);
distrito_act= atoi(menu[1].valor);
municipio_act= atoi(menu[2].valor);
seccion_act= atoi(menu[3].valor);
localidad_act= atoi(menu[4].valor);

memset(manzana_act.arr,'\0',sizeof(manzana_act.arr));
char_to_varchar(manzana_act,menu[5].valor);

EXEC SQL SELECT NUM_PADRON
        INTO :num_padron
        FROM EDMS_PDI
        WHERE ESTADO_ACT=:estado_act AND DISTRITO_ACT=:distrito_act AND
            MUNICIPIO_ACT=:municipio_act AND SECCION_ACT=:seccion_act AND
            LOCALIDAD_ACT=:localidad_act AND MANZANA_ACT=:manzana_act;
```

```
EXEC SQL SELECT VALIDAR
        INTO :validar2:validar_nulo
        FROM EDMS_PDI
        WHERE ESTADO_ACT=:estado_act AND DISTRITO_ACT=:distrito_act AND
           MUNICIPIO_ACT=:municipio_act AND SECCION_ACT=:seccion_act AND
           LOCALIDAD_ACT=:localidad_act AND MANZANA_ACT=:manzana_act;

if(num_padron >= 0 && validar_nulo == -1){
        m_mvprintf(11,48,"TOT CIUDADANOS %04d",num_padron);
        return(OK);
}
else
        return(-1);
}


/*__ Valida estado, distrito, municipio para datos nuevos, en caso de ser
   correcta la clave retorna OK en caso contrario retorna -1 ___*/
validar_edm_val1()
{

total= 0;
estado_nue= atoi(menu[6].valor);
distrito_nue= atoi(menu[7].valor);
municipio_nue= atoi(menu[8].valor);


EXEC SQL SELECT count(*)
        INTO :total
        FROM EDM_VAL1
        WHERE ESTADO=:estado_nue AND DISTRITO=:distrito_nue AND
        MUNICIPIO=:municipio_nue;

if(total > 0)
        return(OK);

else
        return(-1);
}

/*___ Realiza el UPDATE a la tabla edms_pdi con los datos nuevos ___*/
grabar_informacion()
{
 int key;

 pon_mensaje2();
```

```
while ((key = sgetch()) != IS_K3 && key != IS_K7 && key != IS_K5)
        m_doble_beep();

if (key == IS_K3 || key == IS_K7){
        pon_mensaje();
        return(key);
}
if (key == IS_K5){
   pon_mensaje_grabar();
   estado_act= atoi(menu[0].valor);
        distrito_act= atoi(menu[1].valor);
        municipio_act= atoi(menu[2].valor);
        seccion_act= atoi(menu[3].valor);
        localidad_act= atoi(menu[4].valor);

        memset(manzana_act.arr,'\0',sizeof(manzana_act.arr));
        char_to_varchar(manzana_act,menu[5].valor);


        estado_nue= atoi(menu[6].valor);
        distrito_nue= atoi(menu[7].valor);
        municipio_nue= atoi(menu[8].valor);
        localidad_nue = atoi(menu[10].valor);
        seccion_nue= atoi(menu[9].valor);
        char_to_varchar(nom_localidad, menu[11].valor);
        manzana_nue= atoi(menu[12].valor);
        char_to_varchar(tipo, menu[13].valor);
        char_to_varchar(validar, menu[14].valor);
        char_to_varchar(observaciones, menu[15].valor);

        EXEC SQL UPDATE EDMS_PDI
        SET ESTADO_NUE=:estado_nue, DISTRITO_NUE=:distrito_nue,
                MUNICIPIO_NUE=:municipio_nue, SECCION_NUE=:seccion_nue,
                LOCALIDAD_NUE=:localidad_nue, NOM_LOCALIDAD=:nom_localidad,
                MANZANA_NUE=:manzana_nue, TIPO=:tipo, VALIDAR=:validar,
                OBSERVACIONES=:observaciones, FECHA_CAPTURA=SYSDATE
        WHERE
                ESTADO_ACT=:estado_act AND DISTRITO_ACT=:distrito_act AND
                MUNICIPIO_ACT=:municipio_act AND SECCION_ACT=:seccion_act
                AND LOCALIDAD_ACT=:localidad_act AND MANZANA_ACT=:manzana_act;
        EXEC SQL COMMIT;
}
pon_mensaje();
return(IS_K7);

}
```

```
/*___ Pone mensaje sentellante al final de la pantalla ___*/
pon_mensaje2(){
        atributo(4);
        m_centray(21,"F3-REGRESAR MENU PRAL.   F5-GRABAR Y CONTINUAR"
          "    F7-NO GRABAR Y CONTINUAR");
        atributo(PAN_NORMAL);
}

/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
pon_mensaje_grabar(){
                atributo(7);
                m_mvprintf(13,54,"GRABANDO INFORMACION");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,54,"                ");
}

/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
pon_mensaje_actualizado(){
                atributo(7);
                m_mvprintf(13,54,"REGISTRO YA ACTUALIZADO");
                sleep(3);
                atributo(PAN_NORMAL);
                m_mvprintf(13,54,"                ");
}
```

2.CAMBIOS A LA CARTOGRAFIA CAPTURADA Captura_PDI2.pc

```
/*****************************************************************************************
**********************************

  PROGRAMA  : Permite realizar cambios a la cartografía nueva, a
              partir de la cartografía actual.

  ENTRADAS  : Cartografía Actual.

  SALIDA    : Cartografía nueva.

  TABLAS    : EDMS_PDI, GEOGRAFIA_1_TABLE.

  REALIZADO POR: JOSE ALFREDO ZARZA RAMIREZ

*****************************************************************************************
**********************************/


/*...
 INCLUDES
...*/

#include <stdio.h>
#include <string.h>
#include "/usr2/zarza/include/pantalla.h"
#include "/usr2/zarza/include/pan_io.h"
#include "/usr2/zarza/include/act_car.h"
#include "/usr2/zarza/proc/oracle.h"


EXEC SQL BEGIN DECLARE SECTION;

    VARCHAR uid[20];
    VARCHAR pwd[20];
```

```
    int estado_act; VARCHAR estado[30];
    int distrito_act;
    int municipio_act; VARCHAR municipio[30];
    int seccion_act;
    int localidad_act;

    int total;
    int num_padron;

    VARCHAR manzana_act[2];


    int estado_nue;
    int distrito_nue;
    int municipio_nue;
    int seccion_nue;
    int localidad_nue;
    VARCHAR nom_localidad[30];
    int manzana_nue;
    VARCHAR tipo[2];
    VARCHAR validar[1];
    VARCHAR validar2[1];
    VARCHAR observaciones[30];

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

/*---
 COMENZAMOS
---*/


main()
{

        int fin, continua;
        int register cont;
        fin= 0;


        strcpy(uid.arr,"ops$dbareg");
        uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg");pwd.len=strlen(pwd.arr);

        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
```

```
        for (cont= 0; cont < NUM_DAT_CAP; cont++)
                memset(menu[cont].valor,'\0',sizeof(menu[cont].valor));

        /*
        ** Inicializamos CURSES
        */
        inicio_curses();

        m_pon_pantalla("DEPPAD","P1.0","VER 1.0", "OPCION 4: VALIDACION CARTOGRAFICA");


    init_kinf();
        pon_mensaje_consulta();
        while(fin != IS_K3){
                pantalla_inicial();
                fin= captura_informacion_actual();
        }

    fin_curses();

        exit(0);

        error:
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        printf ("error al tiempo de ejecucion");
        exit(1);
}


captura_informacion_actual()
{
 int xx,kbresult,valor,i,valido,valor_resp;
 char cad[50];
 int register cont;

/*inicializa menu*/
for (cont= 0; cont < NUM_DAT_CAP; cont++)
        memset(menu[cont].valor,'\0',sizeof(menu[cont].valor));

m_pon_hora();

 for (pmenu= &menu[0],i= 0,pact= &act[0]; i<16;){
        memset(cad,'\0',sizeof(cad));
        xx= pact->x+strlen(pact->nombre)+1;
        move(pact->y,xx);
```

```
switch(i){

        case 5:
                valor= obten_cad_dig(i,pact->y,xx,pact->len,cad);
                strcpy(pmenu->valor,cad);
                valido= validar_adms_pdi();
                if (valido == OK)
                        valor= sacar_datos_nuevos();
                else{
                        valor = IS_K7;
                        m_doble_beep();
                        m_doble_beep();
                }
                break;
        case 11:
        case 15:
                valor=obten_cad_dig(i,pact->y,xx,pact->len,cad);
                if(i == 15 && valor != IS_TAB){
                        strcpy(pmenu->valor,cad);
                        valor= grabar_informacion();
                        return(valor);
                }
                break;
        case 14:
                valor= obten_cadena(i,pact->y,xx,pact->len,cad);
                valor_resp = valor;
                valor = validar_validar(cad);
                if (valor != OK)
                        valor = IS_K7;
                else
                        valor = valor_resp;

                break;
        default:
                valor= obten_digito(i,pact->y,xx,pact->len,cad);
}
switch(valor){
        case IS_K3:
                return(IS_K3);
        case IS_K7:
                return(IS_K7);

        case IS_TAB:
                if (i>0){
                        i--;
      pmenu--;
                        pact--;
```

```
                        }
                        break;
                case OK:
                        strcpy(pmenu->valor,cad);
                        cad[0] = '\0';
                        valido = validar_campos(i);
                        if (valido == OK){
                                if(pact->tipo == 0)
                                  if(pact->len == 2)
                                    m_mvprintf(pact->y,xx,"%02d",
                        atoi(pmenu->valor));
                                  else if(pact->len == 3)
                                    m_mvprintf(pact->y,xx,"%03d",
                        atoi(pmenu->valor));
                                  else if(pact->len == 4)
                                    m_mvprintf(pact->y,xx,"%04d",
                        atoi(pmenu->valor));
                                  else if(pact->len == 5)
                                    m_mvprintf(pact->y,xx,"%05d",
                        atoi(pmenu->valor));

                                i++;
                                pmenu++;
                                pact++;
                        }
                        else {
                                m_beep();
                                ilumina(i);
                        }


        }
}

}



validar_campos(int i)
{
int valido = 0;

 switch (i){
        case 0:
                valido= validar_estado_act();
                break;
        case 2:
```

```
                valido= validar_municipio_act();
                break;
        case 6:
                valido= validar_estado_nue();
                break;
        case 8:
                valido= validar_municipio_nue();
                break;

 }
 return(valido);
}

/*____
 Valida que el cambio al campo validar no sea
 de 'N' a 'V' o 'C'
_____*/

validar_valider(char cad[])
{

 EXEC SQL SELECT VALIDAR
        INTO :validar
        FROM EDMS_PDI
        WHERE ESTADO_ACT=:estado_act AND DISTRITO_ACT=:distrito_act AND
            MUNICIPIO_ACT=:municipio_act AND SECCION_ACT=:seccion_act AND
            LOCALIDAD_ACT=:localidad_act AND MANZANA_ACT=:manzana_act;

        strcpy (validar2.arr,cad);
        validar2.len= strlen(validar2.arr);

        if ( (strcmp(validar.arr,"N") == 0) &&
            (strcmp(validar2.arr,"V")==0 || strcmp(validar2.arr,"C")==0))
        {
                mensaje_noprocede();
                return(-1);
        }

        return(OK);
}
```

```
/*___
 Valida estado actual desplegando el nombre del estado
___*/

validar_estado_act ()
{
 char est[30];

 estado_act= atoi(menu[0].valor);
 memset(est,' ',sizeof(est));
 est[sizeof(est)] = '\0';

 m_mvprintf(9,22,"%s",est);

 estado.arr[0]= '\0';
 EXEC SQL SELECT NOMBRE_MUNICIPIO
          INTO :estado
          FROM GEOGRAFIA_1_TABLE
          WHERE ENTIDAD = :estado_act AND MUNICIPIO=0;

 estado.arr[estado.len]= '\0';

 if (strlen(estado.arr) > 0){
  m_mvprintf(9,26,"%s",estado.arr);
  return(OK);
 }
 else
          return(-1);
}



/*___
 Validar municipio actual desplegando el nombre del municipio
___*/

validar_municipio_act ()
{
 char mun[30];

 municipio_act= atoi(menu[2].valor);
 memset(mun,' ',sizeof(mun));
 mun[sizeof(mun)] = '\0';

 m_mvprintf(10,22,"%s",mun);
```

```
 municipio.arr[0]= '\0';
 EXEC SQL SELECT NOMBRE_MUNICIPIO
          INTO :municipio
          FROM GEOGRAFIA_1_TABLE
          WHERE ENTIDAD = :estado_act AND MUNICIPIO= :municipio_act;

 municipio.arr[municipio.len]= '\0';

 if (strlen(municipio.arr) > 0){
   m_mvprintf(10,26,"%s",municipio.arr);
   return(OK);
 }
 else
          return(-1);
}


validar_edms_pdi()
{

num_padron= -1;
municipio_act= atoi(menu[0].valor);
distrito_act= atoi(menu[1].valor);
municipio_act= atoi(menu[2].valor);
seccion_act= atoi(menu[3].valor);
localidad_act= atoi(menu[4].valor);

memset(manzana_act.arr,'\0',sizeof(manzana_act.arr));
char_to_varchar(manzana_act,menu[5].valor);

EXEC SQL SELECT NUM_PADRON
          INTO :num_padron
          FROM EDMS_PDI
          WHERE ESTADO_ACT=:estado_act AND DISTRITO_ACT=:distrito_act AND
              MUNICIPIO_ACT=:municipio_act AND SECCION_ACT=:seccion_act AND
              LOCALIDAD_ACT=:localidad_act AND MANZANA_ACT=:manzana_act;



if(num_padron >= 0){
          m_mvprintf(11,48,"TOT CIUDADANOS %04d",num_padron);

          return(OK);
}
else
          return(-1);
}


sacar_datos_nuevos() {
```

```
int xx[12];
int yy[12];
int i,key;
struct campo_act *ppact1;

                EXEC SQL SELECT ESTADO_NUE, DISTRITO_NUE, MUNICIPIO_NUE,
                                SECCION_NUE, LOCALIDAD_NUE, NOM_LOCALIDAD,
                                MANZANA_NUE, TIPO, VALIDAR, OBSERVACIONES
                        INTO  :estado_nue, :distrito_nue, :municipio_nue,
                                :seccion_nue, :localidad_nue, :nom_localidad,
                                :manzana_nue, :tipo, :validar, :observaciones
                        FROM EDMS_PDI
                WHERE ESTADO_ACT=:estado_act AND
                        DISTRITO_ACT=:distrito_act AND
                        MUNICIPIO_ACT=:municipio_act AND
                        SECCION_ACT=:seccion_act AND
                        LOCALIDAD_ACT=:localidad_act AND
                        MANZANA_ACT=:manzana_act;

        if (estado_nue > 0){
                for(i=0,ppact1= &act[6]; i<11; i++,ppact1++){
                        yy[i]= ppact1->y;
                        xx[i]= ppact1->x+strlen(ppact1->nombre)+1;
                }

                atributo(PAN_INVERTIDO);

                m_mvprintf(yy[0],xx[0],"%02d",estado_nue);
                itoa(estado_nue,menu[6].valor);

                m_mvprintf(yy[1],xx[1],"%02d",distrito_nue);
                itoa(distrito_nue,menu[7].valor);

                m_mvprintf(yy[2],xx[2],"%03d",municipio_nue);
                itoa(municipio_nue,menu[8].valor);

                m_mvprintf(yy[3],xx[3],"%03d",seccion_nue);
                itoa(seccion_nue,menu[9].valor);

                m_mvprintf(yy[4],xx[4],"%03d",localidad_nue);
        itoa(localidad_nue,menu[10].valor);

                nom_localidad.arr[nom_localidad.len]= '\0';
                m_mvprintf(yy[5],xx[5],"%s",nom_localidad.arr);
                strcpy(menu[11].valor, nom_localidad.arr);
```

```
                m_mvprintf(yy[6],xx[6],"%02d",manzana_nue);
                itoa(manzana_nue,menu[12].valor);

                tipo.arr[tipo.len]= '\0';
                m_mvprintf(yy[7],xx[7],"%s",tipo.arr);
                strcpy(menu[13].valor, tipo.arr);

                validar.arr[validar.len]= '\0';
                m_mvprintf(yy[8],xx[8],"%c",validar.arr[0]);
                strcpy(menu[14].valor, validar.arr);

                observaciones.arr[observaciones.len]= '\0';
                m_mvprintf(yy[9],xx[9],"%s",observaciones.arr);
                strcpy(menu[15].valor, observaciones.arr);

                atributo(PAN_NORMAL);

        return(OK);
        }
        return(-1);

}



/* PONE MENSAJE FINAL */
pon_mensaje_consulta(){
        atributo(PAN_INVERTIDO);
        m_centray(21,"     F3-REGRESAR MENU PRINCIPAL     F7-CONTINUAR CONSULTA     ");
        atributo(PAN_NORMAL);
}
```

```
/*___
 Validar estado nuevo desplegando el nombre del estado
___*/

validar_estado_nue()
{
 char est[30];

 estado_nue = atoi(menu[6].valor);
 memset(est,' ',sizeof(est));
 est[sizeof(est)] = '\0';

 m_mvprintf(15,22,"%s",est);

 estado.arr[0]= '\0';
 EXEC SQL SELECT NOMBRE_MUNICIPIO
          INTO :estado
          FROM GEOGRAFIA_1_TABLE
          WHERE ENTIDAD = :estado_nue AND MUNICIPIO=0;

 estado.arr[estado.len]= '\0';

 if (strlen(estado.arr) > 0){
  m_mvprintf(15,26,"%s",estado.arr);
  return(OK);
 }
 else
          return(-1);
}



/*___
 Validar municipio nuevo desplegando el nombre del municipio
___*/

validar_municipio_nue ()
{
 char mun[30];

 estado_nue = atoi(menu[6].valor);
 municipio_nue = atoi(menu[8].valor);
 memset(mun,' ',sizeof(mun));
 mun[sizeof(mun)] = '\0';
```

```
 municipio.arr[0]= '\0';
 EXEC SQL SELECT NOMBRE_MUNICIPIO
          INTO :municipio
          FROM GEOGRAFIA_1_TABLE
          WHERE ENTIDAD = :estado_nue AND MUNICIPIO= :municipio_nue;

 municipio.arr[municipio.len]= '\0';

 if (strlen(municipio.arr) > 0){
  m_mvprintf(16,26,"%s",municipio.arr);
  return(OK);
 }
 else
          return(-1);
}


ilumina (int i)
{
 int register j;

 struct campo_act *ppact;

 ppact= &act[i];
 atributo (PAN_INVERTIDO);
 for (j=0; j < ppact->len; j++)
 {
          m_mvprintf(ppact->y,ppact->x+strlen(ppact->nombre)+1+j," ");
 }
 atributo(PAN_NORMAL);
}



grabar_informacion()
{
 int key;

 pon_mensaje2();


 while ((key = sgetch()) != IS_K3 && key != IS_K7 && key != IS_K5)
          m_doble_beep();

 if (key == IS_K3 || key == IS_K7){
          pon_mensaje();
          return(key);
 }
```

```
if (key == IS_K5){
    pon_mensaje_grabar();
    estado_act= atoi(menu[0].valor);
        distrito_act= atoi(menu[1].valor);
        municipio_act= atoi(menu[2].valor);
        seccion_act= atoi(menu[3].valor);
        localidad_act= atoi(menu[4].valor);

        memset(manzana_act.arr,'\0',sizeof(manzana_act.arr));
        char_to_varchar(manzana_act,menu[5].valor);


        estado_nue= atoi(menu[6].valor);
        distrito_nue= atoi(menu[7].valor);
        municipio_nue= atoi(menu[8].valor);
        localidad_nue = atoi(menu[10].valor);
        seccion_nue= atoi(menu[9].valor);
        char_to_varchar(nom_localidad, menu[11].valor);
        manzana_nue= atoi(menu[12].valor);
        char_to_varchar(tipo, menu[13].valor);
        char_to_varchar(validar, menu[14].valor);
        char_to_varchar(observaciones, menu[15].valor);

        EXEC SQL UPDATE EDMS_PDI
        SET ESTADO_NUE=:estado_nue, DISTRITO_NUE=:distrito_nue,
                MUNICIPIO_NUE=:municipio_nue, SECCION_NUE=:seccion_nue,
                LOCALIDAD_NUE=:localidad_nue, NOM_LOCALIDAD=:nom_localidad,
                MANZANA_NUE=:manzana_nue, TIPO=:tipo, VALIDAR=:validar,
                OBSERVACIONES=:observaciones, FECHA_CAPTURA=SYSDATE
        WHERE
                ESTADO_ACT=:estado_act AND DISTRITO_ACT=:distrito_act AND
                MUNICIPIO_ACT=:municipio_act AND SECCION_ACT=:seccion_act
                AND LOCALIDAD_ACT=:localidad_act AND MANZANA_ACT=:manzana_act;
        EXEC SQL COMMIT;
 }
 pon_mensaje();
 return(IS_K7);

}

pon_mensaje2(){
        atributo(4);
        m_centray(21,"F3-REGRESAR MENU PRAL.   F5-GRABAR Y CONTINUAR"
          "    F7-NO GRABAR Y CONTINUAR");
        atributo(PAN_NORMAL);
}
```

```
pon_mensaje_grabar(){
                    atributo(7);
                    m_mvprintf(13,54,"GRABANDO INFORMACION");
                    sleep(2);
                    atributo(PAN_NORMAL);
                    m_mvprintf(13,54,"                ");
}

mensaje_noprocede(){
                    atributo(PAN_INVERTIDO);
                    m_mvprintf(13,50,"Cambio N a C o V no permitido");
                    sleep(3);
                    atributo(PAN_NORMAL);
                    m_mvprintf(13,50,"                          ");
}
```

3. CONSULTA A LA CARTOGRAFIA CAPTURADA. Captura_PDI3.pc

```
/*****************************************************************************

  PROGRAMA : Permite realizar consultas a la cartografía nueva capturada
             a partir de la cartografía actual.

  ENTRADAS : Estado, Distrito, Municipio, Seccion, Localidad, Manzana
        Actuales

  SALIDAS  : Despliega la informacion de Estado, Distrito, Municipio,
             Sección, Localidad, Manzana Nuevos.

  TABLAS   : EDMS_PDI, GEOGRAFIA_1_TABLE.

  REALIZADO POR: JOSE ALFREDO ZARZA RAMIREZ

*****************************************************************************/


/*...
 INCLUDES
---*/

#include <stdio.h>
#include <string.h>
#include "/usr2/zarza/include/pantalla.h"
#include "/usr2/zarza/include/psn_io.h"
#include "/usr2/zarza/include/act_car.h"
#include "/usr2/zarza/proc/oracle.h"


EXEC SQL BEGIN DECLARE SECTION;

    VARCHAR uid[20];
    VARCHAR pwd[20];

    int estado_act;  VARCHAR estado[30];
    int distrito_act;
    int municipio_act; VARCHAR municipio[30];
    int seccion_act;
    int localidad_act;
```

```
    int total;
    int num_padron;

    VARCHAR manzana_act[2];


    int estado_nue;
    int distrito_nue;
    int municipio_nue;
    int seccion_nue;
    int localidad_nue;
    VARCHAR nom_localidad[30];
    int manzana_nue;
    VARCHAR tipo[2];
    VARCHAR validar[1];
    VARCHAR observaciones[30];


EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

/*...
 COMENZAMOS
...*/
main()
{

        int fin, continua;
        int register cont;
        fin= 0;


        strcpy(uid.arr,"ops$dbareg");
        uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg");pwd.len=strlen(pwd.arr);

        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;



        for (cont= 0; cont < NUM_DAT_CAP; cont++)
                memset(menu[cont].valor,'\0',sizeof(menu[cont].valor));
```

```
        /*
        ** Inicializamos CURSES
        */
        inicio_curses();

        m_pon_pantalla("DEPPAD","P1.0","VER 1.0", "OPCION 2: CONSULTA CARTOGRAFICA");


    init_kinf();
        pon_mensaje_consulta();
        while(fin != IS_K3){
                pantalla_inicial();
                fin= captura_informacion();
        }

    fin_curses();

        exit(0);

        error:
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        printf ("error al tiempo de ejecucion");
        exit(1);
}

/*
  Módulo que realiza la captura de la información (estado, distrito, municipio
  sección, localidad y manzana) tanto actuales como nuevas.
*/
captura_informacion()
{
 int xx,kbresult,valor,i,valido;
 char cad[50];
 int register cont;

/*inicializa menu*/
for (cont= 0; cont < NUM_DAT_CAP; cont++)
        memset(menu[cont].valor,'\0',sizeof(menu[cont].valor));

 for (pmenu= &menu[0],i= 0,pact= &act[0]; i<6;){
        memset(cad,'\0',sizeof(cad));
        xx= pact->x+strlen(pact->nombre)+1;
        while((kbresult = kbhit_delay(0,100000)) <= 0){
                m_pon_hora();
                move(pact->y,xx);
        }
```

```
switch(i){
        case 5:
                valor= obten_cad_dig(i,pact->y,xx,pact->len,cad);
                strcpy(pmenu->valor,cad);
                valido= validar_edms_pdi();
                if (valido == OK){
                        valor= sacar_datos_nuevos();
                        if (valor == -1){
                                pon_mensaje_error();
                                valor = IS_K7;
                        }
                }
                else{
                        valor = IS_K7;
                        m_doble_beep();
                        m_doble_beep();
                }
                break;
        default:
                valor= obten_digito(i,pact->y,xx,pact->len,cad);
}
switch(valor){
        case IS_K3:
                return(IS_K3);
        case IS_K7:
                return(IS_K7);
        case IS_TAB:
                if (i>0){
                        i--;
      pmenu--;
                        pact--;
                }
                break;
        case OK:
                strcpy(pmenu->valor,cad);
                cad[0] = '\0';
                valido = validar_campos(i);
                if (valido == OK){
                        if(pact->tipo == 0)
                          if(pact->len == 2)
                            m_mvprintf(pact->y,xx,"%02d",
            atoi(pmenu->valor));
                          else if(pact->len == 3)
                            m_mvprintf(pact->y,xx,"%03d",
            atoi(pmenu->valor));
```

```
                        i++;
                        pmenu++;
                        pact++;
                }
                else {
                        m_beep();
                        ilumina(i);
                }


        }
 }

}


/*
 Ilumina el campo cuando no se ocupa el total de el
*/
ilumina (int i)
{
 int register j;

 struct campo_act *ppact;

 ppact= &act[i];
 atributo (PAN_INVERTIDO);
 for (j=0; j < ppact->len; j++)
 {
        m_mvprintf(ppact->y,ppact->x+strlen(ppact->nombre)+1+j," ");
 }
 atributo(PAN_NORMAL);
}


/*
 Rutina que se encarga de validar los campos de estado, distrito, sección,
  localidad, manzana. Desplegando datos segun sea el caso.
*/

validar_campos(int i)
{
int valido = 0;
```

```
 switch (i){
        case 0:
                valido= validar_estado_act();
                break;
        case 2:
                validar_municipio_act();
                break;
        case 6:
                validar_estado_nue();
                break;
        case 8:
                validar_municipio_nue();
                break;

 }
 return(valido);
}


/*____
 Valida estado actual desplegando el nombre del estado
____*/

validar_estado_act ()
{
 char est[30];

 estado_act= atoi(menu[0].valor);
 memset(est,' ',sizeof(est));
 est[sizeof(est)] = '\0';

 m_mvprintf(9,22,"%s",est);

 estado.arr[0]= '\0';
 EXEC SQL SELECT NOMBRE_MUNICIPIO
        INTO :estado
        FROM GEOGRAFIA_1_TABLE
        WHERE ENTIDAD = :estado_act AND MUNICIPIO=0;

 estado.arr[estado.len]= '\0';

 if (strlen(estado.arr) > 0){
  m_mvprintf(9,26,"%s",estado.arr);
  return(OK);
 }
 else
        return(-1);
}
```

```
/*___
 Validar municipio actual desplegando el nombre del municipio
___*/

validar_municipio_act ()
{
char mun[30];

 municipio_act= atoi(menu[2].valor);
 memset(mun,' ',sizeof(mun));
 mun[sizeof(mun)] = '\0';

 m_mvprintf(10,22,"%s",mun);

 municipio.arr[0]= '\0';
 EXEC SQL SELECT NOMBRE_MUNICIPIO
          INTO :municipio
          FROM GEOGRAFIA_1_TABLE
          WHERE ENTIDAD = :estado_act AND MUNICIPIO= :municipio_act;

 municipio.arr[municipio.len]= '\0';

 if (strlen(municipio.arr) > 0){
  m_mvprintf(10,26,"%s",municipio.arr);
  return(OK);
 }
 else
          return(-1);
}


/*
  Valida la clave total (edmsim) en la tabla EDMS_PDI
*/

validar_edms_pdi()
{

 num_padron= -1;
 municipio_act= atoi(menu[0].valor);
 distrito_act= atoi(menu[1].valor);
 municipio_act= atoi(menu[2].valor);
 seccion_act= atoi(menu[3].valor);
 localidad_act= atoi(menu[4].valor);
```

```
memset(manzana_act.arr,'\0',sizeof(manzana_act.arr));
char_to_varchar(manzana_act,menu[5].valor);

EXEC SQL SELECT NUM_PADRON
        INTO :num_padron
        FROM EDMS_PDI
        WHERE ESTADO_ACT=:estado_act AND DISTRITO_ACT=:distrito_act AND
            MUNICIPIO_ACT=:municipio_act AND SECCION_ACT=:seccion_act AND
            LOCALIDAD_ACT=:localidad_act AND MANZANA_ACT=:manzana_act;

if(num_padron >= 0){
        m_mvprintf(11,48,"TOT CIUDADANOS %04d",num_padron);
        return(OK);
}
else
        return(-1);
}


/*
  Saca datos nuevos a partir de los datos actuales
*/

sacar_datos_nuevos() {

int xx[12];
int yy[12];
int i,key;
struct campo_act *ppact1;

                EXEC SQL SELECT ESTADO_NUE, DISTRITO_NUE, MUNICIPIO_NUE,
                                SECCION_NUE, LOCALIDAD_NUE, NOM_LOCALIDAD,
                                MANZANA_NUE, TIPO, VALIDAR, OBSERVACIONES
                        INTO  :estado_nue, :distrito_nue, :municipio_nue,
                                :seccion_nue, :localidad_nue, :nom_localidad,
                                :manzana_nue, :tipo, :validar, :observaciones
                        FROM EDMS_PDI
                WHERE ESTADO_ACT=:estado_act AND
                        DISTRITO_ACT=:distrito_act AND
                        MUNICIPIO_ACT=:municipio_act AND
                        SECCION_ACT=:seccion_act AND
                        LOCALIDAD_ACT=:localidad_act AND
                        MANZANA_ACT=:manzana_act;

        if (estado_nue > 0){
                for(i=0,ppact1= &act[6]; i<11; i++,ppact1++){
                        yy[i]= ppact1->y;
                        xx[i]= ppact1->x+strlen(ppact1->nombre)+1;
```

```
                }

                atributo(PAN_INVERTIDO);

                m_mvprintf(yy[0],xx[0],"%02d",estado_nue);
                m_mvprintf(yy[1],xx[1],"%02d",distrito_nue);
                m_mvprintf(yy[2],xx[2],"%03d",municipio_nue);
                m_mvprintf(yy[3],xx[3],"%03d",seccion_nue);
                m_mvprintf(yy[4],xx[4],"%03d",localidad_nue);

                nom_localidad.arr[nom_localidad.len]= '\0';
                m_mvprintf(yy[5],xx[5],"%s",nom_localidad.arr);

                m_mvprintf(yy[6],xx[6],"%02d",manzana_nue);

                m_mvprintf(yy[7],xx[7],"%c",tipo.arr[0]);

                m_mvprintf(yy[8],xx[8],"%c",validar.arr[0]);

                observaciones.arr[observaciones.len]= '\0';
                m_mvprintf(yy[9],xx[9],"%s",observaciones.arr);

                atributo(PAN_NORMAL);

                pon_mensaje_consulta2();
                move(21,76);
                while ((key=sgetch()) != IS_K3 && key != IS_K7){
                        m_doble_beep();
                }
                pon_mensaje_consulta();
                return (key);
        }
        return(-1);

}



/* PONE MENSAJE FINAL */
pon_mensaje_consulta(){
        atributo(PAN_INVERTIDO);
        m_centray(21,"      F3-REGRESAR MENU PRINCIPAL      F7-CONTINUAR CONSULTA      ");
        atributo(PAN_NORMAL);
}
```

```
/* PONE MENSAJE FINAL RESALTADO */
pon_mensaje_consulta2(){
        atributo(4);
        m_centray(21,"    F3-REGRESAR MENU PRINCIPAL    F7-CONTINUAR CONSULTA    ");
        atributo(PAN_NORMAL);
}




/*___
 Validar estado nuevo desplegando el nombre del estado
___*/

validar_estado_nue()
{
 char est[30];

 estado_nue = atoi(menu[6].valor);
 memset(est,' ',sizeof(est));
 est[sizeof(est)] = '\0';

 m_mvprintf(15,22,"%s",est);

 estado.arr[0]= '\0';
 EXEC SQL SELECT NOMBRE_MUNICIPIO
        INTO :estado
        FROM GEOGRAFIA_1_TABLE
        WHERE ENTIDAD = :estado_nue AND MUNICIPIO = 0;

 estado.arr[estado.len]= '\0';

 if (strlen(estado.arr) > 0){
  m_mvprintf(15,26,"%s",estado.arr);
  return(OK);
 }
 else
        return(-1);
}
```

```
/*___
 Validar municipio nuevo desplegando el nombre del municipio
___*/

validar_municipio_nue ()
{
 char mun[30];

 estado_nue= atoi(menu[6].valor);
 municipio_nue= atoi(menu[8].valor);
 memset(mun,' ',sizeof(mun));
 mun[sizeof(mun)] = '\0';


 municipio.arr[0]= '\0';
 EXEC SQL SELECT NOMBRE_MUNICIPIO
          INTO :municipio
          FROM GEOGRAFIA_1_TABLE
          WHERE ENTIDAD = :estado_nue AND MUNICIPIO= :municipio_nue;

 municipio.arr[municipio.len]= '\0';

 if (strlen(municipio.arr) > 0){
  m_mvprintf(16,28,"%s",municipio.arr);
  return(OK);
 }
 else
          return(-1);
}



/*___ Pone mensaje de error cuando se requiere consultar datos nuevos y
    estos no han sido tecleados ___*/

pon_mensaje_error()
{
 atributo (PAN_INVERTIDO_SUB);
 m_mvprintf(13,50,"NO HAY DATOS NUEVOS <RET>");
 sgetch();
 atributo (PAN_NORMAL);
 m_mvprintf(13,50,"                    ");
}
```

4. CARTOGRAFIA INVALIDA. (Cambio cartográfico a nivel ciudadano) Captura_CORRIGE.pc

```
/*___
        PROGRAMA: Realiza la actualización de ciudadanos que en su cartografía contengan la
bandera
        "N" ( No estan validados) y los actualiza sobre la tabla EDMS_CORRIGE.
        ENTRADAS: Estado, Distrito, Municipio,Seccion,Localidad;Manzana Actual
                Sección,Localidad,Manzana Nueva.
        SALIDAS:
        TABLAS: EDMS_PDI, EDMS_CORRIGE, GEOGRAFIA_1_TABLE,CATALOG.
        PROGRAMA REALIZADO POR:
                JOSE ALFREDO ZARZA RAMIREZ
___*/



#define EXITO_ORA 0
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "/usr2/zarza/include/pantalla.h"
#include "/usr2/zarza/include/pan_io.h"
#include "/usr2/zarza/ac/act_ciud_cap.h"
#include "/usr2/zarza/proc/oracle.h"

int append_EDMS_CORRIGE();

EXEC SQL BEGIN DECLARE SECTION;

        varchar pwd[10], uid[10];

        /*___ clave elector para catalog ___*/
        char alfa[3000][7];          VARCHAR alfax[7];
        int fech[3000],   fechx;
        int luga[3000],   lugax;
      char sexo[3000][2]; VARCHAR       sexox[2];
        int digi[3000],   digix;
        int homo[3000], homox;
        int form[3000],  formx;
```

```
        int esta, estan;
        int dist, distn;
        int muni, munin;
        int secc, seccn;
        int loca, locan;
        int manzn;
        VARCHAR manz[3];
        /*VARCHAR obsen [30];*/
        VARCHAR nloc[30];

        VARCHAR mate[32];
        VARCHAR pate[32];
        VARCHAR nomb[32];

        int num_no_modif;
        int num_n;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;


/*___
        PROGRAMA PRINCIPAL
                        ___*/

main(){

        int register i, cont_ciud;
        int resp;

        strcpy(uid.arr,"ops$dbareg"); uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg"); pwd.len=strlen(pwd.arr);

        /* intenta la connexión a oracle */

        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;


        inicio_curses();


        init_kinf();


        while (resp != IS_K3){
```

```
                clear ();
                m_pon_pantalla("DEPPAD","P1.0","VER 1.0",
                        "OPCION 1: ACTUALIZACION CARTOGRAFICA");

                pantalla_inicial();
                resp = captura_informacion();

        }

        fin_curses();

        exit(0);

        error:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error en tiempo de ejecucion :\n");
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                EXEC SQL ROLLBACK RELEASE;
                exit(1);
}


/*___ Rutina que realiza la captura de la información de datos de entrada ___*/
captura_informacion(){
        int resp, kbresult, sigue;
        int register i;


        /*___ Inicializa las variables de captura todas a '\0'___*/
        for (pmenu= &var_menu[0]; pmenu->x > 0 ; pmenu++)
                memset(pmenu->valor,'\0',sizeof(pmenu->valor));


        m_pon_hora();

        for (pmenu= &var_menu[0]; pmenu->x > 0 ; ){


                switch (pmenu->tipo){

                        case 0:
                                resp = input_xy_digito(pmenu);
                                break;
                        case 1:
                                resp = input_xy_cadena(pmenu);
```

```
                        break;
                case 2:
                        resp = input_xy_cad_dig(pmenu);
                        break;
        }

        switch (resp){
                case IS_K3:
                case IS_K7:
                        return(resp);
                case IS_TAB:
                        if (pmenu > &var_menu[0])
                                pmenu--;
                        break;
                case OK:
                        pmenu++;
                        break;
        }

        /*___ validación inserta registros a EDMS_CORRIGE ___*/
        if (pmenu == &var_menu[10]){

                sigue = inserta_datos ();
                if (sigue == IS_K7)
                        return (IS_K7);
                else {
                        pmenu = &var_menu[5];
                }
        } /*___ end if ___*/
    }/*__ end for __*/
}


/*__ inserta datos nuevos a la tabla EDMS_CORRIGE ___*/
inserta_datos (){
  int registros = 0;
```

```
        esta = atoi(var_menu[0].valor);
        dist = atoi(var_menu[1].valor);
        muni = atoi(var_menu[2].valor);
        secc = atoi(var_menu[3].valor);
        loca = atoi(var_menu[4].valor);
        char_to_varchar(manz , var_menu[5].valor);

    /* Datos Nuevos */
    seccn = atoi(var_menu[6].valor);
    locan = atoi(var_menu[7].valor);
        manzn = atoi(var_menu[8].valor);
        char_to_varchar (nloc, var_menu[9].valor) ;


if ( (val_cve_EDMS_CORRIGE()) == NO_OK ){


        if ( (val_cve_EDMS_PDI()) == OK ){
                /*___ existe con 'N' ___*/
                registros = append_EDMS_CORRIGE();
        }
        else{
                m_beep();
                mensaje_no_edms_pdi();
                return(IS_K7);
        }
}
else{

  mensaje_si_edms_corrige();
  memset (var_menu[5].valor,'\0', strlen(var_menu[6].valor));
  ilumina(5);
  memset (var_menu[6].valor,'\0', strlen(var_menu[6].valor));
  ilumina(6);
  memset (var_menu[7].valor,'\0', strlen(var_menu[7].valor));
  ilumina(7);
  memset (var_menu[8].valor,'\0', strlen(var_menu[8].valor));
  ilumina(8);
  memset (var_menu[9].valor,'\0', strlen(var_menu[9].valor));
  ilumina(9);
```

```
 }
 return(OK);
}

/*___ A&ade información ciudadana a EDMS_CORRIGE ___*/
int append_EDMS_CORRIGE()
{

int register i, cont_ciud;


        for (i = 0; i < 3000 ; i++){
                memset (alfa[i], '\0', sizeof(alfa[i]));
                memset (sexo[i], '\0', sizeof(sexo[i]));
                fech[i] = 0;
                luga[i] = 0;
        }

                EXEC SQL
                SELECT          ALFA_CLAVE_ELECTORAL,
                        FECHA_NACIMIENTO,
                        LUGAR_NACIMIENTO,
                        SEXO,
                        DIGIT_VERIFICADOR,
                        CLAVE_HOMONIMIA,
                        NUM_FORMA
                INTO    :alfa,:fech,:luga,:sexo,:digi,:homo,:form
                FROM    CATALOG
                WHERE   ESTADO = :esta AND DISTRITO = :dist AND
                        MUNICIPIO = :muni AND SECCION = :secc AND
                        LOCALIDAD = :loca AND MANZANA = :manz;



        for (i = 0; i < 3000 && fech[i] > 0; i++)
        {
                alfa[i][6] = '\0';
                sexo[i][1] = '\0';
        }
        cont_ciud = i;
```

```
atributo( PAN_INVERTIDO );
m_mvprintf(10,50,"Insertando:%d ciudadanos.",i);

if (strlen(alfa[0]) > 0){

        for (i = 0; (i < cont_ciud)&&(fech[i] > 0); i++){

                char_to_varchar (alfax, alfa[i]);
                char_to_varchar (sexox, sexo[i]);
                fechx = fech[i];
                lugax = luga[i];
                digix = digi[i];
                homox = homo[i];
                formx = form[i];


                EXEC SQL INSERT INTO EDMS_CORRIGE
                (
                        ALFA_CLAVE_ELECTORAL,
                        FECHA_NACIMIENTO,
                        LUGAR_NACIMIENTO,
                        SEXO,
                        DIGIT_VERIFICADOR,
                        CLAVE_HOMONIMIA,
                        NUM_FORMA,
                        ESTADO_ACT,
                        DISTRITO_ACT,
                        MUNICIPIO_ACT,
                        SECCION_ACT,
                        LOCALIDAD_ACT,
                        MANZANA_ACT,

                        ESTADO_NVO,
                        DISTRITO_NVO,
                        MUNICIPIO_NVO,
                        SECCION_NVO,
                        LOCALIDAD_NVO,
                        MANZANA_NVO,
                        LOCALIDAD,
                        MODIFICADO,
                        FECHA
                )
```

```
                VALUES
                (
                        :alfax,
                        :fechx,
                        :lugax,
                        :sexox,
                        :digix,
                        :homox,
                        :formx,
                        :esta,
                        :dist,
                        :muni,
                        :secc,
                        :loca,
                        :manz,

                        :esta,
                        :dist,
                        :muni,
                        :seccn,
                        :locan,
                        :manzn,
                        :nloc,
                        'M',
                        SYSDATE
                );
                if (sqlca.sqlcode != EXITO_ORA){
                        EXEC SQL WHENEVER SQLERROR CONTINUE;
                        fin_curses();
                        printf("Error en tiempo de ejecucion :\n");
                        printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                        EXEC SQL ROLLBACK RELEASE;
                        exit(1);
                }

        } /*___ End for OK ___*/

        EXEC SQL COMMIT;

} /*___ End if OK informacion ___*/
else
        return ( 0 );
```

```
        atributo( PAN_NORMAL );
        m_mvprintf(10,50,"                    ");

        memset (var_menu[5].valor,'\0', strlen(var_menu[6].valor));
        ilumina(5);
        memset (var_menu[6].valor,'\0', strlen(var_menu[6].valor));
        ilumina(6);
        memset (var_menu[7].valor,'\0', strlen(var_menu[7].valor));
        ilumina(7);
        memset (var_menu[8].valor,'\0', strlen(var_menu[8].valor));
        ilumina(8);
        memset (var_menu[9].valor,'\0', strlen(var_menu[9].valor));
        ilumina(9);

}


/*** checa si existe ya en EDMS_CORRIGE ***/
int val_cve_EDMS_CORRIGE ()
  {

    num_no_modif = -1;

    EXEC SQL SELECT COUNT (*)
       INTO :num_no_modif
       FROM EDMS_CORRIGE
       WHERE ESTADO_ACT = :esta  AND DISTRITO_ACT = :dist   AND
          MUNICIPIO_ACT = :muni  AND SECCION_ACT = :secc  AND
                LOCALIDAD_ACT = :loca  AND MANZANA_ACT = :manz ;


    if ( num_no_modif > 0 )
      return ( OK );
    else
      return ( NO_OK );
   }


/*** checa si existe ya en EDMS_PDI ***/
int val_cve_EDMS_PDI ()
   {
        char cadena[49];
```

```
    num_n = -1;

    EXEC SQL SELECT COUNT (*)
        INTO :num_n
        FROM EDMS_PDI
        WHERE ESTADO_ACT = :esta  AND DISTRITO_ACT = :dist  AND
                MUNICIPIO_ACT = :muni  AND SECCION_ACT = :secc  AND
                LOCALIDAD_ACT = :loca  AND MANZANA_ACT = :manz  AND
                VALIDAR = 'N';

    if ( num_n > 0 )
      return ( OK );
    else{

        atributo ( PAN_NORMAL );
        memset (cadena,' ', sizeof(cadena));
        cadena[49] = '\0';
        m_mvprintf(10,30,"%s",cadena);

      return ( NO_OK );
    }
  }


/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
mensaje_no_edms_pdi(){
                atributo(7);
                m_mvprintf(13,50,"NO HAY CLAVE 'N' EN EDMS_PDI");
                sleep(1);
                atributo(PAN_NORMAL);
                m_mvprintf(13,50,"                          ");
}


/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
mensaje_si_edms_corrige(){
        char cadena[50];

                atributo(PAN_INVERTIDO);
                m_mvprintf(13,49,"LA CLAVE YA HA SIDO ACTULIZADA");
                sleep(1);
                atributo(PAN_NORMAL);
                m_mvprintf(13,49,"                          ");
```

```
}

/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
mensaje_no_edms_corrige(){
        char cadena[50];

                atributo(7);
                m_mvprintf(13,49,"NO HAY MAS DATOS QUE ACTUALIZAR");
                sleep(1);
                atributo(PAN_NORMAL);
                m_mvprintf(13,49,"                              ");
                memset (cadena,' ', sizeof(cadena));
                cadena[49] = '\0';
                m_mvprintf(10,30,"%s",cadena);
}
/* ___ pone mensaje al finel de la línea ___ */
pon_mensaje1 (){
        atributo(PAN_INVERTIDO);
        m_centray(21,"              F3-REGRESAR MENU PRINCIPAL"
            "                   ");
        atributo(PAN_NORMAL);
}

/*___ Pone mensaje sentellante al final de la pantalla ___*/
pon_mensaje2(){
        atributo(4);
        m_centray(21,"F3-REGRESAR MENU PRAL.   F5-GRABAR Y CONTINUAR"
            "   F7-NO GRABAR Y CONTINUAR");
        atributo(PAN_NORMAL);
}

/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
pon_mensaje_grabar(){
                atributo(7);
                m_mvprintf(13,54,"GRABANDO INFORMACION");
                sleep(1);
                atributo(PAN_NORMAL);
                m_mvprintf(13,54,"                  ");
}
```

```
/*___ ilumina para el campo i su longitud aceptable ___*/
ilumina (int i)
{
 int register j;

 struct menu *ppact;

 ppact= &var_menu[i];
 atributo (PAN_INVERTIDO);
 for (j=0; j < ppact->len; j++)
 {
        m_mvprintf(ppact->y+1,ppact->x+j," ");
 }
 atributo(PAN_NORMAL);
}
```

5. CAMBIO A LA CARTOGRAFIA DEL CIUDADANO Captura_CORRIGE2

```
/*____

        PROGRAMA: Permite relizar cambios a datos capturados en la tabla EDMS_CORRIGE .
        ENTRADA: Clave electoral del ciudadano. y su cartografía nueva.
        SALIDA:
        TABLAS: EDMS-CORRIGE, GEOGRAFIA_1_TABLE.
        PROGRAMA REALIZADO POR:
                JOSE ALFREDO ZARZA RAMIREZ

____*/

#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "/usr2/zarza/include/pantalla.h"
#include "/usr2/zarza/include/pan_io.h"
#include "/usr2/zarza/ac/act_ciud2.h"
#include "/usr2/zarza/proc/oracle.h"


int append_EDMS_CORRIGE();
```

```
EXEC SQL BEGIN DECLARE SECTION;

        varchar pwd[10], uid[10];

        /*___ clave elector para catalog ___*/
        VARCHAR alfax[7];
        int fechx;
        int lugax;
        VARCHAR      sexox[2];
        int digix;
        int  homox;
        int  formx;


        int  esta, estan;
        int  dist, distn;
        int  muni, munin;
        int  secc, seccn;
        int  loca, locan;
        int  manzn;
        VARCHAR manz[3];
        /*VARCHAR obsen [30]; */
        VARCHAR nloca[30];

        VARCHAR mate[32];
        VARCHAR pate[32];
        VARCHAR nomb[32];

        int num_no_modif;
        int num_n;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;


/*___
        PROGRAMA PRINCIPAL
                        ___*/
```

```
main(){

        int register i, cont_ciud;
        int resp;

        strcpy(uid.arr,"ops$dbareg"); uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg"); pwd.len=strlen(pwd.arr);

        /* intenta la connexión a oracle */

        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;


        inicio_curses();


        init_kinf();

        while (resp != IS_K3){
                m_pon_pantalla("DEPPAD","P1.0","VER 1.0",
                        "OPCION 2: CAMBIOS CARTOGRAFICOS EN CIUDADANOS");
                pantalla_inicial();
                resp = captura_informacion();

        }

        fin_curses();

        exit(0);

        error:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error en tiempo de ejecucion :\n");
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                EXEC SQL ROLLBACK RELEASE;
                exit(1);
}
```

```
/*___ Rutina que realiza la captura de la información de datos de entrada ___*/
captura_informacion(){
        int resp, kbresult, sigue;
        int register i;


        /*___ Inicializa las variables de captura todas a '\0'___*/
  for (i = 0; i < NUM_DAT_CAP2; i++)
                memset(var_menu[i].valor,'\0',sizeof(var_menu[i].valor));


        m_pon_hora();

        for (pmenu= &var_menu[0]; pmenu < &var_menu[NUM_DAT_CAP2]; ){

                if (pmenu == &var_menu[0]){
                 pantalla_inicial();
                 for (i = 0; i < NUM_DAT_CAP2; i++)
                   memset(var_menu[i].valor,'\0',sizeof(var_menu[i].valor));
                }
                atributo(PAN_NORMAL);
                switch (pmenu->tipo){

                        case 0:
                                resp = input_xy_digito(pmenu);
                                break;
                        case 1:
                                resp = input_xy_cadena(pmenu);
                                break;
                        case 2:
                                resp = input_xy_cad_dig(pmenu);
                                break;
                }

                switch (resp){
                        case IS_K3:
                        case IS_K7:
                                 return(resp);
                        case IS_TAB:
                                if (pmenu > &var_menu[7])
                                        pmenu--;
                                break;
                        case OK:
```

```
                        pmenu++;
                        break;
                }

                /*___ validación inserta registros a EDMS_CORRIGE ___*/
                if (pmenu == &var_menu[1]){
                        if (saca_claves() == NO_OK){
                                m_beep();
                                mensaje_no_edms_corrige();
                                return(IS_K7);
                        }
                        pmenu = &var_menu[7];
                } /*___ end if ___*/
                if (pmenu == &var_menu[15]){
                        pon_mensaje2();
                        sigue = grabar_informacion ();
                        if (sigue == IS_K3 || sigue == IS_K7)
                                return (sigue);
                        pon_mensaje1();
                        pmenu = &var_menu[0];
                        continue;
                }
        }/*__ end for __*/
}


/*___ Saca los datos de un ciudadano de CATALOG ___*/
saca_claves(){
 char buffer[18];
 struct menu *ppmenu;


  substr(alfax.arr, var_menu[0].valor, 0, 5);
  alfax.len = strlen(alfax.arr);

  substr(buffer, var_menu[0].valor, 6, 11);
  fechx = atoi(buffer);

  substr(buffer, var_menu[0].valor, 12, 13);
  lugax = atoi(buffer);

  substr(sexox.arr, var_menu[0].valor,14, 14);
  sexox.len = strlen(sexox.arr);


  substr(buffer, var_menu[0].valor,15, 15);
```

```
digix = atoi(buffer);

substr(buffer, var_menu[0].valor, 16, 15);
homox = atoi(buffer);

esta = -1;

        EXEC SQL
        SELECT ESTADO_ACT, DISTRITO_ACT, MUNICIPIO_ACT, SECCION_ACT ,
            LOCALIDAD_ACT , MANZANA_ACT, ESTADO_NVO, DISTRITO_NVO,
            MUNICIPIO_NVO, SECCION_NVO, LOCALIDAD_NVO, MANZANA_NVO,
            LOCALIDAD
        INTO  :esta, :dist, :muni, :secc, :loca, :manz, :estan, :distn, :munin,
            :seccn, :locan, :manzn, :nloca
        FROM  EDMS_CORRIGE
        WHERE  ALFA_CLAVE_ELECTORAL = :alfax AND FECHA_NACIMIENTO = :fechx
            AND LUGAR_NACIMIENTO = :lugax AND SEXO = :sexox AND
            DIGIT_VERIFICADOR = :digix AND CLAVE_HOMONIMIA = :homox;


        if (esta <= 0)
                return(NO_OK);



        itoa(esta,var_menu[1].valor);
        itoa(dist,var_menu[2].valor);
        itoa(muni,var_menu[3].valor);
        itoa(secc,var_menu[4].valor);
        itoa(loca,var_menu[5].valor);
    manz.arr[manz.len] = '\0';
        strcpy(var_menu[6].valor, manz.arr);

        itoa(estan,var_menu[7].valor);
        itoa(distn,var_menu[8].valor);
        itoa(munin,var_menu[9].valor);
        itoa(seccn,var_menu[10].valor);
        itoa(locan,var_menu[11].valor);
        itoa(manzn,var_menu[12].valor);
        /*___ aqui va nombre localidad ___*/
```

```
        nloca.arr[nloca.len] = '\0';
        strcpy (var_menu[13].valor, nloca.arr);

        atributo(PAN_INVERTIDO);
        for (ppmenu = &var_menu[1]; ppmenu->x >0; ppmenu++){
                if (ppmenu->tipo == 0){

                switch (ppmenu->len){
                case 2:
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%02d",atoi(ppmenu->valor));
                break;
                case 3:
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%03d",atoi(ppmenu->valor));
                break;
                case 4:
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%04d",atoi(ppmenu->valor));
                break;
                }

                }
                else
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%s",ppmenu->valor);
        }
        atributo(PAN_NORMAL);
        move(var_menu[1].y+1, var_menu[1].x);
        return (OK);
}
```

```
/*___ Graba la información del ciudadano actualizada en edms_corrige __*/
grabar_informacion(){
        int key;

        while ((key = sgetch()) != IS_K3 && key != IS_K5 && key != IS_K7)
                m_beep();
        if (key == IS_K3 || key == IS_K7)
                return(key);
        else if (key == IS_K5){
                estan = atoi(var_menu[7].valor);
                distn = atoi(var_menu[8].valor);
                munin = atoi(var_menu[9].valor);
                seccn = atoi(var_menu[10].valor);
                locan = atoi(var_menu[11].valor);
                manzn = atoi(var_menu[12].valor);
                char_to_varchar(nloca, var_menu[13].valor);
                pon_mensaje_grabar();

                EXEC SQL UPDATE EDMS_CORRIGE SET
                        ESTADO_NVO   = :estan,
                        DISTRITO_NVO  = :distn,
                        MUNICIPIO_NVO = :munin,
                        SECCION_NVO   = :seccn,
                        LOCALIDAD_NVO = :locan,
                        MANZANA_NVO   = :manzn,
                        LOCALIDAD     = :nloca,
                        MODIFICADO    = 'M',
                        FECHA         = SYSDATE
                WHERE  ALFA_CLAVE_ELECTORAL = :alfax AND
                        FECHA_NACIMIENTO    = :fechx AND
                        LUGAR_NACIMIENTO    = :lugax AND
                        SEXO                = :sexox AND
                        DIGIT_VERIFICADOR   = :digix AND
                        CLAVE_HOMONIMIA     = :homox;

                EXEC SQL COMMIT;
                return (OK);
        }

}



/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
```

```
mensaje_no_edms_pdi(){
                atributo(7);
                m_mvprintf(13,50,"NO EXISTEN DATOS CON ESA CLAVE");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,50,"                    ");
}



/**** trae de edms_corrige la clave electoral ****/
obtiene_clave()
  {
        memset (alfax.arr,'\0',sizeof(alfax.arr));
        fechx = 0;
        lugax = 0;
        memset (sexox.arr,'\0',sizeof(sexox.arr));
        digix = 0;
        homox = 0;
   EXEC SQL SELECT ALFA_CLAVE_ELECTORAL, FECHA_NACIMIENTO,
LUGAR_NACIMIENTO,
            SEXO, DIGIT_VERIFICADOR, CLAVE_HOMONIMIA
      INTO :alfax, :fechx, :lugax, :sexox, :digix, :homox
      FROM EDMS_CORRIGE
      WHERE ESTADO_ACT = :esta AND DISTRITO_ACT = :dist AND
         MUNICIPIO_ACT = :muni AND SECCION_ACT = :secc AND
               LOCALIDAD_ACT = :loca AND MANZANA_ACT = :manz AND
               MODIFICADO = 'N';

        obtiene_nombre();
  }


/**** trae de la tabla CATALOG el nombre y apellido ****/
obtiene_nombre()
  {
        memset (pate.arr, '\0',sizeof(pate.arr));
        memset (mate.arr, '\0',sizeof(mate.arr));
        memset (nomb.arr, '\0',sizeof(nomb.arr));
   EXEC SQL SELECT APELLIDO_PATERNO, APELLIDO_MATERNO, NOMBRE
      INTO :pate, :mate, :nomb
      FROM CATALOG
      WHERE ALFA_CLAVE_ELECTORAL = :alfax AND FECHA_NACIMIENTO = :fechx
```

```
                AND LUGAR_NACIMIENTO = :lugax AND SEXO= :sexox  AND
          DIGIT_VERIFICADOR = :digix  AND  CLAVE_HOMONIMIA = :homox;

   return(OK);

  }


/* ___ pone mensaje al final de la linea ___ */
pon_mensaje1 (){
        atributo(PAN_INVERTIDO);
        m_centray(21,"      F3-REGRESAR MENU PRAL.       "
             "    F7-NO GRABAR Y CONTINUAR       ");
        atributo(PAN_NORMAL);
}

/*___ Pone mensaje sentellante al final de la pantalla ___*/
pon_mensaje2(){
        atributo(4);
        m_centray(21,"F3-REGRESAR MENU PRAL.   F5-GRABAR Y CONTINUAR"
             "    F7-NO GRABAR Y CONTINUAR");
        atributo(PAN_NORMAL);
}

/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
pon_mensaje_grabar(){
                atributo(7);
                m_mvprintf(13,54,"GRABANDO INFORMACION");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,54,"                    ");
}


/*___ función que guarda en var_origen el substring de var_total desde
    la posición inicial hasta la posición final                ___*/

substr (char *var_origen, char var_total[], int inic, int final)
{
 int register j = 0;

 for (; inic <= final; inic++)
   var_origen[j++] = var_total[inic];


 var_origen[j] = '\0';
```

```
}


/*___ entero a alfanumérico ___*/

itoa (int n, char s[])
{
 int i,j,c;


        i=0;
        j=0;

        do{
                s[i++]= n%10+'0';
        }while ((n /= 10) > 0);

        for (i=0,j= strlen(s)-1; i<j; i++,j--)
                c= s[i], s[i]= s[j], s[j]= c;
}

/*___ Despliega el mensaje de clave no valida en edms_pdi ___*/
mensaje_no_edms_corrige(){
                atributo(7);
                m_mvprintf(13,49,"CLAVE INVALIDA");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,49,"                    ");
}
```

6. CONSULTA CARTOGRAFICA DE CIUDADANOS Captura_CORRIGE3.pc

```
/*****************************************************************************

PROGRAMA: Permite relizar consultas de información de ciudadanos dada su clave electoral.
ENTRADAS:  Clave Electoral.
SALIDAS: cartografía actual, cartografía nueva, nombre de localidad.
TABLAS: EDMS_CORRIGE, CATALOG.
PROGRAMA REALIZADO POR: Jose Alfredo Zarza Ramírez.

*****************************************************************************/


#include <stdio.h>
#include <string.h>
#include "/usr2/lozada/oracle.h"
#include "/usr2/lozada/act_cart/act_ciud2.h"
#include "/usr2/zarza/include/pan_io.h"
#include "/usr2/zarza/include/pantalla.h"

EXEC SQL BEGIN DECLARE SECTION;

    VARCHAR uid [15];   /* variables pasword */
    VARCHAR pwd [15];
                        /* variables de edmslm */
    int esta_a, dist_a, muni_a, secc_a, loca_a;
    int esta_n, dist_n, muni_n, secc_n, loca_n, manz_n;
    VARCHAR manz_a [2], nloc[30], modi[1];

    int res ;

    VARCHAR alfa[6], sexo[1];  /*variables para clave electoral */
    int fech, luga, digi, homo, form;

    VARCHAR pate[32], mate[32], nomb[32], apeynom[35]; /*vars.nombre catalog */

    VARCHAR call[32], nume[8], col[32], direccion[45] ; /* direccion  */

    struct menu  cve_elect[1], *pcve_elect ;/* clave electoral */
```

```
EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;

 main ()
 {
   char str_aux [10];
   int opcion;
   strcpy ( uid.arr,"ops$dbareg" );
   uid.len = strlen ( uid.arr );
   strcpy ( pwd.arr,"ops$dbareg" );
   pwd.len = strlen ( pwd.arr );

 EXEC SQL WHENEVER SQLERROR GOTO error;

  EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;


  pcve_elect = &cve_elect[0];       /* inicializar pcve_elect */
  strcpy ( pcve_elect->nombre, "clave_electoral");
  strcpy ( pcve_elect->valor, "\0");
  pcve_elect->x = 2;
  pcve_elect->y = 7;
  pcve_elect->len = 18;
  pcve_elect->tipo = 2;


do
{
   inicio_curses();
   init_kinf ();
   m_pon_pantalla ( "DEPPAD", "P1.0", "VER 1.0",
            "OPCION 2: CONSULTA CARTOGRAFICA POR CIUDADANO ");
   m_mvprintf ( 7,4,  "CLAVE ELECTORAL" );
   res = input_xy_cad_dig ( pcve_elect );/* capturar clave electoral */
   substr ( alfa.arr, pcve_elect->valor, 0, 5);/* dividirla en sus partes */
   alfa.len = strlen(alfa.arr);
   substr ( str_aux, pcve_elect->valor, 6, 11);
   fech = atoi ( str_aux );
   substr ( str_aux, pcve_elect->valor, 12, 13);
   luga = atoi ( str_aux );
   substr ( sexo.arr, pcve_elect->valor, 14, 14);
   sexo.len = strlen(sexo.arr);
   substr ( str_aux, pcve_elect->valor, 15, 15);

   digi = atoi ( str_aux );
```

```
substr ( str_aux, pcve_elect->valor, 16, 17);
homo = atoi  ( str_aux );

if ( trae_de_corrige() == NO_OK )
 {  atributo ( PAN_INVERTIDO );
    m_mvprintf (14,20, " NO SE ENCONTRO PERSONA CON ESA CLAVE ");
    atributo ( PAN_NORMAL );
 }
 else
 {
       res= trae_de_catalog ();
         direccion.arr[0]='\0'; /* unir datos de direccion */
    m_quita_espacios (call.arr);
    strcat ( direccion.arr, call.arr);
    m_quita_espacios (direccion.arr);
    strcat ( direccion.arr, ", ");
    m_quita_espacios (nume.arr);
    strcat ( direccion.arr, nume.arr);
    m_quita_espacios (direccion.arr);
    strcat ( direccion.arr, ", ");
    m_quita_espacios (col.arr);
    strcat ( direccion.arr, col.arr);
    m_quita_espacios (direccion.arr);
     apeynom.arr[0] = '\0';   /* y nombre y apellidos */
    m_quita_espacios (pate.arr) ;
    strcat ( apeynom.arr, pate.arr) ;
    m_quita_espacios (apeynom.arr) ;
    strcat ( apeynom.arr, " " );
    m_quita_espacios (mate.arr) ;
    strcat ( apeynom.arr, mate.arr) ;
    m_quita_espacios (apeynom.arr) ;
    strcat ( apeynom.arr, " " );
    m_quita_espacios (nomb.arr) ;
    strcat ( apeynom.arr, nomb.arr );
    m_quita_espacios (apeynom.arr) ;

    res = despliega();  /* desplegar en pantalla todos los datos */
  }
atributo ( PAN_INVERTIDO );
m_centray (21," F3-REGRESAR AL MENU PRINCIPAL "
          "              F7-CONTINUAR CONSULTA ");
atributo ( PAN_NORMAL );
do

{ opcion = sgetch();
  if ( opcion != IS_K3 && opcion != IS_K7 ) m_beep();}
while ( opcion != IS_K3  && opcion != IS_K7 );
```

```
   fin_curses();
  }
  while ( opcion == IS_K7 );
  exit(0);
  error:
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf ( "Error en la ejecucion");
    exit ( 1 );
}


          /**** trae de edms_corrige los datos ****/
trae_de_corrige()
  {
     manz_a.arr[0] = '\0';
     esta_n=0; dist_n=0; muni_n=0; secc_n=0; loca_n=0;
     manz_n = 0; nloc.arr[0]= '\0'; modi.arr[0]='\0';
     EXEC SQL SELECT  NUM_FORMA, ESTADO_ACT, ESTADO_NVO, DISTRITO_ACT,
           DISTRITO_NVO, MUNICIPIO_ACT, MUNICIPIO_NVO, SECCION_ACT,
           SECCION_NVO, LOCALIDAD_ACT, LOCALIDAD_NVO, MANZANA_ACT,
           MANZANA_NVO, LOCALIDAD, MODIFICADO
       INTO  :form, :esta_a, :esta_n, :dist_a, :dist_n, :muni_a,
           :muni_n, :secc_a, :secc_n, :loca_a, :loca_n, :manz_a,
           :manz_n, :nloc, :modi
       FROM  EDMS_CORRIGE
       WHERE ALFA_CLAVE_ELECTORAL = :alfa  AND
           FECHA_NACIMIENTO = :fech   AND
           LUGAR_NACIMIENTO = :luga   AND  SEXO= :sexo   AND
           DIGIT_VERIFICADOR = :digi  AND  CLAVE_HOMONIMIA = :homo;

     if ( strlen (manz_a.arr) > 0 )
       return ( OK );
     else
       return ( NO_OK );
  }
```

```
        /**** trae de catalog el nombre, apellido y direccion ****/
trae_de_catalog ()
  {
    EXEC SQL SELECT APELLIDO_PATERNO, APELLIDO_MATERNO, NOMBRE,
         CALLE, NUM_EXTERIOR, COLONIA
      INTO  :pate, :mate, :nomb, :call, :nume, :col
      FROM  CATALOG
      WHERE ALFA_CLAVE_ELECTORAL = :alfa  AND
         FECHA_NACIMIENTO = :fech   AND
         LUGAR_NACIMIENTO = :luga   AND  SEXO= :sexo   AND
         DIGIT_VERIFICADOR = :digi  AND  CLAVE_HOMONIMIA = :homo;
   return(1);
  }


despliega ()
 {
  atributo ( PAN_NORMAL);
  m_mvprintf ( 7,4,  "CLAVE ELECTORAL" );
  m_mvprintf ( 7,26, "FOLIO NAL." );
  m_mvprintf ( 7,42, "APELLIDO PATERNO, MATERNO, NOMBRE" );
  m_mvprintf ( 10,47, "DIRECCION" );
  m_mvprintf ( 16,3, "EDO DTTO MPIO SECC LOC  MZA" );
  atributo (PAN_INVERTIDO);
  m_mvprintf ( 14,9, "DATOS ACTUALES" );

  m_mvprintf ( 8,2,"%s",alfa.arr );
  m_mvprintf ( 8,8,"%d", fech );
  m_mvprintf ( 8,14,"%d", luga );
  m_mvprintf ( 8,16,"%s", sexo.arr );
  m_mvprintf ( 8,17,"%d", digi );
  m_mvprintf ( 8,18,"%02d", homo );
  m_mvprintf ( 8,26,"%d", form );
  m_mvprintf ( 8,41,"%s", apeynom.arr );
  m_mvprintf ( 11,31,"%s",direccion.arr );

  m_mvprintf ( 17,4, "%d",esta_a );
  m_mvprintf ( 17,8, "%d",dist_a );
  m_mvprintf ( 17,12,"%d",muni_a );
  m_mvprintf ( 17,17,"%d",secc_a );
  m_mvprintf ( 17,22,"%d",loca_a );
  m_mvprintf ( 17,27,"%s",manz_a.arr );
```

```
    if ( strcmp ( modi.arr, "M" ) == 0 )
      {
       m_mvprintf ( 14,51,"DATOS NUEVOS" );
       m_mvprintf ( 17,46,"%d",esta_n );
       m_mvprintf ( 17,50,"%d",dist_n );
       m_mvprintf ( 17,54,"%d",muni_n );
       m_mvprintf ( 17,59,"%d",secc_n );
       m_mvprintf ( 17,64,"%d",loca_n );
       m_mvprintf ( 17,69,"%d",manz_n );
       m_mvprintf ( 19,48,"%s",nloc.arr );
       atributo (PAN_NORMAL);
       m_mvprintf ( 16,45,"EDO DTTO MPIO SECC LOC  MZA" );
       m_mvprintf ( 19,38,"LOCALIDAD" );
      }
    else
      { atributo ( 4 );
        m_centray (19, "ESTA PERSONA NO HA SIDO MODIFICADA");
        atributo ( PAN_NORMAL );
      }

    return(1);
   }


/*___ función que guarda en var_origen el substring de var_total desde
     la posicion inicial hasta la posicion final                    ___*/
substr (char var_origen [], char var_total[], int inic, int final)
{
  int register j = 0;

  for (; inic <= final; inic++)
    var_origen[j++] = var_total[inic];
  var_origen[j] = '\0' ;
}
```

7 HOMONIMIAS
baja_homo.pc

```
/*****************************************************************************

PROGRAMA: Permite actualizar un ciudadano como homonimia lo cual causara una
          baja del Padrón Electoral
ENTRADAS: La clave electoral del ciudadano.
SALIDAS:  La cartografía del ciudadano asi como su domicilio.
TABLAS:   HOMONIMIAS, CATALOG, CARTOGRAFIA_1_TABLE
PROGRAMA REALIZADO POR: Jose Alfredo Zarza Ramírez.

*****************************************************************************/



#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "/usr2/zarza/fd/homonimias.h"
#include "/usr2/zarza/include/pantalla.h"
#include "/usr2/zarza/include/pan_io.h"
#include "/usr2/zarza/proc/oracle.h"

EXEC SQL BEGIN DECLARE SECTION;

        varchar pwd[10], uid[10];

        /*___ clave electo para catalog ___*/
        VARCHAR alfax[7];
        int fechx;
        int lugax;
      VARCHAR sexox[2];
        int digix;
        int  homox;
        int  formx;
```

```
        int esta;
        int dist;
        int muni;
        int secc;
        int loca;
        VARCHAR manz[3];

        VARCHAR mate[32];
        VARCHAR pate[32];
        VARCHAR nomb[32];

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;


main(){

        int register i, cont_ciud;
        int resp, pasa;

        strcpy(uid.arr,"ops$dbareg"); uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg"); pwd.len=strlen(pwd.arr);

        /* intenta la connexión a oracle */

        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;


        inicio_curses();
        init_kinf();

    pasa = passwd();
        if (pasa == NO_OK){
                fin_curses();
                exit ( 0 );
        }
```

```
        while (resp != IS_K3){
                clear();
                m_pon_pantalla("DEPPAD","P1.0","VER 1.0",
                        "OPCION 1: BAJA A CIUDADANO POR HOMONIMIA");
                pantalla_inicial();
                atributo ( PAN_NORMAL );
                resp = captura_informacion();

        }

        fin_curses();

        exit(0);

        error:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error en tiempo de ejecucion :\n");
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                EXEC SQL ROLLBACK RELEASE;
                exit(1);
}

/*___ PASSWD ____*/
passwd(){

        char *password;
        int chances=3, cont;

        clear();

        for (cont= 0; cont < chances; cont++){
                m_mvprintf(5,2,"TECLEE PASSWD: ");
             atributo ( PAN_INVISIBLE );
                password = m_inputxy(17,5,80);
                atributo ( PAN_NORMAL );
                if ( ! strcmp (password, "regional") )
                        return ( OK );
        }
        return ( NO_OK );
}
```

/*___ Rutina que realiza la captura de la información de datos de entrada ___*/

```
int captura_informacion(){
        int resp, kbresult, sigue;
        register int i;

        for (i=0; i<11; i++)
                memset(var_menu[i].valor,'\0',sizeof(var_menu[i].valor));

        m_pon_hora();

        pmenu = &var_menu[0];

        resp = input_xy_cad_dig( pmenu );
        if ( resp == IS_K7 || resp == IS_K3 )
                return ( resp );

        else if ( resp == OK ){

                resp = saca_datos ();
                if ( resp != OK) {
                        mensaje_no_catalog ();
                        return ( IS_K7 );
                }

                else {

                        resp = m_confirma("DESEAS DARLO DE BAJA    SI/NO ");
                        if (resp == 1){
                                EXEC SQL INSERT INTO HOMONIMIAS
                                values(:
                                m_mvprintf(21,2,"BORRANDO REGISTRO");
                                sleep(2);
                        }
                        else
                                return ( IS_K7 );

                }
        }

}
```

/* Rutina que se encarga de sacar los datos de una persona con una clave
de elector determinada */

```
int saca_datos(){

 char buffer[18],nombrex[50];
 struct menu *ppmenu;
 unsigned int respuesta;

  /*___ Particiona la clave electoral ___*/
  memset (nombrex,'\0',sizeof(nombrex));
  substr(alfax.arr, var_menu[0].valor, 0, 5);
  alfax.len = strlen(alfax.arr);

  substr(buffer, var_menu[0].valor, 6, 11);
  fechx = atoi(buffer);

  substr(buffer, var_menu[0].valor, 12, 13);
  lugax = atoi(buffer);

  substr(sexox.arr, var_menu[0].valor,14, 14);
  sexox.len = strlen(sexox.arr);

  substr(buffer, var_menu[0].valor,15, 15);
  digix = atoi(buffer);

  substr(buffer, var_menu[0].valor, 16, 15);
  homox = atoi(buffer);

  esta = -1;
     /*___ Saca datos de un ciudadano ___*/
         EXEC SQL
         SELECT NUM_FORMA, APELLIDO_PATERNO, APELLIDO_MATERNO, NOMBRE,
              ESTADO, DISTRITO, MUNICIPIO, SECCION, LOCALIDAD, MANZANA
         INTO   :formx, :pate, :mate, :nomb,
              :esta, :dist, :muni, :secc, :loca, :manz
         FROM   CATALOG
         WHERE  ALFA_CLAVE_ELECTORAL = :alfax AND FECHA_NACIMIENTO = :fechx
              AND LUGAR_NACIMIENTO = :lugax AND SEXO = :sexox AND
              DIGIT_VERIFICADOR = :digix AND CLAVE_HOMONIMIA = :homox;
```

```
    if (esta <= 0)
            return(NO_OK);



    itoa(formx,var_menu[1].valor);
/*__ conjuntar nombre __*/
    pate.arr[pate.len] = '\0';
    mate.arr[mate.len] = '\0';
    nomb.arr[nomb.len] = '\0';
    m_quita_espacios( pate.arr );
    m_quita_espacios( mate.arr );
    m_quita_espacios( nomb.arr );
    strcpy  ( nombrex,pate.arr );
    strcat  ( nombrex," " );
    strcat  ( nombrex,mate.arr );
    strcat  ( nombrex," " );
    strcat  ( nombrex,nomb.arr );
    if (strlen( nombrex ) > 35 )
                    nombrex[35] = '\0';
    strcpy (var_menu[2].valor,nombrex);
    itoa ( esta,var_menu[3].valor );
    itoa ( dist,var_menu[4].valor );
    itoa ( muni,var_menu[5].valor );
    itoa ( secc,var_menu[6].valor );
    itoa ( loca,var_menu[7].valor );
manz.arr[manz.len] = '\0';
    strcpy ( var_menu[8].valor, manz.arr );

    atributo (PAN_INVERTIDO);
    for (ppmenu = &var_menu[1]; ppmenu->x >0; ppmenu++){
            if (ppmenu->tipo == 0){

            switch ( ppmenu->len ) {
            case 2:
            m_mvprintf(ppmenu->y+1,ppmenu->x,"%02d",atoi(ppmenu->valor));
            break;
            case 3:
            m_mvprintf(ppmenu->y+1,ppmenu->x,"%03d",atoi(ppmenu->valor));
            break;
            case 4:
            m_mvprintf(ppmenu->y+1,ppmenu->x,"%04d",atoi(ppmenu->valor));
            break;
            case 8:

            m_mvprintf(ppmenu->y+1,ppmenu->x,"%08d",atoi(ppmenu->valor));
```

```
                    break;
                    }

                    }
                    else
                    m_mvprintf(ppmenu->y+1,ppmenu->x,"%s",ppmenu->valor);
          }
     /* confirma no duplicados */
     for(ppmenu = &var_menu[9]; ppmenu->x > 0;){
                    respuesta = input_xy_cadena(ppmenu);
                    if (respuesta == IS_K3 || respuesta == IS_K7)
                              return( respuesta );
                    else if (respuesta == OK)
                              ppmenu++;
                    else if (respuesta == IS_TAB && ppmenu > &var_menu[9])
                              ppmenu--;
          }

          atributo(PAN_NORMAL);
          move(var_menu[1].y+1, var_menu[1].x);
          return (OK);
}




/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
void mensaje_no_catalog(){
                    atributo(7);
                    m_mvprintf(13,50,"NO EXISTE ESA CLAVE");
                    sleep(2);
                    atributo(PAN_NORMAL);
                    m_mvprintf(13,50,"                    ");
}
```

```
        /**** trae de edms_corrige la clave elctoral ****/
obtiene_clave()
  {
       memset (alfax.arr,'\0',sizeof(alfax.arr));
       fechx = 0;
       lugax = 0;
       memset (sexox.arr,'\0',sizeof(sexox.arr));
       digix = 0;
       homox = 0;
   EXEC SQL SELECT ALFA_CLAVE_ELECTORAL, FECHA_NACIMIENTO,
LUGAR_NACIMIENTO,
          SEXO, DIGIT_VERIFICADOR, CLAVE_HOMONIMIA
      INTO  :alfax, :fechx, :lugax, :sexox, :digix, :homox
      FROM  EDMS_CORRIGE
      WHERE ESTADO_ACT = :esta  AND DISTRITO_ACT = :dist  AND
         MUNICIPIO_ACT = :muni  AND SECCION_ACT = :secc AND
              LOCALIDAD_ACT = :loca  AND MANZANA_ACT = :manz AND
              MODIFICADO = 'N';

       obtiene_nombre();
  }


        /**** trae de la tabla CATALOG el nombre y apellido ****/
obtiene_nombre()
  {
       memset (pate.arr, '\0',sizeof(pate.arr));
       memset (mate.arr, '\0',sizeof(mate.arr));
       memset (nomb.arr, '\0',sizeof(nomb.arr));
   EXEC SQL SELECT APELLIDO_PATERNO, APELLIDO_MATERNO, NOMBRE
      INTO  :pate, :mate, :nomb
      FROM  CATALOG
      WHERE ALFA_CLAVE_ELECTORAL = :alfax AND FECHA_NACIMIENTO = :fechx
              AND LUGAR_NACIMIENTO = :lugax AND SEXO= :sexox  AND
        DIGIT_VERIFICADOR = :digix  AND  CLAVE_HOMONIMIA = :homox;

  return(OK);

  }
```

```
/* ___ pone mensaje al final de la linea ___ */
pon_mensaje1 (){
        atributo(PAN_INVERTIDO);
        m_centray(23,"      F3-REGRESAR MENU PRAL.      "
            "    F7-NO GRABAR Y CONTINUAR      ");
        atributo(PAN_NORMAL);
}

/*___ Pone mensaje sentellante al final de la pantalla ___*/
pon_mensaje2(){
        atributo(4);
        m_centray(21,"F3-REGRESAR MENU PRAL.   F5-GRABAR Y CONTINUAR"
            "    F7-NO GRABAR Y CONTINUAR");
        atributo(PAN_NORMAL);
}

/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
pon_mensaje_grabar(){
                atributo(7);
                m_mvprintf(13,54,"GRABANDO INFORMACION");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,54,"                ");
}


/*___ función que guarda en var_origen el substring de var_total desde
    la posición inicial hasta la posición final                ___*/

substr (char *var_origen, char var_total[], int inic, int final)
{
 int register j = 0;

 for (; inic <= final; inic++)
   var_origen[j++] = var_total[inic];

 var_origen[j] = '\0';
}
```

```
/*___ entero a alfanumerico ___*/

itoa (int n, char s[])
{
 int i,j,c;


        i=0;
        j=0;

        do{
                s[i++]= n%10+'0';
        }while ((n /= 10) > 0);

        for (i=0,j= strlen(s)-1; i<j; i++,j--)
                c= s[i], s[i]= s[j], s[j]= c;
}



mensaje_no_edms_corrige(){
                atributo(7);
                m_mvprintf(13,49,"CLAVE INVALIDA");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,49,"                    ");
}
```

8. CANCELA HOMONIMIA
canc_homo.pc

```
/*****************************************************************************

PROGRAMA: Permite realizar consultas a  un ciudadano que fue marcado como
          homonimia
ENTRADAS: La clave electoral del ciudadano.
SALIDAS:  La cartografía del ciudadano asi como su domicilio.
TABLAS:   HOMONIMIAS, CATALOG, CARTOGRAFIA_1_TABLE
PROGRAMA REALIZADO POR: Jose Alfredo Zarza Ramírez.

*****************************************************************************/



#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "/usr2/zarza/fd/homonimias.h"
#include "/usr2/zarza/include/pantalla.h"
#include "/usr2/zarza/include/pan_io.h"
#include "/usr2/zarza/proc/oracle.h"


EXEC SQL BEGIN DECLARE SECTION;

        varchar pwd[10], uid[10];

        /*___ clave elector para catalog ___*/
        VARCHAR alfax[7];
        int fechx;
        int lugax;
      VARCHAR sexox[2];
        int digix;
        int  homox;
        int  formx;



        int  esta;
```

```
        int dist;
        int muni;
        int secc;
        int loca;
        VARCHAR manz[3];

        VARCHAR prob_dup[2];
        VARCHAR nolocalizado[2];

        VARCHAR mate[32];
        VARCHAR pate[32];
        VARCHAR nomb[32];

        int contador;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;


main(){

        int register i, cont_ciud;
        int resp, pasa;

        strcpy(uid.arr,"ops$dbareg"); uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg"); pwd.len=strlen(pwd.arr);

        /* intenta la conneccion a oracle */

        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;


        inicio_curses();
        init_kinf();

    pasa = passwd();
        if (pasa == NO_OK){
                fin_curses();
                exit ( 0 );
        }
```

```
        while (resp != IS_K3){
                clear();
                m_pon_pantalla("DEPPAD","P1.0","VER 1.0",
                        "OPCION 2: CANCELACION DE BAJA POR HOMONIMIA");
                pantalla_inicial();
                atributo ( PAN_NORMAL );
                resp = captura_informacion();

        }

        fin_curses();

        exit(0);

        error:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error en tiempo de ejecucion :\n");
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                EXEC SQL ROLLBACK RELEASE;
                exit(1);
}

/*___ PASSWD ____*/
passwd(){

        char *password;
        int chances=3, cont;

        clear();

        for (cont= 0; cont < chances; cont++){
                m_mvprintf(5,2,"TECLEE PASSWD: ");
             atributo ( PAN_INVISIBLE );
                password = m_inputxy(17,5,80);
                atributo ( PAN_NORMAL );
                if ( ! strcmp (password, "regional") )
                        return ( OK );
        }
        return ( NO_OK );
}
```

/*___ Rutina que realiza la captura de la información de datos de entrada ___*/

```
captura_informacion(){
        int resp, kbresult, sigue;

        memset(var_menu[1].valor,'\0',sizeof(var_menu[1].valor));
        memset(var_menu[9].valor,'\0',sizeof(var_menu[9].valor));
        memset(var_menu[10].valor,'\0',sizeof(var_menu[10].valor));
        m_pon_hora();

        pmenu = &var_menu[0];

        resp = input_xy_cad_dig( pmenu );
        if ( resp == IS_K7 || resp == IS_K3 )
                return ( resp );

        else if ( resp == OK ){

                resp = saca_datos ();
                if ( resp != OK) {
                        return ( IS_K7 );
                }

                else {

                        resp = m_confirma("DESEAS CANCELAR LA BAJA   SI/NO ");
                        if (resp == 1)
                                cancela_homonimias();
                        return ( IS_K7 );

                }
        }

}


saca_datos()
{

 char buffer[18],nombre[32];
 struct menu *ppmenu;
 unsigned int respuesta;

  memset(var_menu[9].valor,'\0',sizeof(var_menu[9].valor));
  memset(var_menu[10].valor,'\0',sizeof(var_menu[10].valor));
```

```
/*___ Particiona la clave electoral ___*/
substr(alfax.arr, var_menu[0].valor, 0, 5);
alfax.len = strlen(alfax.arr);

substr(buffer, var_menu[0].valor, 6, 11);
fechx = atoi(buffer);

substr(buffer, var_menu[0].valor, 12, 13);
lugax = atoi(buffer);

substr(sexox.arr, var_menu[0].valor,14, 14);
sexox.len = strlen(sexox.arr);

substr(buffer, var_menu[0].valor,15, 15);
digix = atoi(buffer);

substr(buffer, var_menu[0].valor, 16, 17);
homox = atoi(buffer);

        memset (prob_dup.arr,'\0',sizeof(prob_dup.arr));
        prob_dup.len = strlen(prob_dup.arr);

        EXEC SQL SELECT PROB_DUP,DOCTOS_NO_LOC
        INTO :prob_dup,:nolocalizado
        FROM HOMONIMIAS
        WHERE ALFA_CLAVE_ELECTORAL=:alfax AND FECHA_NACIMIENTO=:fechx
AND
        LUGAR_NACIMIENTO=:lugax AND SEXO=:sexox AND DIGIT_VERIFICADOR=:digix
        AND CLAVE_HOMONIMIA=:homox;

        if ( strlen(prob_dup.arr) == 0){
                mensaje_no_homonimia();
                return( NO_OK );
        }

        esta = -1;

    /*___ Saca datos de un ciudadano ___*/
        EXEC SQL
        SELECT NUM_FORMA, APELLIDO_PATERNO, APELLIDO_MATERNO, NOMBRE,
            ESTADO, DISTRITO, MUNICIPIO, SECCION, LOCALIDAD, MANZANA
        INTO   :formx, :pate, :mate, :nomb,
            :esta, :dist, :muni, :secc, :loca, :manz
```

```
    FROM  CATALOG
    WHERE  ALFA_CLAVE_ELECTORAL = :alfax AND FECHA_NACIMIENTO = :fechx
        AND LUGAR_NACIMIENTO = :lugax AND SEXO = :sexox AND
        DIGIT_VERIFICADOR = :digix AND CLAVE_HOMONIMIA = :homox;


    if (esta <= 0)
            return( NO_OK );



    itoa(formx,var_menu[1].valor);
/*__ conjuntar nombre __*/
    pate.arr[pate.len] = '\0';
    mate.arr[mate.len] = '\0';
    nomb.arr[nomb.len] = '\0';
    m_quita_espacios( pate.arr );
    m_quita_espacios( mate.arr );
    m_quita_espacios( nomb.arr );
    strcpy  ( nombre,pate.arr );
    strcat  ( nombre," " );
    strcat  ( nombre,mate.arr );
    strcat  ( nombre," " );
    if (strlen( nombre ) > 30 ){
                    nombre[30] = '\0';
    } else {
            strcat  ( nombre,nomb.arr );
            if (strlen( nombre ) > 30 )
                    nombre[30] = '\0';
    }

    strcpy (var_menu[2].valor,nombre);

    itoa ( esta,var_menu[3].valor );
    itoa ( dist,var_menu[4].valor );
    itoa ( muni,var_menu[5].valor );
    itoa ( secc,var_menu[6].valor );
    itoa ( loca,var_menu[7].valor );
manz.arr[manz.len] = '\0';
    strcpy ( var_menu[8].valor, manz.arr );

prob_dup.arr[prob_dup.len] = '\0';
    strcpy ( var_menu[9].valor, prob_dup.arr );


nolocalizado.arr[nolocalizado.len] = '\0';
```

```
        strcpy ( var_menu[10].valor, nolocalizado.arr );

        atributo (PAN_INVERTIDO);
        for (ppmenu = &var_menu[1]; ppmenu->x >0; ppmenu++){
                if (ppmenu->tipo == 0){

                switch ( ppmenu->len ) {
                case 2:
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%02d",atoi(ppmenu->valor));
                break;
                case 3:
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%03d",atoi(ppmenu->valor));
                break;
                case 4:
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%04d",atoi(ppmenu->valor));
                break;
                case 8:
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%08d",atoi(ppmenu->valor));
                break;
                }

                }
                else
                m_mvprintf(ppmenu->y+1,ppmenu->x,"%s",ppmenu->valor);
        }

        atributo(PAN_NORMAL);
        move(var_menu[1].y+1, var_menu[1].x);
        return (OK);
}
```

```
/*_ Insertar ciudadano para baja por homonimias _*/
cancela_homonimias (){

        EXEC SQL DELETE FROM HOMONIMIAS
        WHERE
         ALFA_CLAVE_ELECTORAL = :alfax and
         FECHA_NACIMIENTO = :fechx and
         LUGAR_NACIMIENTO = :lugax and
         SEXO = :sexox and
         DIGIT_VERIFICADOR = :digix and
         CLAVE_HOMONIMIA = :homox;

        EXEC SQL COMMIT;
        return ( OK );

}



/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
mensaje_no_catalog(){
                atributo(7);
                m_mvprintf(13,50,"NO EXISTE ESA CLAVE");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,50,"                ");
}



        /**** trae de edms_corrige la clave electoral ****/
obtiene_clave()
  {
        memset (alfax.arr,'\0',sizeof(alfax.arr));
        fechx = 0;
        lugax = 0;
        memset (sexox.arr,'\0',sizeof(sexox.arr));
        digix = 0;
        homox = 0;
```

```
    EXEC SQL SELECT ALFA_CLAVE_ELECTORAL, FECHA_NACIMIENTO,
LUGAR_NACIMIENTO,
            SEXO, DIGIT_VERIFICADOR, CLAVE_HOMONIMIA
        INTO  :alfax, :fechx, :lugax, :sexox, :digix, :homox
        FROM  EDMS_CORRIGE
        WHERE ESTADO_ACT = :esta  AND DISTRITO_ACT = :dist  AND
            MUNICIPIO_ACT = :muni  AND SECCION_ACT = :secc AND
                LOCALIDAD_ACT = :loca  AND MANZANA_ACT = :manz AND
                MODIFICADO = 'N';

         obtiene_nombre();
  }



          /**** trae de la tabla CATALOG el nombre y apellido ****/
obtiene_nombre()
  {
         memset (pate.arr, '\0',sizeof(pate.arr));
         memset (mate.arr, '\0',sizeof(mate.arr));
         memset (nomb.arr, '\0',sizeof(nomb.arr));
    EXEC SQL SELECT APELLIDO_PATERNO, APELLIDO_MATERNO, NOMBRE
        INTO  :pate, :mate, :nomb
        FROM  CATALOG
        WHERE ALFA_CLAVE_ELECTORAL = :alfax AND FECHA_NACIMIENTO = :fechx
                AND LUGAR_NACIMIENTO = :lugax AND SEXO= :sexox  AND
          DIGIT_VERIFICADOR = :digix  AND  CLAVE_HOMONIMIA = :homox;

   return(OK);

  }


/* ___ pone mensaje al finel de la linea ___ */
pon_mensaje1 (){
         atributo(PAN_INVERTIDO);
         m_centray(23,"       F3-REGRESAR MENU PRAL.        "
              "    F7-NO GRABAR Y CONTINUAR       ");
         atributo(PAN_NORMAL);
}
```

```
/*___ Pone mensaje sentellante al final de la pantalla ___*/
pon_mensaje2(){
        atributo(4);
        m_centray(21,"F3-REGRESAR MENU PRAL.   F5-GRABAR Y CONTINUAR"
             "    F7-NO GRABAR Y CONTINUAR");
        atributo(PAN_NORMAL);
}

/*___ Pone el mensaje de espera por que los datos se estan grabando ___*/
pon_mensaje_grabar(){
                atributo(7);
                m_mvprintf(13,54,"GRABANDO INFORMACION");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,54,"                ");
}



/*___ funcion que guarda en var_origen el substring de var_total desde
    la posición inicial  hasta la posición final                ___*/

substr (char *var_origen, char var_total[], int inic, int final)
{
 int register j = 0;

 for (; inic <= final; inic++)
   var_origen[j++] = var_total[inic];

 var_origen[j] = '\0';
}


/*___ entero a alfanumerico ___*/

itoa (int n, char s[])
{
 int i,j,c;


        i=0;
        j=0;
```

```
        do{
                s[i++]= n%10+'0';
        }while ((n /= 10) > 0);

        for (i=0,j= strlen(s)-1; i<j; i++,j--)
                c= s[i], s[i]= s[j], s[j]= c;
}


mensaje_no_edms_corrige(){
                atributo(7);
                m_mvprintf(13,49,"CLAVE INVALIDA");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,49,"                          ");
}

mensaje_no_homonimia(){
                atributo(7);
                m_mvprintf(13,49,"CLAVE NO DADA DE BAJA");
                sleep(2);
                atributo(PAN_NORMAL);
                m_mvprintf(13,49,"                  ");
}
```

CAPITULO VI

SISTEMA DE COMPUTO PARA REALIZAR LA RENUMERACION DE SECCIONES ELECTORALES EN LA BASE DE DATOS DEL PADRON ELECTORAL NACIONAL

- VI.1 INTRODUCCION
- VI.2 SISTEMA PARA LA CAPTURA DE SECCIONES RESULTANTES
- VI.3 MENUS, REPORTES Y PANTALLAS DE CAPTURA.

VI. 1 INTRODUCCION.

A partir del proceso de consolidación del marco cartográfico electoral, se fusionaron y se crearon nuevas cartografías. El sistema de renumeración de secciones trata de numerar la nueva cartografía y en especifico las secciones.
Este proceso cambia la sección de 3 dígitos a una de 4 dígitos, y esto da un mejor control, las secciones pasaron a ser consecutivas, no es necesario conocer toda la cartografía para conocer la sección, ahora basta con la sección para conocer su ubicación.

El sistema dió la solución a dicho problema, realizando un catálogo cartográfico nuevo, y con una pantalla de captura permitió insertar la sección resultante, de esta forma se conformó un mapeo entre una sección de 3 dígitos a una sección de 4 dígitos consecutiva.

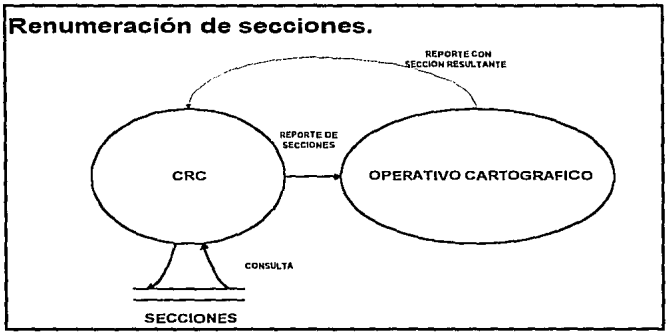
VI.2. SISTEMA PARA LA RENUMERACION DE SECCIONES

VI.2.1 AMBITO

VI.2.1.1 OBJETIVO:

- Desarrollar e instalar un sistema que sea capaz de generar un catálogo de la cartografía nueva y capturar la sección resultante.

VI.2.1.2 DIAGRAMA DE FLUJO DE DATOS GENERAL.



El Centro Regional de Cómputo es responsable de generar la tabla de secciones y de emitir un reporte de dicho catálogo.
Una vez que el operativo cartográfico ha revisado a detalle el reporte lo envía anotando la sección resultante correspondiente.
El Centro Regional de Cómputo, captura dichas secciones resultantes generando así el catálogo de secciones continuas.

VI.2.1.3 HARDWARE Y SOFTWARE.

El hardware que se utilizó para el desarrollo de este sistema fue una IBM RISC/6000 con AIX ver 3.1 como sistema operativo, con una capacidad de 96 Mb y 28 Gb en memoria y disco respectivamente. Conectada a la red nacional del Registro Federal de Electores.

El software que se utilizó fue SQL*FORMS, SQL*PLUS, SQL*REPORT y PRO*C. Todas estas, herramientas que ORACLE nos proporciona para explotar la base de datos.

VI.2.1.4 BASE DE DATOS DEFINIDA.

La base de datos esta definida en los Centros Regionales de la siguiente forma.

TABLESPACE_NAME	MAX(BYTES)	SUM(BYTES)
SYSTEM	48,969,696	57,909,191
RBS	100,000,000	100,000,000
TEMPORAL	54,212,567	642,32,526
INDI_CAR	71,099,425	81,189,568
CARTOGRA	123,121,568	134,123,456

Este tipo de configuración fue tratada ampliamente en el capitulo IV para el sistema de consolidación del marco cartográfico.

VI.2.1.5 BASE DE DATOS PARA EL SISTEMA.

Las tablas e índices que se utilizaron en el diseño son las siguientes;

NOMBRE TABLA	DUEÑO	TABLESPACE	INITIAL (Mb)
SECCIONES	OPS$DBAREG	CARTOGRA	20
EDMS_PDI2	OPS$DBAREG	CARTOGRA	20
EDMS_CORRIGE2	OPS$DBAREG	CARTOGRA	10

La tabla EDMS_PDI2 contiene un catálogo de la nueva cartografía derivado de la tabla EDMS_PDI (descrita ampliamente en el capítulo anterior). Tiene la siguiente descripción.

ESTADO	NUMBER(2)
DISTRITO	NUMBER(2)
MUNICIPIO	NUMBER(3)
SECCION	NUMBER(3)
LOCALIDAD	NUMBER(4)
MANZANA	NUMBER(2)
NUM_PADRON	NUMBER(4)

Contiene la cartografía completa, pero ésta es la cartografía nueva sacada de la tabla EDMS_PDI. También tiene un campo NUM_PADRON que indica el número de ciudadanos en dicha cartografía.
Así mismo como se generó un catálogo de cartografía nueva para EDMS_PDI, se realizó igualmente para EDMS_CORRIGE para complementar totalmente la cartografía. Para esto se creó una tabla llamada EDMS_CORRIGE2 la cual tiene la siguiente descripción.

ESTADO	NUMBER(2)
DISTRITO	NUMBER(2)
MUNICIPIO	NUMBER(3)
SECCION	NUMBER(3)
NUM_PADRON	NUMBER(4)

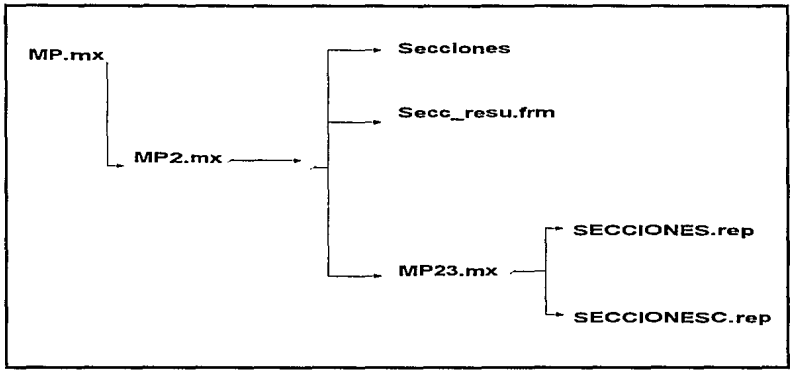
La tabla secciones, es la que contiene el mapeo de la sección de 3 dígitos a la sección de 4 dígitos, tiene la siguiente descripción. La cartografía se llena utilizando las tablas EDMS_PDI2 y EDMS_CORRIGE2 las cuales contienen el catálogo de la nueva cartografía.

ESTADO	NUMBER(2)
DISTRITO	NUMBER(2)
MUNICIPIO	NUMBER(3),
SECCION	NUMBER(3)
NUM_PADRON	NUMBER(4),
TIPO	CHAR(1),
SECC_RESU	NUMBER(4)

Como podemos observar, contiene la cartografía hasta sección, luego NUM_PADRON es el número de ciudadanos en dicha sección, el campo TIPO indica el tipo de sección (1 urbana, 2 rural, 3 mixta) y por último SECC_RESU no indica la sección resultante.
Es importante notar que se mapea una sección de 3 dígitos a una de 4 dígitos, esto por que la sección resultante es consecutiva e independiente del total del marco cartográfico.

VI.2.2 DESCRIPCIÓN DEL DISEÑO.
VI.2.2.1 INTERFACES Y MENUS.

El diagrama de los menús es el siguiente:



Para mayor ilustración ver las figuras en la opción de MENUS al final de este capítulo.

La notación de los menús MP*xy1y2*.. significa:

x. Opción elegida del menú principal

y1,y2. Opciones seleccionadas en los menús subsecuentes.

VI.2.3 MODULOS.

VI.2.3.1 CREACION DE TABLAS PARA OPERACION DEL SISTEMA.

CREA EDMS_PDI2.

```
create table EDMS_PDI_2
(
        ESTADO          NUMBER(2) NOT NULL,
        DISTRITO  NUMBER(2) NOT NULL,
        MUNICIPIO NUMBER(3) NOT NULL,
        SECCION   NUMBER(3) NOT NULL,
        LOCALIDAD       NUMBER(4) NOT NULL,
        MANZANA NUMBER(2) NOT NULL,
        NUM_PADRON      NUMBER(4) NOT NULL
)
tablespace cartogra
storage (initial 20M next 1M)
```

CREA EDMS_CORRIGE2.

```
create table EDMS_CORRIGE_3
(
        ESTADO          NUMBER(2) NOT NULL,
        DISTRITO  NUMBER(2) NOT NULL,
        MUNICIPIO NUMBER(3) NOT NULL,
        SECCION   NUMBER(3) NOT NULL,
        NUM_PADRON      NUMBER(4) NOT NULL
)
tablespace cartogra
storage (initial 10M next 1M)
```

CREA SECCIONES

```
create table SECCIONES
(
        ESTADO      NUMBER(2) NOT NULL,
        DISTRITO    NUMBER(2) NOT NULL,
        MUNICIPIO   NUMBER(3) NOT NULL,
        SECCION     NUMBER(3) NOT NULL,
        NUM_PADRON NUMBER(4) NOT NULL,
        TIPO        CHAR(1),
        SECC_RESU   NUMBER(4)
)
tablespace cartogra
storage (initial 20M next 2M)
```

VI.2.3.2 REPORTES Y PANTALLAS DE CAPTURA.

Los reportes que se utilizan para esta aplicación son los siguientes:

1. **REPORTE DE SECCIONES(SECCIONES.rep).** Este reporte contiene el catálogo de secciones tomando en cuenta la nueva cartografía.

INSTITUTO FEDERAL ELECTORAL
REGISTRO FEDERAL DE ELECTORES
COORDINACION DE INFORMATICA

REPORTE DE SECCIONES

ESTADO	DISTRITO	MUNICIPIO	SECCIÓN	NUM_CIUD	SECC_RESU
9	1	15	40	779	
9	2	15	45	656	
9	3	15	30	1185	
9	4	17	52	1305	
9	5	17	51	1113	

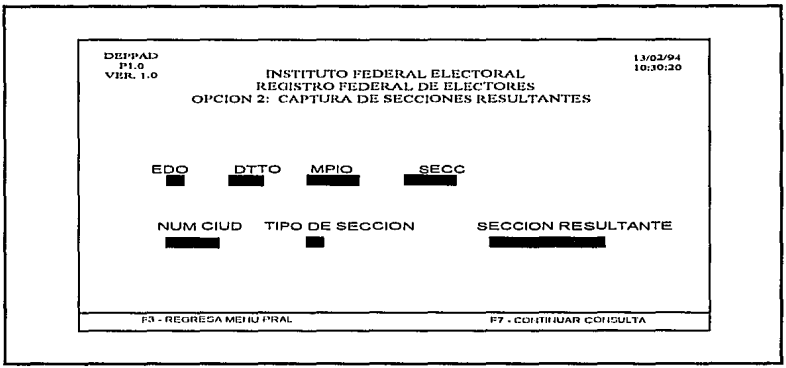
2. **REPORTE DE SECCIONES CON SU CORRESPONDIENTE SECCIÓN RESULTANTE(SECCIONESC.rep).** Este reporte contiene además de lo del anterior, la sección resultante.

INSTITUTO FEDERAL ELECTORAL
REGISTRO FEDERAL DE ELECTORES
COORDINACION DE INFORMATICA

REPORTE DE SECCIONES

ESTADO	DISTRITO	MUNICIPIO	SECCIÓN	NUM_CIUD	SECC_RESU
9	1	15	40	779	100
9	2	15	45	656	101
9	3	15	30	1185	102
9	4	17	52	1305	103
9	5	17	51	1113	104

La pantalla de captura que se utiliza es sólo para capturar la sección resultante y es la siguiente.



En esta pantalla sólo capturamos el ESTADO, DISTRITO, MUNICIPIO, SECCIÓN y nos aparece el número de ciudadanos (NUM CIUD) y el tipo de sección (TIPO DE SECCIÓN 1 urb 2 rur 3 mix), el cursor se posiciona en el campo SECCIÓN RESULTANTE para que capturar la sección resultante.

VI.2.4 INSTALACION

Una vez concluido el desarrollo y las pruebas del sistema, se procede a realizar el envío de los programas a los Centros Regionales a través de la red X.25 con la que cuenta el Registro Federal de Electores. Tomando en cuenta los siguientes pasos:

1. Estar como usuario root del host de desarrollo.
2. Localizarse en el directorio /usr2/zarza/carto/ SECCIONES
3. Teclear el siguiente comando.
   $ rcp SECCIONES.Z regionalx:/u/dbareg/carto/
4. Conectarse al regional y teclear lo siguiente.
   $ rlogin regional
   $ uncompress /u/dbareg/carto/SECCIONES.Z
   $ tar -xvpf /u/dbareg/carto/SECCIONES
5. Desconectarse del regional.
   $ exit

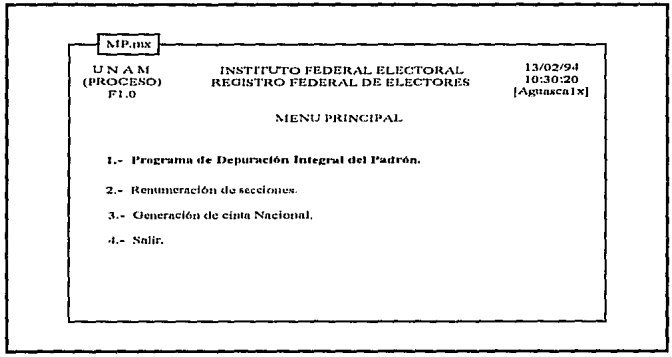
VI.2.5 COMENTARIOS.

Este proyecto permitió ubicar con más facilidad y rápidez las secciones electorales, debido a que las hizó independientes del conjunto cartográfico, pudiendo ubicar la sección, sin la necesidad de todo el conjunto cartográfico.
Este proceso tomó en cuenta todas y cada una de las secciones electorales del país. Numerando más de 63,000 secciones que componen al territorio nacional.

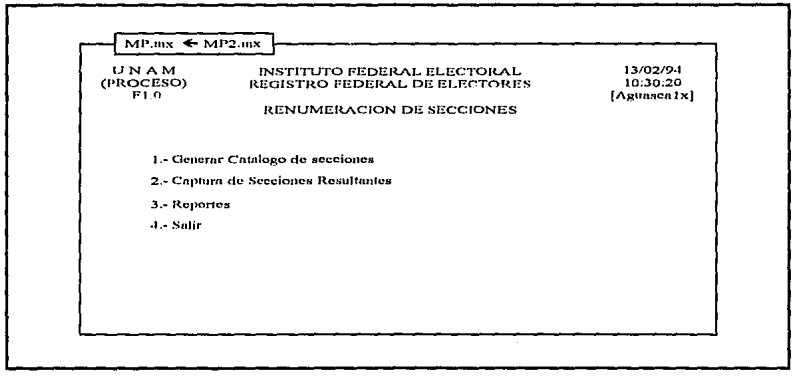
ANEXO

MENUS.

MP.mx



MP2.mx



PROGRAMAS

CREA TABLAS PARA EL CATALOGO DE SECCIONES
secc_rang.pc

```
/*___

        PROGRAMA: Generar tablas para la el catálogo de cartografía
                  nueva
        ENTRADA: Estado a procesar.
        SALIDA:  Ninguna
        TABLAS:  EDMS_PDI2, EDMS_CORRIGE2, EDMS_PDI, EDMS_CORRIGE
        PROGRAMA REALIZADO POR: Jose Alfredo Zarza Ramírez.

___*/


#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include "/usr2/zarza/proc/oracle.h"


EXEC SQL BEGIN DECLARE SECTION;

        varchar pwd[10], uid[10];

        /*___ clave elector para catalog ___*/
        int estado[20000];
        int distrito[20000];
        int municipio[20000];
        int seccion[20000];
        int localidad[20000];
        int manzana[20000];
        int num_padron[20000];
```

```
        int estadox;
        int distritox;
        int municipiox;
        int seccionx;
        int localidadx;
        int manzanax;
        int num_padronx, num_padrony;
        int estadoy;
        int distritoy;
        int municipioy;
        int secciony;
        int localidady;
        int manzanay;
        int contador;
        int tot_num_padron;

        char programa[100]="sqlplus / @/u/cartografia/spool/crea_edms_2 ";
        char estadoxx[4]="";
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;


/*___
        PROGRAMA PRINCIPAL
                        ___*/

main(){

   int register i,j;
   int actualiza,inserta;
        for (i=0; i<20000; i++){
                estado[i]=-1;
                distrito[i]=-1;
                municipio[i]=-1;
                seccion[i]=-1;
                localidad[i]=-1;
                manzana[i]=-1;
                num_padron[i]=0;
        }

        strcpy(uid.arr,"ops$dbareg"); uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg"); pwd.len=strlen(pwd.arr);
```

```
/* intenta la connexión a oracle */

EXEC SQL WHENEVER SQLERROR GOTO error;
EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;



system("clear");
printf("Estado a generar ? ");
scanf("%d",&estadox);

itoa(estadox,estadoxx);
strcat (programa,estadoxx);

inserta= 0;
actualiza= 0;

EXEC SQL DELETE FROM EDMS_PDI_2
WHERE ESTADO=:estadox;

EXEC SQL COMMIT;
system(programa);

EXEC SQL SELECT ESTADO,DISTRITO,MUNICIPIO,SECCION,LOCALIDAD,
MANZANA,NUM_PADRON
INTO :estado,:distrito,:municipio,:seccion,:localidad,
manzana,:num_padron
FROM EDMS_CORRIGE_2
WHERE ESTADO= :estadox;

EXEC SQL SELECT COUNT(*)
INTO :contador
FROM EDMS_CORRIGE_2
WHERE ESTADO=:estadox;

printf("\ncontador :%d",contador);

for (i=0; i<contador; i++){
```

```
estadox= estado[i];
distritox = distrito[i];
municipiox = municipio[i];
seccionx  = seccion[i];
localidadx = localidad[i];
manzanax  = manzana[i];
num_padronx= num_padron[i];

num_padrony  = 999;
estadoy = 0;
distritoy = 0;
municipioy = 0;
seccion y = 0;

EXEC SQL SELECT NUM_PADRON
INTO :num_padrony
FROM EDMS_PDI_2
WHERE ESTADO=:estadox and DISTRITO=:distritox and
MUNICIPIO=:municipiox and SECCION=:seccionx and
LOCALIDAD=:localidadx and MANZANA=:manzanax;

/* Si esta en corrige_3 y no en secciones */
if (num_padrony==999){
        EXEC SQL INSERT INTO EDMS_PDI_2
        (ESTADO,DISTRITO,MUNICIPIO,SECCION,LOCALIDAD,
         MANZANA,NUM_PADRON)
        VALUES(:estadox,:distritox,:municipiox,:seccionx,
                :localidadx,:manzanax,:num_padronx);

        printf("\ninserto %d %d %d %d %d %d     %d",estadox,
        distritox,municipiox,seccionx,localidadx,manzanax
        ,num_padronx);
        inserta++;
} else{

        EXEC SQL UPDATE EDMS_PDI_2
        SET NUM_PADRON=NUM_PADRON + :num_padronx
        WHERE ESTADO=:estadox and DISTRITO=:distritox and
        MUNICIPIO=:municipiox and SECCION=:seccionx  and
        LOCALIDAD=:localidadx and MANZANA=:manzanax;
```

```
                    printf("\nactuali %d %d %d %d %d %d    %d",estadox,
                    distritox,municipiox,seccionx,
                    localidadx,manzanax,num_padronx);
                    actualiza++;
                }
                tot_num_padron = tot_num_padron + num_padronx;
                num_padronx = 0;

                EXEC SQL COMMIT;

        }

        EXEC SQL COMMIT;
        printf("\n\n\nregistros en edms_pdi_2 %d",contador);
        printf("\nregistros insertados edms_pdi_2 %d",inserta);
        printf("\nregistros actualizados edms_pdi_2 %d",actualiza);
        printf("\nnumero de ciudadanos en actualizados  %d",tot_num_padron);


        exit(0);

        error:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error en tiempo de ejecucion :\n");
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                printf("\nerror en %d %d %d %d %d %d     %d",estadox,
                distritox,municipiox,seccionx,localidadx,manzanax,num_padronx);
                EXEC SQL ROLLBACK RELEASE;
                exit(1);
}


itoa (int n, char *s)
{
 int i,j,c;


        i=0;
        j=0;
```

```
        do{
                s[i++]= n%10+'0';
        }while ((n /= 10) > 0);
        s[i]='\0';

        for (i=0,j= strlen(s)-1; i<j; i++,j--)
                c= s[i], s[i]= s[j], s[j]= c;
}

lpad (char *c,int l){

        while ( strlen(c) < l ){
                strcpy (&c[1],&c[0]);
                c[0] = '0';
        }
}
```

□

CREA EL CATALOGO DE SECCIONES
secciones.pc

```
/*___

        PROGRAMA: Crea el catálogo de secciones (SECCIONES) a partir de
                las tablas del proceso secc_rang.pc (EDMS_PDI_2)
        ENTRADA:  Estado a procesar
        SALIDA:   Ninguna
        TABLAS:   EDMS_PDI_2, EDMS_CORRIGE_2, SECCIONES.
        PROGRAMA REALIZADO POR : Jose Alfredo Zarza Ramírez.

____*/


#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "/usr2/zarza/proc/oracle.h"


EXEC SQL BEGIN DECLARE SECTION;

        varchar pwd[10], uid[10];

        /*___ clave elector para catalog ___*/
        int estado[30000];
        int distrito[30000];
        int municipio[30000];
        int seccion[30000];
        int num_padron[30000];

        int estadox;
        int distritox;
        int municipiox;
        int seccionx;
        int num_padronx, num_padrony;
        int estadoy;
        int distritoy;
        int municipioy;
```

```
        int seccionv;
        int contador;

        /* estado que se va a procesar */
        int estadoxy;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;


/*___
        PROGRAMA PRINCIPAL
                        ___*/

main(){

    int register i,j;
    int actualiza,inserta;

        strcpy(uid.arr,"ops$dbareg"); uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg"); pwd.len=strlen(pwd.arr);

        /* intenta la connexión a oracle */

        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;



        inserta= 0;
        actualiza= 0;
        system("clear");
        system("more /u/cartografia/spool/secciones.help");
        printf("H <enter> @");
        getchar();
        system("clear");
        printf("\n\nEstado que vas a procesar?_ ");
        scanf("%d",&estadoxy);

        EXEC SQL DELETE FROM SECCIONES
        WHERE ESTADO=:estadoxy;

        EXEC SQL COMMIT;


        EXEC SQL INSERT INTO SECCIONES
```

```
(estado,distrito,municipio,seccion,num_padron)
SELECT estado,distrito,municipio,seccion,sum(num_padron)
FROM edms_pdi_2
WHERE estado=:estadoxy
GROUP BY estado,distrito,municipio,seccion;

EXEC SQL COMMIT;


exit(0);

error:
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        printf("Error en tiempo de ejecucion :\n");
        printf("%s\n",sqlca.sqlerrm.sqlerrmc);
        EXEC SQL ROLLBACK RELEASE;
        exit(1);
```

INSTITUTO FEDERAL ELECTORAL

CAPITULO VII

SISTEMA DE COMPUTO PARA LA GENERACION DE INFORMACION PARA LA IMPRESION DE LA CREDENCIAL PARA VOTAR CON FOTOGRAFIA EN CINTA DE 8mm. DE 2.3 Gb EN FORMATO ASCII

- VII.1 INTRODUCCION
- VII.2 PREPARACION DEL AMBIENTE ORACLE PARA LA OPERACION DEL PROGRAMA
- VII.3 MENUS, REPORTES, PROGRAMAS Y PANTALLAS DE CAPUTURA.

VII. 1. INTRODUCCION

Dando respuesta a la demanda social y política, de contar con instrumentos electorales mas confiables, se instrumentó el proyecto de una credencial para votar con fotografía, contando con:

- Un sello a manera de holograma con el logotipo del Padrón Electoral que cubre parcialmente la parte inferior izquierda de la fotografía.
- Fotografía del titular.
- Integración molecular de sus componentes que destruyen la credencial ante cualquier intento de alteración.
- Firma.
- Un código de barras cubierto por una banda negra sensible a la luz infrarroja.
- Un número único consecutivo que sirve como número de control individual para cada credencial.
- Huella digital.

Tal como lo muestra la siguiente figura.

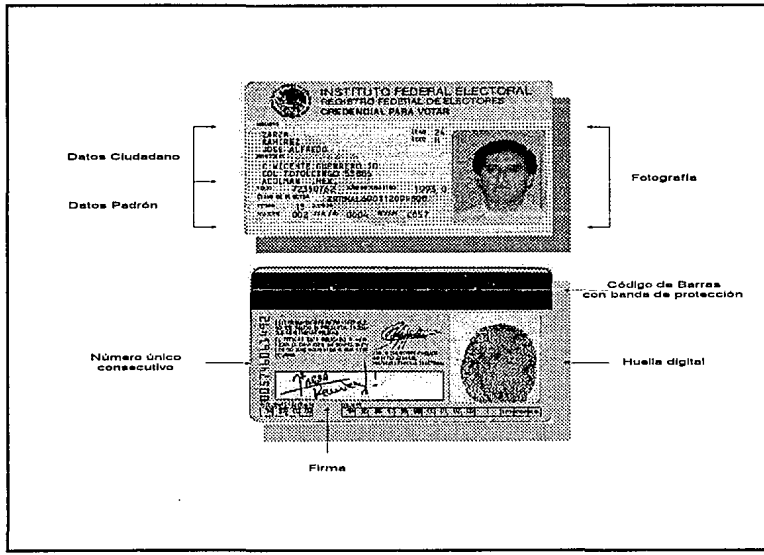


FIGURA VII.1 CREDENCIAL PARA VOTAR CON FOTOGRAFIA.

A través de una licitación pública internacional se logró contar con un proveedor (POLAROID) para la impresión de las micas de la credencial, recibos y equipos necesarios para los módulos. El Registro Federal de Electores tenía la misión de entregar una cinta de 8mm. de 2.1 Gb con registros procesados y dispuestos para su impresión.

El sistema solventó este proyecto procesando más de 40,000,000 de registros y a su vez aplicando todos los programas de depuración realizadas en proyectos anteriores.

El sistema será descrito de la siguiente forma:

1.0. Ambito.
- 1.1. Objetivos del sistema.
- 1.2. Hardware, software de desarrollo.
- 1.4. Base de datos definida.

2.0. Descripción del diseño.
- 2.1. Interfaces.
  - 2.1.1. Menús.
  - 2.1.2. Pantallas.

3.0. Módulos.
- Para cada módulo.
- 3.1. Texto explicativo .
- 3.2. Organización de los datos.

4.0. Instalación del sistema.
5.0. Comentarios.

VII.2 GENERACION DE CINTA PARA LA CREDENCIAL PARA VOTAR CON FOTOGRAFIA.

VII.2.1 AMBITO

VII.2.1.1 OBJETIVO.

Diseñar una sistema para generar una cinta con información lista a ser impresa para la credencial para votar con fotografía, aplicando todos los proyectos de depuración.

VII.2.1.2 DIAGRAMA DE FLUJO DE DATOS DEL SISTEMA.

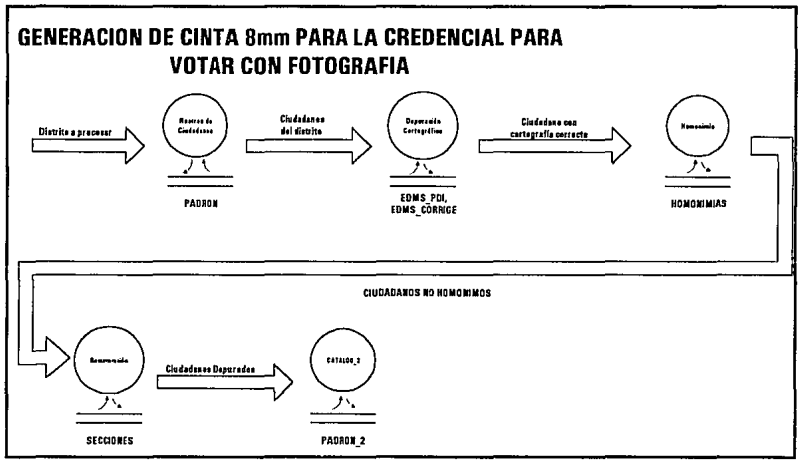


FIGURA VII.2 DIAGRAMA DE FLUJO DE DATOS DEL SISTEMA.

Como podemos observar en el diagrama, tenemos como entrada el distrito electoral a procesar, si el distrito es válido se procede a obtener todos los ciudadanos pertenecientes a dicho distrito, se aplica un proceso de corrección cartográfica, con lo cual se obtiene la ubicación correcta de los ciudadanos. Posteriormente pasan por un filtro de homonimias, esto es, en el caso de que un ciudadano sea homonimia no pasa a la siguiente fase. Después se actualiza a su sección correspondiente.
Una vez que los ciudadanos pasaron por estos filtros, se genera la cinta correspondiente y se entrega a Polaroid para la impresión de la mica de la credencial

VII.2.1.3. HARDWARE Y SOFTWARE

El hardware que se utilizó para el desarrollo del sistema es una computadora IBM RISC/6000 con AIX 3.1 y ORACLE 6.0.37, con una capacidad de almacenamiento en disco de 28 Gb y una capacidad en memoria de 96 Mb.

El software que se utilizó fue PRO*C y SQL*PLUS. Estas son herramientas propias de ORACLE.

VII.2.1.4. BASE DE DATOS DEFINIDA.

TABLESPACE_NAME	MAX(BYTES)	SUM(BYTES)
SYSTEM	48,969,696	57,909,191
RBS	100,000,000	100,000,000
TEMPORAL	54,212,567	64,232,526
INDI_CAR	71,099,425	81,189,568
CARTOGRA	123,121,568	134,123,456

Como podemos observar existen 5 espacios para tablas, el espacio para tablas SYSTEM es de uso especial, lo utiliza ORACLE para almacenar tablas propias de su operación. El tablespace RBS es utilizado para almacenar las últimas operaciones, datos y resultados de los querys de usuario, a este espacio se le llama espacio para ROLL-BACK. El espacio de tabla TEMPORAL es utilizado por ORACLE para almacenar temporalmente una serie de información que es generada por ciertas operaciones de los usuarios, tal como un ordenamiento, una búsqueda extensa, etc. El espacio de tabla INIDI_CAR es utilizado para almacenar todos los índices que se generen en las tablas de cartografía, las tablas se almacenan en CARTOGRA.

VII.2.1.5 DESCRIPCION DE LA BASE DE DATOS DEL SISTEMA.

Las tablas e índices que se utilizaron en el diseño son las siguientes;

NOMBRE TABLA	DUEÑO	TABLESPACE	INITIAL (Mb)
SECCIONES	OPS$DBAREG	CARTOGRA	5
GEOGRAFIA_1_TABLE	OPS$DBAREG	CARTOGRA	2
EDMS_PDI	OPS$DBAREG	CARTOGRA	40
HOMONIMIAS	OPS$DBAREG	CARTOGRA	30
CATALOG	OPS$DBAREG	CARTOGRA	300
LDI_01_CAPTURA	OPS$DBAREG	CARTOGRA	30
LDI_FOLIOS_CORRECTOS	OPS$DBAREG	CARTOGRA	30
EDMS_CORRIGE	OPS$DBAREG	CARTOGRA	30
CATALOG_2	OPS$DBAREG	CARTOGRA	80

En este caso la tabla que contiene la información que será almacenada en cinta es CATALOG_2, todas las demás almacenan información de los sistemas de depuración del padrón, la tabla CATALOG_2 tiene la siguiente descripción.

ALFA_CLAVE_ELECTORAL	CHAR(6)
FECHA_NACIMIENTO	NUMBER(6)
LUGAR_NACIMIENTO	NUMBER(2)
SEXO	CHAR(1)
DIGIT_VERIFICADOR	NUMBER(1)
CLAVE_HOMONIMIA	NUMBER(2)
NUMERO_CREDENCIAL	NUMBER(1)
MUNICIPIO	NUMBER(3)
MANZANA	NUMBER(3)
CALLE	CHAR(32)
NOMBRE	CHAR(32)
APELLIDO_PATERNO	CHAR(32)
APELLIDO_MATERNO	CHAR(32)
COLONIA	CHAR(32)
ESTADO	NUMBER(2)
DISTRITO	NUMBER(2)
NOMBRE_MUNICIPIO	CHAR(32)
LOCALIDAD	NUMBER(4)
SECCION	NUMBER(4)
CONSECUTIVO	NUMBER(5)
MODULO	NUMBER(4)

EDAD	NUMBER(3)
FOLIO	NUMBER(8)
ANO_DE_REGISTRO	NUMBER(4)
MODIFICADO	CHAR(1)

La tabla CATALOG_2 esta compuesta por la clave electoral que se integra de:

- ALFA_CLAVE_ELECTORAL
- FECHA_NACIMIENTO
- LUGAR_NACIMIENTO
- SEXO
- DIGIT_VERIFICADOR
- CLAVE_HOMONIMIA

y también por la cartografía:

- ESTADO
- DISTRITO
- MUNICIPIO
- SECCION
- LOCALIDAD
- MANZANA

, datos personales como:

- NOMBRE
- APELLIDO PATERNO
- APELLIDO MATERNO
- NOMBRE
- CALLE
- COLONIA
- EDAD
- NOMBRE_MUNICIPIO

, datos de control:

- CONSECUTIVO
- FOLIO
- MODULO
- ANO_DE_REGISTRO

El dato de control CONSECUTIVO es un número consecutivo sectorizado, es decir, que para cierta entidad se asignó un rango definido de números consecutivos.
El dato FOLIO , es el folio padrón asignado desde que el ciudadano fue dado de alta en el padrón electoral.
El dato MODULO, es el módulo físico al que le corresponde dicha credencial.
Y AÑO_DE_REGISTRO es el campo que contiene la fecha en que el ciudadano se inscribió al Padrón Electoral.

VII.2.2 DESCRIPCION DEL DISEÑO.

VII.2.2.1 INTERFACES Y MENUS

La estructura del menú, para la generación de cinta es el siguiente.

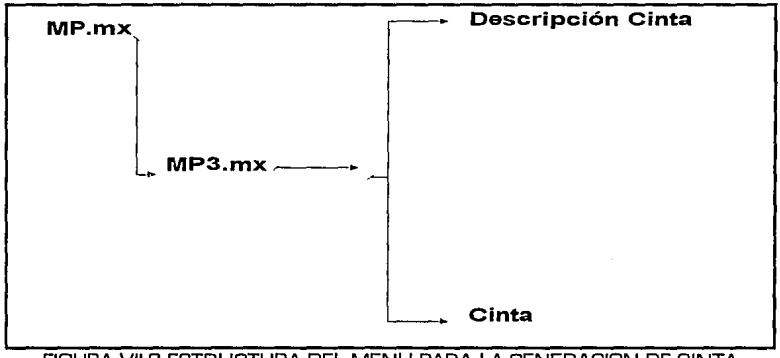


FIGURA VII.3 ESTRUCTURA DEL MENU PARA LA GENERACION DE CINTA

VII.2.3 MODULOS

VII.2.3.1 CREACION DE TABLAS PARA LA OPERACION DEL SISTEMA

CREA CATALOG_2.

```
CREATE TABLE CATALOG_2
(
        ALFA_CLAVE_ELECTORAL        CHAR (6) NOT NULL,
        FECHA_NACIMIENTO            NUMBER (6) NOT NULL,
        LUGAR_NACIMIENTO            NUMBER (2) NOT NULL,
        SEXO                        CHAR(1) NOT NULL,
        DIGIT_VERIFICADOR           NUMBER (1) NOT NULL,
        CLAVE_HOMONIMIA             NUMBER (2) NOT NULL,
        NUMERO_CREDENCIAL           NUMBER (1),
        MUNICIPIO                   NUMBER (3),
        MANZANA                     NUMBER (3),
        CALLE                       CHAR (32),
        NOMBRE                      CHAR (32),
        APELLIDO_PATERNO            CHAR (32),
        APELLIDO_MATERNO            CHAR  (32),
        COLONIA                     CHAR  (32),
        ESTADO                      NUMBER  (2),
        DISTRITO                    NUMBER  (2),
        NOMBRE_MUNICIPIO            CHAR  (32),
        LOCALIDAD                   NUMBER  (4),
        SECCION                     NUMBER  (4),
        CONSECUTIVO                 NUMBER  (5),
        MODULO                      NUMBER  (4),
        EDAD                        NUMBER  (3),
        FOLIO                       NUMBER  (8) NOT NULL,
        ANO_DE_REGISTRO             NUMBER  (4) ,
        MODIFICADO                  CHAR  (1)

)
TABLESPACE ZARZA
STORAGE (INITIAL 60M NEXT 1M)
```

Como se vió en la figura anterior el programa contiene dos módulos.
El primero es una ayuda para que el usuario comprenda el proceso que se lleva a cabo para la operación del sistema. Esta ayuda describe en tareas principales el proceso que se lleva a cabo para la generación de cinta.

El shell de esta opción es el siguiente:

```
clear
echo "          GENERACION DE CINTA  "
echo ""
echo "La generación de la cinta para Centro Nacional de Cómputo,"
echo "se realiza de la siguiente forma:"
echo ""
echo ""
echo ""
echo "1.- Se hace un cruce de las tablas EDMS_PDI vs"
echo "    CATALOG para sacar los datos correspondientes"
echo "    de cada ciudadano(CATALOG) para cada manzana(PDI),"
echo "    la información se almacena en la tabla CATALOG_2."
echo "    Para mayor información de que datos son los que se"
echo "    utilizan de CATALOG y EDMS_PDI ver la última página."
echo ""
echo "2.- Se crea un índice único en  CATALOG_2 por clave"
echo "    electoral para garantizar consistencia en la base,"
echo "    esto para no permitir ciudadanos que ya se insertaron"
echo "    de EDMS_PDI sean insertados de EDMS_CORRIGE."
echo ""
read xx
echo ""
echo ""
echo ""
echo ""
echo ""
echo "Recomendación:"
echo "Es importante tomar nota de cuantos ciudadanos "
echo "son insertados en la tabla CATALOG_2 hasta este"
echo "paso para llevar un control de la información. "
echo "El programa muestra cuantos ciudadanos "
echo "altera para cada proceso."
echo ""
echo ""
echo "3.- Se realiza un Join de las tablas EDMS_CORRIGE vs"
echo "    CATALOG para completar los datos correspondientes de "
```

```
echo "    cada ciudadano."
echo ""
read xx
echo "3.1. Para este caso se puede presentar un error"
echo "     y es que ya se haya insertado algun(os) "
echo "     ciudadano(s) de EDMS_PDI y además esten "
echo "     capturados en EDMS_CORRIGE, esto debido a"
echo "     que en algun momento se contaba(n) con manzana(s)"
echo "     con validar igual a 'N' y se procedió a su"
echo "     captura  y después Cartografía les "
echo "     pidió que cambiaran esa manzana a tipo "
echo "     'C' o 'V', y solo procedieron a realizar"
echo "     el cambio en EDMS_PDI mas no en EDMS_CORRIGE."
echo ""
echo "4.- Se procede una vez que ya estan todos los ciudadanos, "
echo "    a actualizar su sección a la sección resultante que "
echo "    se  encuentra en la tabla de SECCIONES."
read xx
echo ""
echo "Recomendación:"
echo "Por favor compare cuantos ciudadanos actualizó la    "
echo "seccion y cuantos hay en la tabla para no dejar      "
echo "ciudadanos sin seccion resultante ya que a partir de"
echo "esto se le asigna su módulo correspondiente en el    "
echo "Centro Nacional. "
echo "En caso de resultar diferentes revise:"
read xx
echo ""
echo "4.1.- La tabla de CATALOG_2 par ver que "
echo "      ciudadano(s) fueron los que no le asignó "
echo "      sección resultate y revise:"
echo ""
echo "4.1.1.- La tabla de SECCIONES para ver "
echo "       si fueron capturadas todas las "
echo "       secciones y en caso de que esten"
echo "       todas capturadas cheque que "
echo "       esten bien. "
echo ""
echo "4.1.2.- Si en lo anterior todo esta "
echo "      correcto revise si el(los) "
echo "      ciudadano(s) estan mal cap-"
echo "      turados en EDMS_CORRIGE."
read xx
echo ""
```

```
echo "4.1.3.- En caso de no encontrar el "
echo "        error revise EDMS_PDI para"
echo "        comprobar si la(s) manzanas"
echo "        fueron capturadas correcta-"
echo "        mente. "
echo ""
echo "4.1.4.- En caso de persistir el "
echo "        error comunicarse al "
echo "        Centro Nacional de Computo con "
echo "        Ing Jose Alfredo Zarza Ramírez."
echo ""
echo ""
read xx
echo "5.- Se dan de baja de CATALOG_2 ciudadanos que se detectaron "
echo "    como homonimos, son los que se encuentran en la tabla de"
echo "    HOMONIMIAS "
echo "Recomendación: "
echo "Favor de llevar nota de cuantos ciudadanos se dan de baja"
echo "por HOMONIMIA."
echo ""
read xx
echo ""
echo "      INFORMACION UTILIZADA DE CATALOG "
echo ""
echo "a) NUM_FORMA"
echo "b) ALFA_CLAVE_ELECTORAL"
echo "c) FECHA_NACIMIENTO "
echo "d) LUGAR_NACIMIENTO"
echo "e) SEXO"
echo "f) DIGIT_VERIFICADOR"
echo "g) CLAVE_HOMONIMIA"
echo "h) NUMERO CREDENCIAL (0)"
echo "i) NOMBRE"
echo "j) APELLIDO_PATERNO"
echo "k) APELLIDO_MATERNO"
echo "l) CALLE,NUM_EXTERIOR y NUM_INTERIOR (CALLE)"
echo "m) COLONIA y CODIGO_POSTAL (COLONIA)"
echo "n) FECHA_NACIMIENTO (EDAD)"
echo "o) FECHA_INSCRIPCION_PADRON"
echo ""
read xx
echo "    "
echo "    "
echo ""
```

```
echo "          INFORMACION UTILIZADA DE EDMS_PDI y EDMS_CORRIGE "
echo "    "
echo "a) ESTADO_NUE"
echo "b) DISTRITO_NUE"
echo "c) MUNICIPIO_NUE"
echo "d) SECCION_NUE"
echo "e) LOCALIDAD_NUE"
echo "f) MANZANA_NUE"
echo ""
echo ""
echo " fin "
read xx
```

El segundo módulo es construcción del lo descrito anteriormente.
El programa se muestra al final de este capítulo.

VII.2.3.2 REPORTES Y PANTALLAS DE CAPTURA.

Como este programa es solo de ejecución, el usuario interactúa con el, sólo tecleando el número del distrito electoral.
Cuando el programa esta en ejecución, muestra el número de registros que ya proceso y una barra de avance del 0% al 100% tal como lo muestra la siguiente figura.

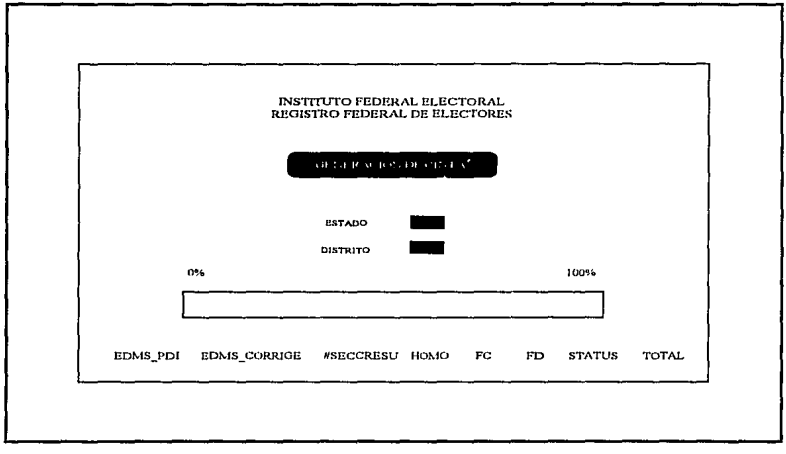


FIGURA VII.4 PANTALLA PRINCIPAL DE SISTEMA DE GENERACION DE CINTA DE 8 mm

Este programa crea una bitácora en */u/carto/spool/cinta.log* con la hora , fecha y los resultados obtenidos en el proceso. También crea otra bitácora */u/carto/spool/errores.log* en donde quedan los errores que se presentaron durante la ejecución del programa.

VII.2.4 INSTALACION DEL SISTEMA

Una vez concluido el desarrollo y las pruebas del sistema, se procede a realizar el envío de los programas a los Centros Regionales a través de la red X.25 con la que cuenta el Registro Federal de Electores. Tomando en cuenta los siguientes pasos:

1. Estar como usuario root del host de desarrollo.
2. Localizarse en el directorio /usr2/zarza/carto/CINTA
3. Teclear el siguiente comando.
   $ rcp CINTA.Z regionalx;/u/dbareg/carto/
4. Conectarse al regional y teclear lo siguiente.
   $ rlogin regional
   $ uncompress /u/dbareg/carto/CINTA.Z
   $ tar -xvpf /u/dbareg/carto/CINTA
5. Desconectarse del regional.
   $ exit

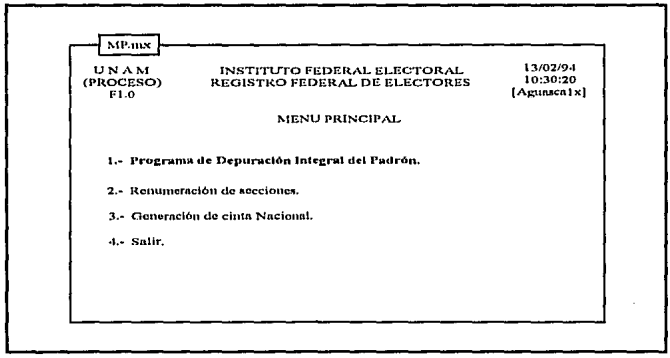
VII.2.5 COMENTARIOS

Este sistema dio soporte a la iniciativa ciudadana y de los partidos políticos de expedir una credencial para votar con fotografía.
A partir de la cual se generaron mas de 40 millones de credenciales para votar con fotografía

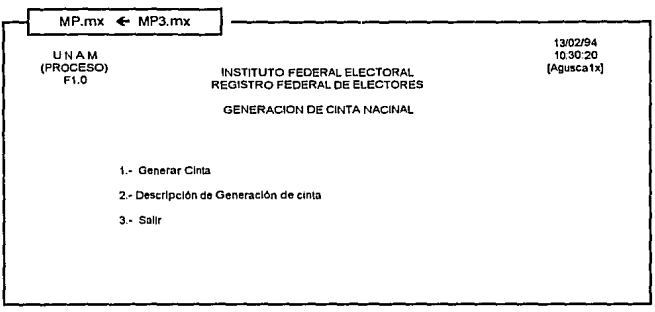
ANEXO

MENUS.

MP.mx



MP3.mx


MP.mx ← MP3.mx
U N A M
(PROCESO)
F1.0
13/02/94
10:30:20
[Aguscatx]
INSTITUTO FEDERAL ELECTORAL
REGISTRO FEDERAL DE ELECTORES
GENERACION DE CINTA NACINAL
1.- Generar Cinta
2.- Descripción de Generación de cinta
3.- Salir

PROGRAMAS

CREA LA PORTADA PRINCIAPAL DEL PROGRAMA. portada.c

```
/*************************************************************************
*************************************************************************

Programa: Realiza el despliegue de la pantalla principal del programa.

Entradas:

Salidas:  Despliega la información con caracteristicas especiales para la
          terminal IBM 3151

Realizado por: Jose Alfredo Zarza Ramírez.

*************************************************************************
*************************************************************************/
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "/usr2/zarza/proc/oracle.h"
#define  NUM_CURSOR  10000
main(){

        system (" clear ");

        system (" cuadro  0 0 22 78");
        system (" cursor 3  26");
        printf("HINSTITUTO FEDERAL ELECTORAL@\n");

        system (" cursor 4 24");
        printf("H REGISTRO FEDERAL DE ELECTORES@\n");

        system (" cursor 6 29");
        printf("IGENERACION DE CINTA@\n");
```

```
        system(" cuadro 15 26 17 47 ");
        system (" cursor 14 26 ");
        printf (" 0%%\n");
        system (" cursor 14 44 ");
        printf (" 100%%\n");

        system(" cursor 21 70 ");
        printf("IZARZA@\n");

        system (" cursor 18 4");
        printf("EDMS_PDI  EDMS_CORRIGE  #SECCRESU  HOMO  FC  FD  S13-21
TOTAL");

}
```

PROCESA TODAS LAS TABLAS DEL PDI Y GENERA LA CINTA DE 8mm.
catalog_2.pc

```
/*****************************************************************************
*****************************************************************************
        Programa: Aplica el resultado obtenido del Programa de Depuración Integral
                  al Padron Electoral Nacional, para obtener los registros reales
                  e imprimir su credencial para votar con fotografía.
                  Estos registros son distribuidos por Seccion Electoral en cintas
                  de 8mm, con estructura y formato definidos por Polaroid. Empresa
                  encargada de imprimir las micas.

        Entradas: Estado y Distrito a procesar.

        Salidas:  Grabación en cinta de 8mm con información para generar su
                  credencial para votar con fotografía.

        Tablas:   tablas EDMS_PDI, EDMS_CORRIGE, HOMONIMIAS, SECCIONES,
LDI_01_CAPTURA
                  FOLIOS_CORRECTOS, LDI_FOLIOS_DUPLICADOS.

        Realizado por: Jose Alfredo Zarza Ramirez

*****************************************************************************
*****************************************************************************/
```

```
/*_____
Includes
_______*/


#define OK   0
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "oracle.h"
#define  NUM_CURSOR  10000
char *zar_time();

EXEC SQL BEGIN DECLARE SECTION;

        varchar pwd[10], uid[10];

        /*___ clave elector para catalog ___*/
        int estadox;
        int distritox;
        int municipiox;
        int seccionx;
        int localidadx;
        varchar manzanax[2];
        varchar tipox[2];

        int estadoy;
        int distritoy;
        int municipioy;
        int secciony;
        int localidady;
        int manzanay;
        int secc_resul;


        int  num_form;
        int  num_form_correcto;
        VARCHAR alfa[7];
        VARCHAR alfax[7];
        int  fech,     fechx;
        int  luga,     lugax;
        VARCHAR sexo[2];
        VARCHAR sexox[2];
```

```
        int digi,     digix;
        int homo,     homox;
        int form,     formx;
        int nume;
        VARCHAR nomb[32];
        VARCHAR pate[32];
        VARCHAR mate[32];
        VARCHAR call[32];
        VARCHAR exte[8];
        VARCHAR inte[8];
        VARCHAR colo[32];
        int codi;
        int insc;
        int yest;
        int ydis;

        int xest;
        int xdis;
        int xmun;
        int xsec;
        int xloc;


        VARCHAR xman[2];

        int total_pdi;
        int total_corrige;
        int total_seccresu;
        int total_bajahomo;
        int total_foliocorrecto;
        int total_folioduplicado;
        int total_status1322;
        int total_catalog;

        char *tiempo;


        int ilumina_pdi;
        int ilumina_corrige;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

int saca_catalog();
```

```
FILE *fpcinta;
FILE *fperrores;

/*___
        PROGRAMA PRINCIPAL
                        ___*/

main(){

    int register i,j;
    int actualiza,inserta;

        strcpy(uid.arr,"ops$dbareg"); uid.len=strlen(uid.arr);
        strcpy(pwd.arr,"ops$dbareg"); pwd.len=strlen(pwd.arr);

        /* intenta la connexión a oracle */

        EXEC SQL WHENEVER SQLERROR GOTO error;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;

        EXEC SQL DROP TABLE CATALOG_2;

        if ((fpcinta = fopen("/u/carto/spool/cinta.log","at"))
                        == NULL ){
                printf("\n\n NO PUDE ABRIR  cinta.log");
                exit (0);
        }

        if ((fperrores = fopen("/u/carto/spool/errores.log","at"))
                        == NULL ){
                printf("\n\n NO PUDE ABRIR  errores.log");
                exit (0);
        }

                total_pdi          = 0;
                total_corrige      = 0;
                total_seccresu     = 0;
                total_bajahomo     = 0;
                total_foliocorrecto = 0;
                total_folioduplicado = 0;
                total_status1322   = 0;
                tiempo = zar_time();

                /*___ PORTADA ____*/
                system("/u/carto/bin/portada");
```

```
        /*____ Leer variables de entrada ____*/
        system ("cursor 10 32 ");
        printf("ESTADO   = I @\n");
        printf("I\n");
        system ("cursor 10 44");
        scanf("%d",&yest);
        printf("@\n");

        system ("cursor 11 32 ");
        printf("DISTRITO  = I @\n");
        printf("I\n");
        system ("cursor 11 44");
        scanf("%d",&ydis);
        printf("@\n");


        /*____ CORRER RUTINAS DE INSERCION ___*/

        inserta_pdi     ();
        inserta_corrige ();
        actualiza_seccion();
        baja_homo       ();
        folio_correcto  ();
        folio_duplicado ();
        status1322      ();


fclose (fpcinta);
fclose (fperrores);
exit(0);

error:
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        printf("\nError en el programa principal :\n");
        printf("%s\n",sqlca.sqlerrm.sqlerrmc);

        /* Para bitacora de errores */
        fprintf(fperrores,"\n%s Error programa principal %d %d:\n",
                tiempo,yest,ydis);
        fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);
```

```
                EXEC SQL ROLLBACK;
                fclose (fpcinta);
                fclose (fperrores);
                getchar ();
                exit(1);
}



/* Se define la rutina para insertar datos de EDMS_PDI */
/* a CATALOG_2 */

inserta_pdi(){
        EXEC SQL BEGIN DECLARE SECTION;
                int i;
                int edadc;
                varchar callc[80];
                varchar colc[180];
        EXEC SQL END DECLARE SECTION;


        EXEC SQL WHENEVER SQLERROR GOTO error_pdi;
        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_pdi;

        EXEC SQL DECLARE cur_pdi CURSOR FOR
                SELECT estado_act,distrito_act,municipio_act,seccion_act,
                localidad_act,manzana_act,estado_nue,distrito_nue,municipio_nue,
                seccion_nue,localidad_nue,manzana_nue,tipo
                FROM edms_pdi
                WHERE  ESTADO_NUE  =:yest    AND
                        DISTRITO_NUE =:ydis    AND
                        (validar='C' or validar='V');


        EXEC SQL OPEN cur_pdi;

        /*___ Carga Ilumina ___*/
        EXEC SQL SELECT SUM(NUM_PADRON)
        INTO :ilumina_pdi
        FROM EDMS_PDI
        WHERE  ESTADO_NUE    =:yest  AND
                DISTRITO_NUE   =:ydis  AND
                VALIDAR IN ('C','V');
```

```
        EXEC SQL SELECT SUM(NUM_PADRON)
        INTO :ilumina_corrige
        FROM EDMS_PDI
        WHERE  ESTADO_NUE    =:yest  AND
               DISTRITO_NUE  =:ydis  AND
               VALIDAR = 'N';

        /*____ Crea Tabla CATALOG_2 _____*/
        system("sqlplus /
@/u/cartografia/cinta/bin/crea_tabla_catalog_2>/u/cartografia/cinta/spool/crea_tabla.log");

        xsec = 0;
        while (1){

                EXEC SQL WHENEVER NOT FOUND GOTO no_mas_pdi;

                EXEC SQL FETCH cur_pdi INTO :estadox,:distritox,:municipiox,
                :seccionx,:localidadx,:manzanax,:estadoy,:distritoy,:municipioy,
                :secciony,:localidady,:manzanay,tipox;

                saca_catalog_pdi();
        }

        no_mas_pdi:
                EXEC SQL CLOSE cur_pdi;
                fprintf(fpcinta,"\n\n\n########################################\n");
                fprintf(fpcinta,"%s",tiempo);
                fprintf(fpcinta,"insertados de EDMS_PDI  %d\n",total_pdi);
                return(1);

        error_pdi:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error al sacar datos de EDMS_PDI :\n");
                printf("act %d %d %d %d %d %s",estadox,distritox,municipiox,
                        seccionx,localidadx,manzanax.arr);
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                getchar();

                fprintf(fperrores,"%s Error al sacar datos de EDMS_PDI:\n",
                        tiempo);
                fprintf(fperrores,"act %d %d %d %d %d %s\n",estadox,distritox,
                        municipiox,seccionx,localidadx,manzanax.arr);
                fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);

                EXEC SQL ROLLBACK;
```

```
                fclose (fpcinta);
                fclose (fperrores);

                printf("ver errores y soluciones <enter>");
                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}



int saca_catalog_pdi()
{

EXEC SQL BEGIN DECLARE SECTION;
        int edade;
        varchar calle[80];
        varchar colc[180];
        int porcentaje_pdi;

        static int yy=27;
        char y[4]="";
        char coory[40]="";
        int porcentaje_hold = 0;
EXEC SQL END DECLARE SECTION;

        EXEC SQL WHENEVER SQLERROR GOTO error_cata_pdi;

        EXEC SQL WHENEVER NOT FOUND GOTO no_catalog_pdi;

        EXEC SQL DECLARE cur_cat CURSOR FOR

                SELECT NUM_FORMA,ALFA_CLAVE_ELECTORAL,FECHA_NACIMIENTO,
                LUGAR_NACIMIENTO,SEXO,DIGIT_VERIFICADOR,CLAVE_HOMONIMIA,

        NUMERO_CREDENCIAL,NOMBRE,APELLIDO_PATERNO,APELLIDO_MATERNO,
                CALLE,NUM_EXTERIOR,NUM_INTERIOR,COLONIA,CODIGO_POSTAL,
                FECHA_INSCRIPCION_PADRON
                FROM catalog
                WHERE ESTADO=:estadox and DISTRITO=:distritox and
                MUNICIPIO=:municipiox and SECCION=:seccionx and
                LOCALIDAD=:localidadx and MANZANA=:manzanax;

        EXEC SQL OPEN cur_cat;
```

```
while (1) {

        EXEC SQL FETCH cur_cat INTO
        :num_form,:alfa,:fech,:luga,:sexo,:digi,:homo,:nume,:nomb,
        :pate,:mate,:call,:exte,:inte,:colo,:codi,:insc;


        /* Calcula edad */
        edadc = calc_edad();

        /*__ TIPO RURURAL DOMICILIO LOCALIDAD ___*/
        if (tipox.arr[0] == '2'){

                EXEC SQL SELECT NOM_LOCALIDAD
                   INTO :callc
                   FROM CATALOGO_LOCALIDADES
                   WHERE ESTADO = :estadoy   AND
                         DISTRITO  = :distritoy  AND
                         MUNICIPIO = :municipioy AND
                         SECCION   = :seccion y  AND
                         LOCALIDAD = :localidady;

        }else{
                /* Saca calle */
                callc.arr[0]= '\0';
                calc_call(&callc.arr[0]);
                callc.len = strlen(callc.arr);
        }

        /* Saca colonia */
        colc.arr[0]= '\0';
        calc_col(&colc.arr[0]);
        colc.len = strlen(colc.arr);

        /* Fecha inscripcion */
        insc = (int) (insc/10000);

        alfa.arr[alfa.len] = '\0';
        pate.arr[pate.len] = '\0';
        mate.arr[mate.len] = '\0';
        nomb.arr[nomb.len] = '\0';
        sexo.arr[sexo.len] = '\0';
        callc.arr[callc.len] ='\0';
        colc.arr[colc.len] = '\0';
```

```
EXEC SQL INSERT INTO CATALOG_2
(FOLIO,ALFA_CLAVE_ELECTORAL,FECHA_NACIMIENTO,LUGAR_NACIMIENTO,
SEXO,DIGIT_VERIFICADOR,CLAVE_HOMONIMIA,NUMERO_CREDENCIAL,
NOMBRE,APELLIDO_PATERNO,APELLIDO_MATERNO,CALLE,COLONIA,EDAD,
ANO_DE_REGISTRO,ESTADO,DISTRITO,MUNICIPIO,SECCION,LOCALIDAD,
MANZANA)
VALUES(:num_form, :alfa, :fech, :luga,
:sexo, :digi, :homo, 0,
:nomb, :pate, :mate, :calle, :cole, :edade,
:inse, :estadoy, :distritoy, :municipioy, :secciony,
:localidady, :manzanay);


if (sqlca.sqlcode == OK){
        total_pdi++;
        system("cursor 19 4");
        printf("%d\n",total_pdi);
}


porcentaje_pdi = (int)(total_pdi*100)/ilumina_pdi;
yy=0;
if (porcentaje_pdi%10 == 0 && porcentaje_pdi != 0){
        switch(porcentaje_pdi){
                case 10: yy=27;
                        break;
                case 20: yy=29;
                        break;
                case 30: yy=31;
                        break;
                case 40: yy=33;
                        break;
                case 50: yy=35;
                        break;
                case 60: yy=37;
                        break;
                case 70: yy=39;
                        break;
                case 80: yy=41;
                        break;
```

```
                        case 90: yy=43;
                                break;
                        case 100: yy=45;
                                break;
                }
                if (yy != 0){
                        coory[0] = '\0';
                        y[0] = '\0';
                        itoa (yy,y);
                        strcpy (coory,"cursor 16 ");
                        strcat (coory,y);
                        system (coory);
                        printf ("I @\n");
                }
        }


}

return(1);

no_catalog_pdi:
        EXEC SQL CLOSE cur_cat;

        if (xsec != seccionx){
                EXEC SQL COMMIT;
                xsec = seccionx;
        }
        return(0);

error_cata_pdi:

        EXEC SQL WHENEVER SQLERROR CONTINUE;
        EXEC SQL CLOSE cur_cat;

        printf("Error al sacar datos de CATALOG_PDI :\n");
        printf("%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                sexo.arr,digi,homo);
        printf("%s\n",sqlca.sqlerrm.sqlerrmc);
        getchar();

        fprintf(fperrores,"%s Error en proceso CATALOG_PDI %d %d:\n",
                tiempo,ycst,ydis);
        fprintf(fperrores,"%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                sexo.arr,digi,homo);
        fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);
```

```
                EXEC SQL ROLLBACK;

                fclose (fperrores);
                fclose (fpcinta);

                printf("ver errores y soluciones <enter>");
                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}

/*____

        INSERTA A CATALOG_2 LOS DATOS QUE SE OBTENGAN DE EDMS_CORRIGE

____*/


/* Se define la rutina para la insercion de datos de EDMS_PDI */
/* a CATALOG_2 */

inserta_corrige(){
        EXEC SQL BEGIN DECLARE SECTION;
                int i;
                int edadc;
                varchar callc[80];
                varchar colc[180];
                int count_corrige = 0;
        EXEC SQL END DECLARE SECTION;


        EXEC SQL WHENEVER SQLERROR GOTO error_corrige;
        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_corrige;

        EXEC SQL DECLARE cur_corrige CURSOR FOR
                SELECT  alfa_clave_electoral,fecha_nacimiento,lugar_nacimiento,
                sexo,digit_verificador,clave_homonimia,estado_act,
                distrito_act,municipio_act,seccion_act,localidad_act,
                manzana_act,estado_nvo,distrito_nvo,municipio_nvo,
                seccion_nvo,localidad_nvo,manzana_nvo
                FROM edms_corrige
                WHERE ESTADO_NVO  =:yest     AND
                   DISTRITO_NVO =:ydis     ;
```

```
        EXEC SQL OPEN cur_corrige;


        /*___ Limpia Avance ___*/
        system(" cursor 16 27 ");
        printf("                \n");

        EXEC SQL COMMIT WORK;
        /*____ Creacion de Indices para CATALOG_2 _____*/
        system("sqlplus / @/u/cartografia/cinta/bin/crea_indice_catalog_2 >
/u/cartografia/cinta/spool/crea_indice.log");

        xscc = 0;
        while (1){

                EXEC SQL WHENEVER NOT FOUND GOTO no_mas_corrige;

                EXEC SQL FETCH cur_corrige INTO :alfa,:fech,:luga,:sexo,
                :digi,:homo,:estadox,:distritox,:municipiox,:seccionx,
                :localidadx,:manzanax,:estadoy,:distritoy,:municipioy,:secciony,
                :localidady,:manzanay;


                count_corrige++;

                saca_catalog_corrige();
        }

        no_mas_corrige:
                EXEC SQL CLOSE cur_corrige;

                /*___ Limpia Avance ___*/
                system(" cursor 16 27 ");
                printf("                \n");

                fprintf(fpcinta,"total insertados de EDMS_CORRIGE %d\n",
                        total_corrige);
                return(1);

        error_corrige:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error al sacar datos de EDMS_CORRIGE :\n");
                printf("%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                        sexo.arr,digi,homo);
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
```

```
                fprintf(fperrores,"%s Error al sacar datos de CORRIGE %d %d:\n",
                        tiempo,yest,ydis);
                fprintf(fperrores,"%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                        sexo.arr,digi,homo);
                fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);
                fclose (fperrores);
                fclose (fpcinta);

                EXEC SQL ROLLBACK;

                printf("ver errores y soluciones <enter>");
                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}



int saca_catalog_corrige()
{

EXEC SQL BEGIN DECLARE SECTION;
        int i;
        int edadc;
        varchar callc[80];
        varchar colc[180];


        static int yy=27;
        char y[4]="";
        int porcentaje_corrige;
        char coory[40]="";
        int porcentaje_hold = 0;

EXEC SQL END DECLARE SECTION;

        EXEC SQL WHENEVER SQLERROR GOTO error_cata_corrige;

        EXEC SQL WHENEVER NOT FOUND GOTO no_catalog_corrige;
```

```
        EXEC SQL SELECT
NUM_FORMA,NUMERO_CREDENCIAL,NOMBRE,APELLIDO_PATERNO,
        APELLIDO_MATERNO,CALLE,NUM_EXTERIOR,NUM_INTERIOR,
        COLONIA,CODIGO_POSTAL,FECHA_INSCRIPCION_PADRON
        INTO :num_form,:nume,:nomb,:pate,:mate,:call,:exte,
           :inte,:colo,:codi,:insc
        FROM catalog
        WHERE( alfa_clave_electoral=:alfa  AND
               fecha_nacimiento   =:fech  AND
               lugar_nacimiento   =:luga  AND
               sexo               =:sexo  AND
               digit_verificador  =:digi  AND
               clave_homonimia    =:homo
          ) AND
          ( ESTADO   =:estadox   AND
               DISTRITO =:distritox  AND
               MUNICIPIO=:municipiox AND
               SECCION  =:seccionx   AND
               LOCALIDAD=:localidadx AND
               MANZANA=:manzanax
          );

        EXEC SQL WHENEVER NOT FOUND GOTO no_seccion;

        tipox.len = 0;
        EXEC SQL SELECT TIPO
        INTO :tipox
        FROM SECCIONES
        WHERE  ESTADO   =:estadoy   AND
               DISTRITO =:distritoy  AND
               MUNICIPIO=:municipioy AND
               SECCION  =:seccion y;

        no_seccion:
               if (tipox.len == 0)
                      strcpy(tipox.arr,"1");

        /* Calcula edad */
        edadc = calc_edad();

        if (tipox.arr[0] == '2'){
```

```
        EXEC SQL SELECT NOM_LOCALIDAD
                INTO :calle
                FROM CATALOGO_LOCALIDADES
                WHERE ESTADO  = :estadoy  AND
                  DISTRITO   = :distritoy  AND
                  MUNICIPIO  = :municipioy AND
                  SECCION    = :seccion y  AND
                  LOCALIDAD  = :localidady;

}else{
        /* Saca calle */
        calle.arr[0]= '\0';
        calc_call(&calle.arr[0]);
        calle.len = strlen(calle.arr);
}


/* Saca colonia */
colc.arr[0]= '\0';
calc_col(&colc.arr[0]);
colc.len = strlen(colc.arr);

/* Fecha inscripción */
insc = (int) (insc/10000);

EXEC SQL INSERT INTO CATALOG_2
(FOLIO,ALFA_CLAVE_ELECTORAL,FECHA_NACIMIENTO,LUGAR_NACIMIENTO,
 SEXO,DIGIT_VERIFICADOR,CLAVE_HOMONIMIA,NUMERO_CREDENCIAL,
 NOMBRE,APELLIDO_PATERNO,APELLIDO_MATERNO,CALLE,COLONIA,EDAD,
 ANO_DE_REGISTRO,ESTADO,DISTRITO,MUNICIPIO,SECCION,LOCALIDAD,
 MANZANA)
VALUES(:num_form, :alfa, :fech, :luga,
 :sexo, :digi,  :homo, 0,
 :nomb, :pate, :mate, :calle, :colc, :edadc,
 :insc, :estadoy, :distritoy, :municipioy, :seccion y,
 :localidady, :manzanay);

if (sqlca.sqlcode == OK){
        total_corrige++;
        system("cursor 19 15");
        printf("%d\n",total_corrige);
}
```

```
porcentaje_corrige = (int)(total_corrige*100)/ilumina_corrige;
yy=0;
if (porcentaje_corrige%10 == 0 && porcentaje_corrige != 0){
        switch(porcentaje_corrige){
                case 10: yy=27;
                        break;
                case 20: yy=29;
                        break;
                case 30: yy=31;
                        break;
                case 40: yy=33;
                        break;
                case 50: yy=35;
                        break;
                case 60: yy=37;
                        break;
                case 70: yy=39;
                        break;
                case 80: yy=41;
                        break;
                case 90: yy=43;
                        break;
                case 100: yy=45;
                        break;
        }
        if (yy != 0){
                y[0] = '\0';
                coory[0] = '\0';
                itoa (yy,y);
                strcpy (coory,"cursor 16 ");
                strcat (coory,y);
                system (coory);
                printf ("I  @\n");
        }
}
if (xsec != seccionx){
        EXEC SQL COMMIT;
        xsec = seccionx;
}
return(1);

no_catalog_corrige:
        return(1);

error_cata_corrige:
```

```
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        printf("Error al sacar datos de CATALOG_CORRIGE :\n");
        printf("%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                sexo.arr,digi,homo);
        printf("%s\n",sqlca.sqlerrm.sqlerrmc);
        getchar();

        fprintf(fperrores,"%s Error proceso CATALOG_CORRIGE %d %d:\n",
                tiempo,yest,ydis);
        fprintf(fperrores,"%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                sexo.arr,digi,homo);
        fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);

        EXEC SQL ROLLBACK;

        fclose (fperrores);
        fclose (fpcinta);

        printf("ver errores y soluciones <enter>");
        getchar();
        system("clear");
        system("more /u/cartografia/cinta/ayuda/errores.help");
        exit(1);
}

/*_____ ACTUALIZACION DE SECCION RESULTANTE _____*/

/* Se define la rutina para inssertar datos de EDMS_PDI */
/* a CATALOG_2 */

actualiza_seccion(){
    EXEC SQL BEGIN DECLARE SECTION;
        int i;
    EXEC SQL END DECLARE SECTION;


    EXEC SQL WHENEVER SQLERROR GOTO error_seccion;
    EXEC SQL WHENEVER NOT FOUND GOTO no_mas_seccion;

    EXEC SQL DECLARE cur_secciones CURSOR FOR
        SELECT estado,distrito,municipio,seccion,
        secc_resu
        FROM secciones
        WHERE  ESTADO   =:yest     AND
                DISTRITO  =:ydis     ;
```

```
EXEC SQL COMMIT;
EXEC SQL OPEN cur_secciones;

while (1){

        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_seccion;

        EXEC SQL FETCH cur_secciones INTO :estadoy,:distritoy,
        :municipioy,:secciony,:secc_resul;

        total_seccresu += actualiza_catalog_2() ;

        system("cursor 19 30");
        printf("%d\n",total_seccresu);
}

no_mas_seccion:
        EXEC SQL CLOSE cur_secciones;
        fprintf (fpcinta,"procesados %d %d en secc resul   %d \n",
                yest,ydis,total_seccresu);

        /* No seles asigno a todos seccion resultante */
        if (total_seccresu != (total_pdi+total_corrige)){
                system ("cursor 20 4");
                printf("L NO SE LES ASIGNO A TODOS LOS CIUDADANOS");
                printf(" SECCION RESULTANTE@\n");
                system ("cursor 21 4");
                printf("H          <enter> @");
                getchar();
                getchar();
                system(" clear ");
                system("more /u/cartografia/cinta/ayuda/errores.help");

                fprintf(fpcinta,"NO SE LES ASIGNO A TODOS LOS ");
                fprintf(fpcinta," CIUDADANOS SECCION RESULTANTE\n");

                fclose (fpcinta);
                fclose (fperrores);

                exit(1);
}

        return(1);
```

```
        error_seccion:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error al sacar datos de SECCIONES :\n");
                printf("%d %d %d %d \n",estadoy,distritoy,municipioy,secciony);
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                getchar ();
                EXEC SQL ROLLBACK;

                fprintf(fperrores,"%s Error al sacar datos de SECCIONES :\n",
                        tiempo);
                fprintf(fperrores,"%d %d %d %d \n",estadoy,distritoy,
                        municipioy,secciony);
                fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);
                printf("ver errores y soluciones <enter>");

                fclose (fperrores);
                fclose (fpcinta);

                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}


/* Graba la seccion resultante en catalog_2 */
int actualiza_catalog_2()
{

EXEC SQL BEGIN DECLARE SECTION;
        int i;
        int edadc;
        varchar callc[80];
        varchar colc[180];
        int actualizados=0;
EXEC SQL END DECLARE SECTION;

        EXEC SQL WHENEVER SQLERROR GOTO error_cata_2;

        EXEC SQL WHENEVER NOT FOUND GOTO no_catalog_2;

        EXEC SQL COMMIT WORK;

        EXEC SQL UPDATE CATALOG_2
        SET SECCION = :secc_resul,
          MODIFICADO = 'M'
```

```
        WHERE MODIFICADO IS NULL      AND (
            ESTADO   =:estadoy    AND
            DISTRITO =:distritoy  AND
            MUNICIPIO =:municipioy AND
            SECCION  =:secciony       );

        actualizados = (int)(sqlca.sqlerrd[2]);

        EXEC SQL COMMIT WORK;
        return(actualizados);

        no_catalog_2:
                return(0);

        error_cata_2:

                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error al acutalizar CATALOG_2 :\n");
                printf("%d %d %d %d \n",estadoy,distritoy,municipioy,secciony);
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                getchar();

                EXEC SQL ROLLBACK;

                fprintf(fperrores,"%s Error al acutalizar CATALOG_2 :\n",
                        tiempo);
                fprintf(fperrores,"%d %d %d %d \n",estadoy,distritoy,
                        municipioy,secciony);
                fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);

                fclose (fperrores);
                fclose (fpcinta);

                printf("ver errores y soluciones <enter>");
                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}
```

```
/*_____ BAJAS POR HOMONIMIA _____*/

/*____

        PROGRAMA PARA DAR DE BAJA POR HOMONIMIA
        A LOS CIUDADANOS QUE SE ENCUENTREN EN CATALOG_2;

____*/


/* Se define la rutina para la insercion de datos de EDMS_PDI */
/* a CATALOG_2 */

baja_homo(){
        EXEC SQL BEGIN DECLARE SECTION;
                int i;
        EXEC SQL END DECLARE SECTION;


        EXEC SQL WHENEVER SQLERROR GOTO error_bajahomo;
        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_bajahomo;

        EXEC SQL DECLARE cur_bajahomo CURSOR FOR
                SELECT  alfa_clave_electoral,fecha_nacimiento,lugar_nacimiento,
                sexo,digit_verificador,clave_homonimia
                FROM homonimias
                WHERE ESTADO_ACT    =:yest;


        EXEC SQL OPEN cur_bajahomo;

        /*___ generar reportes originales ___*/
        system ("cursor 21 4 ");
        printf("                    ");
        system ("cursor 21 4 ");
        printf("IGENERANDO REPORTE\n@");
        gen_rep_orig();
        system ("cursor 21 4 ");
        printf("                    ");

        while (1){

                EXEC SQL WHENEVER NOT FOUND GOTO no_mas_bajahomo;
```

```
        EXEC SQL FETCH cur_bajahomo INTO :alfa,:fech,:luga,:sexo,
        :digi,:homo;


        EXEC SQL WHENEVER NOT FOUND GOTO no_catalog_2;

        EXEC SQL DELETE FROM CATALOG_2
        WHERE alfa_clave_electoral=:alfa        AND
                fecha_nacimiento =:fech         AND
                lugar_nacimiento =:luga         AND
                sexo             =:sexo         AND
                digit_verificador =:digi        AND
                clave_homonimia  =:homo;

                total_bajahomo++;
                system("cursor 19 42");
                printf("%d\n",total_bajahomo);

        no_catalog_2:
                continue;
}

no_mas_bajahomo:
        EXEC SQL CLOSE cur_bajahomo;
        EXEC SQL COMMIT;
        fprintf(fpcinta,"bajas %d %d por homonimia   %d\n",yest,
                ydis,total_bajahomo);

        system("cursor 19 42");
        printf("%d\n",total_bajahomo);

        return(1);

error_bajahomo:
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        printf("Error al sacar datos %d %d de HOMONIMIAS :\n",
                yest,ydis);
        printf("%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                sexo.arr,digi,homo);
        printf("%s\n",sqlca.sqlerrm.sqlerrmc);
        getchar();
        EXEC SQL ROLLBACK;
        fprintf(fperrores,"%s Error al sacar datos %d %d HOMONIMIAS:\n",
                tiempo,yest,ydis);
```

```
                fprintf(fperrores,"%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                        sexo.arr,digi,homo);
                fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);

                fclose (fperrores);
                fclose (fpcinta);


                printf("ver errores y soluciones <enter>");
                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}

/*_____ ACTUALIZACION DE FOLIOS _____*/
/*____

        PROGRAMA PARA DAR DE ACTUALIZACION DE FOLIOS
LDI_FOLIOS_CORRECTOS
        A LOS CIUDADANOS QUE SE ENCUENTREN EN CATALOG_2;

____*/

/* Se define la rutina para la insercion de datos de EDMS_PDI */
/* a CATALOG_2 */

folio_correcto(){
        EXEC SQL BEGIN DECLARE SECTION;
                int i;
        EXEC SQL END DECLARE SECTION;


        EXEC SQL WHENEVER SQLERROR GOTO error_foliocorrecto;
        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_foliocorrecto;

        EXEC SQL DECLARE cur_foliocorrecto CURSOR FOR
                SELECT alfa_clave_electoral,fecha_nacimiento,lugar_nacimiento,
                sexo,digit_verificador,clave_homonimia,folio_correcto
                FROM ldi_folios_correctos;
```

```
EXEC SQL OPEN cur_foliocorrecto;
system("cursor 19 49");
printf("%d\n",total_foliocorrecto);
while (1){

        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_foliocorrecto;

        EXEC SQL FETCH cur_foliocorrecto INTO :alfa,:fech,:luga,:sexo,
        :digi,:homo,:num_form_correcto;


        EXEC SQL WHENEVER NOT FOUND GOTO no_catalog_2;


        EXEC SQL UPDATE CATALOG_2
        SET FOLIO = :num_form_correcto
        WHERE alfa_clave_electoral=:alfa        AND
                fecha_nacimiento =:fech         AND
                lugar_nacimiento =:luga         AND
                sexo             =:sexo         AND
                digit_verificador =:digi        AND
                clave_homonimia  =:homo;

        total_foliocorrecto++;
        system("cursor 19 49");
        printf("%d\n",total_foliocorrecto);

        no_catalog_2:
                continue;
}

no_mas_foliocorrecto:
        EXEC SQL CLOSE cur_foliocorrecto;
        EXEC SQL COMMIT;

        fprintf(fpcinta,"actualizacion %d %d folio correcto  %d\n",
                yest,ydis,total_foliocorrecto);

        total_foliocorrecto++;
        system("cursor 19 49");
        printf("%d\n",total_foliocorrecto);

        return(1);
```

```
        error_foliocorrecto:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error al sacar datos de LDI_FOLIOS_CORRECTOS :\n");
                printf("%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                        sexo.arr,digi,homo);
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);
                getchar();

                fprintf(fperrores,"%s Error en  LDI_FOLIOS_CORRECTOS :\n",
                        tiempo);
                fprintf(fperrores,"%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                        sexo.arr,digi,homo);
                fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);

                fclose(fpcinta);
                fclose(fperrores);

                EXEC SQL ROLLBACK;
                printf("ver errores y soluciones <enter>");
                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}


/*____ BAJAS DE FOLIO DUPLICADO ____*/
/*___

        PROGRAMA PARA DAR DE BAJA POR FOLIO DUPLICADO
        A LOS CIUDADANOS QUE SE ENCUENTREN EN CATALOG_2;

___*/


/* Se define la rutina para la insercion de datos de EDMS_PDI */
/* a CATALOG_2  */

folio_duplicado(){
        EXEC SQL BEGIN DECLARE SECTION;
                int i;
        EXEC SQL END DECLARE SECTION;
```

```
        EXEC SQL WHENEVER SQLERROR GOTO error_folioduplicado;
        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_folioduplicado;

        EXEC SQL DELETE FROM CATALOG_2
        WHERE folio = 99999999;

        total_folioduplicado = (int)(sqlca.sqlerrd[2]);
        system("cursor 19 56");
        printf("%d\n",total_folioduplicado);

        no_mas_folioduplicado:
                EXEC SQL COMMIT;

                system("cursor 19 56");
                printf("%d\n",total_folioduplicado);


                fprintf(fpcinta,"bajas %d %d por folio duplicado   %d\n",
                        yest,ydis,total_folioduplicado);
                return(1);

        error_folioduplicado:
                EXEC SQL WHENEVER SQLERROR CONTINUE;
                printf("Error al sacar datos de LDI_FOLIOS_CORRECTOS :\n");
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);

                getchar ();
                EXEC SQL ROLLBACK;

                fprintf(fperrores,"%s Error en LDI_FOLIOS_CORRECTOS :\n",
                        tiempo);
                fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);

                fclose (fperrores);
                fclose (fpcinta);

                printf("ver errores y soluciones <enter>");
                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}
```

```
/*_____ PROGRAMA PARA DAR DE BAJA POR STATUS 13-22 Y 25 _____*/

/*  Rutina para dar de baja ciudadanos con status 13 al 22 y 25 */

status1322(){
        EXEC SQL BEGIN DECLARE SECTION;
                int i;

                char y[4];
                int yy;
                char coory[40];
                int porcentaje_status1322;
        EXEC SQL END DECLARE SECTION;


        EXEC SQL WHENEVER SQLERROR GOTO error_status1322;
        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_status1322;

        /*___ Limpia Avance ___*/
        system(" cursor 16 27 ");
        printf("                \n");

        EXEC SQL DECLARE cur_status1322 CURSOR FOR
                SELECT  alfa_clave_electoral,fecha_nacimiento,lugar_nacimiento,
                sexo,digit_verificador,clave_homonimia
                FROM ldi_01_captura
                WHERE status_adq='13' or
                    status_adq='14' or
                    status_adq='15' or
                    status_adq='16' or
                    status_adq='17' or
                    status_adq='18' or
                    status_adq='19' or
                    status_adq='20' or
                    status_adq='21' or
                    status_adq='22' or
                    status_adq='25' ;


        EXEC SQL OPEN cur_status1322;

        system("cursor 19 63");
        printf("%d\n",total_status1322);

        while (1){
```

```
        EXEC SQL WHENEVER NOT FOUND GOTO no_mas_status1322;

        EXEC SQL FETCH cur_status1322 INTO :alfa,:fech,:luga,:sexo,
        :digi,:homo;


        EXEC SQL WHENEVER NOT FOUND GOTO no_catalog_2;

        EXEC SQL DELETE FROM CATALOG_2
        WHERE alfa_clave_electoral=:alfa      AND
                fecha_nacimiento =:fech       AND
                lugar_nacimiento =:luga       AND
                sexo             =:sexo       AND
                digit_verificador =:digi      AND
                clave_homonimia  =:homo;

                total_status1322++;
                system("cursor 19 63");
                printf("%d\n",total_status1322);


        no_catalog_2:
                continue;
    }

    no_mas_status1322:
        EXEC SQL CLOSE cur_status1322;
        EXEC SQL COMMIT;

        system ("cursor 21 4 ");
        printf("                    ");
        system ("cursor 21 4 ");
        printf("IGENERANDO REPORTE\n@");
        gen_rep_fina();
        system ("cursor 21 4 ");
        printf("                    ");

        system("cursor 19 72");
        printf("%d\n",(total_pdi+total_corrige-total_bajahomo-total_folioduplicado-
total_status1322));



        fprintf(fpcinta,"bajas %d %d status 13-22y25  %d\n",
                yest,ydis,total_status1322);
        fprintf(fpcinta,"\n########################################");
```

```
                system("cursor 19 63");
                printf("%d\n",total_status1322);

                system(" cursor 21 4");
                printf("HREALIZAR EL EXPORT DE CATALOG_2  <enter>@\n");
                system(" cursor 21 44");
                getchar();
                getchar();
                system("clear");
                system("sh /u/cartografia/cinta/bin/export_catalog_2");
                return(1);

        error_status1322:
                EXEC SQL WHENEVER SQLERROR CONTINUE;

                printf("Error en bajas por STATUS 13-22 Y 25 :\n");
                printf("%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                        sexo.arr,digi,homo);
                printf("%s\n",sqlca.sqlerrm.sqlerrmc);

                fprintf(fperrores,"%s Error bajas STATUS 13-22 Y 25 :\n",
                        tiempo);
                fprintf(fperrores,"%s%d%d%s%d%d\n",alfa.arr,fech,luga,
                        sexo.arr,digi,homo);
                fprintf(fperrores,"%s\n",sqlca.sqlerrm.sqlerrmc);

                fclose(fperrores);
                fclose(fpcinta);

                EXEC SQL ROLLBACK;
                printf("ver errores y soluciones <enter>");
                getchar();
                system("clear");
                system("more /u/cartografia/cinta/ayuda/errores.help");
                exit(1);
}
```

```
/*___ GENERA REPORTES ORIGINALES DE LA CINTA ___*/

gen_rep_orig(){
        char path[50]="sh /u/cartografia/cinta/bin/";
        char rep_orig_cinta[50] ="rep_orig_cinta";
        char rep_estad_orig_cinta[50] ="rep_estad_orig_cinta";
        char rep_bajas_cinta[50] ="rep_bajas_cinta";
        char reportes[1000];
        char estado[4]="";
        char distrito[4]="";


        reportes[0] = '\0';
        itoa(yest,estado);
        itoa(ydis,distrito);
        strcpy(reportes,path);
        strcat(reportes,rep_orig_cinta);
        strcat(reportes," ");
        strcat(reportes,estado);
        strcat(reportes," ");
        strcat(reportes,distrito);
        strcat(reportes," S");
        system(reportes);

        reportes[0] = '\0';
        itoa(yest,estado);
        itoa(ydis,distrito);
        strcpy(reportes,path);
        strcat(reportes,rep_estad_orig_cinta);
        strcat(reportes," ");
        strcat(reportes,estado);
        strcat(reportes," ");
        strcat(reportes,distrito);
        system(reportes);

        reportes[0] = '\0';
        itoa(yest,estado);
        itoa(ydis,distrito);
        strcpy(reportes,path);
        strcat(reportes,rep_bajas_cinta);
        strcat(reportes," ");
        strcat(reportes,estado);
        strcat(reportes," ");
        strcat(reportes,distrito);
        system(reportes);
}
```

```
gen_rep_fina(){
        char path[50]="sh /u/cartografia/cinta/bin/";
        char rep_final_cinta[50] ="rep_final_cinta";
        char rep_estad_final_cinta[50] ="rep_estad_final_cinta";
        char reportes[1000];
        char estado[4]="";
        char distrito[4]="";


        reportes[0] = '\0';
        itoa(yest,estado);
        itoa(ydis,distrito);
        strcpy(reportes,path);
        strcat(reportes,rep_final_cinta);
        strcat(reportes," ");
        strcat(reportes,estado);
        strcat(reportes," ");
        strcat(reportes,distrito);
        strcat(reportes," S");
        system(reportes);

        reportes[0] = '\0';
        itoa(yest,estado);
        itoa(ydis,distrito);
        strcpy(reportes,path);
        strcat(reportes,rep_estad_final_cinta);
        strcat(reportes," ");
        strcat(reportes,estado);
        strcat(reportes," ");
        strcat(reportes,distrito);
        system(reportes);

}


/*_____ RUTINAS PARA CATALOG_2 _____*/
calc_edad(){
        int fechca;

        fechca = (int)(insc/10000)-(int)(fech/10000 + 1900) ;
        if ( (fechca-16) >= 0)
                return(fechca);
        else
                return(fechca+100);
}
```

```
calc_call(char *c){
        char calle_num[130];
        char num_inte[64];
        char num_exte[64];
        int longitud;


        /* Inicializa variables */
        c[0] = '\0';
        /*_ fin de cadena _*/
        inte.arr[inte.len] = '\0';
        exte.arr[exte.len] = '\0';
        call.arr[call.len] = '\0';

        /*_ quitar espacios a id */
        lrtrim(inte.arr);
        lrtrim(exte.arr);
        lrtrim(call.arr);

        num_inte[0] = '\0';
        num_exte[0] = '\0';
        calle_num[0] = '\0';

        if (strlen(inte.arr) > 0){
                strcpy(num_inte," INT.");
                strcat(num_inte,inte.arr);
        }
        /* anade num_interior a num_exterior */
        strcpy(num_exte,exte.arr);
        strcat(num_exte,num_inte);

        /* junta el domicilio total */
        strcpy (calle_num,call.arr);
        strcat (calle_num," ");
        strcat (calle_num,num_exte);

        longitud = strlen(calle_num);
        if ( (longitud-32) >0){
                /*_ quito exedente de numeros con calle para 32 _*/
                call.arr[strlen(call.arr)-(longitud-32)] = '\0';
                strcat (call.arr," ");
                strcat (call.arr,num_exte);
                strcpy (c,call.arr);
        }
        else
```

```
                strcpy (c,calle_num);
}


calc_col(char *colc){
        char codic[16];

        /*___ Inicializa valores ___*/
        codic[0] = '\0';
        colc[0]  = '\0';
        colo.arr[colo.len] = '\0';

        lrtrim (colo.arr);
        strcpy (colc,colo.arr);
        if (tipox.arr[0] == '2' && codi == 0)
                return;
        strcat (colc," ");

        itoa(codi,&codic[0]);
        lpad (&codic[0],5);
        strcat (colc,codic);

        if ( strlen (colc) > 32){
                colo.arr[( strlen(colo.arr)-(strlen(colc)-32) )] = '\0';
                memset (colc,'\0',sizeof(colc));
                strcpy (colc,colo.arr);
                strcat (colc," ");
                itoa(codi,&codic[0]);
                strcat (colc,codic);
        }
}


lrtrim(char *c){

        while ( c[0] && c[strlen(c)-1] == ' ')
                c[strlen(c)-1 ] = '\0';

        while ( c[0] == ' ')
                strcpy (c,&c[1]);
}



itoa (int n, char *s)
{
```

```
 int i,j,c;


        i=0;
        j=0;

        do{
                s[i++]= n%10+'0';
        }while ((n /= 10) > 0);
        s[i]='\0';

        for (i=0,j= strlen(s)-1; i<j; i++,j--)
                c= s[i], s[i]= s[j], s[j]= c;
}

lpad (char *c,int l){

        while ( strlen(c) < l ){
                strcpy (&c[1],&c[0]);
                c[0] = '0';
        }
}

char *zar_time(){
        time_t segundos ;
        struct tm *struct_tiempo;
        char *tiempo;

        time(&segundos);
        struct_tiempo = localtime(&segundos);
        tiempo = asctime(struct_tiempo);

        printf("%s\n",tiempo);
        return(tiempo);
}
```