UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS (SDTAR).

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

DAVID JIMENEZ MIMILA

JOSE ADOLFO AVILA RAMIREZ

DIRECTOR DE TESIS:

M.I.: JUAN CARLOS ROA BEIZA

MEXICO, D. F. 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

A MI MADRE

Es difícil encontrar la forma de agradecerte.

A BETY HORUS

Por que así debe ser, Gracias.

A DAVID, EVA Y MIS HERMANOS

Gracias por su apoyo.

ADOLFO AVILA

A MIS PADRES

Cuyo esfuerzo y preocupación por mí y el de nuestra familia me han permitido obtener una de tantas satisfacciones como lo puede representar esta tesis, y la cual ofrezco simbólicamente como una forma de agradecimiento por su amor y apoyo incondicional en cualquier etapa de mi vida.

A MI ESPOSA

Gracias por su amor y paciencia en momentos difíciles.

A ADOLFO

Por su tolerancia y colaboración para concluir este trabajo escrito.

DAVID JIMENEZ MIMILA

ÍNDICE

CAPITULO I

FUNDAMENTOS TEORICOS

I.1 CONCEPTOS DE BASES DE DATOS

La expresión Base de Datos comenzó a popularizarse a principios de la década de los 60´s. Antes de esa época, en el mundo de la informática se hablaba de archivos y de conjuntos de datos. Como ocurre a menudo, hubo quienes quisieron subir de categoría sus archivos llamándolos bases de datos sin preocuparse de proporcionarles las características de no redundancia, independencia de datos, interconectividad, seguridad y accesibilidad en tiempo real. Estas características comenzaron a diseminarse al mismo tiempo que se empleaba de un software más eficaz para la administración de datos. La ***base de datos puede definirse como una colección de datos interrelacionados y almacenados en conjunto sin redundancia***; su finalidad es la de servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que se usan; se emplean métodos bien determinados para incluir datos nuevos y para modificar o extraer los datos almacenados. Dícese que un sistema comprende una colección de datos cuando éstos son totalmente independientes desde el punto de vista estructural.

La organización de las Bases de Datos debe representar el significado de fondo (o semántica de los datos) en forma correcta y eficiente. En los programas convencionales, la estructura de los datos se arregla de acuerdo con la conveniencia del programa. Una Base de Datos contiene información que será utilizada por muchos y diferentes programas. Debido a esto, la organización de una Base de Datos no puede estar

exclusivamente determinada por decisiones tomadas al programar funciones de propósito particular.

El almacenamiento de información para una Base de Datos se logra empleando uno o más archivos y debe ser posible tener acceso a todos ellos.

Los archivos no sólo se caracterizan de acuerdo con su tamaño, sino que se distinguen aún más de acuerdo con su organización. Las diferencias en las organizaciones de archivos llevan a grandes diferencias en el desempeño al almacenar y recuperar información.

A grandes rasgos los tipos de operaciones que se pueden ejecutar sobre una Base de Datos contemplan :

- La construcción del conjunto de datos,
- La actualización de los elementos dato en el conjunto de datos,
- La recuperación de datos, y,
- La reducción de grandes cantidades de datos a forma utilizable.

Los niveles de la estructura de una Base de Datos se definen como :

Conceptual	Datos con significación suficiente para provocar acciones en base a modelos,
Descriptivo	Descripción de elementos o eventos,
Organización	Independencia de datos, y,

Material Correspondiente al Hardware.

Un ***Sistema de Base de Datos*** es la combinación de programas y archivos que se utilizan conjuntamente. Un conjunto integrado de programas para dar apoyo a Bases de Datos, forma un sistema manejador de Bases de Datos.

Los objetivos de una Base Integrada de Datos son :

1. Obtener relaciones con el mayor grado de claridad semántica.
2. Conservar independencia de visión para simplificar la distribución posterior.
3. Tener el menor número de relaciones.
4. Tener el menor número de registros.
5. Hacer que el número de conexiones entre relaciones y atributos compartidos sea mínimo.
6. Hacer que el número de elementos dato almacenados sea mínimo.
7. Hacer mínima la actividad a lo largo de todas las conexiones entre relaciones.

El ***Esquema Global de Datos*** persigue como objetivo, mostrar las entidades de datos que existen actualmente para apoyar funciones o procesos específicos y determinar las estructuras de datos que necesitarán el apoyo del sistema propuesto estándar.

Utilizando los resultados del estudio de planificación de datos, o a partir de entrevistas, se determinan las entidades núcleo para posteriormente iniciar el modelo Entidad-Relación. El modelo Entidad-Relación ayuda a estructurar los datos necesarios para las tareas de análisis, y las relaciones entre las entidades de datos proporcionan una idea general de las funciones operacionales que el sistema debe desempeñar. Llegando a este punto en el modelo Entidad-Relación se registran únicamente las entidades y elementos de datos claves.

Antes que nada debe partirse del inventario de datos, el cual deberá utilizarse para determinar las entidades de datos del sistema propuesto. El inventario de datos indicará las entidades y sus relaciones para todo el negocio actual. Puede que sea necesario reducirlas para aislar aquellas entidades de datos que son pertinentes a la aplicación del área en cuestión.

Aunque las entidades determinadas reflejarán en gran parte el sistema actual, se supone que los datos que el sistema y la organización necesitan son relativamente estables con relación al tiempo, por lo que los datos del sistema actual pueden utilizarse como la fuente principal para definir el esquema global de datos.

Si existe la certeza de que se necesitarán nuevas entidades, éstas deben indicarse ahora. El modelo de Entidad-Relación es un esbozo de las entidades que el sistema propuesto necesita y debe depurarse y reorganizarse a medida que se reúne información en fases posteriores.

Partiendo del inventario de datos o de las entrevistas realizadas, el primer paso consiste en definir y documentar las posibles ***entidades núcleo*** o ***principales entidades de datos***. Estas constituyen básicamente aquellas "áreas" de un proceso sobre las que deben registrarse datos para que el sistema funcione mejor. Como regla general, las entidades núcleo pueden determinarse a través de conversaciones con los usuarios, ya que suelen mencionarse en forma de sustantivos.

La ***cardinalidad*** nos expresa el número de ocurrencias de una entidad relacionadas con ocurrencias específicas de la otra entidad expresada como una o muchas.

El carácter facultativo es un concepto y un símbolo de la modelación de la relación entre entidades. ***Carácter facultativo*** es el criterio que define si una entidad (ocurrencia) debe o no estar relacionada con otra entidad cuando se crea la primera entidad.

Las ***relaciones entre entidades*** definen las dependencias entre las entidades núcleo en lo que concierne a la cardinalidad de la relación y determine si la relación entre las dos entidades núcleo es de uno-a-muchos, de uno-a-uno o de muchos-a-muchos. Deberá determinarse, para ambas partes de la relación, si la naturaleza de esta relación es facultativa u obligatoria.

A continuación, en las secciones que componen el fundamento teórico de nuestra tésis, presentaremos con mayor profundidad cada uno de estos temas, su uso y su repercución en el diseño de sistemas.

I.1.1 OBJETIVOS DE LA ORGANIZACIÓN DE LAS BASES DE DATOS

Las bases de datos pueden organizarse de muchas maneras. Los principios para la selección de técnicas de organización han sido estudiadas ampliamente. La comisión de sistemas CODASYL (Conference on Data System Languages) ha emitido varios informes sobre el tema, además de un informe clásico sobre los requerimientos que debe satisfacer un sistema de administración de bases de datos. A continuación se presenta un resumen de sus opiniones:

- ***Los datos podrán utilizarse de múltiples maneras.*** Es decir, diferentes usuarios, perciben de manera distinta los mismos datos, y pueden emplearlos de diversas formas.
- ***Se protegerá la inversión intelectual.*** No será necesario rehacer los programas y las estructuras lógicas existentes (que representan muchas horas-hombres de trabajo) aún cuando se modifique el esquema de la base de datos.
- ***Bajo costo.*** Se refiere al bajo costo de almacenamiento y uso de los datos así como a la minimización del costo de los cambios.
- ***Menor proliferación de datos.*** Las necesidades de las nuevas aplicaciones serán satisfechas con los datos existentes en lugar de crear nuevos archivos, evitándose así la excesiva proliferación de datos que se advierte en los malos diseños de las bases de datos.
- ***Desempeño.*** Los requerimientos de información se atenderán con la rapidez adecuada según el uso que habrá de hacerse de la misma.

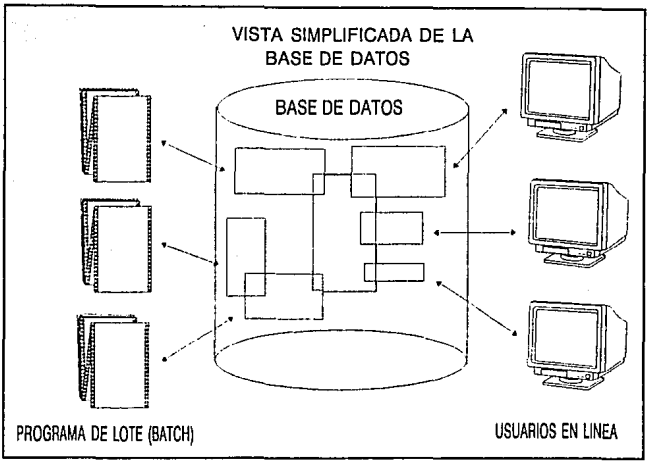


Figura 1.1 Vista simplificada de la Base de Datos

- ***Claridad.*** Los usuarios sabrán que datos se encuentran a su disposición y los comprenderán sin dificultad.
- ***Facilidad de uso.*** Los usuarios tendrán fácil acceso a los datos. Las complejidades internas son ajenas al usuario, gracias al sistema de administración de la base.
- ***Flexibilidad.*** Los datos podrán ser utilizados o explorados de manera flexible, con diferentes caminos de acceso.
- ***Rápida atención de interrogantes no previstas.*** Los requerimientos espontáneos de información se atenderán sin necesidad de escribir un

programa de aplicación específico (lo que significa un cuello de botella por la pérdida de tiempo) sino utilizando un lenguaje de alto nivel para averiguación o generación de reportes no planeados.

- ***Facilidad para el cambio.*** La base de datos puede crecer y variar sin interferir con las maneras establecidas de usar los datos.
- ***Precisión y coherencia.*** Se utilizarán controles de precisión. El sistema evitará las versiones múltiples de los mismos datos con diferentes estados de actualización.
- ***Seguridad de acceso.*** Se evitará el acceso no autorizado de datos. Los mismos datos podrán estar sujetos a diferentes restricciones de acceso para distintos usuarios.
- ***Integridad.*** Los datos estarán protegidos contra fallas y catástrofes, y contra delincuentes, vándalos, incompetentes y personas que intenten falsificarlos.
- ***Disponibilidad.*** Los datos se hallarán inmediatamente disponibles para los usuarios cuando los necesiten.
- ***Independencia física de los datos.*** El hardware de almacenamiento y las técnicas físicas de almacenamiento podrán ser alteradas sin obligar a la modificación de los programas de aplicación.
- ***Independencia lógica de los datos.*** Podrán agregarse nuevos datos, o expandirse la estructura lógica general, sin que sea necesario re-escribir los programas de aplicación existentes.
- ***Redundancia controlada.*** Los datos serán almacenados una sola vez, excepto cuando existan razones técnicas o económicas que aconsejen el almacenamiento redundante.

- ***Adecuada rapidez de acceso.*** Los mecanismos de acceso y los métodos de direccionamiento serán lo suficientemente rápidos, para satisfacer las necesidades de los usos previstos.
- ***Adecuada rapidez de exploración.*** La conveniencia y necesidad de la exploración espontánea se incrementarán en la medida que se difunda el uso interactivo de los sistemas.
- ***Normalización de los datos dentro de un organismo.*** Se necesita un acuerdo interdepartamental sobre los formatos y las definiciones de datos. La normalización es indispensable porque de otro modo se crearían datos incompatibles o con significados distintos.
- ***Diccionario de datos.*** Se necesita un diccionario de datos que defina las estructuras de las Base de datos.
- ***Interfaz de alto nivel con los programadores.*** Los programadores de aplicaciones deben disponer de medios sencillos para pedir datos y estar aislados de las complejidades internas de organización y direccionamiento de los archivos.
- ***Lenguaje del usuario final.*** Un lenguaje de consulta de alto nivel o un lenguaje para la generación de reportes permitirán que los usuarios finales se vean libres de tener que escribir un programa de aplicación convencional.
- ***Controles de integridad.*** Siempre que sea posible, se recurrirá a revisiones de límites y otros controles para asegurar la exactitud de los datos.
- ***Fácil recuperación en caso de falla.*** Recuperación automática sin pérdida de información.

- ***Afinación.*** La base de datos debe ser afinable, para mejorar su desempeño sin exigir la reescritura de los programas de aplicación.
- ***Ayuda para el diseño y la supervisión.*** Se refiere a las ayudas que permitan al diseñador o al administrador de datos predecir y optimizar el desempeño del sistema.
- ***Migración o reorganización automática.*** Migración de datos u otra reorganización física previstas para la mejora del desempeño.

I.1.2 ENTIDADES Y ATRIBUTOS

Llamaremos entidades a los objetos sobre los cuales se almacena información. Una entidad puede ser un objeto tangible, por ejemplo un alumno, pero también puede ser algo intangible, como un suceso o un concepto abstracto. Toda entidad tiene propiedades que eventualmente conviene registrar. A menudo, en el procesamiento de datos nos interesan las colecciones de entidades similares y necesitamos registrar información acerca de las mismas propiedades de cada una de ellas. **A las colecciones de objetos similares las llamaremos conjuntos de entidades.**

Así como las entidades describen el muno real, existen términos que se utilizan para describir la información acerca de ellas. Por lo general mantenemos un registro para cada entidad y agrupamos en conjunto todos los registros pertinentes a entidades similares. Los registros se refieren a atributos de las entidades y contienen los valores de estos atributos.

Cuando hablamos de información podemos referirla a tres diferentes campos, y tendemos, a veces confusamente, a saltar de uno a otro sin advertencia previa. El primero de estos campos es el del mundo real, en el que hay entidades y estas exhiben ciertas propiedades. El segundo es el dominio de las ideas y la información existente en las mentes de las personas y los programadores. Aquí hablamos de los atributos de las entidades y nos referimos a éstos simbólicamente. El tercer campo es el de los datos, en el que usamos caracteres o bits para codificar información.
La manera más común de asociar un valor con un dato y de asociar datos con atributos de entidad, consiste en almacenar juntos los datos en una secuencia fija, por ejemplo, en un registro de clientes. En este podría usarse una codificación para representar los valores del atributo. Para esta entidad se indicará si el atributo es de longitud fija o de longitud variable. Los nombres de los atributos y las representaciones de valor no se registran en el archivo, aunque se deben encontrar registrados en alguna parte, por ejemplo, en el diccionario de datos, en el que se listen los nombres y tipos de los diferentes datos de la base.

Podemos determinar entonces un registro de entidad para cada uno de los elementos y de los atributos que lo acompañan. A este tipo de distribución bidimensional de datos se le llama a menudo disposición plana.

Por otra parte un identificador de identidad es aquel atributo que es único a esa entidad y que sirve para reconocerlo, y en ocasiones se requiere más de un atributo para identificar un registro.

Una llave es el atributo o conjunto de estos que se utilizan para reconocer un registro. Una llave primaria es la que se utiliza para definir unívocamente un registro, es decir el identificador de entidad formado por uno o más atributos. También se usan llaves que no identifican registros únicos, sino todos aquellos que tienen cierta propiedad, y a estas se les llama llaves secundarias.

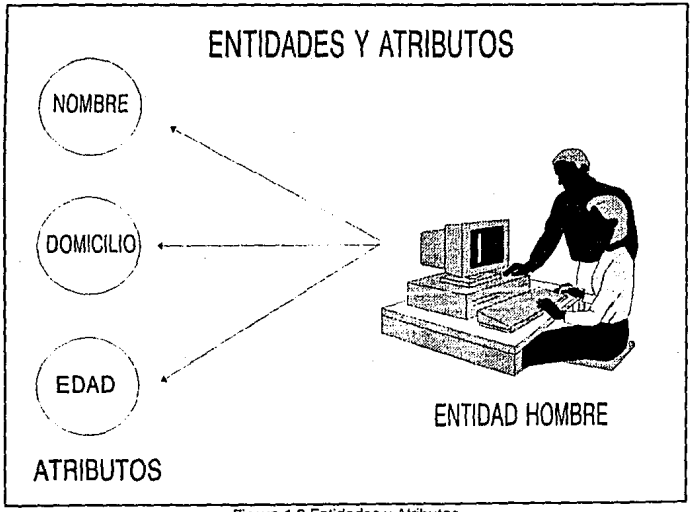


Figura 1.2 Entidades y Atributos

I.1.3 ESQUEMAS Y SUBESQUEMAS

Es preciso describir la organización de los datos de una manera formal. Las descripciones, lógica y física, de la base de datos son indispensables para el software de administración con el fin de extraer de la base los datos que pide el usuario.

ESQUEMA

Llamamos esquema a la descripción lógica de la base de datos. El esquema es un diagrama de los tipos de datos que se usan; proporciona los nombres de las entidades y sus atributos especificando las relaciones que existen entre ellos. Es un marco en el que se escriben los valores de los datos. Por ejemplo, un estado de cuenta podría ser considerado un esquema si se borraran los valores de los datos. Cuando el marco del esquema se completa con los valores de los datos, entonces hablamos de una instancia del esquema.

Los esquemas se presentan a menudo en forma de diagramas de bloques. Las líneas que unen ciertos bloques representan relaciones. Estas agregan información que no es inherente a los datos indicados en el esquema.

Del mismo esquema se pueden derivar muchos subesquemas. Los programadores no tienen que conocer la totalidad del esquema, en cambio, el administrador de datos debe asegurar que los subesquemas que se usan se deriven del esquema principal. El software de administración de datos

deriva estos de los subesquemas automáticamente, y los pone a disposición del programa de aplicación.

Ni los esquemas ni los subesquemas reflejan la forma como los datos se almacenan físicamente y hay muchos estilos de organización física para cada organización lógica definida. Tenemos así tres distintas descripciones de datos:

SUBESQUEMA

Es el diagrama de una porción de los datos, orientado a satisfacer uno o más programas de aplicación, esto es, una organización de archivos del programador.

***La descripción lógica global de la base de datos o esquema*:** es el diagrama lógico de la base de datos completa. Esta es la vista general de los datos como los contempla el administrador o los analistas de sistemas que usan toda la base.

***La descripción de la base de datos*:** es el diagrama de la distribución física de los datos en los dispositivos de almacenamiento. La vista de los datos que tienen los programadores y los diseñadores de sistemas que se ocupan del desempeño y de cómo se ubican los datos en el hardware, de cómo se localizan, y de que técnica de compactación emplean.

Correspondencia entre datos, simple y compleja

La relación que existe entre dos conjuntos de datos puede ser simple o compleja. Por simple queremos decir que hay una correspondencia biunívoca (uno a uno) entre los datos. Cuando a cada dato le corresponden más de uno del otro conjunto, entonces la correspondencia es compleja.

- ***Correspondencia uno-a-uno***. Una entidad **A** está asociada, cuando más, con una entidad en **B**, y una entidad en **B** está asociada cuando más con una entidad en **A**.

- ***Correspondencia uno-a-muchos.*** Una entidad en **A** está asociada con cualquier número de entidades en **B**. Una entidad en **B**, sin embargo, puede estar asociada, cuando más, con una entidad en **A**.

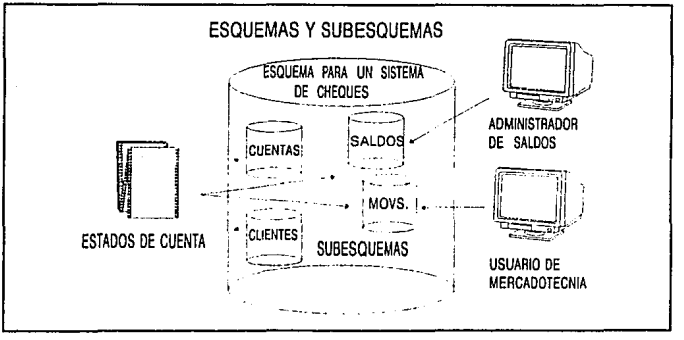


Figura 1.3 Esquemas y subesquemas

- ***Correspondencia muchos-a-uno.*** Una entidad en **A** está asociada, cuando más, con una entidad en **B**. Sin embargo, una entidad en **B**, puede estar asociada con cualquier número de entidades en **A**.

- ***Correspondencia muchos-a-muchos.*** Una entidad en **A** está asociada con cualquier número de entidades en **B** y una entidad en **B** está asociada con cualquier número de entidades en **A**.

Reglas para dibujar un esquema

1. El diagrama debe distinguir claramente los nombres de registro, de datos y otros.
2. Debe quedar clara la agrupación de datos en conjuntos.
3. Debe aclararse la distinción entre conjuntos de datos y registros.
4. Deben mostrarse los identificadores de registro.
5. El diagrama debe indicar claramente qué relaciones son simples y cuales son complejas.
6. Las relaciones simples deben diferenciarse de las relaciones complejas.
7. Las relaciones entre registros deben nombrase o numerarse.
8. No deben utilizarse nombres duplicados.

I.1.4 ABSTRACCIONES DE DATOS

Un sistema manejador de base de datos es una colección de archivos interrelacionados y un conjunto de programas que permiten a varios

usuarios accesar y modificar estos archivos. Uno de los principales fines de un sistema de bases de datos es proporcionar a los usuarios una representación simbólica de las piezas de información, a la cual llamaremos abstracción. Para que un sistema sea útil, los datos deben ser recuperados eficientemente. Con el fin de aumentar la eficiencia, se diseñan estructuras complejas de datos para representarlos en una base de datos. Sin embargo, como la mayoría de los usuarios de las bases de datos no son expertos en esta materia deben definirse varios niveles de abstracción desde los cuales se puede ver la base de datos. Estos son:

- **Nivel Interno,**
- **Nivel Conceptual, y,**
- **Nivel Externo**

Estos niveles serán desarrollados en el contexto de las Arquitecturas de Bases de Datos.

I.1.5 INDEPENDENCIA DE DATOS

Es la habilidad para modificar la definición de un esquema en un nivel sin afectarlo. Hay dos niveles de independencia de datos:

- ***Independencia física de datos.*** Es la habilidad para modificar el esquema físico sin tener que re-escribir los programas de aplicación. Las modificaciones al nivel físico son necesarias ocasionalmente para mejorar el funcionamiento del sistema.
- ***Independencia lógica.*** Es la habilidad para modificar el esquema conceptual sin que se tenga que alterar el programa de aplicación.

Estos cambios a nivel conceptual son necesarios cuando la estructura lógica de la base de datos es alterada.

La independencia lógica de los datos es más difícil de implementar, ya que los programas de aplicación dependen de la estructura lógica de los datos que accesa. El concepto de independencia de datos es similar en muchos aspectos al concepto de tipos abstractos de datos en los lenguajes modernos de programación. Ambos esconden los detalles de implementación a los usuarios, permitiéndoles concentrarse en la estructura general, en lugar de fijarse en detalles de implementación de bajo nivel.

I.1.6 ELEMENTOS DEL SISTEMA DE BASE DE DATOS

LENGUAJE DE DEFINICIÓN DE DATOS

Un esquema de bases de datos se especifica por un conjunto de definiciones que son expresadas por un lenguaje especial llamado ***"Data Definition Language"*** (DDL). El resultado de la compilación del código de DDL es un conjunto de tablas. Sus estructuras son almacenadas en un archivo especial llamado diccionario de datos.

Un diccionario de datos es un archivo que contiene "metadatos", es decir "datos sobre los datos". Este archivo es consultado antes de que los datos reales sean leídos o modificados en el sistema de bases de datos.
La estructura de almacenamiento y métodos de acceso usados por el sistema de bases de datos son especificados por un conjunto de definiciones en un tipo especial de DDL llamado lenguaje de definición y

almacenamiento. El resultado de la compilación de estas definiciones es un conjunto de instrucciones para especificar la implementación de detalles de los esquemas de bases de datos que están ocultos a los usuarios.

LENGUAJE DE MANIPULACIÓN DE DATOS

Los niveles de abstracción se aplican también a la manipulación de los datos. Por manipulación de datos debemos entender:

- La recuperación de información almacenada en una base de datos.
- La inserción de información nueva en la base de datos.
- El borrado de la información de la base de datos.

A nivel físico debemos definir algoritmos que nos permitan el acceso eficiente a los datos. En un nivel de abstracción mayor, se da más énfasis a la facilidad de uso. El objetivo es proporcionar una interacción humana eficiente con el sistema.

Un sistema de manipulación de datos (***DML, Data Manipulation Language***) es un lenguaje que permite a los usuarios accesar o manipular datos organizados de acuerdo al modelo de datos apropiados. Existen dos tipos básicos de DML:

- ***De procedimientos***. Este requiere que el usuario especifique que datos son necesarios y como obtenerlos.

- ***De no procedimientos.*** Este requiere que el usuario especifique qué datos son necesarios, sin especificar cómo obtenerlos.

Los DMLs de no procedimientos son generalmente más fáciles de aprender que los procedurales, sin embargo, como el usuario no tiene que señalar como obtener los datos, estos lenguajes pueden generar código que no será tan eficiente como el producido por los lenguajes procedurales. Esta dificultad se salva usando técnicas de optimización.

Un "query" es una declaración que indica la recuperación de información. La porción del DML que se ocupa de la recuperación de la información se llama lenguaje de "query". Aunque es técnicamente incorrecto, comúnmente se usan los términos lenguaje de "query" y DML indistintamente.

MANEJADOR DE BASE DE DATOS

Un manejador de Bases de Datos es un módulo de programa que proporciona la interface entre los datos de bajo nivel almacenados en la base de datos, los programas de aplicación y las consultas del sistema. El manejador de la base de datos es responsable de las siguientes tareas:

- ***Interacción con el manejador de archivos.*** Los datos son almacenados en discos usando el sistema de archivos que es proporcionado por un sistema operativo convencional. El manejador

de la base de datos traduce las instrucciones de DML en comandos de sistema de archivos de bajo nivel. Por lo tanto, el manejador es responsable del almacenamiento, recuperación y actualización de los datos.

- ***Garantizar la Integridad.*** El valor de los datos almacenados en la base de datos debe satisfacer ciertos requerimientos de consistencia. Estos deben ser especificados por el administrador de la base de datos y así, el manejador de la base de datos puede checar si las actualizaciones de la base de datos violan estas restricciones, tomando como consecuencia las medidas necesarias para evitarlas.
- ***Garantizar la seguridad.*** No todos los usuarios podrán accesar todos los datos.
- ***Respaldo y recuperación.*** Es responsabilidad del manejador detectar las fallas que pudieran afectar los datos y restaurar la base de datos al estado en que se encontraba antes de la ocurrencia de la falla. Esto se lleva a cabo mediante la iniciación de procedimientos de respaldo y recuperación.
- ***Control de concurrencia.*** Cuando varios usuarios accesan la base de datos concurrentemente, la consistencia de los datos puede ser dañada. Es necesario que se controle la interacción entre los usuarios concurrentes.

ESTRUCTURA GENERAL DEL SISTEMA

Un sistema de base de datos se divide en módulos que se encargan de cada una de las funciones del sistema. Algunas de estas pueden ser

proporcionadas por el sistema operativo de la máquina. En la mayoría de los casos, el sistema operativo provee solamente los servicios básicos y el sistema de bases de datos debe ser construido sobre éste.

Un sistema de base de datos consiste en un número de componentes funcionales, los cuales incluyen:

- ***Manejador de archivos:*** Controla la distribución de espacio de almacenamiento en disco y las estructuras de datos usadas para representar la información almacenada en disco.
- ***Manejador de la base de datos:*** Suministra la interface entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y queries que se envían al sistema.
- ***Procesador de queries:*** Traduce instrucciones de un lenguaje de query a instrucciones de bajo nivel que sean entendidas por el manejador de la base de datos. En adición, el procesador de queries intenta trasformar las solicitudes del usuario a una forma equivalente con mayor desempeño buscando una buena estrategia para ejecutar el query.
- ***Precompilador de DML:*** Convierte las instrucciones de DML incluidas en un programa de aplicación a llamadas normales de procedimientos en el lenguaje de programación. El precompilador debe interactuar con el procesador de query para generar el código apropiado.
- ***Compilador de DDL:*** Transforma las instrucciones de DDL a un conjunto de tablas que contienen metadatos. Estas tablas son almacenadas en el diccionario de datos.

Adicionalmente se requieren también varias estructuras de datos como parte de la implementación del sistema físico. Estas incluyen:

- ***Archivos de datos:*** Los cuales almacenan la base de datos.
- ***Diccionario de datos:*** Almacena información sobre la estructura de la base de datos. El diccionario de datos es usado frecuentemente por los componentes funcionales, por ello se debe dar gran énfasis al logro de un buen diseño y una implementación eficiente del diccionario de datos.
- ***Índices:*** Proporcionan un acceso rápido a los datos con valores particulares.

ADMINISTRADOR DE LA BASE DE DATOS

Una de las principales razones por las cuales es necesario contar con un sistema de manejo de bases de datos es tener un control central de los datos y los programas que los accesan. La persona que lleve el control del sistema es el administrador de la base de datos (DBA, Data Base Administrator). Las funciones de un DBA son las siguientes:

- ***Definición de esquemas.*** La creación del esquema original de la base de datos. Esto se lleva a cabo escribiendo un conjunto de definiciones que son traducidas por el compilador de DDL a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.
- ***Estructura de almacenamiento y definición de métodos de acceso.*** La creación de estructuras adecuadas de almacenamiento y métodos

de acceso. Esto se logra escribiendo un conjunto de definiciones que son traducidas por el compilador.

- ***Modificación de la organización física y el esquema.*** Estos cambios se llevan a cabo escribiendo un conjunto de definiciones que son usadas por el compilador de DDL o el compilador de definición del lenguaje para generar modificaciones a las tablas internas apropiadas.
- ***Permitir la autorización para el acceso de datos.*** Diferentes niveles de acceso a los diferentes usuarios. Esto le permite regular qué partes de la base pueden accesar los diferentes usuarios.
- ***Especificación de las restricciones de Integridad.*** Estas restricciones se mantienen en una estructura especial del sistema que son consultadas por el manejador cuando se hace una actualización.

USUARIOS DE LA BASE DE DATOS

Uno de los objetivos de los sistemas de bases de datos es el de proporcionar un ambiente para recuperar y almacenar información en la base de datos. Hay tres tipos diferentes de usuarios de este sistema, y se diferencian por la manera en que se espera que interactúen con el sistema.

- ***Programadores de Aplicación.*** Son los profesionales de la computación, los cuales interactúan con el sistema a través de llamadas de DML, las mismas que están escritas en algún lenguaje. Estos programas son referidos comúnmente como programas de aplicación. Como la sintaxis de DML es generalmente diferente del

lenguaje, por lo regular estas llamadas están precedidas de un carácter especial, de modo que el código apropiado pueda ser generado por un preprocesador llamado precompilador de DML. Este convierte las instrucciones de DML a llamadas de procedimientos normales en el lenguaje de programación. El programa resultante se corre entonces con el compilador del mismo, el cual genera el código objeto apropiado. Existen algunos tipos especiales de lenguajes de programación que combinan estructuras de control para el manejo de objetos en las bases de datos.

- ***Usuario Final.*** Estos son los usuarios que interactúan con el sistema sin escribir programas, actualizan los datos y validan que se hayan registrado en forma correcta. Sus consultas son atendidas a través de pantallas y reportes predefinidos, cuando han adquirido cierta experiencia son capaces de formular "querys".
- ***Usuarios especializados.*** Estos usuarios escriben aplicaciones que no encajan en la estructura tradicional del procesamiento de datos. Entre estas aplicaciones están los sistemas de CAD, los sistemas expertos que almacenan tipos de datos complejos (por ejemplo audio y gráficas), y sistemas de modelaje de ambiente.

La siguiente figura muestra estos componentes y las conexiones entre ellos.

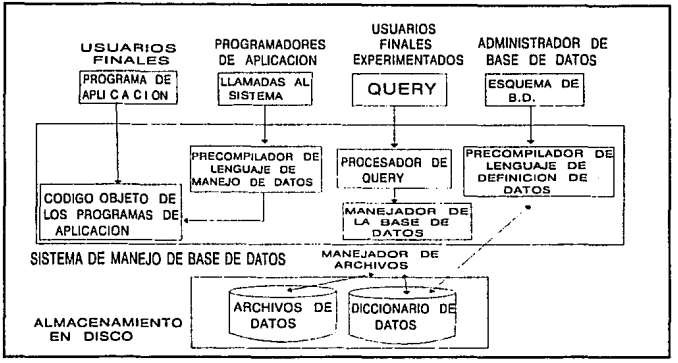


Figura 1.4 Estructura general del Sistema de Base de Datos

I.1.7 BASES DE DATOS EN AMBIENTES MULTIUSUARIOS

Para entender mejor lo que es un ambiente de bases de datos multiusuario podríamos imaginarnos que estamos escribiendo un libro en colaboración con otra persona, donde cada uno es responsable de hacer ciertos capítulos. Como resultado de este trabajo "concurrente", el trabajo avanza rápidamente. Trabajando de esta manera, podemos leer los capítulos del otro, pero no podemos editar o escribir en el capítulo que no nos corresponde, ya que de hacerlo así, el resultado sería un caos y un manuscrito que tal vez quedaría incompleto. Cuando necesitamos generar un índice, la otra persona debe dejar de escribir y darnos control absoluto del manuscrito hasta que la generación del índice haya terminado.

MANEJO DE LAS BASES DE DATOS MULTIUSUARIO

Para los sistemas de Bases de Datos Multiusuario, los DBMS proporcionan un balance entre dos objetivos conflictivos: la concurrencia y la integridad de los datos. Asegurar la integridad de los datos implica bloquear (lock) los archivos y los registros, lo cual reduce la habilidad de trabajar concurrentemente.

A diferencia de las bases de datos de un solo usuario, una aplicación multiusuario debe estar preparada para manejar las situaciones en las cuales un archivo no esté disponible cuando se le necesita, y en el cual un archivo puede contener datos modificados la próxima vez que se le necesite. Al convertir las aplicaciones de un solo usuario a aplicaciones multiusuario, el principal objetivo será el de maximizar la concurrencia tratando de mantener al mínimo el nivel de bloqueo de archivos que se requieran, y solamente bloquearlos mientras dure la operación.

Solamente el usuario que ha bloqueado un archivo puede escribir en él. Otros usuarios solo pueden leer los registros o archivos. Un archivo bloqueado puede ser accesado de nuevo dándole un comando para desbloquearlo (unlock). Este comando deberá liberar el bloqueo que se encuentre sobre el archivo. Los locks deben desaparecer también cuando se cierre el archivo o cuando el programa termine normalmente.

Procesamiento Alternado

En el ambiente multiusuario, es común que varios usuarios necesiten usar un dato al mismo tiempo, por lo cual un programa multiusuario necesita saber que hacer cuando un usuario no puede obtener el lock deseado sobre un recurso. El programa podría seguir intentándolo hasta que se produzca una interrupción, o bien deberá reintentar un cierto número de veces antes de reportarlo al usuario. La aplicación debe incluir un procedimiento de error para manejar actividades de procesamiento alternas cuando un usuario no pueda accesar el archivo.

Control de Concurrencia

Los sistemas multiusuarios bloquean los archivos y registros para prevenir que dos o más usuarios actualicen el mismo dato al mismo tiempo, lo cual causaría actualizaciones no integras. En una red típica, la solicitud de datos de un usuario requiere que se copie el archivo de disco a la memoria de la estación de trabajo. Esto puede causar actualizaciones erróneas aún si los archivos tienen locks, porque varios usuarios pueden tener copias de los mismos datos en la memoria de sus estaciones de trabajo. Supongamos que dos usuarios tienen un archivo en sus estaciones. Un usuario modifica los datos y los vuelve a escribir a disco, pero el otro usuario puede actualizar la copia anterior de los datos que aún está en su memoria. Cuando este usuario lo escriba nuevamente a disco, éste va a reemplazar la actualización anterior. Algunos de los métodos utilizados para manejar las actualizaciones concurrentes son:

- ***La apertura de archivos exclusivos.*** Este es el método más sencillo para controlar la concurrencia de actualizaciones y consiste en abrir los archivos involucrados en la actualización para uso exclusivo mientras dure la operación de lectura/escritura. Esto impide que otros usuarios puedan obtener copias de los archivos mientras estén ocurriendo las actualizaciones. Este método controla la concurrencia efectivamente pero restringe el acceso de otros usuarios a archivos enteros por períodos largos e innecesarios. Por esta razón es altamente ineficiente.
- ***Marcar los archivos en uso (flagging).*** Como parte de una aplicación, se puede crear un archivo maestro que almacene en cada registro la llave primaria y una bandera que indique si un registro está en uso. Cada solicitud de acceso (lectura o actualización) a un registro debe checar este archivo. Si un registro está en uso, se niega el acceso de otros usuarios a éste. Esto es más eficiente que el uso de archivos exclusivos, ya que solamente pone lock a los registros en uso y permite que otros usuarios accesen los registros restantes. Su inconveniente es la complejidad de programación, especialmente para una aplicación existente, porque cada solicitud de acceso a un registro debe llamar primero a una rutina que revise la disponibilidad del registro. Si un registro está disponible, la bandera debe ser actualizada para indicar que ahora está en uso, cuando se libera un registro, debe actualizarse la bandera para indicar que el registro se ha liberado.
- ***Uso de semáforos.*** Se puede implementar una función para controlar el acceso de registros basados en sus llaves primarias. Un semáforo es un mecanismo que asocia una etiqueta definida por el usuario (representando un recurso) con un contador que puede tomar un valor

de 0 a 127. Se usa para controlar el acceso a un recurso. Se definen operaciones de:

- OpenSemaphore,
- ExamineSemaphore,
- WaitOnSemaphore,
- SignalSemaphore, y,
- CloseSemaphore.

Cada semáforo tiene un contador abierto asociado a él. Un contador abierto es el número de procesos que tiene un semáforo abierto, es decir, el número de aplicaciones que han usado la operación de OpenSemaphore y que todavía no han llamado la de CloseSemaphore. La información del contador de apertura tiene poco uso, ya que solamente indica el número de procesos que pueden pedir acceso a los recursos.

El valor del semáforo es el número de recursos disponibles. Este número es puesto inicialmente por el primer proceso que llama a OpenSemaphore para un recurso particular. El valor del semáforo es mantenido por el sistema, y es el elemento crucial para controlar el acceso a un recurso. Cuando el valor se hace negativo, el proceso debe esperar a que se libere el registro, es decir debe esperar a que uno o más recursos sean liberados con la instrucción SignalSemaphore.

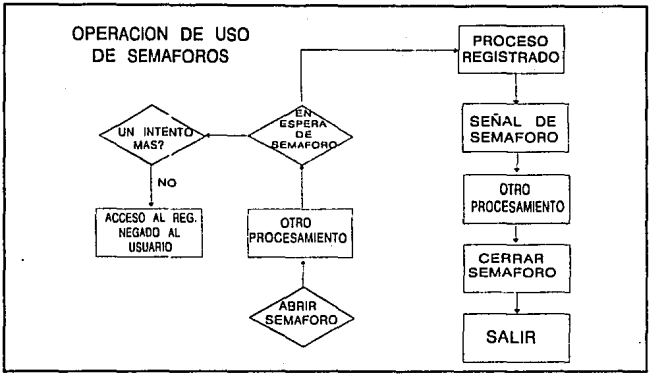


Figura 1.5 Operación de Uso de Semáforos

Una aplicación que use semáforos para controlar el acceso concurrente debe llamar a OpenSemaphore con la llave primaria como etiqueta, y un contador inicial de 1. Debe llamar WaitOnSemaphore cuando esté listo para accesar un registro; si el recurso no está disponible, la operación de WaitOnSemaphore pondrá al programa en cola hasta que el recurso esté disponible o que expire el tiempo límite. Cuando una aplicación deje de usar un registro, debe llamar a SignalSemaphore para incrementar el contador y permitir a otro usuario accesar el CloseSemaphore antes de terminar. Esto se ilustra en las figuras 1.5 y 1.6

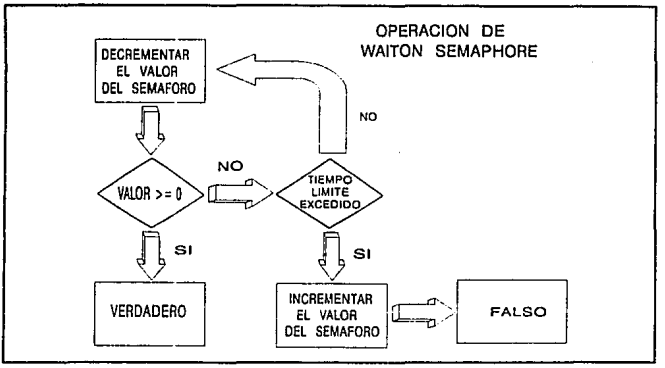


Figura 1.6 Operación de WaitOn Semaphore

- ***Comparar los valores de antes y después en arreglos.*** Se pueden implementar procedimientos para verificar que un registro no haya sido modificado por otro usuario entre el momento en que el registro es subido a la memoria de la estación de trabajo y el momento en que se actualiza. Esto es posible en productos que manejen arreglos.

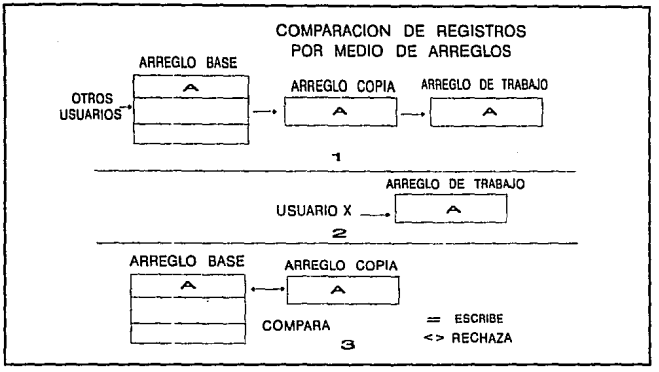


Figura 1.7 Comparación de registros por medio de arreglos

En el momento en que el registro es leído (arreglo base), su contenido debe ser copiado en un arreglo (arreglo copia). Luego se copia el registro base a un archivo temporal en el cual un usuario puede modificar los datos (arreglo de trabajo). Antes de escribir el registro nuevamente (arreglo de trabajo) se compara el arreglo copia con el arreglo base. Si hay diferencias, es porque el registro base se cambio entre el momento en que el usuario lo tomo y lo volvió a escribir. La edición que completó el último usuario queda desactualizada y necesita volverse a capturar. Este método se ilustra en la figura 1.7.

Prevención de Estancamientos (DEADLOCKS)

A menos de que se tenga cuidado, una aplicación multiusuario que lleva a cabo operaciones de lock de registros y archivos podría causar un problema de contención de recursos llamado estancamiento (deadlock).

Por ejemplo, tomemos las tablas, cuentas y clientes y los usuarios *A* y *B*. (Figura 1.8) El usuario *A* tiene un lock sobre la cuenta XXX. El usuario *B* tiene lock sobre el cliente YYY. Si XXX es la cuenta del cliente YYY y *A* necesita al cliente YYY no podría accesarlas porque B la tiene en lock, y a su vez no soltará el lock sobre XXX porque no puede terminar su operación. Al mismo tiempo *B* tiene que accesar la cuenta de su cliente pero no puede hacerlo ya que *A* lo tiene en estado de lock, por lo tanto al no poder terminar su operación se quedará esperando a que *A* lo libere, pero *A* está esperando a que *B* libere el suyo.

Esto ocurre cuando dos o más procesos están esperando poner lock a un recurso al que otro proceso ya le puso lock. Cuando esto pasa los procesos esperarán indefinidamente a menos que el estancamiento sea detectado y resuelto por el sistema.

Algunas alternativas para prevenir estos estancamientos son:

- ***Abrir archivos en un orden preestablecido***

  Una aplicación puede adoptar un estándar que pida que todos los archivos de la base sean abiertos y que se les ponga lock en un orden

predeterminado, sin importar el orden en el que serán usados. Una desventaja de esta alternativa es que requiere que la aplicación abra los archivos en un orden que probablemente no vaya con sus necesidades. El diseño de la aplicación es muy compleja para que los archivos sean puestos en lock de manera secuencial y que, además, algunos archivos pueden permanecer en lock por períodos prolongados si son abiertos mucho antes de que sean necesitados.

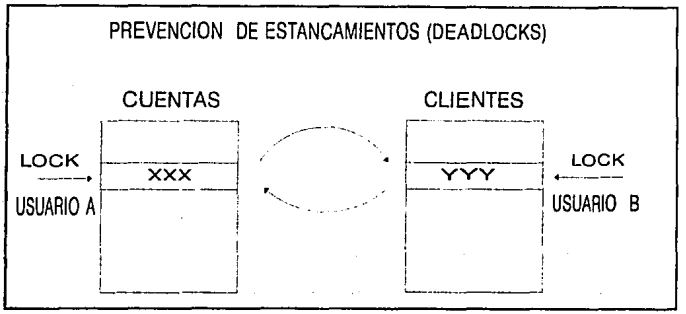


Figura 1.8 Prevención de estancamientos (DEADLOCKS)

- ***Preasignación de archivos en áreas de trabajo específicas***

Esta opción también requiere que se creen estándares en la aplicación. Cada archivo debe ser abierto sólo en áreas de trabajo preasignadas, las cuales deben ser seleccionadas en secuencia numérica.

Esta alternativa tiene varias desventajas. En primer lugar no es apta para aplicaciones grandes que usen un gran número de archivos que excedan el número máximo de áreas de trabajo. En segundo lugar, el estándar puede ser más difícil de aplicar que la apertura de archivos en orden preestablecido. Cada programador debe tener acceso a una lista que indique el área de trabajo asignada a cada archivo en el sistema. Mantener este sistema puede ser un verdadero problema cuando se tengan que cambiar las áreas preasignadas o cuando se agreguen nuevos archivos. Finalmente, este método evita los estancamientos cuando se trabaja con archivos, pero no con registros.

- ***Poner lock a los recursos necesarios desde la inicialización del programa***

Otra alternativa para evitar los estancamientos es hacer que cada programa de aplicación ponga lock en todos los archivos/registros, el programa debe liberar los otros. Esto evita los estancamientos resultantes de poner lock a los archivos y registros porque un programa no esperará a que estén disponibles.

Esta solución obliga a la aplicación a saber que recursos va a necesitar, lo cual puede presentar problemas si el procesamiento varía dependiendo de los datos proporcionados por los usuarios. También hace el mantenimiento más difícil. Esta solución tiene el potencial de monopolizar recursos porque estos son puestos en lock mientras dure el

proceso. Si se requieren muchos recursos, un programa puede esperar mucho para que todos los recursos estén disponibles simultáneamente.

- ***Implementar servicios de sincronización***

Este esquema se centra en una tabla que mantiene una lista de todos los registros a los que se va poner lock. Una aplicación puede requerir que al conjunto de archivos contenidos en la tabla se les ponga lock, de modo que todo el conjunto de archivos quede con lock, o ninguno de ellos. La aplicación debe liberar los archivos antes de que termine. Este concepto es útil porque el conjunto de archivos puede ser liberado pero mantenido en la tabla para uso posterior. Esto se puede hacer físicamente especificando los registros en un archivo DOS con el offset de inicio y la longitud del registro. Los servicios de lock lógico presentan una solución más razonable, realizando un seguimiento de los locks lógicos colocados en los nombres de los registros. Esto coordina al acceso multiusuario a los datos compartidos sin poner locks físicos. Las aplicaciones necesitan un mayor grado de control interno para implementar efectivamente el lock lógico. La clave de esto es establecer un estándar para asignar nombres a los registros a los que se va a poner el lock.

Borrado de registros en una red

El borrado de registros en un sistema multiusuario presenta otros problemas. Cuando se borren registros, se debe planear empacar (pack) el archivo para remover los registros permanentemente, recapturando el espacio que ocupan. Como esta operación incluye el uso exclusivo del archivo, presenta un problema de concurrencia en un ambiente multiusuario en el cual los usuarios frecuentemente requieren acceso a varios archivos.

En el momento que se está empacando un archivo, los usuarios no tienen acceso a éste, por lo tanto, se necesita una mejor alternativa. Una opción sería la de reciclar primero un registro, es decir ponerlo en blanco, y luego marcarlo como borrado. Cuando la aplicación necesite agregar un registro, puede "desborrar" el borrado y regresarlo a la aplicación como un registro nuevo. Cuando se acaben estos registros reciclados se necesitarán más registros nuevos. Lo que se recomienda en estos casos es anexar un lote de registros en blanco, y marcarlos como borrados, exceptuando los que se van a regresar a la aplicación.

BASES DE DATOS DISTRIBUIDAS

Debido al crecimiento de los volúmenes de datos, al gran número de usuarios que los accesan, y a las mejoras en los medios de comunicaciones ha sido necesario distribuir las Bases de Datos en distintos puntos geográficos para facilitar su acceso y darle mayor rapidez a las

operaciones que se realizan sobre ellas. Esto ha traído como consecuencia el concepto de las Bases de Datos Distribuidas.

En la figura 1.5 se observa un ambiente multiusuario en el que los datos se encuentran distribuidos, y la comunicación se hace a través de una red. Las características de los sistemas de Bases de datos distribuidos multiusuarios son las siguientes:

- Los datos están dispersos en varios lugares.
- La localización geográfica de los datos es transparente para los programas que los utilizan.
- Cada nodo puede trabajar en forma autónoma.
- La integridad de los datos es controlada por el DBMS.

Según Date, para que una base de datos distribuida y multiusuario cumpla con los objetivos de su creación, debe cumplir con las siguientes reglas:

- Autonomía local.
- No debe depender de un nodo centralizado.
- Debe operar continuamente.
- Debe ser independiente de su localización.
- Debe ser independiente de la fragmentación.
- Debe ser independiente en la generación de réplicas.
- Debe soportar el procesamiento de queries distribuidos.
- Debe ser independiente del Hardware en que corra.
- Debe ser independiente del sistema operativo.

- Debe ser independiente de la red.
- Debe ser independiente del DBMS.

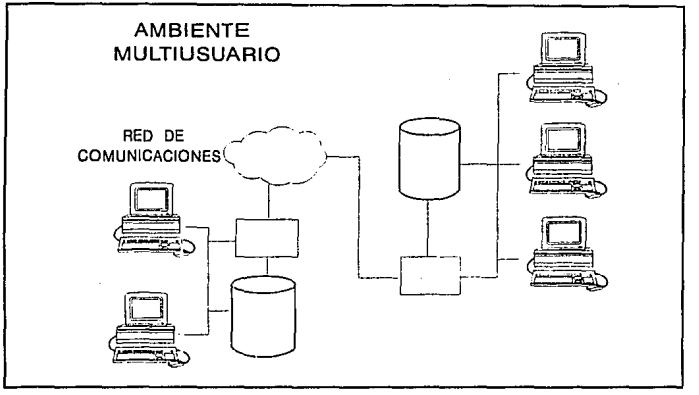


Figura 1.9 Ambiente Multiusuario

I.2 ARQUITECTURAS DE BASES DE DATOS

En esta sección se hace una revisión a las principales arquitecturas de bases de datos como son la CODASYL y la ANSI/SPARC, que fueron definidas por grupos de expertos para tener un marco de referencia para la descripción de conceptos generales sobre las bases de datos. Después se presentan las características principales de algunos modelos de organización de datos que son: el Modelo Jerárquico o de Árbol y el Modelo Reticular o de Red, el Modelo Entidad - Relación y el Modelo Semántico, además nos ocuparemos del modelo relacional que será presentado detalladamente en la siguiente sección.

I.2.1 NIVELES DE ARQUITECTURA DE ORGANIZACIÓN DE DATOS

El manejo y organización de la información en una Base de Datos puede ser vista en 4 niveles de Arquitectura que van desde el nivel de Organización Lógico hasta su correspondiente en el nivel de Organización Físico, desde el más independiente del equipo hasta el más dependiente.

En el nivel 4, el más alto, es donde se hacen las consideraciones lógicas de datos, el análisis de necesidades del usuario, de los datos con que cuenta y los que necesita y donde se especifica el flujo de la información y sus transformaciones. Posteriormente se definen entidades u objetos y sus relaciones funcionales.

A partir de éstas entidades y relaciones funcionales se construye o modela la base utilizando los modelos lógicos de organización, que serán explicados posteriormente. Cabe mencionar que éste nivel es el más independiente del computador donde se va a implementar la base de datos.

El siguiente nivel, se orienta a la implementación de los modelos de organización, se abordan acciones orientadas al desempeño, se empiezan a tomar en cuenta las limitaciones de los dispositivos físicos. Aquí se definen las estrategias de acceso para mejorar el desempeño en el manejo de datos (consulta, actualización). Por ejemplo, en este nivel se decide qué organizaciones de archivo se utilizarán, si son de acceso secuencial, random ó por organizaciones secundarias (listas invertidas, multilista, etc.) y sus posibles variaciones.

A los niveles 3 y 4 se les llama interface física-lógica. El nivel 2 refleja alternativas de organización Física y formatos de almacenamiento de datos en dispositivos de almacenamiento lineal para cada una de las organizaciones definidas en el nivel 3. Algunos autores sólo reconocen 3 niveles y manejan como uno solo al 3 y al 2.

El nivel 1, es más dependiente que los anteriores del equipo donde se implementa la Base de Datos; se refiere a las características y aspectos particulares de los dispositivos físicos de almacenamiento. En este nivel se ven aspectos como la organización de los dispositivos en subdivisiones; el almacenamiento de áreas de sobreflujo y características dependientes de

cada máquina, así como la localización de datos de control, verificación de integridad, etc.

I.2.2 ARQUITECTURA CODASYL

El Grupo de Trabajo para Bases de Datos (DBTG) de Codasyl presentó su reporte a inicios de 1971. En 1973 y 1978 su propuesta fue revisada y modificada sin que cambiaran sus conceptos básicos, muchos manejadores de Bases de Datos la tomaron como estándar en la década de los setenta y parte de los ochenta.

En Codasyl se proponen 3 niveles de organización de datos con su lenguaje asociado, más un lenguaje para procesar esos datos. Estos lenguajes son:

- El lenguaje de definición de datos a nivel de esquemas (Schema DDL), en el cual se hace la definición Lógica de la Base entera.

- El lenguaje de definición de datos a nivel subesquema (Subschema DDL). Aquí se tiene la definición lógica de vistas de usuario.

- El lenguaje de manipulación de Datos (DML), estilo Cobol con comandos como OPEN, CLOSE, GET, FIND, STORE, MODIFY, etc.

- El lenguaje de control de dispositivos/medios (DMCL), que finalmente fue reestructurado y renombrado como lenguaje de descripción de almacenamiento de Datos (DSDL), en el que se tiene una descripción única y formal de las estrategias de almacenamiento físico y de controles expresada en instrucciones de alto nivel.

La arquitectura Codasyl se muestra en la figura 1.10. En ésta arquitectura el esquema es la descripción lógica de la base de datos entera. Está formado por una descripción de los varios tipos de registro involucrados y de los tipos de conjuntos (SET TYPES) que los relacionan. Un tipo de registro esta formado por uno o más elementos dato; el elemento dato es la parte mas pequeña de la base de datos.

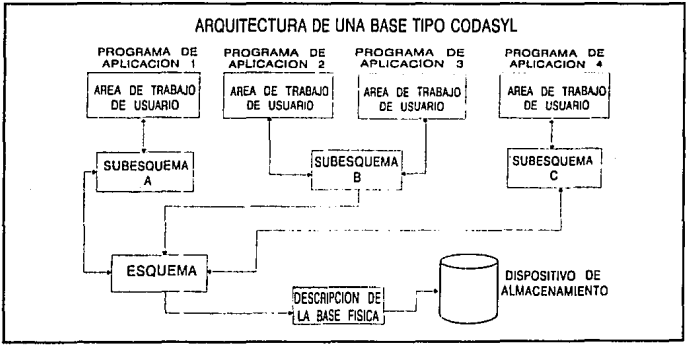


Figura 1.10 Arquitectura de una base tipo CODASYL

Un subesquema es un subconjunto lógico del esquema, o sea un subconjunto de los tipos de relaciones (sets), tipos de registros y elementos dato de los tipos de registro del esquema. La figura 1.11 presenta un esquema de ejemplo. El subesquema representa la vista o parte correspondiente de la base que le pertenece o utiliza un usuario. Todos los tipos de registro, tipos de relaciones y elementos dato que no estén en un esquema no podrán ser definidos ni introducidos a la base mediante un subesquema. En un esquema se pueden definir cualquier número arbitrario de subesquemas.

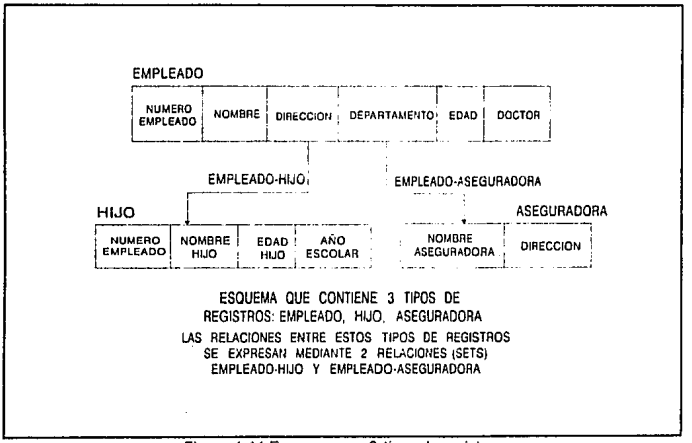


Figura 1.11 Esquema con 3 tipos de registros

I.2.3 ARQUITECTURA ANSI/SPARC

En 1972 fue fundado el Grupo de Estudio de ANSI/X3/SPARC sobre Sistemas de Administración de Bases de Datos por el Standards Planning and Requirements Committee (SPARC) de ANSI/X3 (American National Standards Committe on Computers and Information Processing), para determinar las áreas, si las había, de la tecnología de bases de datos para los cuales la estandarización era adecuada y producir un conjunto de recomendaciones de acción en cada una de las mismas. Este grupo definió una arquitectura generalizada que presentó en un reporte provisional en 1975.

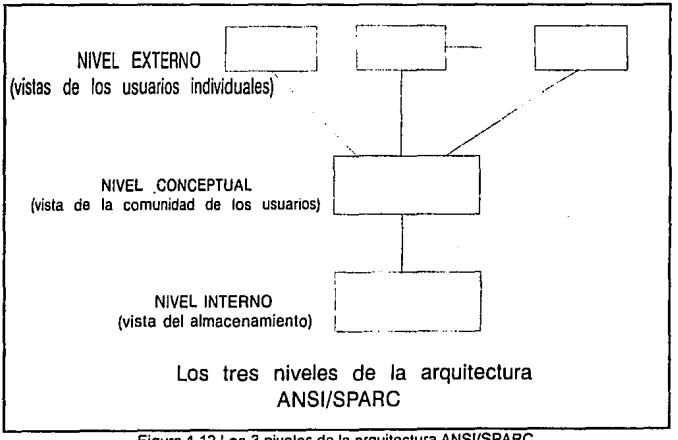


Figura 1.12 Los 3 niveles de la arquitectura ANSI/SPARC

En esta Arquitectura se contemplan 3 niveles:

- ***Nivel externo***

  En este nivel se aborda la definición de subesquemas de la base de datos, es donde se revisan y se definen las vistas de usuario; se cuenta con la lista de entidades, sus atributos, sus características. Aquí se tiene el control de acceso a la base de datos y cada usuario verá o accesará su información de uno o varios esquemas externos. Los esquemas externos pueden traslaparse para reflejar modelos de datos.

  En este nivel se cuenta con diversos lenguajes orientados a aplicaciones, por ejemplo: COBOL, FORTRAN, PASCAL, etc, que cuentan con sublenguajes para la definición de datos (DDL) que sirven para la descripción de los objetos de la base tal como los ve el usuario y sublenguajes de manipulación de datos (DML) que apoyan el manejo o procesamiento de esos objetos.

- ***Nivel conceptual***

  En este nivel se implanta la base de datos en su forma más general, se define el esquema de la base de datos, sus vínculos, relaciones, etc. La función principal en este nivel es el diseño y la generación de esquema.

Los esquemas reflejan el mundo real, el negocio, la sociedad, y si éstos cambian, habrá que modificar el esquema conceptual y con esto redefinir o ajustar algunos esquemas externos.

Todavía en este nivel se representa el contenido total de la base de datos en forma relativamente abstracta, en comparación con la forma en que los datos son almacenados físicamente.

Un registro en este nivel es llamado registro conceptual y no siempre es idéntico a un registro del nivel externo ni a un registro almacenado.

Para la definición de esquemas se utiliza el lenguaje de definición de datos (DDL) conceptual. Si se desea lograr independencia de los datos, estas definiciones no deben incluir ninguna consideración sobre la estructura de almacenamiento ni sobre la estrategia de acceso, únicamente definiciones de contenido de información.

- ***Nivel interno***

En este nivel se implantan las definiciones de almacenamiento a nivel registro almacenado; la vista interna aún se mantiene a un paso del nivel físico ya que no se involucra con registros físicos o bloques, ni a ninguna restricción específica de dispositivos tales como capacidades de cilindros o pistas. Aquí se define dónde están colocados los valores de atributo de la base de datos y cómo se obtiene acceso a ellos; se ve el desempeño y las afinaciones para equilibrar los tiempos de

respuesta de algunas transacciones, y se manejan los dispositivos físicos de almacenamiento.

En la figura 1.12 se muestra un diagrama de los niveles de arquitectura de ANSI/SPARC, en relación a los niveles de arquitectura de Codasyl.

I.2.4 MODELOS LÓGICOS

Una base de datos es un modelo en computadora de un sistema del mundo real. El contenido de la base de datos corresponde al estado del sistema de aplicación mientras que los cambios a la base de datos corresponden a eventos del sistema. En general, si el modelo puede ser descrito en términos de estructuras naturales, el trabajo del diseñador de la base de datos se simplificará tanto para la definición inicial como para los cambios subsecuentes. Algo muy importante, es que al usuario de la base de datos le será muy fácil formular consultas a la misma si el modelo refleja el ambiente de aplicación y usa términos y conceptos familiares para él.

NIVEL	CODASYL 71	CODASYL 78	ANSI/SPARC
NIVEL 4 LOGICO DE USUARIO	LENGUAJE DE DEFINICION DE DATOS (D.D.L.)	LENGUAJE DE DEFINICION DE DATOS (D.D.L.)	MODELO EXTERNO
NIVEL 3 LOGICO GLOBAL	LENGUAJE DE DEFINICION DE DATOS (D.D.L.)	LENGUAJE DE DEFINICION DE DATOS (D.D.L.)	MODELO CONCEPTUAL
NIVEL 2 DE ORGANIZACION FISICA	LENGUAJE DE DEFINICION DE DATOS (D.D.L.) LENGUAJE DE CONTROL DE DISPOSITIVOS/ MEDIOS	LENGUAJE DE DESCRIPCION DE ALMACENAMIENTO DE DATOS	MODELO INTERNO
NIVEL 1 DE DISPOSITIVOS FISICOS	LENGUAJE DE CONTROL DE DISPOSITIVOS/ MEDIOS	LENGUAJE DE DESCRIPCION DE ALMACENAMIENTO DE DATOS	

Figura 1.13 Niveles de la arquitectura CODASYL y ANSI/SPARC

El mecanismo formal utilizado para expresar la estructura lógica de los datos así como la semántica asociada es llamado modelo lógico de datos. Para que un modelo sea efectivo, debe representar lo más cercanamente posible los conceptos del mundo real que son usados para estructurar la información en una organización. Los ejemplos más conocidos de modelos de datos son: el Jerárquico, el de Red, el Relacional, el de Entidad-Relación y el Semántico. Los tres primeros son los tradicionales, orientados a registros y los dos últimos son orientados a entidades.

A. ***Modelo Jerárquico o de Árbol.***- Abarca tanto estructuras jerárquicas, jerárquicas invertidas y de árbol tipo CODASYL.

B. ***Modelo Reticular o de Red.***- Abarca las estructuras de red tipo CODASYL. Cabe señalar que las estructuras de árbol o jerárquicas son un caso especial del modelo de red.

C. ***Modelo Relacional.***- Está basado en el álgebra relacional. Concibe las estructuras de datos como conjuntos de tablas en la que se representan las entidades y sus relaciones.

D. ***Modelo Entidad-Relación.***- Ve al mundo en términos de entidades y relaciones entre estas.

E. ***Modelo Semántico.***- Utiliza construcciones orientadas al usuario que capturan la semántica del ambiente de la aplicación. Para lograr esto, utiliza conceptos como "generalización", "agregación", "datos derivados" y reglas de integridad para la definición de la base de datos.

La naturaleza de la organización de datos y sus relaciones en la mayoría de las situaciones prácticas es tal que se puede representar ya sea por árboles o por relaciones, sin mayor dificultad. Cualquier estructura de red puede ser convertida a una jerárquica introduciendo elementos redundantes, también cualquier estructura de red o jerárquica puede convertirse en una estructura relacional haciendo que las relaciones sean explícitas en lugar de implícitas. Una excepción de esto son las relaciones M:N (MUCHOS A MUCHOS).

Desde las décadas pasadas y sobre todo en equipos grandes existen infinidad de sistemas manejadores de bases de datos que usan el modelo jerárquico o el de red y no fue sino hasta los 80's que se han empezado a implementar comercialmente los manejadores relacionales con gran éxito; tanto en computadoras personales (PC'S) como en equipos mainframe.

MODELO JERÁRQUICO O DE ÁRBOL

Haciendo una analogía con las estructuras de árbol, en las cuales se tienen nodos y una relación de jerarquía se parte de un nodo padre a un nodo hijo, se define un conjunto (set) como una jerarquía de dos niveles de registros. El registro padre es llamado el propietario (owner). Cada propietario puede tener una ocurrencia del mismo y cualquier número de ocurrencias de los registros hijos llamados miembros (member). Las relaciones de propietario a miembro pueden ser 1:1 (uno a uno), 1:N (uno a muchos) pero no M:N (muchos a muchos).

Un registro propietario y uno miembro pueden ser el mismo. Una base de datos es descrita por un esquema consistente en uno o varios conjuntos (set) arreglados en una forma de árbol de muchos niveles, entonces un registro podrá ser miembro de un conjunto pero propietario de algún otro. Es fundamental, en cuanto a la vista jerárquica de los datos que cualquier ocurrencia de un registro específico se vea bajo el contexto de la relación padre-hijo.

Figura 1.14 Ejemplo de un árbol de dos niveles

Las estructuras jerárquicas representan bastante bien algunas estructuras de la vida real, pero no todas las estructuras de la vida real se pueden representar con estructuras jerárquicas, sobre todo donde se dan relaciones M:N.

MODELO RETICULAR O DE RED

En una estructura de árbol no se permite que un nodo hijo tenga más de un nodo padre. En una estructura tipo red, según CODASYL un miembro si puede tener más de un propietario siempre y cuando cada uno esté en un

conjunto diferente. Una relación M:N es una red en si misma. El modelo de red permite modelar en forma directa relaciones M:N, pero en éste hay un nuevo elemento que se llama conector, que se puede representar como un registro que contiene datos que describen la asociación entre propietarios y miembros.

Todas las ocurrencias de un conector para un propietario se colocan en una cadena que parte del mismo y retornan a él, igual sucede con las ocurrencias de un miembro. De esta manera cada ocurrencia del conector está en dos cadenas, en una de su propietario y en una de su miembro. Esto hace que la estructura interna de un archivo sea muy compleja, pues contiene muchos apuntadores. Un problema que presentan, es que para una misma pregunta, se puede tener acceso a la información por dos caminos, uno de los cuales, según las condiciones específicas de la pregunta, será mejor que el otro. Esto se debe tomar en cuenta cuando la programación es muy rígida.

EL MODELO ENTIDAD RELACIÓN (E-R)

El modelo entidad relación (E:R) está basado en una percepción del mundo real, que consiste en un conjunto de objetos básicos llamados "entidades" y la "relaciones" entre estos objetos. Fue desarrollado para facilitar el diseño de las bases de datos permitiendo la especificación de un esquema. Tal esquema representa la estructura lógica de la base de datos.

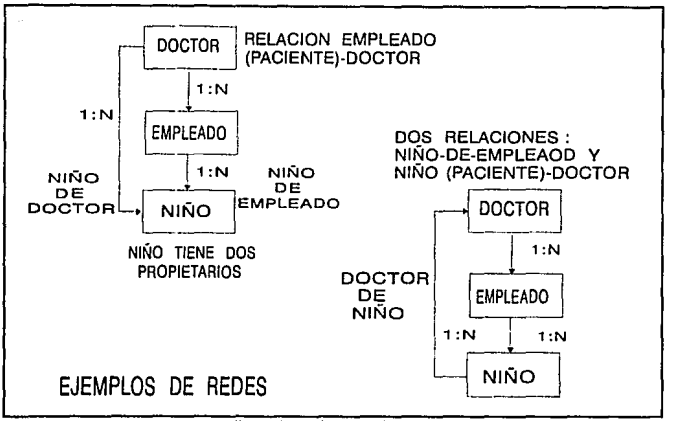


Figura 1.15 Ejemplos de redes

Diagrama Entidad-Relación

El modelo Entidad-Relación, está asociado con su respectivo diagrama de Entidad-Relación para representar las entidades y las relaciones entre estas. Esta es la mejor manera de expresar la vista global de la base de datos. Por consiguiente, se ha desarrollado una terminología para realizar Diagramas de Entidad-Relación.

En un Diagrama Entidad-Relación, las entidades están representadas por un rectángulo y una relación se representa con un rombo. Por ejemplo en la figura 1.17 **Cuenta** y **Transacción** son las entidades y **LOG** es la relación

entre estas, que describe el número de transacciones asociadas a una cuenta bancaria en particular. El **1** y la **n** que están junto a los arcos indican que el tipo de relación entre estas entidades es de 1 a muchos. Esto es, la cuenta bancaria de un cliente puede incluir diversas transacciones (depósito, abono, etc.)

Las entidades y las relaciones poseen propiedades. Estas propiedades pueden expresarse en términos del valor de sus atributos. Por lo tanto, opcionalmente, estas pueden representarse dentro del Diagrama de Entidad-Relación mediante círculos y los valores de estos se expresan en los arcos que van de la entidad al respectivo atributo.

Por cada función de una aplicación, se dibuja un diagrama de Entidad-Relación para describir los requerimientos de datos. De este modo, en el mismo diagrama se describe cómo se procesarán los datos, el punto de inicio del proceso y su secuencia.

Entidades y Conjuntos de Entidad

Una entidad es un objeto que existe y es distinguible de cualquier otro. Por ejemplo, el Sr. García con el número de seguro social 890-12-3456 es una entidad, ya que el número identifica únicamente a una persona en particular dentro de este universo.

Un conjunto de entidad contiene entidades del mismo tipo. El conjunto de todas las personas que poseen una cuenta bancaria, por ejemplo, puede ser definida como el conjunto de entidad denominado Clientes. De manera similar, el conjunto de entidad cuentas debe representar el conjunto de todas las Cuentas, de un banco en particular.
Los conjuntos de entidad no deben disociarse. Por ejemplo, es posible definir el conjunto de entidad de todos los empleados del banco (Empleados) y el conjunto de entidad de todos los clientes del banco (Clientes).

Una entidad está representada por un conjunto de atributos. Los posibles atributos del conjunto de entidad Clientes son **nombre**, **seguro social**, **calle** y **ciudad.** Los posibles atributos para el conjunto de entidad Cuenta son el **número** y **saldo.** Para cada atributo hay un conjunto de valores permitidos, llamado el Dominio de ese atributo. Por ejemplo, el domino del atributo **nombre** debe ser el conjunto de todas las cadenas de texto de una cierta longitud. De forma análoga el dominio del atributo número debe ser el conjunto de todos los enteros positivos.

Formalmente, un atributo es una función que mapea a un conjunto de entidad dentro de un dominio. Así, cada entidad está descrita por un conjunto de parejas (atributos o valores de dato); un par de cada atributo del conjunto de entidad. De este modo, la entidad de Cliente en particular está descrita por el conjunto {(**nombre** García), (**seguro social**, 890-123456), (**calle** Norte 41), (**ciudad** Monterrey)}, mediante la cual la entidad define a una persona llamada García con número de seguro social 890-12-

3456, que reside en la calle Norte 41 en la ciudad de Monterrey. Para ilustrar la diferencia entre un conjunto de entidad y una entidad particular de un conjunto, haremos una analogía con las nociones de lenguajes de programación. Un conjunto de entidad correspondería a una definición "Type". Una variable (un "Type") determinado tiene un valor particular en un instante determinado de tiempo. Así que, una variable de un lenguaje de programación correspondería al concepto de una entidad en el modelo E-R.

Relaciones y Conjuntos de Relación

Una relación es una asociación entre varias entidades. Por ejemplo, definimos una relación la cual asocia al cliente "García" con la cuenta 401. Esta especifica que García es un cliente con número de cuenta 401.

Un conjunto de relación es un conjunto de relación del mismo tipo. Formalmente, es una relación matemática sobre n>=2 conjuntos de entidad. Si (e_1, e_2,...,e_n), son conjuntos de entidad, entonces un conjunto de relación R es un subconjunto de {(e_1,e_2,...,e_n)}, donde (e_1, e_2,...,e_n) es una relación.

Para ilustrar esto, consideremos las dos entidades, Clientes y Cuentas de la Figura 1.16 donde se define la relación Cliente-Cuenta para denotar la asociación entre clientes y cuentas bancarias.

La relación Cliente-Cuenta es un ejemplo de conjunto de relación binaria, esto es, una en la cual se involucran dos conjuntos de entidad. Muchos de los conjuntos de relación en una base de datos son binarios. Sin embargo, ocasionalmente, hay conjuntos de relación que involucran más de un conjunto de entidad. Por ejemplo consideremos la relación ternaria (García, 401, Centro) que especifica que el cliente García tiene la cuenta 401 en la sucursal Centro. Esta relación es una instancia en una relación Cliente-Cuenta-Sucursal que involucra los conjuntos de entidad Cliente, Cuenta y Sucursal.

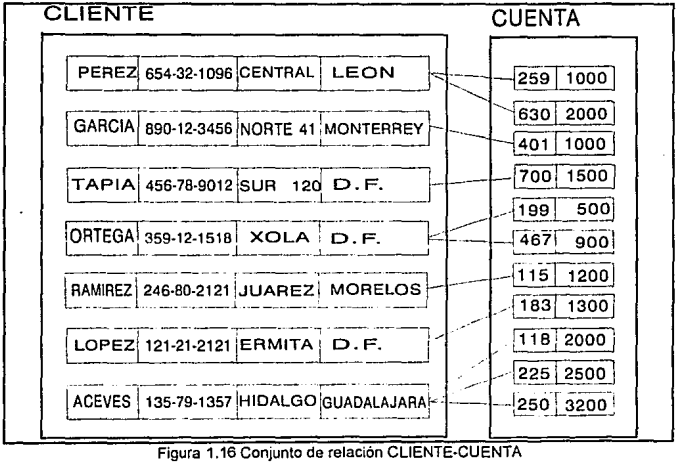


Figura 1.16 Conjunto de relación CLIENTE-CUENTA

La función que juega una entidad en una relación se denomina su rol. Los roles son normalmente implícitos y usualmente no son especificados. Sin embargo, son utilizados cuando el manejo de una relación necesita clarificación. Tal es el caso cuando el conjunto de entidad no está bien definido.

Una relación además puede tener atributos descriptivos. Por ejemplo, la fecha debe ser un atributo de conjunto de relación Cliente-Cuenta. Esto específica que la última fecha en la cual el cliente tuvo acceso a la cuenta. La relación Cuenta-Cliente de (García, 401) está descrita por {(fecha, abril 21 1994)}, que indica que la última vez que García accesó la cuenta 401 fue en abril 21 de 1994.

Tipos de Relación

Un esquema implementado por el modelo E-R define ciertas restricciones que deben conformar el contenido de la base de datos. Una de estas restricciones es el tipo de relación entre entidades, estas relaciones pueden comprenderse como un mapeo cardinal que expresa el número de entidades a la cual otra entidad puede ser asociada vía una relación.

Los mapeos cardinales son los más utilizados en la descripción de conjuntos de relación binaria, además, ocasionalmente contribuyen a la descripción de conjuntos de relación que involucran más de dos conjuntos

de entidad. Para un conjunto de relación binaria entre los conjuntos de entidad **A** y **B**, el mapeo cardinal debe ser uno de los siguientes:

- **Uno a Uno**
- **Uno a Muchos**
- **Muchos a Uno**
- **Muchos a Muchos**

Para ejemplificar lo anterior, considere el conjunto de relación Cliente-Cuenta. Si en un banco en particular una cuenta pertenece únicamente a un cliente, y un cliente puede tener varias cuentas, entonces el conjunto de relación es uno a muchos en el sentido cliente a cuenta. Si una cuenta puede pertenecer a varios clientes (como cuentas entre familiares), el conjunto de relación es muchos a muchos. Existen dependencias entre entidades. Específicamente, si la existencia de una entidad **X** depende de la existencia de la entidad **Y**, entonces se dice que **X** está dependiendo existencialmente de **Y**. Operativamente si **Y** es eliminada, **X** también lo será. Entonces se dice que **Y** es la entidad dominante y **X** es la entidad subordinada.

Para ilustrar esto, consideremos los conjuntos de entidad Cuenta y Transacción. Formaremos la relación Log entre estos dos conjuntos, la cual especifica que para una cuenta en particular hay varias transacciones. Esta relación es uno-a-muchos desde Cuenta hacia Transacción. Cada una de las entidades de Transacción asociadas con una en Cuenta deben eliminarse también. En cambio, las entidades de Transacción pueden

eliminarse desde la base de datos sin afectar cualquier Cuenta. Por lo tanto, el conjunto de entidad Cuenta es dominante y el de Transacción es el subordinado en la relación Log. La figura 1.17 muestra el diagrama E-R de la relación Log. En notación matemática esto puede definirse de la siguiente manera:

Sea **A** un conjunto de entidad dependiente con atributos descriptivos a_1, a_2,...,a_n. Sea **B** el conjunto de entidad dominante **B** sobre el cual **A** es dependiente. Si la llave primaria de B consiste de los atributos b_1, b_2,...b_n. Representamos al conjunto de entidad **A** por una tabla llamada **A** con una columna por cada atributo de el conjunto:

$$\{a_1, a_2,...,a_n\} \cup \{b_1, b_2,...,b_n\}$$

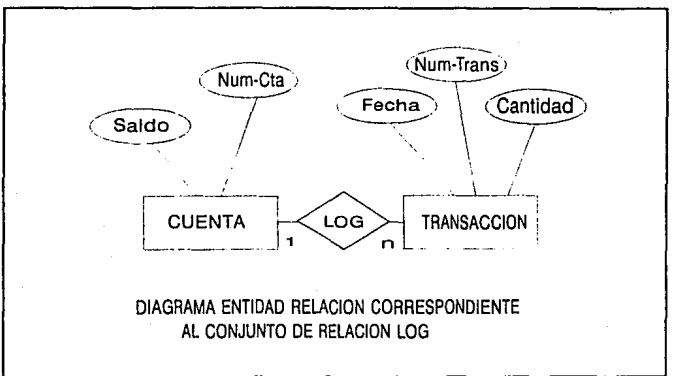


Figura 1.17 Diagrama entidad-relación correspondiente al conjunto de relación LOG.

Para explicar lo anterior, considere el mismo diagrama Entidad-Relación de la figura 1.17 correspondiente al conjunto de relación LOG. Como puede apreciarse, el conjunto de entidad Transacción posee tres atributos: Num-Trans, Fecha y Cantidad. La llave primaria de el conjunto de entidad Cuenta, mediante la cual Transacción es dependiente, es Num-Cta (figura 1.18). Por lo tanto, de acuerdo con la teoría anteriormente expuesta, Transacción es representada por una tabla de cuatro columnas denominadas Num-Cta, Num-Trans, Fecha y Cantidad, la cual corresponde a la unión de sus atributos y la llave primaria de el conjunto de entidad dominante "Cuenta". La figura 1.19 muestra la relación de dependencia de Transacción respecto de Cuenta.

Llaves Primarias

Una tarea importante en el modelado de una base de datos es especificar como distinguir las relaciones y las entidades. Conceptualmente, las entidades individuales y las relaciones son distintas, pero desde la perspectiva de las bases de datos, la diferencia entre ellas debe expresarse en términos de sus atributos. Para hacer tales distinciones, una llave se asigna para cada conjunto de entidad. Esta llave es un conjunto de uno o más atributos, que se toman colectivamente, permitiéndonos identificar únicamente a una entidad en el conjunto de entidad. Por ejemplo, el atributo Seguro-Social del conjunto de entidad Clientes es suficiente para distinguirlo de otra entidad Clientes. Por lo tanto, Seguro-Social es una llave. Similarmente, la combinación de Nom-Clien y Seguro-Social es una

llave para el conjunto de entidad Clientes. Sin embargo, el atributo Nom-Clien no es una llave ya que varias personas pueden tener el mismo nombre. Dado que la llave puede ser una combinación de atributos, esta puede llegar a ser muy grande, por lo que se desea minimizarla. A esta llave se le denomina ***llave candidata.***

Es posible que haya varios conjuntos distintos de atributos los cuales pueden ser llaves candidatas. Usaremos el término de llave primaria para denotar a una llave candidata que haya sido seleccionada por el diseñador de la base de datos como el medio principal de identificación de entidades entre un conjunto de entidad.

Es posible que una entidad no posea los suficientes atributos para formar una llave primaria. Por lo tanto ésta debe ser dependiente de la entidad dominante o que posee la llave primaria. Esto se relaciona con la dependencia existencial que se mencionó con anterioridad, en el ejemplo de la relación LOG que se estableció entre las entidades Cuenta y Transacción.

CUENTA

NUM-CTA	SALDO
259	1000
630	2000
401	1500
700	1500
199	500
467	900
115	1200
183	1300
218	2000
225	2500
210	2200

Figura 1.18 La Tabla de Cuenta

TRANSACCION

NUM-CTA	NUM-TRANS	FECHA	CANTIDAD
259	5	11 MAY 85	+50
630	11	17 MAY 85	+70
401	22	23 MAY 85	-300
700	69	28 MAY 85	-500
199	103	03 JUN 85	+900
259	6	07 JUN 85	-44
115	53	07 JUN 85	+120
199	104	13 JUN 85	-200
259	7	17 JUN 85	-79

Figura 1.19 La tabla de Transacción.

Los conjuntos de relación tienen, además, llaves primarias. Estas llaves primarias están formadas tomando todos los atributos que comprenden las llaves primarias de los conjuntos de entidad que definen al conjunto de

relación. Por ejemplo, Seguro-Social, es la llave primaria de Clientes y Num-Cta es la llave primaria de cuenta. Por lo tanto, la llave primaria del conjunto de relación Clien-Cta es (Seguro-Social, Num-Cta).

MODELO SEMÁNTICO

Como se dijo antes, los medios tradicionales están orientados a registros, e incluso las entidades u objetos del mundo real todavía no pueden ser directamente expresados en los modelos tradicionales. Una de la ventajas del modelo de datos semántico es que está más orientado al usuario y que no lo restringe a usar ninguna estructura de implementación en particular.

En otras palabras, tanto el diseñador de la base de datos como el usuario de ésta deben pensar en el nivel abstracto (entidades y atributos) en el que están acostumbrados sin tener que preocuparse por pensar a nivel estructura de datos (archivos, registros, tuplos etc.).

Conceptos de Modalado Semántico

Los componentes primarios del modelo semántico son la representación directa de objetos, atributos, relaciones entre objetos, redes de generalización, componentes derivados del esquema; así como los medios para especificar en forma natural la semántica de la aplicación. Todo esto basándonos en el Modelo de Datos Semántico de Hammer y

McLeod(1981). Las siguientes definiciones nos podrán ayudar a entender mejor este modelo:

- ***Clases y Subclases***

En el Modelo Semántico los tipos de objetos o entidades son llamados clases.

Una clase es un medio de estructuración y no implica ninguna implementación física.

Cada entidad es un miembro de alguna clase. Las entidades en una base semántica están organizadas dentro de una colección de clases, donde cada clase corresponde a una colección significativa de entidades que comparte características comunes.

El diagrama de clase de un esquema se representa con un óvalo. El nombre de la clase identifica y establece el conjunto de entidades que son miembros de ella.

En el esquema las "Subclases" también son representadas con un óvalo pero están conectadas a su "Superclase" inmediata por una flecha doblemente marcada.

En este trabajo se usará el término "Clase" y "Subclase" indistintamente cuando la diferencia no sea importante.

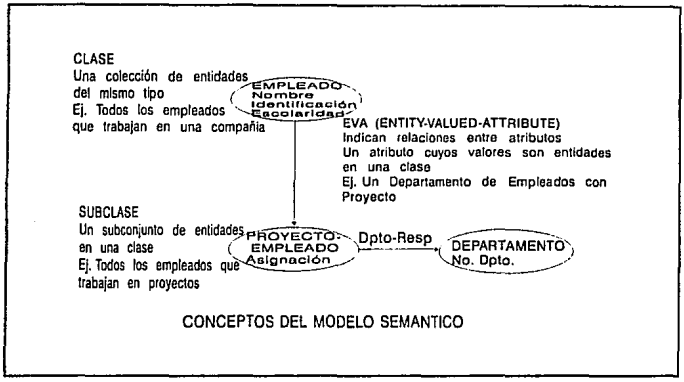


Figura 1.20 Conceptos del Modelo Semántico

Una clase tiene una colección de atributos que describen a los miembros de la clase completamente.

- ***Atributos***

Una clase tiene asociados un conjunto de atributos si la clase está definida por la identificación de sus atributos. Un atributo es por lo tanto, una característica descriptiva de la clase.

Hay dos tipos de atributos de entidad o miembro y atributos de clase.

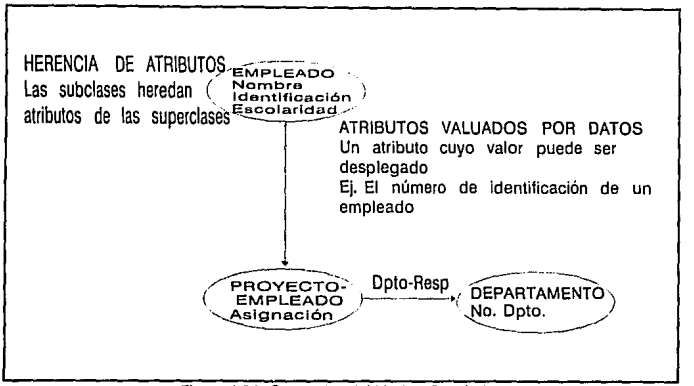


Figura 1.21 Conceptos del Modelo Semántico

Algunas de las características más importantes de los atributos en el Modelo Semántico son:

1. Un atributo es identificado por un nombre y es de un tipo específico. El tipo de atributo puede ser primitivo, como entero, real, carácter, fecha, etc., y puede ser definido por el usuario (igual al concepto de tipos de registro en Pascal).

2. Un atributo puede ser univaluado ó multivaluado; Si es multivaluado, se pueden especificar sus valores máximo y mínimo. También los atributos multivaluados pueden ser definidos para ser "DISTINTOS" (DISTINCT) o DISJUNTOS "DISJOINT". DISTINCT significa que no hay dos valores de un atributo para una entidad específica (común) mientras que

DISJOINT implica que el valor del atributo no se repite en dos entidades diferentes.

3. Los atributos pueden ser atributos miembro o atributos de clase como se definió anteriormente.

4. El atributo puede ser especificado como requerido (REQUIRED) en cuyo caso no se permite un valor nulo. Los atributos pueden ser definidos de sólo lectura (READONLY) esto es, no pueden ser cambiados y se puede especificar un valor inicial (INITIALVALUE) para ellos.

5. Un atributo puede ser derivado (DERIVED) de los otros atributos del esquema. Esto puede ser muy útil para definir vistas de usuario simplificadas.

6. El uso más importante de los atributos en el Modelo Semántico es que pueden ser utilizados para definir relaciones (RELATIONSHIPS). Para esto, pueden ser clasificados en:

   - Atributos valuados por datos (DATA VALUED ATTRIBUTES-DVA) que definen relaciones explícitas, y,
   - Atributos valuados por entidades (ENTITY VALUED ATRIBUTES-EVA) que definen relaciones implícitas.

Los valores asociados con los DVAs son desplegables. Los valores EVAs son algunas otras entidades o conjuntos de entidades-si el EVA es multivaluado- de la base de datos.

En el diagrama del esquema, los EVAs se representan por flechas. Una flecha con doble punta se utiliza para representar un EVA multivaluado.

- ***Herencia de Atributos***

En una base de datos de Modelo Semántico, una entidad puede pertenecer sólo a una clase, pero puede ser miembro de mas de una subclase, si las subclases tienen al menos una clase padre común. Cualquier miembro de una subclase automáticamente hereda todos los atributos de todas las superclases, esto es, la herencia de todos los atributos de todas las superclases que permite hacer más directa la representación de muchas situaciones del mundo real y ayuda a la formulación de consultas (QUERIES) a la base de datos.

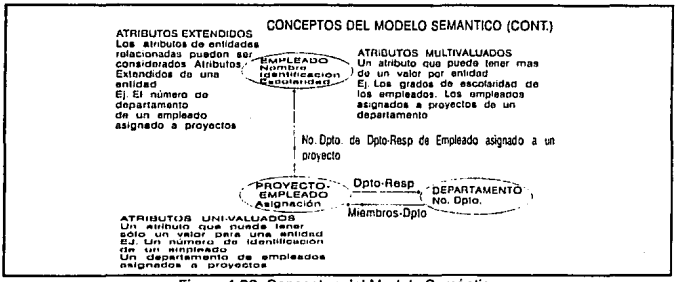


Figura 1.22 Conceptos del Modelo Semántico

I.3 EL ENFOQUE RELACIONAL DE LAS BASES DE DATOS

Un sistema de Base de Datos debe ser capaz de representar y manipular entidades (registros o segmentos) y sus relaciones de manera fácil y conveniente. En el enfoque jerárquico o de árbol, se representa la relación entre dos segmentos por la posición relativa de arriba hacia abajo y de derecha a izquierda de los tipos de segmentos involucrados. En el enfoque de red, las relaciones se representan mediante mecanismos de "set" con uso de apuntadores, los cuales enlazan a un tipo de registro propietario con un tipo de registro miembro. Para el caso de una base de datos grande y complicada, el modelo lógico y la manera como pueden tener acceso a él los usuarios por medio de un lenguaje de manipulación de datos (DML) pueden volverse muy complejos. Aún más, el acceso a los datos se inclina demasiado a las rutas de acceso en términos de los enlaces o posiciones jerárquicas que estableció el diseñador. De esta manera puede ocurrir que muchos cambios a la base de datos violen la independencia de los datos o afecten los programas de aplicación.

El enfoque relacional a bases de datos concebido por E.F. Cood, constituye un enfoque muy diferente para la descripción y manipulación "lógica" de los datos. Se esfuerza por evitar muchas de las desventajas que se han mencionado. En forma concisa, visualiza la base de datos lógica como una simple colección de tablas bidimensionales llamadas "Relaciones". Estas tablas son planas, en el sentido de que no hay grupos repetitivos. Los usuarios las comprenden y manejan fácilmente con poco o ningún entrenamiento en programación, y no implican consideración alguna sobre

aspectos posicionales, de apuntadores o de rutas de acceso. Aparentemente es más fácil visualizar y manipular una tabla como la de la figura 1.23A que en su forma convencional equivalente dada en la figura 1.23B, así mismo, es posible transformar cualquier base de datos a una tabla del tipo plana o relacional, mediante la introducción de redundancia adicional.

El enfoque relacional introduce terminología propia y exhibe una tendencia a usar términos poco convencionales relacionados con las matemáticas. El enfoque relacional se fundamenta en la teoría matemática de las relaciones, por lo cual, posee un buen fundamento teórico.

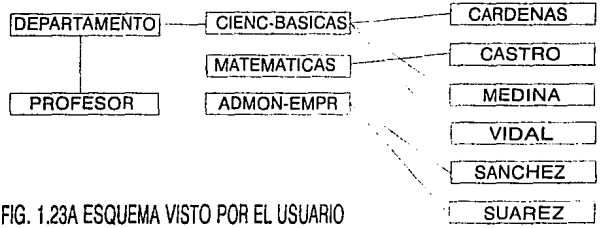


FIG. 1.23A ESQUEMA VISTO POR EL USUARIO

DEPARTAMENTO	PROFESOR
CIENC-BASICAS	CARDENAS
CIENC-BASICAS	CASTRO
CIENC-BASICAS	MEDINA
MATEMATICAS	VIDAL
ADMON-EMPR	SANCHEZ
ADMON-EMPR	SUAREZ

FIG. 1.23B ESQUEMA TABULAR, RELACIONAL O PLANO

La figura 1.23C muestra la arquitectura relacional que se propone. La Base de Datos Global es un conjunto de relaciones, al que se hace referencia generalmente como modelo relacional de datos, relaciones base o base de datos relacional. El modelo de datos lo define el administrador de la base de datos (DBA) mediante un lenguaje de descripción del modelo relacional de datos.

Al modelo particular de datos de un usuario, que se extrae del modelo de datos global, se le llama el submodelo relacional de datos o vista. El submodelo de datos lo define el DBA y/o lo llama el usuario vía un lenguaje de descripción del submodelo relacional de datos. El submodelo de datos es una colección de relaciones que pueden derivarse de las del modelo de datos mediante ciertas operaciones relacionales. El lenguaje de descripción del submodelo de datos incluirá las operaciones o mecanismos relacionales necesarios para formar submodelos permisibles a partir del modelo. De hecho, el lenguaje de descripción de submodelos es básicamente el mismo sublenguaje de datos, con ciertas adiciones para definición de datos.

Puede definirse un número arbitrario de submodelos sobre un modelo de datos dado. Un submodelo podría ser el modelo de datos completo. Un número arbitrario de usuarios puede compartir un modelo de datos dado vía los submodelos.

I.3.1 EL MODELO RELACIONAL

Desde una perspectiva histórica, el modelo de datos relacional es relativamente nuevo. Los primeros sistemas de base de datos fueron diseñados utilizando los modelos jerárquico y de red.

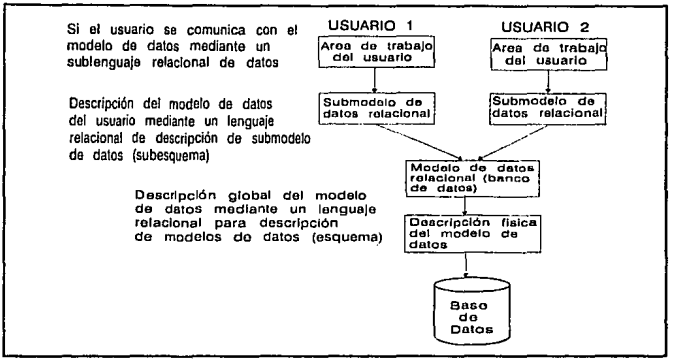


Figura 1.23C Arquitectura relacional

El modelo de datos relacional representa, como ya se mencionó, a la base de datos como una colección de tablas. Aunque las tablas son simples y de noción intuitiva, están en correspondencia directa entre el concepto de tabla y el concepto matemático de relación.

En años posteriores a la introducción del modelo relacional, se desarrollo una teoría para bases de datos relacionales. Esta teoría asiste en el diseño

de bases de datos relacional y en el procesamiento eficiente de los requerimientos de información de los usuarios desde la base de datos.
Una base de datos relacional consiste en una colección de tablas, a cada una de las cuales se le asigna un nombre único. Cada tabla tiene una estructura similar a las representadas en el modelo Entidad-Relación. Un renglón en una tabla representa una relación entre un conjunto de valores. Dado que una tabla es una colección de tales relaciones, encierra el concepto de tabla y el concepto matemático de relación, del cual el modelo de datos relacional toma su nombre.

En términos menos formales, una relación es una tabla bidimensional de n columnas constituidas por un conjunto de tuplos de n elementos (n-tuplos). Cada una de las columnas en una relación es un conjunto de valores de elementos de datos (tipo de atributo o campo) al que se le conoce como dominio.

Considere la siguiente tabla de Saldos:

Nom-Suc	Num-Cta	Nom-Clien	Saldo
Centro	101	Orozpe	500
Coyoacan	215	Murillo	700
Taxqueña	102	Avila	400
Valle	222	Jimenez	750

Figura 1.24 La tabla de Saldos

Esta posee cuatro atributos: Nom-suc, Num-cta, Nom-Clien y Saldo. Para cada atributo, hay un conjunto de valores permitidos, llamado dominio de un atributo. Por ejemplo para el atributo Nom-suc, el dominio debe ser el conjunto de todos los nombres de sucursal. Denotemos a este conjunto con D_1, D_2 al conjunto de todos los números de cuenta (Num-cta), D_3 al conjunto de todos los nombres de clientes (Nom-clien) y D_4 al conjunto de todos los Saldos. Cada renglón debe consistir de 4 tuplos (v_1, v_2, v_3, v_4) donde v_1 es el nombre de la sucursal (esto es, v_1 está en el dominio D_1), v_2 es el número de cuenta (v_2 está en el dominio D_2), v_3 es el nombre del cliente (v_3 está en el dominio D_3) y v_4 es el saldo (v_4 está en el dominio D_4).

En el lenguaje formal de las matemáticas, dados n conjuntos D_1, D_2,...,D_n; R es una relación sobre estos n conjuntos si R es un conjunto de n-tuplos cada uno de los cuales tiene su primer elemento en D_1, su segundo elementos en D_2, etc. Si la relación tiene n dominios ó columnas se dice que es de grado n. A las relaciones de grado 2 se les llama binarias, a las de grado 3 ternarias y a las grado n se les denomina enearias. El término tabla se refiere a una colección de tuplos de una relación dada.

Una relación o tabla es un arreglo bidimensional con las siguientes características:

1. Cada entrada en la tabla es un elemento de datos o dato elemental; no hay grupos repetitivos. Es decir, cada dominio debe representar a una sola relación. Se dice que una relación está normalizada si no tiene grupos repetitivos.

2. A cada columna, esto es, al dominio, se le asigna un nombre diferente y está constituido por valores del mismo dato elemental.

3. Todas las hileras o tuplos son distintas; no se permiten duplicados.
4. Las hileras y columnas pueden ordenarse en cualquier secuencia en cualquier momento, sin que esto afecte el contenido de la formación o la semántica aplicada.

Cada tuplo o relación debe poseer una llave que lo identifica unívocamente y lo diferencia de otros tuplos de esa relación. La llave es un dominio simple o una combinación de dominios. Una llave constituida por una combinación de dominios es no redundante si ninguna entidad de la llave puede eliminarse o borrarse sin destruir la habilidad de identificar unívocamente a cada tuplo. Puede existir más de un conjunto de dominios que pueden constituir una llave; es decir, que identifica unívocamente a un tuplo y que es no redundante. A estos conjuntos se les denomina llaves candidatas. La llave primaria es el conjunto de dominios que se selecciona para identificar a los tuplos. Normalmente debería ser el que tuviera el número mínimo de dominios.

VENTAJAS DEL MODELO DE DATOS RELACIONAL

La información es presentada al usuario final con un modelo de datos simple. Sus requerimientos están formulados en términos del contenido de

la información y no refleja ninguna complejidad en los aspectos orientados al sistema. Un modelo de datos relacional es lo que el usuario ve, pero no necesariamente lo que físicamente se implemento.

Requerimientos de no procedimientos. Dado que no hay dependencia posicional entre las relaciones, no requiere reflejar alguna estructura preferida y por lo tanto puede ser de no procedimientos.

Independencia de los datos. El modelo de datos relacional elimina los detalles de estructura de almacenamiento y estrategia de acceso desde la interface del usuario. El modelo proporciona un grado relativamente grande de independencia de datos.

I.4 ANÁLISIS DE HERRAMIENTAS DE SOFTWARE DE BASES DE DATOS PARA PC´s

La aparición de la computadora personal en el mercado, y su rápida evolución tecnológica, así como el abatimiento de costos, han propiciado su uso en gran escala. En consecuencia, ha surgido una gran cantidad de software de bases de datos.

Por la facilidad de implantación del modelo relacional, el desarrollo de los manejadores de bases de datos se han orientado a éste. El poder y la facilidad de uso son los aspectos fundamentales para los usuarios de bases de datos relacionales. Entre las características de poder que ofrecen la mayoría de los fabricantes se cuenta con la posibilidad integrada de enlace en redes, la posibilidad de colocar un programa nuevo de software en una red, para ofrecer acceso compartido a datos y lenguajes de programación que se pueden utilizar para desarrollar aplicaciones adecuadas a las necesidades de cada usuario, incluyendo menús y reportes.

Uno de los aspectos importantes de los manejadores de bases de datos más recientes es que todos logran que ese poder sea más accesible con una serie de menús, editores de toda la pantalla y características automáticas que vuelven relativamente sencilla la realización de operaciones relacionales. Productos pioneros como Dbase por ejemplo, eran predominantemente controlados por línea de comandos. Para aprovechar ese software se tenía que aprender un gran número de comandos y procedimientos. Esto ya no sucede con los productos actuales.

La demanda del usuario de acceso a bases de datos ha apresurado a los productores de software a diseñar interfaces realistas. Recientes adelantos técnicos (CPUs más veloces, memoria expandida, monitores gráficos de alta resolución) han sido de gran ayuda.

Con la mayoría de los productos actuales, se puede colocar de inmediato una base de datos enlazada a una red en entorno multiusuario. No obstante, aún las versiones para redes de éstos productos tienen algunas limitaciones. Por ejemplo a diferencia de sus contrapartes para minicomputadoras, estos no aceptan llamadas a otros lenguajes perdiendo flexibilidad en un entorno que se sustente firmemente en la programación habitual. Si vemos hacia un futuro cercano en el cual el procesamiento distribuido cobra cada vez mayor importancia, con seguridad veremos cada vez más de estos productos que puedan enlazarse con una PC, estación de trabajo o manejador de base de datos de un sistema anfitrión existente. Mientras tanto resulta alentador observar que los usuarios cotidianos y los programadores experimentados ahora pueden utilizar manejadores de bases de datos poderosos en muchos aspectos.

A continuación se hace un análisis de los manejadores de bases de datos más utilizados actualmente.

PRODUCTO [1]	COMPAÑIA	PRECIO	RAM
dBase IV	Ashton-Tate Corp.	795	640K
FoxPro	Fox Software, Inc.	795	640K
Paradox	Borland International	795	640K
Informix	Informix Sotware	795	640K

[1] Fuente : DOS Databases at Work Byte Magazine, January 1992

I.4.1 PARADOX versión 3.5

Paradox está diseñado para usuarios de computadoras con todos los niveles de experiencia, desde principiantes hasta los más experimentados. No se requiere ser un programador para usar Paradox, la interface con el usuario es a través de menús descendentes generales de los cuales se puede lograr el acceso directo a casi todas las características del programa. (Como sucede en Lotus 1-2-3).

Para satisfacer requisitos de uso de bases de datos más complejos, se puede extender el poder de Paradox usando la utilería Paradox Personal Programmer, basada en menús simplifica la creación de códigos para aplicaciones. Constituye también una herramienta del programador para diseñar prototipos, lo que simplifica las tareas de codificación arduas.

Otra herramienta disponible es PAL (Paradox Application Language), que es un lenguaje estructurado que ofrece una programación de procedimientos que viene completa con variables, arreglos y construcciones "if-then-else", y proporciona un acceso a todos los comandos del menú de Paradox, así como a una gran variedad de funciones financieras, numéricas y de manejo de cadenas de caracteres.
Paradox se basa en cuatro elementos: tablas para organizar datos, formas de entrada de datos y de query, formatos de reportes y escritos. Paradox considera que cada uno de estos elementos es un objeto con el cual se puede realizar una tarea, donde a su vez cada objeto se puede emplear en varias funciones.

Paradox opera a través de una serie de vistas (diferentes formas en que se pueden observar los datos) que pueden estar desplegadas en el área de trabajo simultáneamente dependiendo de la configuración del sistema, de manera individual o enlazadas a través de un campo común, en un formato parecido al de una hoja de cálculo con etiquetas de campos colocadas en sentido horizontal en la parte superior de la pantalla, donde cada renglón constituye un registro. Las teclas de control de cursor se utilizan para "navegar" la vista a fin de observar todos los elementos de un registro dado.

Si se prefiere observar la información registro por registro, la opción "Forms" de Paradox en una manera sencilla de hacerlo. Una forma estándar de toda la pantalla se genera automáticamente cuando se crea una nueva tabla (archivo de datos). Además se dispone de editor de pantallas para construir hasta quince formas por tabla de acuerdo a las necesidades específicas del usuario de una manera fácil. Paradox, proporciona la facilidad de crear formas multi-tablas, esto es, se pueden diseñar vistas que contengan información de diferentes tablas.

Asimismo, Borland ha adoptado varias características de otros productos: Paradox 3.5 integra varias vistas tabuladas cruzadas o transversales del manejador de bases de datos de archivos fijos Reflex y agrega los recursos gráficos de presentación de la hoja de cálculo Quatro. (La tabulación cruzada es una manera de resumir o compendiar datos que le permite presentar información en un formato similar al de una hoja de cálculo donde las categorías principales se presentan verticalmente y la información resumida o totalizada en los renglones asociados).

El poder de Paradox reside en la trivialidad con la que se pueden hacer consultas y manipulación de los datos contenidos en las tablas, utilizando el método "query by example".

Las preguntas son llamadas "queries". Se pueden hacer queries de una o más tablas juntas en un sólo query. En un query se pueden definir:

- cuáles tablas contienen la información
- cuáles campos se quieren desplegar
- cuáles registros se quieren seleccionar
- que cálculos se quieren ejecutar

Además, de que en un query se pueden ejecutar operaciones tales como:

- insertar nuevos registros en la tabla
- borrar registros de la tabla
- cambiar valores en campos
- encontrar registros en una tabla

Los resultados del query son usualmente desplegados en una tabla temporal llamada "answer".

Paradox proporciona un diseñador de reportes que permite: formatear, ordenar, integrar y presentar la información contenida en las tablas en tres formas :

- Reportes tabulares. La información está dispuesta en columnas y renglones.
- Reportes de forma libre. Los campos pueden estar dispuestos en cualquier lugar de la forma.
- Etiquetas.

Una de las desventajas de Paradox es la reconstrucción de índices debido principalmente a la naturaleza del programa. A diferencia de algunos manejadores de bases de datos que almacenan registros en el orden en que se hayan capturado y luego establecen archivos de índices para hacer posible su recuperación en un orden especificado, Paradox almacena los datos físicos en el orden en que se indexan.

Algunas operaciones de Paradox requieren mucho espacio en memoria y el programa no siempre se recuperará con seguridad. En caso de que suceda un "overflow" en la memoria, se puede provocar la pérdida de enlaces importantes de múltiples tablas.

I.4.2. Dbase IV 1.1

Para Ashton-Tate sacar la versión 1.1 se ha convertido en una cuestión de honor: la versión original de dBase IV fue prácticamente inutilizable, por lo que la versión 1.1 tenía que ser un producto estable, libre de errores; así como una solución a los problemas producidos por las nuevas características en la versión 1.0. Habían pasado casi tres años y medio

desde la aparición de Dbase III Plus, y los otros fabricantes de bases de datos no estaban inactivos.

Afortunadamente, dBase IV, versión 1.1, es un producto mucho más útil que dBase III Plus, y un reemplazo estable y confiable para la versión 1.0. Pero el mundo de base de datos en la PC ha cambiado enormemente, y la versión 1.1 debe competir con la versión 2.5 de FoxPro de Fox Software, la versión 3.5 de Paradox de Borland y los nuevos productos de Nantucket y Microrim. ¿ Qué es lo que tienen estos productos que no ofrece dBase IV? ¿Qué opciones están presentes en dBase IV 1.1 que son mejoras a las encontradas en la versión 1.0 y dBase III Plus? ¿Y cómo se compara dBase IV con ellas?

UNA MIRADA AL PASADO

Cuando evaluamos dBase IV por primera vez, encontramos que la espera habia sido larga y cansada, pero el producto finalmente representaba nada menos que una revisión completa de este programa inmensamente popular.

Y vaya que fue una revisión completa. Aparte del lenguaje subyacente y la subestructura de archivo, queda poco de dBase III Plus en la versión 1.0 de dBase IV. Para empezar, el Assistant, una interface basada en menús, ha sido reemplazada por el Control Center. El Control Center permite mostrar y manipular cualquier tipo de base de datos: archivos, vistas, modelos,

reportes, etiquetas e incluso programas. El propósito del Control Center fue eliminar la dependencia de la línea de comandos de dBase, que los programadores amaban y los usuarios odiaban. El Control Center ofreció, servicios para simplificar la administración de archivos de dBase para los usuarios. Este incluyó un entorno de DOS que era tan bueno como cualquiera en el mercado en ese tiempo. Ashton-Tate esperaba que los programadores utilizaran al Control Center como un frente para sus aplicaciones.

El Control Center no fue la única mejora de la interfaz, dBase IV 1.0 produjo un verdadero generador de etiquetas y reportes, con interfaces WYSIWYG para diseñar reportes como aparecen, campos calculados, sumarios, cajas, líneas y apoyo para fuentes de impresora. Además, la versión 1.0 ofreció un servicio de Query-By-Example (QBE). Esta permitía a los usuarios buscar entradas seleccionando los campos que deseaban ver, estableciendo condiciones de selección con simples expresiones y fórmulas, y extrayendo las líneas correspondientes, todo esto con pocos pulsos de teclas. Aunque dBase IV no era el único DBMS con QBE (Paradox ha tenido un QBE desde mediados de los 80), fue una alternativa innovadora a usar los comandos de SEEK, FIND o LOCATE.

Para los programadores, la versión 1.0 ofreció una implementación casi de SQL como la que se encuentra bajo DB2, el administrador de base de datos relacionales para *mainframes* de IBM. Lamentablemente, el módulo de SQL de esa versión resultó ser completamente inestable. La versión 1.0 también ofreció un lenguaje de plantillas que los programadores podían

usar para generar programas, etiquetas, reportes y modelos de pantalla. Las facilidades de ventanas con menús horizontales y despegables se introdujeron con la versión 1.0, así como las funciones definidas por el usuario (UDF). Una vez más los UDFs no fueron tan innovadores para los amantes de las bases de datos, FoxBase ya los tenía, pero estos les permiten crear sus propias funciones y extensiones al lenguaje de dBase.

dBase IV 1.0 también ofreció arreglos bidimensionales, así como comandos para copiar entradas a arreglos y generar nuevas entradas de ellos. Otras características incluyeron los servicios automáticos de multiusuarios con bloqueo de entradas y archivos, y repetición automática. La versión 1.0 incluso ofreció la posibilidad de proveer actualizaciones de pantalla en tiempo real, cuando un usuario modifica una entrada mostrada por otro usuario. Esta fue una adición importante de dBase porque había estado disponible en Paradox por dos años. Además, Applications Generator, un compilador y apoyo expandido para procedimientos, estaban disponibles para ayudar a los programadores en el lenguaje de dBase.

Más raro, entonces y hoy, es el apoyo de impresoras de dBase IV. La versión 1.0 es capaz de apoyar hasta cuatro impresoras a la vez, con cinco fuentes apoyadas por manejador de impresoras, y la habilidad de cambiar entre los manejadores sin necesidad de reinstalar el producto. Además, la versión 1.0 introdujó los índices múltiples: un archivo de disco conteniendo hasta 47 índices claves separados, puede abrirse automáticamente cuando el archivo de base de datos correspondiente se abre. Para los usuarios y programadores de dBase, esto significó que un sólo comando USE podía

abrir un archivo de base de datos y su archivo de índices múltiple, y que las 47 claves se mantuvieran a medida que los cambios se hacían en el archivo de base de datos. Estas características, procesos de transacciones, y nuevos comandos de funciones como SCAN, LOOKUP y CALCULATE, mostraron que dBase IV fue un avance importante para los usuarios de este producto.

Sin embargo, como se dijo en aquel entonces: "Las características de dBase IV, que parecen ser revolucionarias para los usuarios de dBase maduros, son simples copias de las ofrecidas por los productos competidores". Además, la versión 1.0 estaba llena de errores. Su requisitos de memoria le impedieron ejecutar hasta las aplicaciones más modestas, y algo tan simple como un Ctrl-Brk podía trabarla completamente.

Por lo tanto, legiones de programadores en el lenguaje de dBase enfocaron su atención en FoxBase y Clipper, y los usuarios y otros programadores de base de datos acudieron a Paradox. Por casi dos años, estos competidores hicieron avances importantes en el mercado de las bases de datos de la PC, principalmente a costa de Ashton-Tate. Con FoxBase (ahora FoxPro) y Paradox, Fox Software y Borland casi duplicaron su porción del mercado.

CONCLUSIONES

Si posee la versión 1.0 de dBase IV, puede aprovechar la posiblidad de adquirir gratis la versión 1.1. Los usuarios de dBase III Plus pueden mejorar a la versión 1.1. por US$175 o la Developers Edition por US$475. (La Developers Edition viene con el módulo de tiempo de ejecución y de distribución ilimitada, libre de derechos de autor, el código original para el lenguaje, y tres licencias de LAN. Su precio de lista es de US$1.295). Si está comprometido con los productos Ashton-Tate, las mejoras son de gran valor.

Si tiene que usar el lenguaje dBase, pero no una implementación particular, es mejor que elija otro. Aunque dBase IV ha mejorado mucho, FoxPro 2.5 de Software da un rendimiento mejor, una excelente implementación del lenguaje dBase y una interfaz de usuario. No se arrepentirá un sólo minuto que pase con él.

Si encontró el QBE de dBase atractivo y quiere considerar un enfoque orientado a la administración de base de datos debe considerar seriamente a Paradox. Aunque la interfaz de usuario no es tan buena como la de Fox Pro, encontrará que las operaciones de base de datos son más intuitivas y que el sistema hará por su cuenta gran parte del trabajo.

Aunque dBase IV, versión 1.1 no supera a Fox Pro y a Paradox, la larga pesadilla de Ashton-Tate ya ha terminado. La compañía ha reestablecido su papel dominante en el mercado de base de datos de la PC, y sus

empleados parecen comprometidos a recuperar el lugar que dBase ocupó en su tiempo. La resurrección de dBase significa más competencia, y eso promete a forzar a los competidores a producir mejores productos de base de datos para todo el mundo.

dBase IV, versión 1.1 Ashton-Tate 20101 Hamilton Ave., Torrance, CA 90509, E.U.A. (123) 329-9989.

Precio de lista:

Standard Edition,

US$795; Developers Edition US$1.295; LAN Pack para cinco usuarios, US$995.

Requiere:

640K de RAM con 450K disponible en tiempo de ejecución, disco duro con por lo menos 3,5MB libres, DOS 2.1 o posterior.

Comentario:

La versión 1.1 de dBase IV es una gran mejora sobre la poca exitosa versión 1.0. Las características de Control Center y QBE funcionan mejor pero la interfaz SQL se pudiera integrar mucho mejor. Un servicio de caché de disco integrado se supone que mejore el rendimiento y lo hace hasta cierto punto, pero Fox Pro es mucho más rápido.

I.4.3 FoxPro Versión 2.5

CARACTERISTICAS GENERALES DEL PRODUCTO

FoxPro de Microsoft puede ser descrito como una herramienta que conjunta una serie de "habilidades". Por ejemplo se puede ver como : un manejador de bases de datos interactivo; un interprete de comandos del lenguaje Xbase; un ambiente de desarrollo de aplicaciones integrado; o como un compilador. FoxPro :

- Es un manejador de Bases de Datos
- Crea y maneja tablas de datos.
- Crea y maneja índices para ordenar tablas de datos.
- Crea ligas en tablas relacionales.
- Ordena tabla de datos.
- Localiza datos.
- Optimiza la recuperación de datos (Rushmore).
- Realiza querys con tablas múltiples.

Lenguaje

En FoxPro se programa con un lenguaje procedural Xbase y consiste de comandos, funciones y sentencias que efectuan un amplio rango de operaciones.

Interface

Al igual que FoxPro de Microsoft puede ser usado como un manejador de bases de datos, la mejor manera de utilizarlo es como un ambiente de desarrollo de aplicaciones integrado.

Cuando se inicia FoxPro, éste despliega dos cosas: la barra de menú del sistema y la ventana de comandos. Si se conoce un lenguaje de programación y se teclea razonablemente bien, teclear en la ventana de comandos ofrece una rapidez mayor que a través de la navegación en el menú del sistema. Debido a que la ventana de comandos es un editor, se puede facilmente reusar y revisar comandos previamente introducidos.

Muchas de las acciones que se seleccionan a través del menú del sistema, en realidad generan código de FoxPro. Este código es pasado al interprete el cual invoca la acción apropiada. Cuando este proceso ocurre, se puede observar en la ventana de comandos el o los comandos procesados.

Los Comandos de Foxpro tiene dos beneficios adicionales: Después de que la interfase de FoxPro envía los comandos al interprete, se puede entrar a la ventana de comandos (que en realidad es un editor de texto), copiar bloques de línea de código y pegarlos en los archivos propios del usuario. Por lo tanto se puede usar el sistema de menúes para aprender los comandos de FoxPro y minimizar los errores de sintaxis.

Algunas opciones del menú disparan las acciones internas de FoxPro, mientras otras activan programas que son suministrados por el paquete de FoxPro pero que son externas a él. Los programas externos incluyen FoxApp, FoxDoc y los generadores de código.

El generador de programas de FoxPro es usado para crear código fuente de pantallas, menúes y tablas de referencias cruzadas. El generador de programas suministra el código fuente de FoxPro de modo que puede ser modificado. Sus nombres son especificados mediante variables de sistema, así que los programas alternativos pueden ser llamados automáticamente.

Compilador

Al igual que FoxPro puede actuar como un interprete, realmente ejecuta código compilado. Una línea de comando tecleada dentro en la ventana de comandos es compilada en la memoria y luego es ejecutada. Sin embargo, una serie de comandos en un archivo de código fuente es siempre compilado en un archivo separado de código objeto antes de ser ejecutado, automáticamente o manualmente. (El código objeto no es compatible con ningún otro formato de código objeto). La compilación es aconsejable por varias razones. Un archivo compilado es pequeño y más rápido de ejecutar. Protege el diseño y operación de las aplicaciones porque no está legible como si fuera un archivo de código fuente y no puede ser fácilmente modificado.

Herramientas de desarrollo

Existen una gran cantidad de herramientas de programación proporcionadas por FoxPro. El manejador de proyectos organiza todos los archivos que pertenezcan a una aplicación. También permite crear varios formatos de archivos de aplicaciones:

1. FXP. Es la extensión que corresponde a un archivo objeto compilado desde en archivo de código fuente PRG.

2. APP. Es un "archivo" que contiene el equivalente de varios FXP.

3. EXE. Es un archivo ejecutable que contiene un APP más componentes de "runtime" de FoxPro.
4. FoxPro incluye un editor de textos que es invocado para los siguientes propósitos:

    - Manejo de ventanas de comandos
    - Visualización y edición de campos "memo"
    - Edición durante la construcción de expresiones
    - Edición de "code snippets"
    - Visualización y edición de archivos de texto
    - Creación y edición de archivos de programa

Constructor de Pantallas

Es una herramienta que permite diseñar interactivamente pantallas de entrada/salida. Permite manipular elementos de pantalla individuales. Las definiciones de pantalla son almacenadas en tablas especiales que son usadas por un generador de programas para crear el código fuente. El código fuente obtenido se puede modificar.

Constructor de Menús

Es una herramienta que permite diseñar interactivamente sistemas de menús. Opera en forma muy parecida al constructor de pantallas. En ambos casos el uso de estas herramientas es opcional, pero elimina una gran cantidad de programación de código tedioso.

Generador de Reportes

Es una herramienta de definición de reportes que tiene la ventaja de no generar código fuente.

Herramientas de "Debugging"

Es un programa que está integrado con el desarrollador de programas. Cuando se encuentra un error en tiempo de ejecución en una rutina, si el código fuente está disponible, FoxPro puede cargar el archivo fuente

automáticamente en el editor y resaltar la parte donde es probable que se encuentre el error.

FoxDoc

Es un programa de utilería que sirve para analizar código de programas. Puede crear referencias cruzadas de variables, crear un módulo llamado árbol y ejecutar otros análisis.

Trace

Es una ventana que despliega el código fuente que se está ejecutando, resalta la línea que se está ejecutando en ese momento. Se pueden usar "break poits" en cualquier línea del programa y ejecutar paso a paso el programa.

Debugg

Es una ventana en la que se pueden introducir expresiones. En el momento en que una aplicación esta corriendo, los valores de las expresiones pueden ser actualizados en tiempo real.

FOXPRO PARA AMBIENTE WINDOWS

Microsoft ha estado durante muchos años sin disponer de ninguna base de datos para su popular entorno gráfico. Sin embargo, actualmente la situación ha cambiado, ya que después de la compra de Fox Software y el lanzamiento de Access, la empresa de Bill Gates se ha metido con fuerza en el mundo de los gestores de datos para Windows. En esta sección analizamos al segundo abanderado de esta revolución: FoxPro 2.5 para Windows.

Generalmente, cuando se realiza una versión para Windows de algún programa nacido en el mundo DOS, lo normal es encontrarse ante una interface de usuario totalmente renovado. Sin embargo, en el caso de FoxPro esta renovación no ha sido tan radical, ya que la versión 2.0 para DOS se basaba en un completo sistema de menús apoyado en multiventanas y el uso de ratón. No obstante, la propia definición de Windows (interface gráfico frente a modo carácter del DOS) nos puede dar una idea de que hay bastantes cosas que han cambiado.

Lo primero que nos viene a la cabeza tras la presentación de una nueva versión de un programa, es la compatibilidad de las aplicaciones que ya tenemos creadas. Es decir, si podemos recompilar antiguos desarrollos con la nueva adquisición, todo ello sin tener que modificar el código fuente. Pues bien, con FoxPro 2.5 para Windows es posible ejecutar programas desarrollados con dBase III Plus, Foxbase Plus o dBase IV. Además, si se

desea, se pueden correr aplicaciones basadas en caracteres sin ninguna modificación.

FoxPro 2.5 está disponible para plataformas DOS, Windows, Macintosh y Unix, con la principal ventaja de que tanto los datos como el código fuente de las aplicaciones se pueden compartir entre estos entornos sin apenas cambios. Esta filosofía de trabajo supone una gran ventaja para los programadores, ya que podrán desarrollar a su antojo sin preocuparse del tipo plataforma que soportará su programa.

Es aquí donde nos encontramos con la herramienta más fascinante y sorprendente de FoxPro 2.5: el <<transportador entre plataformas>>.

Esta utilería permite transformar una antigua aplicación DOS en un nuevo desarrollo para Windows. FoxPro 2.5 añade automáticamente fuentes proporcionales, traspasa los controles a los de Windows y crea las típicas cajas de diálogo de la interface gráfica de Microsoft.

Una vez que tengamos nuestra aplicación Windows (ya sea nueva o transformada), dispondremos de las ventajas del Intercambio Dinámico de Datos (DDE) entre otras aplicaciones Windows, y de la posibilidad de almacenamiento de datos complejos mediante la técnica OLE (*Object Linking and Embedding),* que nos permitirá guardar objetos, tablas de hojas de cálculo, fotografías, sonido o imágenes de vídeo. Un doble <<click>> sobre este tipo de campo lanzará la aplicación asociada y abrirá el documento. FoxPro también soporta llamadas a las bibliotecas de

vinculación dinámica (DLL) y dispone de un *kit* opcional de conectividad que permite el acceso a la interface de programación de aplicaciones del *Open Database Connectivity* (ODBC).
No obstante, todas las presentaciones de FoxPro 2.5 quedan claramente explicadas en la documentación, que aunque se encuentra en inglés incluye un gran número de ejemplos.

Simplifición del trabajo

Para simplificar el trabajo tanto al programador como al usuario final, FoxPro 2.5 dispone de un nutrido grupo de accesorios que consiguen automatizar la construcción de la interface, efectuar interrogaciones *(query)* a ficheros, obtener informes *(reports)*, así como coordinar y componer una aplicación FoxPro. también contamos con otras herramientas como una calculadora, un curioso rompecabezas para los ratos de ocio y un sistema de administración de ficheros.

Todos los usuarios -incluso los que no tengan conocimiento de programación- pueden alcanzar una alta productividad gracias a la generación visual de pantallas e informes. Con el generador de pantallas de FoxPro 2.5, podremos personalizar la interface y crear formatos de pantalla acordes a nuestros gustos: incluiremos objetos como campos de datos, botones interactivos, cajas de verificación, listas de *scroll*... El usuario selecciona y posiciona los objetos a su gusto; tras ello el generador de pantallas establece el código fuente automáticamente.

Algo semejante podemos hacer con el generador de aplicaciones: sólo hay que crear la estructura de la base de datos (campos, tipo, tamaño...) y elegir la opción apropiada para que automáticamente se forme una aplicación que nos permita emplear (rellenar, borrar, buscar, modificar...) la base de datos creada.

Para la interrogación de ficheros se emplea el método RQBE *(Relational Query By Example)* con el cual no es necesario usar ningún comando ni conocer a priori el nombre del campo; algo ciertamente diferente (más fácil) a la forma de realizar una interrogación SQL de toda la vida. Además, los registros encontrados en una base de datos tras efectuar la interrogación RQBE pueden ser relacionados (por eso lo de *Relational)* en un *Report Writer.*

Herramientas mágicas

El generador de informes incluye una barra de herramientas que permite acceder de manera rápida a útiles de diseño de informes y objetos. Ahora pueden especificarse atributos como las fuentes de letra para todos los objetos del informe. El generador de informes permite crear informes en múltiples columnas y añadir elementos del diseñador de etiquetas integrado.

Algo que gustará a la mayoría de los programadores-usuarios de FoxPro 2.5 es que pone a disposición de todos ellos funciones de trazado y de *debug* (depuración). Gracias a estas funciones, el seguimiento y corrección de errores de los programas es mucho más sencillo. Otra utilería muy práctica es el <<FoxDoc>>, una aplicación que automáticamente formatea y documenta los programas o proyectos, basado su funcionamiento en un sistema de referencias cruzadas.

FoxPro 2.5 incluye un sistema de programación por comandos SQL *(Structured Query Lenguage)*. El sistema de interrogación interactivo se basa en el comando <<Select>>, utilizado para recuperar datos provenientes de una o más bases de datos. Se emplea una técnica especial de optimización denominada <<Rushmore>>, la cual consigue reducir de un modo drástico el tiempo de ejecución de los comandos. La técnica <<Rushmore>>, que se activa automáticamente siempre que es posible, consiste en acelerar todas las aplicaciones cíclicas.

FoxPro : CARACTERÍSTICAS TÉCNICAS

DESCRIPCIÓN	Sistema gestor de bases de datos para el entorno Windows que ofrece soporte y compatibilidad con plataformas Mac, Unix y DOS, siempre que las aplicaciones estén desarrolladas en lenguaje Xbase. Por el momento no está traducida al castellano, por lo que tuvimos que evaluar la versión inglesa.

REQUISITOS	Equipo 386 con 4 Mbytes de RAM y Windows instalado. Exige 17 Mbytes de disco duro para la instalación completa. Muy recomendable el uso de ratón.
PRECIO:	$795, se puede actualizar cualquier versión DOS a Windows.
DOCUMENTACIÓN:	La documentación es completa y clara, con muchos ejemplos, aunque tiene el inconveniente de estar en inglés. el proceso de instalación dura de 10 a 15 minutos.
FABRICANTE:	Microsoft Co.

FoxPro 2.5 para DOS también ya existe

Aunque la versión evaluada de FoxPro 2.5 ha sido la de Windows, como ya hemos comentado existen versiones para otras plataformas; entre ellas podemos encontrar FoxPro 2.5 para DOS.

Desde hace unos seis años, la base de datos FoxPro ha estado disponible en versiones de 16 y 32 bits. Esta nueva revisión añade numerosas extensiones a FoxPro 2.0. En la versión extendida de 32 bits, el número de áreas de trabajo se ha aumentado hasta 255, en comparación con las 25 de la 2.0 (FoxPro para Windows también soporta 255 áreas de trabajo). Además, soporta la nueva interface de modo protegido de DOS, que corre eficientemente bajo el modo extendido 386 de Windows en una caja de compatibilidad DOS.

La versión de 32 bits de FoxPro 2.5 para DOS incluye el <<386/DOS Extender>> de Phar Lap Software. Este producto permite a usuarios y programadores superar la barrera de los 640 Kbytes del DOS, y acceder a varios Mbytes de memoria superior para aprovechar las ventajas de potencia y velocidad de los 32 bits. El <<386/DOS Extender>> es un producto líder en la tecnología extendida DOS de 32 bits, ya que permite ejecuciones de FoxPro para DOS como si de una estación de trabajo se tratase.

I.4.4 INFORMIX-SQL

Informix-SQL es un lenguaje estructurado de bases de datos (structured query language). El programa se asemeja más complicado que cualquier otro manejador de bases de datos; no se tiene la presentación de pantallas multicolor con menús descolgantes ni ventanas. El paquete es del estilo de Lotus, las pantallas de menús automatizan la mayoría de las operaciones más significativas de bases de datos, tales como la creación de tablas, definición y modificación de campos, y demás. Informix también incluye un generador de reportes y un sistema para la ejecución de archivos de definición de formas que pueden ser creados empleando una especie de pseudolenguaje. Pero el corazón de Informix es su sistema de SQL.

Se puede construir una forma en Informix no precisamente a través de mover el cursor a través de la pantalla con el ratón o con las teclas de flechas, sino más bien, escribiendo un archivo de *especificación-de-formas*, una especie de definición de pantalla acompañada con instrucciones ejecutables. Este archivo comprende cinco partes : una sección para definición de bases de datos, el cual identifica la base de datos sobre la cual operará la forma; una sección de pantalla, la cual define el formato de la pantalla; una sección de tablas, la cual identifica las tablas que la forma va a accesar; una sección de atributos, la cual describe cada campo desplegado en la forma; y una sección de instrucciones opcional, la cual define las operaciones a ser ejecutadas sobre los campos dentro de la forma. Colocando simplemente, los nombres de las tablas y de la base de datos, le dice al sistema que mostrar, la sección de pantalla indica donde

mostrarlos, la sección de atributos le dice como mostrarlos, y finalmente, la sección de instrucciones le indica al sistema que hacer antes, durante ó después de mostrar la información.

Desafortunadamente, el sistema de formas de Informix sólo puede operar sobre los campos desplegados en la forma. Otro problema es la facilidad con que uno puede generar errores dentro de Informix. En la mayoría de los casos, estos aparecen en la última línea de la pantalla, anunciados por un beep. Si uno no baja la vista rápidamente, los mensajes desaparecen antes de que uno pueda leerlos.

Informix incluye una serie de utilerías adicionales. BCHECK es un programa que verifica la integridad de los índices; si este encuentra una discrepancia entre un archivo de datos y sus índices, este permite recrear el archivo de índices. DBLINK y DBLOAD son altamente útiles para compartir información entre Informix y el mundo externo como podría ser Lotus 1-2-3, DBase o archivos ASCII. Con DBSCHEMA, es posible generar la secuencia de instrucciones SQL necesarias para crear una tabla o base de datos (si la tabla fue creada dentro de Informix),

La documentación de Informix esta correctamente presentada, y el software esta bien respaldado. Informix-SQL corre en una amplia variedad de plataformas; si se esta interesado en la portabilidad, Informix es un DBMS que conviene examinar. Informix vende una gran cantidad de add-ons que amplian la capacidad de Informix-SQL.

I.5 DISEÑO DE BASES DE DATOS

El diseño de bases de datos es el proceso de arreglar en una estructura organizada los campos de datos necesarios para una o más aplicaciones. Esta estructura debe contemplar las relaciones necesarias entre los campos y al mismo tiempo debe adaptarse a las restricciones del sistema de manejo de bases de datos que se este utilizando. Existen dos partes en el proceso: el de diseño lógico y el de diseño físico.

- El diseño lógico es un ejercicio independiente de la implementación que se lleva a cabo en los campos y las relaciones para una o más aplicaciones.

- El diseño físico depende de la implementación, y toma los resultados del diseño lógico y lo refina de acuerdo a las características del sistema manejador de bases de datos que se esta utilizando.

Existen muchas razones que hacen necesario el diseño de la base de datos. Estas incluyen la redundancia de datos, el funcionamiento de la aplicación, la independencia de datos, así como la seguridad y facilidad de programación. Todos estos factores son importantes en el ambiente del procesamiento de datos, y pueden verse afectados cuando el diseño no es el apropiado.

La mayoría de los involucrados en el diseño de bases de datos concuerdan en que existen dos fases separadas en este proceso: 1) el diseño de una

estructura lógica que pueda ser procesada por el manejador de la base de datos (DBMS) y que describa el punto de vista del usuario; y, 2) la selección de la estructura física, tal como los métodos de acceso.

Novak define cuatro componentes básicos que son necesarios para lograr una metodología de diseño de bases de datos:

- Un proceso de diseño estructurado que consiste en una serie de pasos donde se escoge una sola alternativa de las presentadas.

- Técnicas de diseño para realizar la enumeración requerida y criterios de evaluación para seleccionar alternativas en cada paso.

- Requerimientos de información de entrada para el proceso de diseño como un todo y para cada paso del proceso de diseño.

- Un mecanismo descriptivo para representar las entradas de información y los resultados en cada paso del diseño.

I.5.1. ELEMENTOS DEL PROCESO DE DISEÑO

Las principales entradas y resultados del proceso de diseño son los siguientes (ver figura 1.32) :

Entradas :

- Requerimientos de información general
- Requerimientos de procesamiento
- Especificaciones de DBMS
- Configuración de hardware / sistema operativo
- Especificaciones del programa de aplicación

Resultados :

- Estructura lógica de la base de datos (vista del usuario)
- Estructura de Almacenamiento (diseño físico)

Los requerimientos de información general representan las descripciones de varios usuarios de la organización para la cual los datos son reunidos, los objetivos de la base de datos, y las vistas de los usuarios de los cuales los datos deben ser colectados y almacenados en la base de datos. Estos requerimientos son considerados como independientes de los procesos porque no están atados a ningún sistema de manejo de datos específico o aplicación. El diseño de bases de datos en estos requerimientos es considerado ventajoso a largo plazo para las bases de datos que deben ser adaptables a los requerimientos cambiantes del procesamiento.

El procesamiento de requerimientos consiste en tres partes :

- Los datos específicos requeridos para cada aplicación
- El volumen de datos y su crecimiento esperado
- La frecuencia de procesamiento en términos del número de veces en que cada aplicación debe correr por unidad de tiempo.

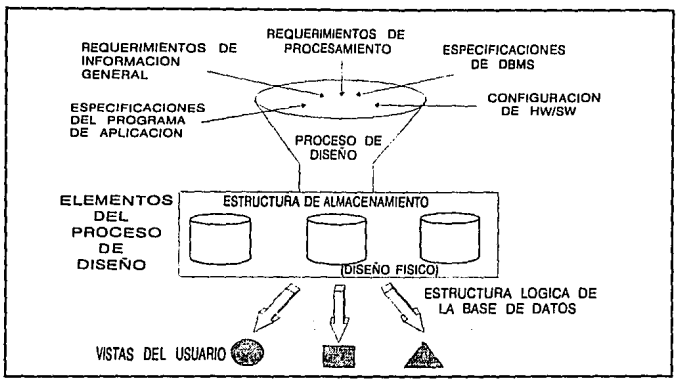


Figura 1.32 Elementos del Proceso de Diseño

Las restricciones y la capacidad de desempeño también influyen en el diseño de la base de datos. Las restricciones pueden ser tiempos de respuesta, recuperación en casos de falla, o datos específicos necesarios para requerimientos de seguridad o integridad.

Los medios que se usan para evaluar el funcionamiento de la estructura final pueden incluir los costos de actualización, almacenamiento y reorganización además de los requerimientos de respuesta.

I.5.2 ANALISIS ESTRUCTURADO PARA EL DISEÑO DE SISTEMAS

El Análisis Estructurado es un nuevo tipo de especificación funcional: la especificación estructurada. El análisis tradicional comparativamente con el análisis estructurado puede resumirse como :

Narrativo	vs	Especificación estructurada
Detallado		Panorama general
No gráfico		Gráfico

Figura 1.33 Análisis narrativo VS Especificación estructurada

¿ Qué es es el Análisis ?

- Es el estudio de un problema
- Es el estudio de una área del negocio o aplicación
- Su producto principal es el documento de especificaciones o documento de objetivos.
- La fase de analisis incluye además:
    a) Seleccionar objetivos
    b) Documentar los objetivos de tal suerte que se pueda evaluar el grado de cumplimiento.
    c) Estimar costos, beneficios y tiempos.
    d) Obtener aprobación de los involucrados.

Tareas del Analista

- Seleccionar objetivos optimos.
- Producir documentación detallada de los objetivos, de tal manera que después de la implementación pueda ser evaluado el grado de cumplimiento.
- Estimar costos, beneficios, duración del proyecto, etc.
- Obtener aprobación, en cada uno de estos puntos, de las partes afectadas.

Características del Análisis

- Los analistas antes fueron programadores.
- Esta tendencia esta cambiando.

ANALISTAS	PROGRAMADORES
No es fácil	Es razonablemente fácil
Relaciones complicadas y conflictivas	Relaciones interpersonales reducidas
No es definitivo	Fácil determinar si está bien o está mal
No es satisfactorio	Es satisfactoria

Figura 1.34 El perfil del Analista VS perfil del Programador

Involucración del Usuario

- Ningún sistema tiene éxito sin la participación del usuario.
- Debemos aprovechar su experiencia en el negocio.
- El analista es responsable de involucrar, entrenar y asesorar al usuario.

Especificaciones

- El documento de objetivos es el puente que usa el analista para comunicar al usuario con el grupo de desarrollo.

Estimación

- Al analista le solicitan estimaciones sobre todo; duración del proyecto, carga de CPU, etc., (ver figura 1.35).
- Problemas al estimar:
    a) Nadie es experto en estimaciones
    b) Nunca guardamos datos históricos
    c) Las estimaciones estan sujetas a decisiones gerenciales y financieras.

La naturaleza defensiva del Análisis

- La mayor preocupación en el análisis no es la de tener éxito, sino la de evitar el fracaso.

- Las herramientas del "análisis estructurado" proveen los medios para minimizar la probabilidad de un error critico en la fase de análisis.

La problemática del Análisis

Problemas del Análisis :

- El excesivo mantenimiento de las aplicaciones se debe a errores en el diseño.
- La frecuencia de las fallas de un sistema se debe a errores en la codificación y pruebas.
- Un error en el análisis se traduce en un incremento exagerado de los costos y en ocasiones no se obtiene el resultado deseado.
- Desgraciadamente los errores en el análisis no pueden solucionarse con mayor inversión o más tiempo, es más, no hay herramientas disponibles para evitarlos.

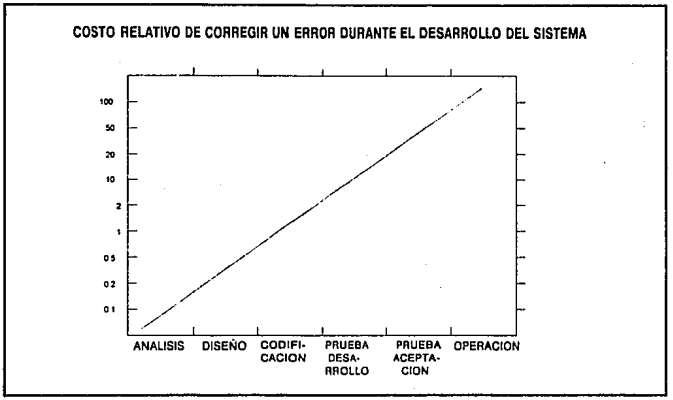


Figura 1.35 Costo de los errores durante la fase de diseño

Problemas de Comunicación :

- Dificultad para describir un proceso.
- Métodos no apropiados (narrativo) "una imagen dice más que mil palabras".
- La falta de un lenguaje común entre el usuario y el analista.
- La falta de un modelo, a priori, del sistema.

Otros problemas del Análisis :

- Es difícil para el analista aprender acerca del negocio para comprender los requerimientos desde la perspectiva del usuario.

- La comunidad de usuarios aún no conoce suficientemente el área de Procesamiento de Datos (DP) como para saber que es factible y que no lo es.
- El analista puede perderse facilmente con los detalles (del negocio y técnicos).
- Es difícil, para el usuario, comprender el documento de especificaciones del sistema.
- Aún con las mejores herramientas algunos problemas son inevitables.
- No hay forma de mostrar al usuario un modelo tangible del nuevo sistema.
- Los narrativos son muy vagos, confusos y extensos.
- Los diagramas de flujo perjudican más de lo que ayudan.
- No existe una forma sistemática de llevar registro de las preferencias del usuario.

La naturaleza cambiante de los Requerimientos

- Los requerimientos cambian debido a que el negocio cambia o a que el usuario entiende mejor el sistema.
- La gerencia de sistemas tiende a querer congelar los requerimientos y dejar para después los cambios:
  - Requiere gran esfuerzo modificar las especificaciones.
  - Prefiere trabajar con objetivos estables.
- El efecto de diferir los cambios puede incrementar los costos hasta el doble.
- Debemos construir un documento de objetivos que sea fácil de mantener.

La falta de herramienta

- Evaluar la calidad de un programa es fácil.
- Evaluar la calidad del diseño es menos fácil.
- Evaluar la calidad del análisis resulta casi imposible.

Problema del Documento de Objetivos

- En el enfoque tradicional:
  I. Es excesivamente redundante.
  II. Es excesivamente largo
  III. Es excesivamente físico.
  IV. Es tedioso.
- Debemos particionar el documento en "mini-especificaciones".

Asignación de Trabajo

- Resulta difícil dividir el trabajo entre los miembros del equipo de análisis.
- Se requiere particionar.

Política

- Un nuevo sistema puede alterar la distribución de poder y autonomía.
- No hay soluciones simples.

- En la medida en que las tareas del analista y lo que se espera de su trabajo estén claramente especificados, se puede esperar un menor impacto por cuestiones políticas.
- El análisis estructurado proporciona un enfoque más formal y preciso.

La relación Analista-Usuario

¿ Qué es un Usuario ?

- Hay tres tipos.
  - El operador del sistema.
  - El usuario responsable (gerencia baja)
  - El propietario (gerencia media)

¿ Qué es un Analista ?

- Es el puente entre usuarios y el equipo de implementación.
- Debe estar familiarizado con la tecnología.
- Debe manejar conceptos del area del negocio.
- Debe poder comunicar dichos conceptos.

División de responsabilidad entre Analista/Usuario

- El usuario debe saber: ***Qué se debe hacer.***
- El analista debe saber: ***Cúal es la mejor forma de hacerlo.***

¿ Qué es el Analista Estructurado ?

Nuevas metas para el Analista :

- Los productos del análisis deben ser facilmente mantenibles.
- Los problemas de tamaño se deben enfrentar con un método de particionar.
- Debemos usar gráficos siempre que sea posible.
- Debemos diferenciar entre consideraciones lógicas y físicas.
- Debemos construir un modelo lógico del sistema para facilitar la comprensión del usuario.

Herramientas del Análisis Estructurado

- Diagrama de flujo de datos (DFD)
- Diccionario de datos (DD)
- Tablas de decisión, arboles de decisión, diagramas de acción, etc.
- Normalización.

Análisis Estructurado

Definición :

El análisis estructurado hace uso de las herramientas mencionadas anteriormente para construir un nuevo tipo de documento de objetivos; las especificaciones estructuradas. Adicionalmente ayuda en:

I. Heuristicos para estimación.
II. Métodos para facilitar la transición del análisis al diseño.
III. Ayuda para las pruebas de aceptación.
IV. Técnicas de inspección.

¿ Qué no es el Análisis Estructurado ?

- Análisis costo/beneficio.
- Estudio de factibilidad.
- Control de proyectos
- Análisis de rendimiento.
- Selección de equipo
- Consideraciones personales.
- Política.

CONDUCCION DE LA FASE DE ANALISIS

El ciclo de vida de los sistemas y el impacto en el Análisis Estructurado

1. Hacer uso de herramientas graficas para mejorar la comunicación con el usuario.
2. Eliminar la redundancia del documento de objetivos.
3. Eliminar los textos narrativos del documento de objetivos, reemplazandolos con un equivalente más formal.
4. Eliminar la información física del documento de objetivos para que sea un documento eminentemente lógico.

Procesos del Análisis Estructurado

Consta de siete estudios:

a) Estudio del sistema físico actual.
b) Derivación del modelo lógico actual.
c) Derivación del modelo lógico nuevo.
d) Modelo (s) físico nuevo (s) tentativo (s)
e) Evaluación de alternativas.
f) Selección de un modelo
g) Especificación estructurada.

Estudio del sistema físico actual

- Determinar el contexto del estudio.
- Identificar usuarios afectados.
- Entrevistas con usuarios.
- Diagrama de flujo de datos.
- Recolección de tipos de datos, archivos, formas, etc.
- Inspecciones con el usuario para validar.
- Publicación de la documentación resultante.

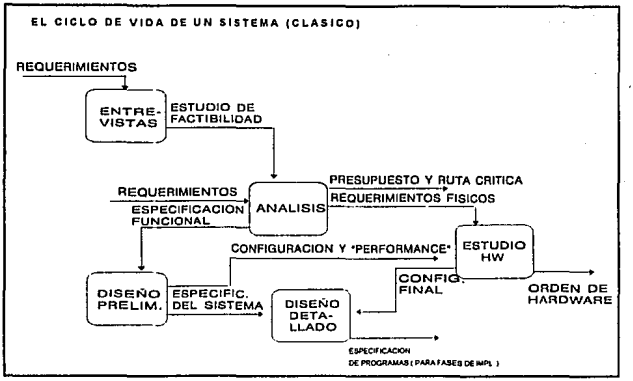


Figura 1.36 El ciclo de vida de un sistema clásico

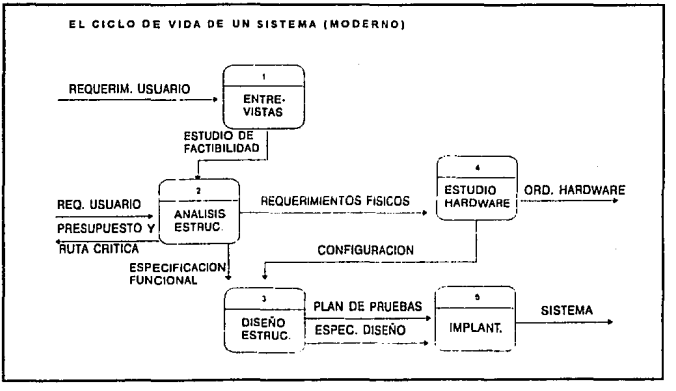


Figura 1.37 El ciclo de vida de un sistema modemo

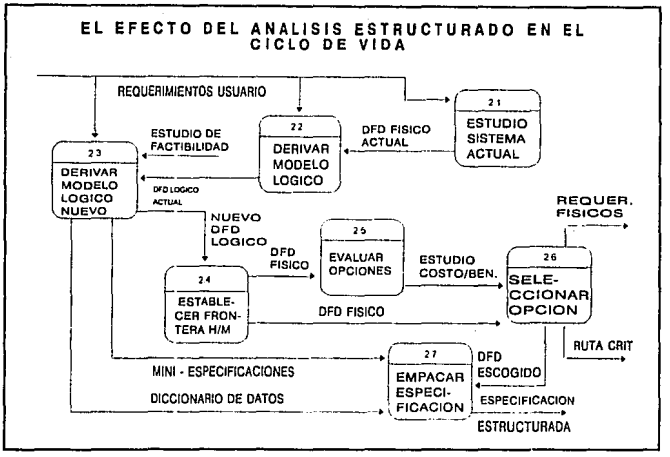


Figura 1.38 Efecto del Análisis Estructurado en el ciclo de vida

Derivando un modelo lógico equivalente

- Consiste en "limpiar" el modelo físico.
- Sustituir nombre de personas, departamentos, numeros de formas, etc. por los conceptos que representan.
- Decir qué y no cómo.
- Dibujar el DFD lógico actual
- Validar con el usuario.

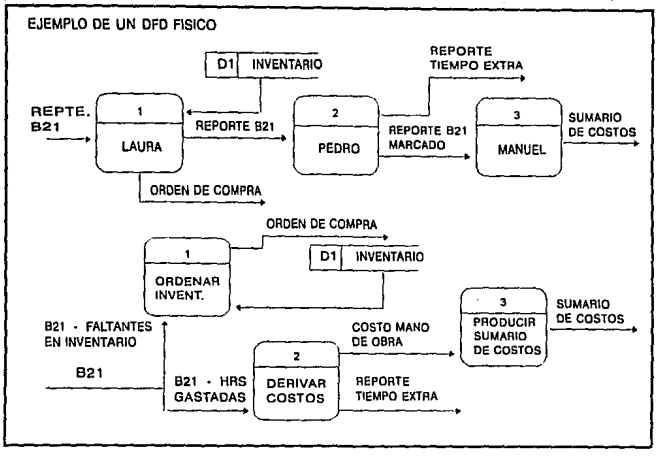


Figura 1.39 Ejemplo de un DFD Físico

Derivando el modelo lógico nuevo

- Incorporar los cambios propuestos por el estudio de factibilidad y describir el nuevo ambiente.
- Debemos producir un modelo en papel del nuevo sistema.
- El modelo se construye usando DFD(s) que muestran la partición y las interfaces. El diccionario de datos y las descripciones de transformación (mini-especificaciones) que domentan el interior de los procesos.
- Decir qué y no cómo.

Modelo(s) físico nuevo(s) tentativo(s)

- Establecer frontera hombre/máquina.
- Introducir consideraciones físicas al modelo lógico nuevo.
- ¿ Qué debemos automatizar?
- El resultado es el modelo "físico" nuevo.
- Se recomienda proponer varias alternativas.

Cuantificar las opciones

- Estudio costo/beneficio.
- Estimación del tiempo de entrega.
- Riesgos, costos de entrenamiento, operación, conversión, imagen, prestigio, etc.

Empacar las especificaciones

- Pulir el documento.

Características de la Especificación Estructurada

Consiste de:

- Diagramas de flujo de datos.
- Diccionario de datos.
- Descripciones de transformación.

Cualidades:

- Es gráfica.
- Es particionada
- Es rigurosa
- Es mantenible
- Es interativa
- Es lógica y no física.
- Es precisa, concisa y legible.

Efectos políticos del Análisis Estructurado

Efecto en el Ciclo de Vida del Proyecto

- Se requiere dedicar más tiempo a la fase de análisis.
- La gerencia es suspicaz con este aspecto.
- Enfatizar los riesgos de un análisis incorrecto.

Efecto en la relación Analista-Usuario

- Propicia una mayor participación del usuario.
- Podemos encontrar resistencia.
- Considerar que:

  - Los usuarios conocen mejor el negocio.
  - Al usuario no le gusta oir tecnicismos.
  - Los usuarios sólo requieren un conocimiento mínimo del análisis estructurado.

División del esfuerzo

- Provee una manera efectiva de particionar el trabajo durante la fase de análisis.

HERRAMIENTAS PARA EL ANALISIS ESTRUCTURADO

Una situación de ejemplo

1. La empresa "Comp-Sultores, S.A." es una compañia mediana que ofrece servicios de consultoría y entrenamiento a empresas impartiendo cursos y seminarios en varias ciudades.

2. Las personas interesadas se inscriben por correo o por teléfono. Cada inscripción se confirma a través de una carta en donde se incluye la Factura del Participante.

3. Los pagos se envian por correo. Cada pago deberá corresponder con una Factura para así registrar el crédito en las Cuentas por Cobrar.

4. Existe un procedimiento para que las personas puedan efectuar cancelaciones en caso de requerirlo.

5. Cuando una persona ha tomado un curso con la compañia, su nombre es agregado a una base de datos para así enviarle información periódica de nuevos ofrecimientos.

6. Además de que la base de datos es usada como herramienta de ventas, tiene otros usos para consultas como :

- ¿ Cuándo es el siguiente curso de análisis y diseño de sistemas ?
- ¿ Qué otras personas de mi organización han asistido al seminario de programación estructurada? ¿Cuándo lo hicieron?
- ¿ Qué instructor esta impartiendo el Workshop de FoxPro en Monterrey?

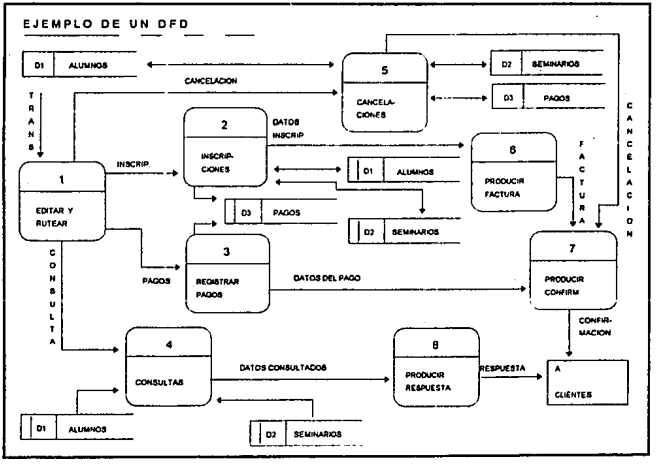
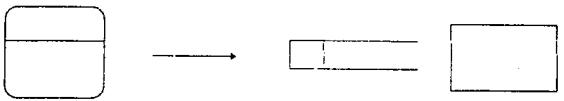


Figura 1.40 Ejemplo de un DFD

Convenciones para los DFD(s)

- El DFD muestra flujo de datos no de control.
- Usa 4 simbolos:



Una ventaja importante del DFD

- Cuando un DFD tiene errores, estos resaltan de manera contundente.

¿ Qué logramos con un DFD?

- El DFD presenta un esquema comprensible del sistema.
- Sirve como modelo de una situación real.
- Ayuda a particionar un sistema

Las herramientas del Analista

- Al ir construyendo los diagramas de flujo de datos, asignamos nombres a los flujos de datos, archivos y procesos.

- En cuanto tratamos de ver más a detalle cada uno de esos elementos nos haremos preguntas como: que es exactamente una orden? que es lo que hace el proceso de "alta a nuevos clientes"?.
- Para manejar este tipo de preguntas se emplean otras herramientas como son:
  - Diccionario de datos
  - Mini-especificaciones.

DIAGRAMAS DE FLUJO DE DATOS

¿Qué es un Diagrama de Flujo de Datos?

- Un diagrama de flujo de datos (DFD) es una representación de un sistema usando una red para identificar sus componentes y las interfaces entre estos.

- A los DFD(s) también se les conoce como: gráfica de flujo de datos o diagrama de burbujas.

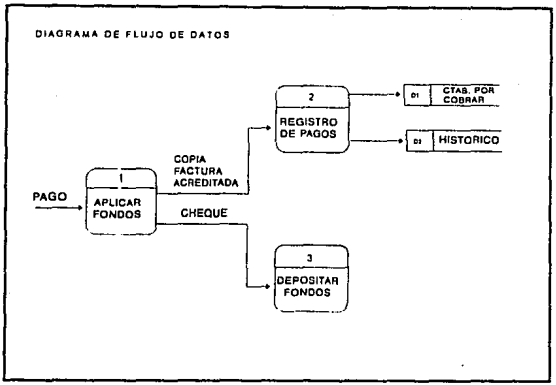


Figura 1.41 Diagrama de Flujo de Datos

Características de un DFD. Cambio del punto de vista.

- Gráfico particionado, enfatiza el flujo de datos, minimiza el flujo de control.

- Presenta la situación desde el punto de vista de los datos y no desde el punto vista de una persona, organización o sistema.

- La ventaja de este enfoque es que los datos ven el panorama global y una persona solo ve una porción.

Convenciones para el DFD.

Elementos de un DFD :

- Flujo de datos → *Representado po un vector.*
- Procesos → *Representado por una burbuja.*
- Archivos → *Representados por rectangulos abiertos.*
- Entidades → *Representados por cajas.*

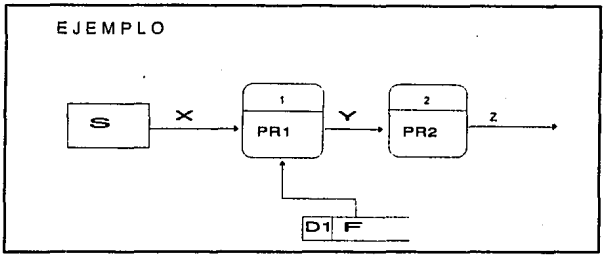


Figura 1.42 Ejemplo de un DFD

FLUJO DE DATOS

- Es un paquete de información que viaja junta y que sirve a un mismo propósito.
- Una buena regla de dedo para poder determinar si algo es un flujo de datos es intentar ponerle un nombre.
- No usar homónimos o sinóminos al nombrar un flujo de datos.

- El nombre del flujo de datos no unicamente debe representar los datos sino también lo que sabemos acerca de ellos.
- Los flujos de datos que van o salen de un archivo no requieren nombre.
- No se debe mostrar flujo de control o transferencia de control.

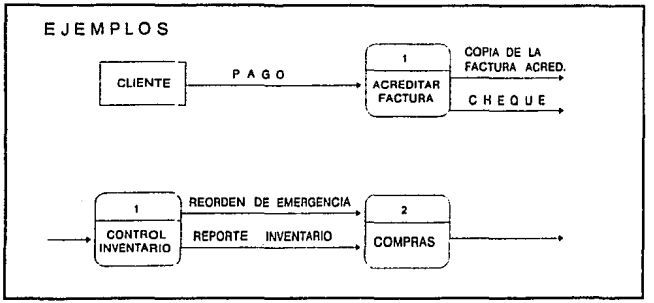


Figura 1.43 Ejemplo de un DFD de pedidos y control de inventarios

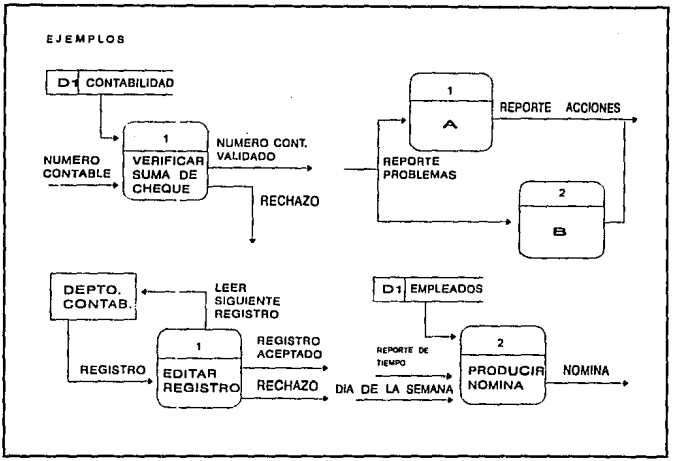


Figura 1.44 Ejemplo de un sistema de nómina en un DFD

EL PROCESO

- Los procesos muestran algún trabajo hecho con los datos.
- Los nombres de los procesos deben hacerse en términos de sus entradas y salidas.
- Un proceso es la transformación de los flujos de datos de entrada a flujos de datos de salida.

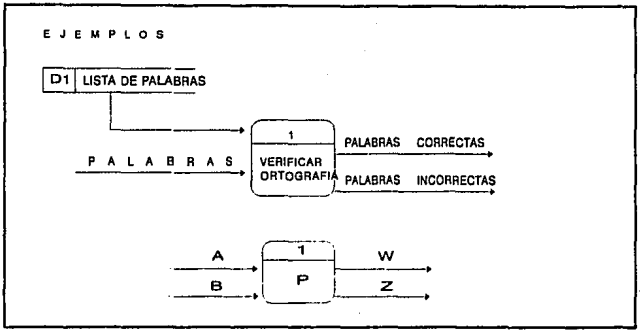


Figura 1.45 Los procesos expresan la transformación de datos en términos de sus entradas y salidas.

EL ARCHIVO

- Un archivo es un depósito temporal de datos.
- Usar nombres descriptivos del contenido del archivo. El detalle se podría ver en el diccionario de datos.
- El sentido del flujo de datos indicaria si un archivo es de INPUT (Flujo Saliendo del Archivo). De OUTPUT (Flujo entrando al archivo) o de UPDATE (Flujo en dos sentidos).

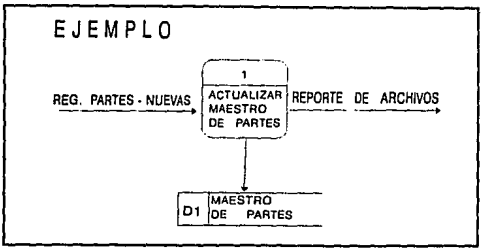


Figura 1.46 DFD mostrando un archivo de salida

LA ENTIDAD EXTERNA O FUENTE/DESTINO

- Es una persona u organización fuera del contexto del sistema que es el originador o receptor neto de los datos.
- Una persona u organización dentro del contexto del sistema estaría representado por un proceso.
- Al hablar de un sistema nos referimos a un conjunto de procesos (manuales y/o automáticos) que se usan para efectuar el fin deseado.

" APRENDIZ DE MUCHO. MAESTRO DE NADA"

- Hasta este momento podemos observar como en el "análisis estructurado" se difieren cierto tipo de consideraciones, especialmente aquellas que tiene que ver con aspectos "físicos" del sistema.

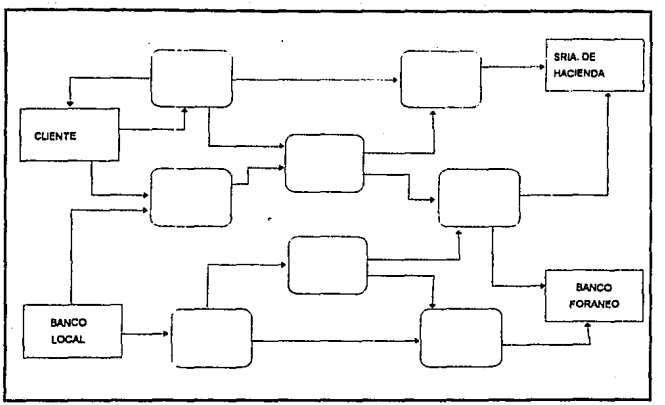


Figura 1.47a Las entidades externas son : Cliente, Banco Local, Banco Foraneo y Sria. de Hacienda

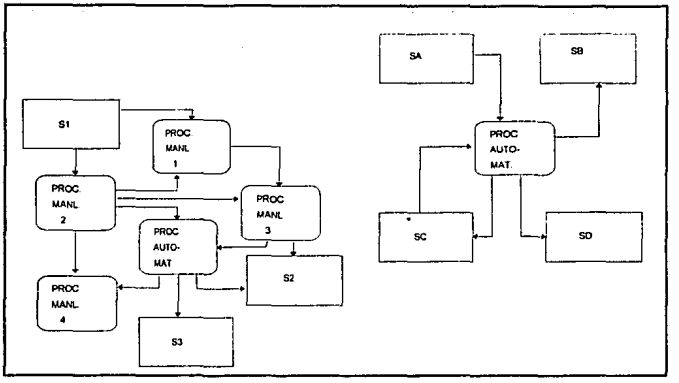


Figura 1.47b Las entidades externas mostradas son : S1, S2, S3, SA, SB, SC y SD.

- "Si queremos aprender algo, no tratemos de aprender todo".
- En el análisis estructurado la regla es diferir cualquier cosa que no sea indispensable.

GUIAS PARA DIBUJAR DFD

Identificar las Entradas y Salidas Netas

- Determinar el contexto del sistema.
- El contexto debe ser suficientemente grande como para incluir todo lo relevante al esfuerzo de desarrollo, pero también suficientemente pequeño para no incluir cuestiones irrelevantes.

- Buscar los flujos de datos que cruzan el contexto, estos constituirán las E/S netas (Fuente/Destino).

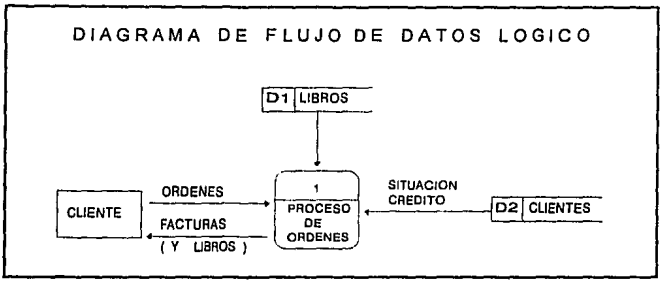


Figura 1.48 El DFD debe mostrar el contexto del sistema

Llenar el cuerpo del DFD

- Concentrarse en los flujos de datos.
- Establecer procesos donde se requiera transformar un flujo de datos.
- Averiguar si dentro de los procesos identificados hay flujos de datos, en tal caso separar ese proceso en 2, 3 o los procesos que sean necesarios.
- Cuestionarse sobre cada flujo de datos:
  - *¿ Qué se necesita para construirlo?*
  - *¿ De dónde provienen sus componentes?*
  - *¿ Qué procesos se requieren para ello?*
  - *¿ Podemos construir este flujo de datos a partir de los flujos de datos de entrada?*

- Introducir en el DFD los archivos que representen los depósitos de información que el usuario ha identificado.
- Checar si no hay flujos de datos faltantes o sobrantes o redes desconectadas y corregir.

Nombrar los Flujos de Datos

- Asignar nombres a todos los flujos de datos.
- Evitar nombres ambiguos.
- Tener cuidado de no mezclar datos que nada tienen que ver entre si.
- Si no se encuentra un nombre preciso dividir el flujo de datos.

Nombrar los Procesos

- Asegurarse de que todos los flujos de datos ya tienen nombre.
- Tratar de nombrar los procesos usando un verbo y un sustantivo.
- Evitar ambiguedades o nombres que abarquen parcialmente el proceso.
- Si se obtienen nombres ambiguos reparticionar.

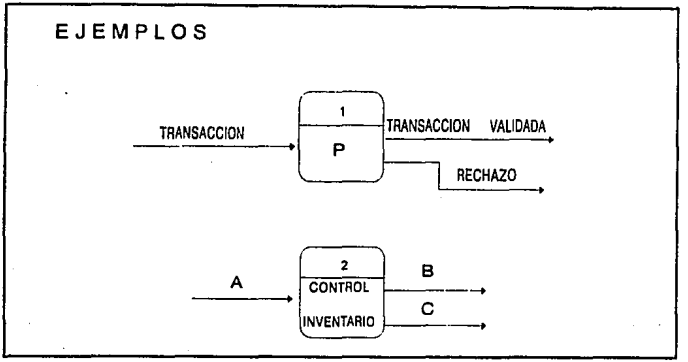


Figura 1.49 Nombrar los Procesos con un verbo y un sustantivo.

Omitir detalles de Manejo de Errores

- Cuando se detecten rutas de error se recomienda dejarlas indicadas y trabajar en ellas posteriormente.
- Si el error no requiere descartar algún proceso ya efectuado, ignorarlo.
- Si el error requiere "*backout*" entonces no ignorarlo.

Mostrar el Flujo de Datos y no de Control

- Verificar que en todos los flujos de datos se este pasando información, si no es así, eliminarlos.
- Verificar que la información en cada flujo de datos este usandose en el proceso al que llega, de no ser así, eliminarla.

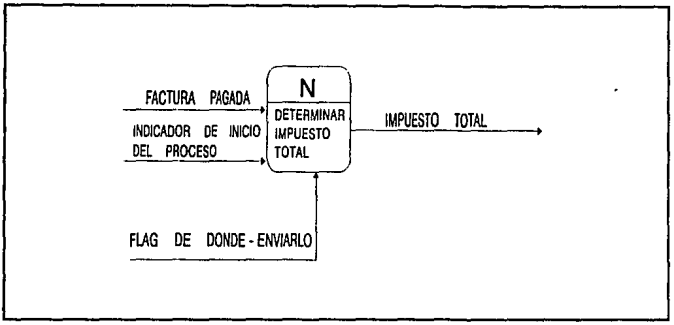


Figura 1.50 Mostrar Flujo de dato y no de Control

Empezar otra vez

- Es imposible que el DFD este correcto la primera vez.
- Si el primer DFD dibujado resulta ser el final, seguramente habrá errores.
- Ejecutar todo este proceso de manera iterativa.
- Este preparado para sustituir el primer DFD por otra (s) versión (s) mejorada (s).
- Este proceso iterativo redundará en grandes beneficios en las fases posteriores del ciclo de vida del sistema.

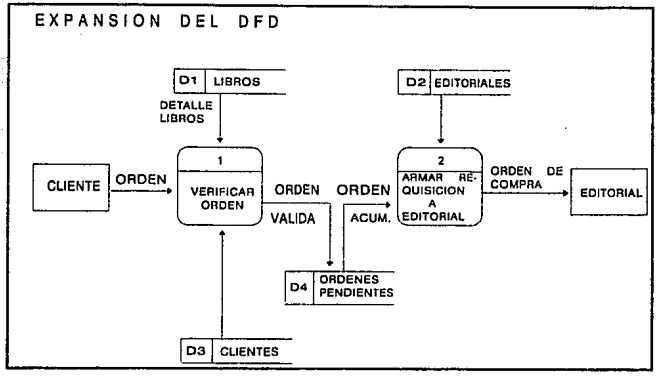


Figura 1.51 Expansión del Diagrama de Flujo de Datos

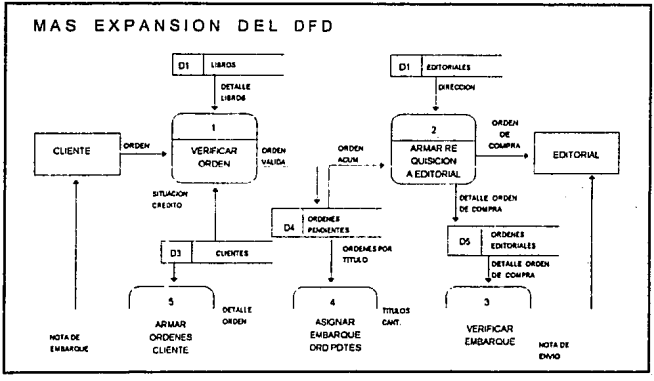


Figura 1.52a A través del análisis se ve necesario expander los DFDs.

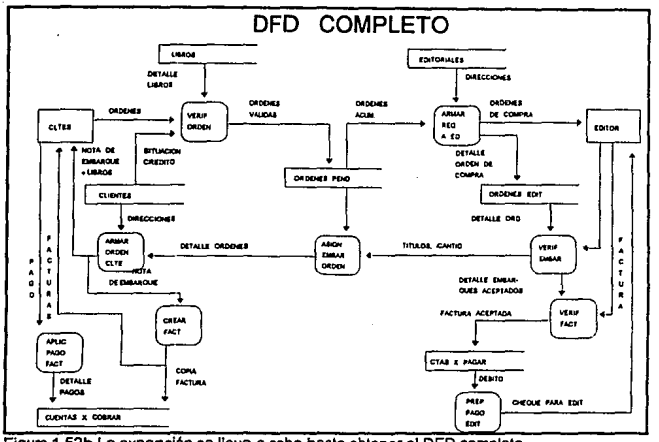


Figura 1.52b La expansión se lleva a cabo hasta obtener el DFD completo.

NIVELES EN LOS DFD

Análisis Top - Down. El concepto de Niveles

- Cuando tenemos un sistema muy grande como para mostrarlo en un DFD en una sola hoja, debemos particionarlo en sub-sistemas.

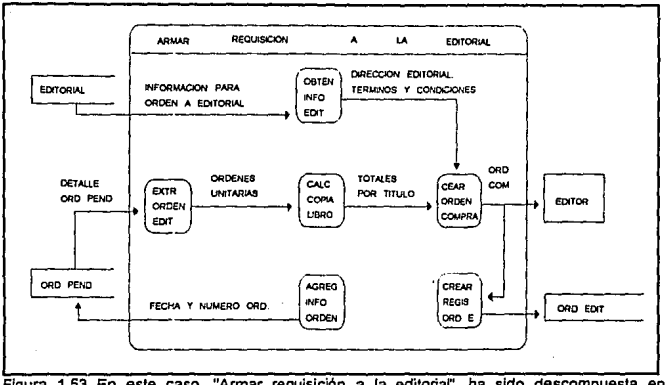


Figura 1.53 En este caso, "Armar requisición a la editorial", ha sido descompuesta en subsistemas.

Elementos de un DFD con Niveles

- El conjunto de niveles de un DFD esta hecho de un nivel superior, uno inferior y uno intermedio.
- Al nivel superior se le llama "diagrama de contexto"
- Al nivel inferior se le llama "funciones primitivas".

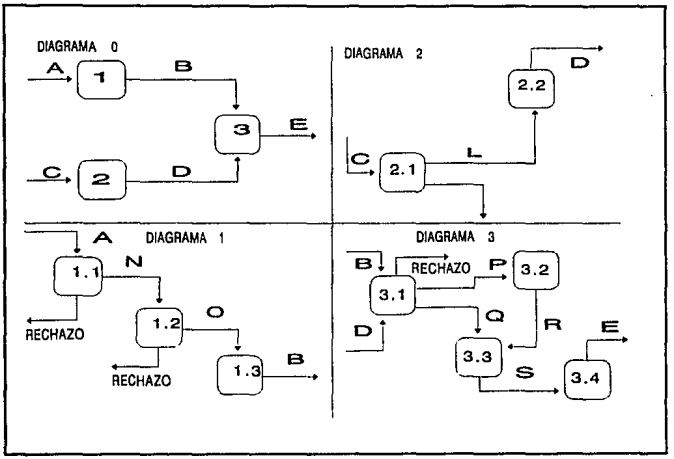


Figura 1.54 El concepto de niveles en los DFDs.

El Diagrama de Contexto

- El diagrama de contexto es una versión sin particionar de todo el sistema, sólo muestra entradas y salidas netas.
- Sirve para delinear el contexto de estudio.
- Se debe validar:

- Si es posible transformar las entradas en las salidas mostradas en el diagrama.

- Si las entradas son suficientes para producir las salidas.

Funciones Primitivas

- Son procesos que ya no se van a descomponer en otros sub-sistemas.

- Determinar cuales son las funciones primitivas y como se relacionan entre sí, es el objetivo de los diagramas de flujo de datos.

El Nivel Intermedio

- Esta formado por todos los niveles (pueden llegar a ser 8 o 9) entre el diagrama de contexto y las primitivas.

- En un sistema muy simple podría no haber nivel intermedio.

Convenciones para los Niveles de un DFD

- La descomposición de DFD en sub-sistemas va creando una jerarquía que se encuentra en el diagrama de contexto.

- El diagrama de contexto es el "padre" de los diagramas de primer nivel: estos a su vez serán los "padres" de los diagramas del siguiente nivel: etc.

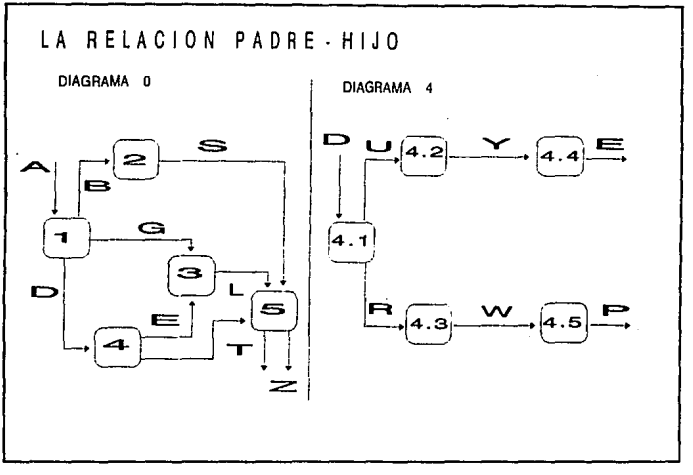


Figura 1.55 La relación Padre-Hijo en el concepto de niveles.

- Habrá tantos diagramas de nivel inmediato inferior, como procesos existan en su diagrama "padre".

Balanceo

- Todos los flujos de datos entrando en un diagrama hijo deben estar representados en el padre por el mismo flujo de datos.

- Las salidas del diagrama hijo deben ser las mismas salidas que las del padre asociado.

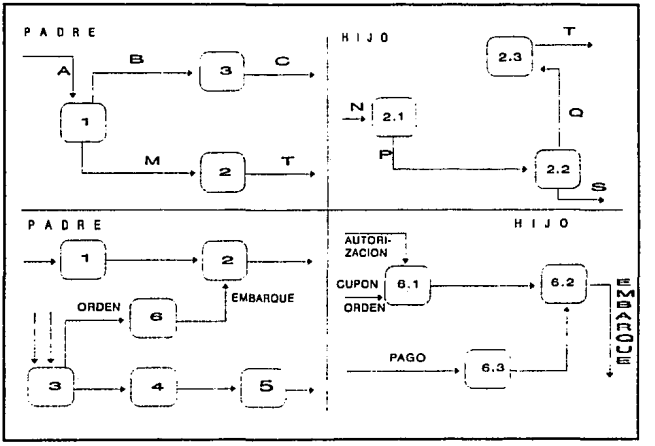


Figura 1.56 Convenciones de numeración de los DFDs.

- Es válido que mientras se efectua las descomposición top-down de los DFD (s) también se haga una descomposición top-down de los flujos de

datos, mostrando más detalle. En estos casos, para verificar el balanceo hay que auxiliarse del diccionario de datos.

Convenciones de Numeración

- La numeración de los procesos se forma concatenando el número de diagrama, más un punto, con un número local único.
- Así resulta fácil decir en que nivel estamos y quien es el padre de un proceso.
- Cuando hay muchos niveles se puede usar una notación abreviada.

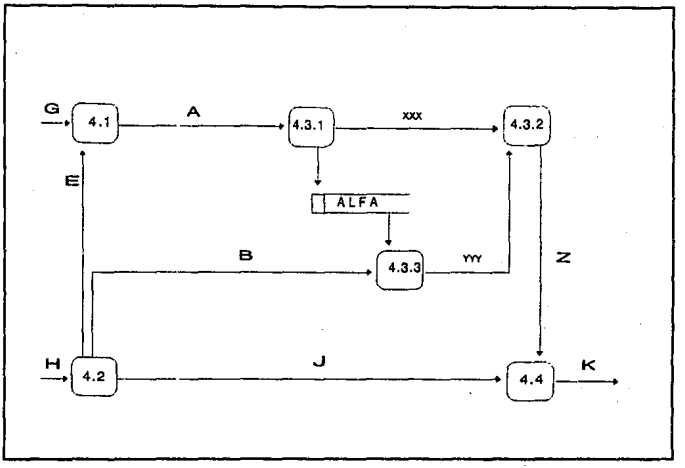


Figura 1.57 Archivos usados por procesos dependientes.

Archivos Locales

- Un archivo se muestra en un DFD en el primer nivel donde se usa como interface entre dos procesos.

- Como consecuencia, en el primer nivel donde se muestra un archivo, todas las referencias a el son mostradas.

- En el diagrama mostrado en la figura 1.57, el archivo "ALFA" no será usado más que por los procesos 4.3.1. 4.3.3 y sus dependientes.

Extensión del Particionamiento

- Al tratar de responder a la pregunta: ¿Cuánto particionar un proceso? una buene regla de dedo es un número alrededor de 7.

- La regla más importante a seguir es la de tratar de conseguir la mayor claridad posible en los DFD (s).

- Particionar para mostrar interfaces significativas.

- Entre más se particione (sin restar claridad) existirán menos diagramas lo cual facilita el mantenimiento.

Consideraciones para el Nivel inferior

- Cómo decidir donde detener el particionamiento?

- Cómo se documentarán los procesos que están en los diagramas de nivel inferior?

- ¿ Cómo se relacionan entre si los diagramas del nivel inférior?

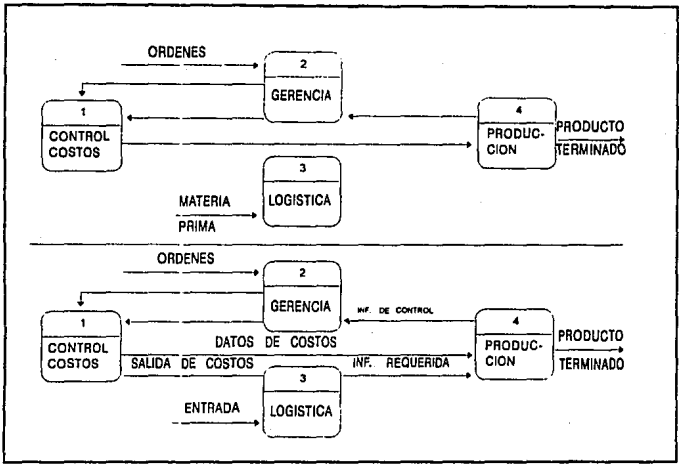


Figura 1.58 Relaciones del nivel inferior.

Determinación del Nivel Inferior

- Detenerse cuando se este en un nivel tal que los procesos puedan documentarse con una mini-especificación que no ocupe más de una hoja.

- Llegar hasta un nivel en el que los procesos tengan un solo flujo de datos de entrada y un sólo flujo de datos de salida.

- Determinarse cuando cada proceso muestre relaciones uno-a-uno o muchos-a-uno entre flujos de datos de entrada y salida.

Documentación de los Procesos

- Deberá haber una nimi-especificación por cada función (proceso) primitiva.

- Las mini-especificaciones las debemos indicar con el mismo número del proceso correspondiente.

- De esta forma no es necesario documentar los procesos de niveles superiores ya que observando a sus correspondientes hijos tendremos una equivalencia rigurosa entre estos y aquellos.

Construcción de un DFD Expandido

- Si se reemplaza cada proceso no primitivo con su nivel inmediato inferior en la red, este deberá encajar perfectamente.

- Si continuamos este proceso bottom-up de manera sistemática, estaremos construyendo un DFD expandido.

- Esto no se efectua en la práctica, pero resulta ilustrativo para entender las relaciones entre los diferentes diagramas.

Ventajas de los Niveles en los DFD(s)

- Permiten un enfoque top-down.
- No hay conectores a otras hojas como en el tradicional diagrama de flujo.
- Todos los diagramas pueden restringirse al tamaño de una hoja.

Prueba de Corrección

- ¿Qué puede estar mal en un DFD?
  - Flujos de datos faltantes
  - Flujos de datos sobrantes
  - Procesos faltantes
  - Niveles incorrectos
  - Nombres ambiguos
  - Puede contener información de control y no de flujo
  - Puede estar mal concebido
- Existen formas para detectar y corregir estos errores.

Nombres en el DFD

- Un error común es olvidar nombrar los flujos de datos.
- Asegurarse que cada flujo de datos tenga nombre.
- Asegurarse de poder definir cada nombre.
- Eliminar flujos de datos nulos.
- Probar los nombres de procesos contra las entradas y salidas.
- Probar los nombres de los procesos contra los niveles inferiores.

Errores de consistencia

- Aplicar cuidadosamente la regla de balanceo para evitar contradicciones, omisiones o procesos redundantes.

Conservación de los datos

- Verificar que no haya información que salga de un proceso y que no haya entrado en el de alguna manera.
- Un proceso debe ser capaz de construir sus salidas a partir de sus entradas.
- Puede ser un error cuando existe una entrada a un porceso y esta no es transformada en alguna salida, sino que simplemente "muere" dentro del proceso.

Problemas con Archivos

- Validar que un archivo no unicamente se este usando de salida y jamás se consulte.
- Para detectar este tipo de errores es importante usar la convención de marcar la dirección de flujo neto.

Errores conceptuales. La revisión del Análisis

- La única forma de evitar errores conceptuales es trabajando de cerca con el usuario y verificando la validez de los DFD(s).

- Recomendaciones para las revisiones en la fase de análisis:
  - Las primeras revisiones las debe hacer el usuario por su cuenta.
  - No enviar DFD(s) al usuario hasta que este familiarizado con la metodología.
  - A los nuevos usuarios mostrarles diagramas de niveles medio o bajo.
- Recomendaciones para las revisiones en la fase de análisis (cont.)
  - Completar la revisión con información "física"
  - Cuando se revise el nivel "N" de un DFD, asegurarse que el equipo de revisión tenga en sus manos el nivel "N-1".
  - Evitar el uso de tecnicismos.

Prueba de Utilidad

- A veces un DFD puede resultar correcto pero difícil de leerlo o complicado.
- Revisar:
  -Complejidad de las interfaces
  -Nombres de los procesos
  -Particionamiento apropiado

Compejidad de las Interfaces

- Entre menos flujos de datos entren y salgan de un proceso mejor.
- Considerar la posibilidad de particionar un proceso que tenga muchas interfaces.

Nombres de los Procesos

- Los nombres ambiguos generalmente indican una mala partición.
- El nombre ideal es uno que consista de un verbo seguido por un objeto concreto.
- En los niveles superiores puede ser difícil seguir esta convención.

Particionamiento Inapropiado

- El mejor particionamiento es aquel que divide en partes de tamaño similar.

Comenzando de Nuevo

El enfoque del "re-particionamiento topológico"

Cuando un diagrama de nivel inferior contiene redes desconectadas. El padre de este diagrama es un candidato para aplicarle el enfoque mencionado.

La metodología se enuncia a continuación:

1. Construir un diagrama expandido combinando los hijos del diagrama que requiere re-particionamiento.

2. Trate de dividir los procesos del diagrama expandido en subconjuntos que minimicen las interfaces entre ellos.

3. Recree el diagrama de nivel superior dibujando un proceso para cada subconjunto.

4. Recree el nivel inferior marcando las fronteras entre cada subconjunto.

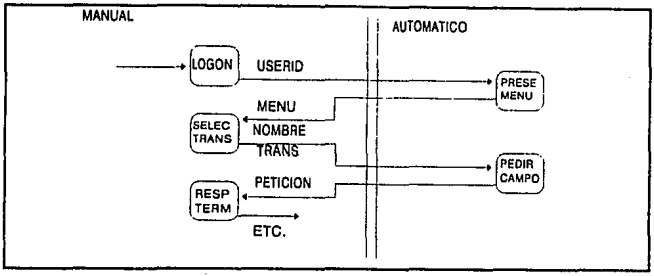
5. Renumere y renombre los diagramas resultantes.

DFD PARA LA ESPECIFICACION DEL SISTEMA

El dialogo Hombre-Máquina

- La fase de análisis no estará conlcuída hasta no tener documentada la interface Hombre-Máquina.

- Una posibilidad para documentar esta interface es usando una DFD extra.



El enfoque del Nivel Superior Integrado

- Se recomienda mostrar al sistema automatizado como un sólo proceso al que fluyen datos de procesos manuales.

- La especificación estructurada estará hecha de procesos manuales y automáticos. En ambos casos se usarán las mismas herramientas.

Documentación de los Procesos Manuales

- Debido a la metodología empleada por el análisis estructurado, los procesos manuales (manual del usuario) se documentan en la fase de análisis y no al final del desarrollo.

- Con esto se elimina la desventaja que tiene el usuario al tratar de discutir algún proceso automatizado.

Problemas Potenciales

- ¿En qué difiere un DFD de un diagrama de flujo del sistema?
- ¿No se vuelven muy complicados los diagramas de nivel superior?
- ¿Tendrán problema los usuarios para manejar los DFD(s)?
- ¿No resulta en ocasiones difícil o imposible seguir un análisis top-down?
- ¿Cuántos niveles debemos esperar?
- ¿Al ver los detalles del nivel "N", no ocurre que haya que regresar a modificar el nivel "N-1"?
- Es posible que encontremos algo en un nivel inferior que invalide todos los niveles superiores.
- ¿ No llevan demasiado tiempo las revisiones iterativas de los DFD(s)?
- ¿Es correcto tener nombres diferentes para el mismo flujo de datos?
- ¿Cómo agrupar el conjunto de DFD (s)? en que orden deberán conservarse?
- ¿Es posible mostrar flujos de cosas "fisicas" en un DFD?

EL DICCIONARIO DE DATOS EN LA FASE DE ANALISIS

Generalidades:

- Un Diccionario de Datos (DD) es un depósito de información donde se guardan datos acerca de los datos.
- Un DD incluye el conjunto de procedimientos necesarios para construirlo y mantenerlo.

- El uso de DD(s) se ha extendido a partir del crecimiento vertiginoso de la tecnología de DBMS(s).

Usos del Diccionario de Datos

- El DD es una parte integral de la especificación estructurada: en el están contenidas las definiciones rigurosas de algunos elementos de los DFD(s).
- En el DD hay definiciones de:
  - Flujos de datos.
  - Datos almacenados (archivos)
  - Componentes de los flujos de datos y/o de los archivos.

- El uso del DD no se limita a la fase de análisis sino que se emplea durante todo el ciclo de vida de los sistemas.

Correlacionando el DD con los DFD(s)

- Hay una entrada en el DD por cada flujo de datos único que aparezca en cualquier DFD.
- Hay una entrada en el DD por cada archivo usado en el conjunto de diagramas.
- Hay una entrada en el DD por cada componente de los flujos de datos y/o archivos documentados.

Consideraciones de Implementación

- Un diccionario de datos puede ser manual, automatizado o semi-automatizado.
- Un DD manual es muy limitado pero de bajo costo.
- Un DD automatizado es una herramienta indispensable en una instalación de cierto tamaño ya que es la única forma de tener el control de los datos que usa una organización.
- Los DD(s) también pueden ser integrados o no integrados.

Requerimientos para un DD

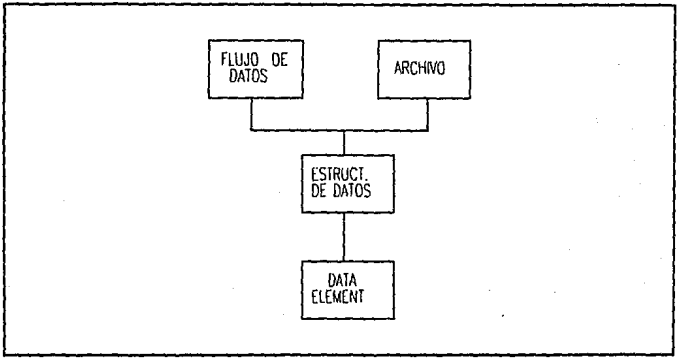
- Listados ordenados de todas las entradas o de cierta clase de entradas con toda su información o con información parcial.
- Facilidad para emitir reportes.
- Posibilidad de hacer referencias cruzadas.
- Encontrar un nombre a partir de su descripción.
- Verificación de consistencia.
- Generación de definiciones para algún lenguaje de programación.
- Extracción de entradas para el DD a partir de programas existentes.
- Facilidad de extensión del DD para que cada organización defina sus nodos de acuerdo con sus necesidades particulares.
- Seguridad de acceso y actualización.
- No deberá ser redundante.

DEFINICIONES EN UN DD

Características de una definición

- Una definición consiste de un genero y una diferencia. El genero establece una clase para lo que se esta definiendo y el conjunto de diferencias lo distingue de otros miembros del mismo genero.

Generos a definir



Data elements: son pedazos de datos que no requieren descoponerse en otros para los propósitos del sistema, por ejemplo, fecha.

Estructuras de datos: Están hechas de data elements o de otras estructuras de datos o de una mezcla de ambas:

ORDEN

Identificación-orden
    Fecha-orden
    Número-orden
Detalle-cliente
    Nombre-de-la-organización
    Persona-que-autoriza
        Nombre
        Apellido-paterno
    Teléfono
    Dirección
        Avenida
        Ciudad
        Código-postal
    Detalle-libros
        Nombre-autor.....etc., etc.

Flujos de datos y datos almacenados: Los flujos de datos son trayectorias por las que viajan las estructuras de datos. Los datos almacenados son lugares donde las estructuras de datos se guardan hasta que se necesiten.

- Los flujos de datos son estructuras de datos en movimiento: los datos almacenados son estructuras de datos en reposo.

¿ Qué información guardar en el DD?

Describiendo un Data Element :

- La información mínima requerida es su nombre y una descripción.
- Adicionalmente:
  - Se le puede asignar un Alias.
  - Las estructuras de datos deben estar relacionadas.
  - Se debe especificar el rango de valores y el significado de los valores.
  - Se debe especificar su longitud.
  - Información de edición.

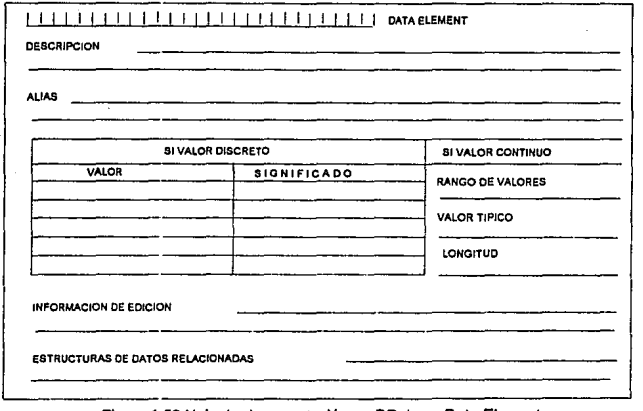


Figura 1.59 Hoja de documentación en DD de un Data Element

Describiendo las Estructuras de Datos :

- Adicionalmente a conocer los componentes de una estructura de datos se necesita información extra:

  - Estructuras opcionales
    [ ]

  - Estructuras alternativas.
    { }

  - Iteración de estructuras
    *(0 - 10)

  - Flujos de datos/archivos relacionados

Describiendo Flujo de Datos :

- Podemos ahora especificar el contenido de flujos de datos, listando los nombres de las estructuras de datos que contienen, además, es importante contar con:

  - La fuente del flujo de datos
  - El destino
  - El volumen por día o por mes

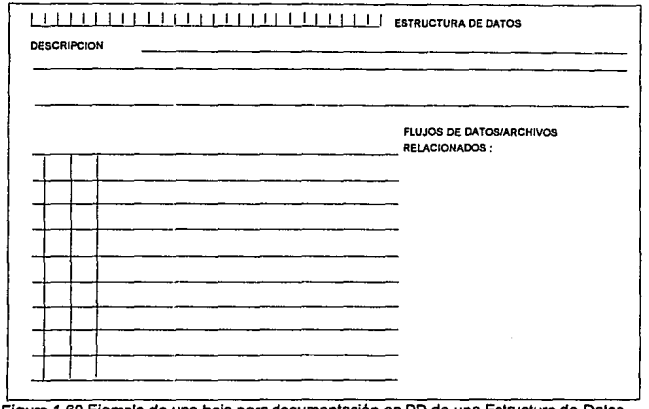


Figura 1.60 Ejemplo de una hoja para documentación en DD de una Estructura de Datos.

Describiendo los Datos Almacenados (Archivos) :

- Aqui también se describe el contenido de los archivos en términos de las estructuras de datos de que se compone, además, resulta conveniente tener:

  - Flujos de datos de actualización.

  - Flujos de datos de consulta y llave de búsqueda.

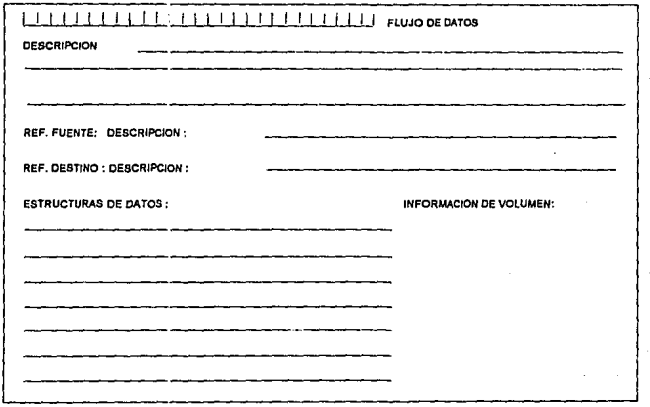


Figura 1.61 Ejemplo de una hoja para documentar Flujos de Datos.

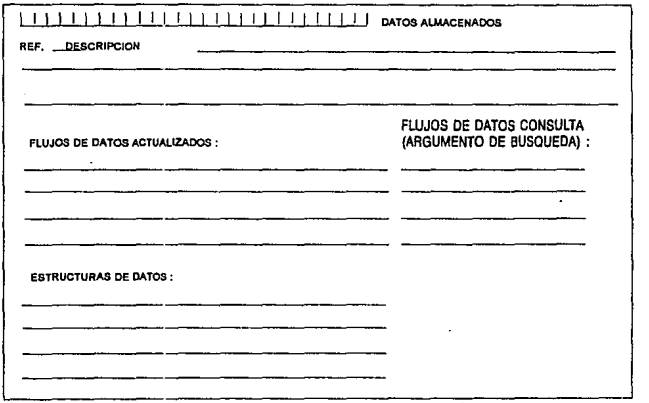


Figura 1.62 Hoja para documentar Datos Almacenados

DESCRIPCION PRIMITIVAS

Objetivos de la Especificación

- Claridad
- Precisión
- Concisa
- Completa
- Debe ser lógica y no física.

El papel de una Mini-Especificación

- Deberá haber una mini-especificación por cada función primitiva en los DFD(s).
- Cada mini-especificación deberá describir las reglas que gobiernan la transformación de los flujos de datos.
- Cada mini-especificación deberá describir la política que rige la transformación pero no el método para implementar esa política.
- Las mini-especificaciones deben establecer las políticas de transformación sin introducir ninguna clase de redundancia en la especificación estructurada.

¿ Qué es una Mini-Especificación ?

Existen varios métodos para hacerla:

- Tablas de decisión
- Arboles de decisión
- Diagramas de acción, etc.

MINI-ESPECIFICACIONES

Problemas al expresar la lógica

1. Agregar ***A*** a ***B*** a menos que ***A*** sea menor que ***B***, en cuyo caso restar ***A*** de ***B***.
2. Agregar ***A*** a ***B*** sin embargo, si ***A*** es menor que ***B***, la respuesta es la diferencia entre ***A*** y ***B***.

3 Sumar ***A*** y ***B***, pero restar ***A*** de ***B*** cuando ***A*** sea menor que ***A***.

4 El total es la suma de ***A*** y ***B*** . A pesar de la afirmación previa, en el caso de que ***B*** sea mayor que ***A*** el resultado será la diferencia entre ***B*** y ***A***.

Ambigüedad en el Y/O. Imprecisión en los Adjetivos

"Los clientes que hagan compras por más de $10,000,000 al año y tengan una buena historia de pagos, o bien, que hayan sido clientes nuestros por más de 20 años, recibirán trato prioritario ".

Más de N$ 10000 al año	y	buena historia de pagos o más de 20 años
Más de N$ 10000 al año y buena historia de pagos	o	Más de 20 años

Diagramas de Acción

Definición

- El diagrama de accion consiste de:
  - Verbos imperativos
  - Terminos definidos en el diccionario de datos
  - Palabras reservadas para expresar la lógica.

- La sintaxis esta limitada a:
  - Oraciones declarativas
  - Estructuras de decisión
  - Estructuras de iteración.

Estructuras del Diagrama de Acción :

- Secuencia
- If - Then - Else
- Do While
- Do until
- Cada estructura se caracteriza por tener una sóla entrada y una sola salida.

Vocabulario del Diagrama de Acción :

- Minimizar el uso de adjetivos y sustantivos
- Emplear verbos precisos (evitar usar verbos como: procesar, manipular, etc).
- Limitarse a:
  - Verbos
  - Objetos - del "DD" (archivos, data elements y flujos de datos)
  - Calificadores - valores de los data elements en el "DD")
  - Conjunciones (IF, WHILE, UNTIL, etc)
  - Operadores relacionales (and, or, equal, etc.)

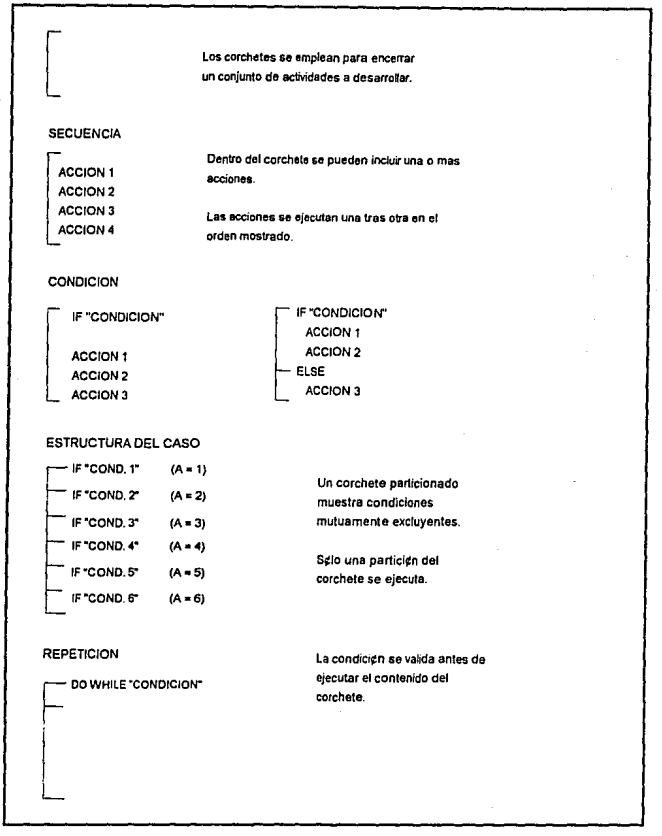
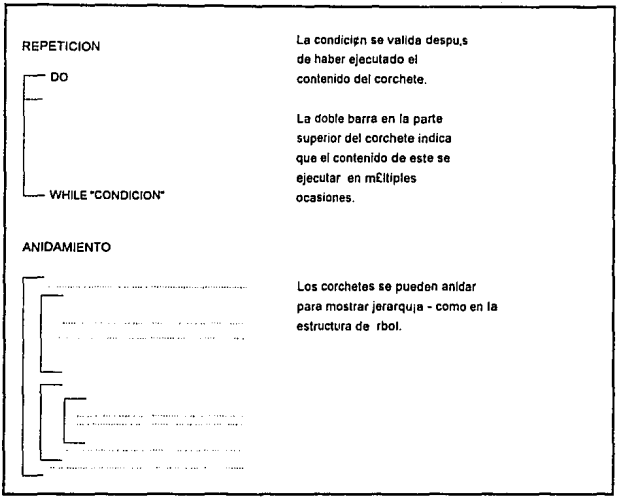


Fig. 1.63a Notación para los diagramas de acción.



Fig, 1.63b Notación para los diagramas de acción.

Ventajas de los Diagramas de Acción :

- Son rápidos y faciles de dibujar y cambiar.
- Se pueden usar para mostrar diferentes niveles de detalle.
- Son faciles de enseñar al usuario

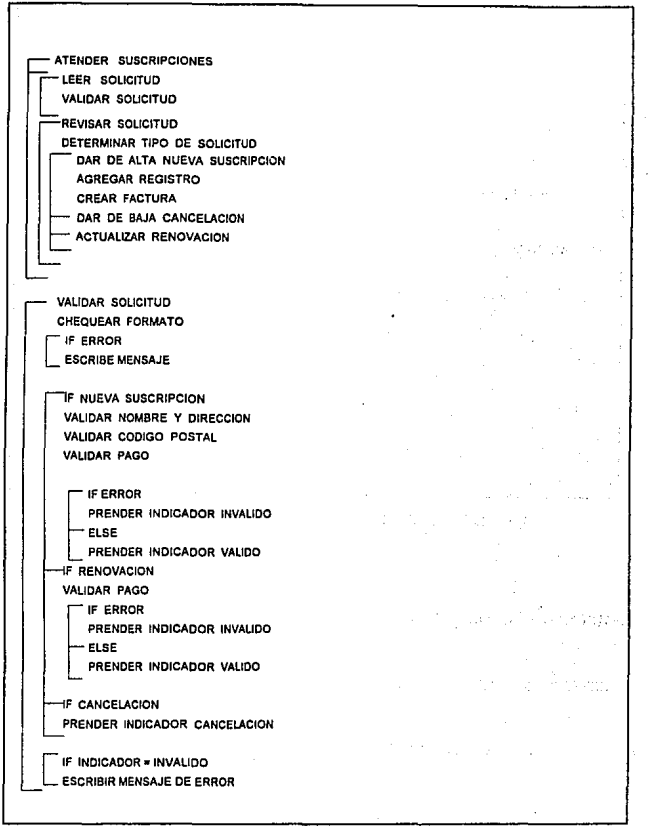


Figura 1.64a Ejemplo del uso de la notación para diagramas de acción

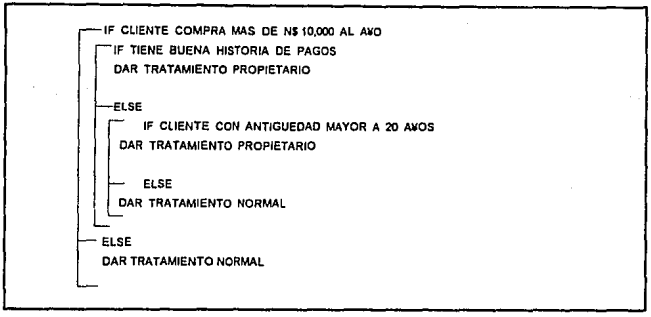
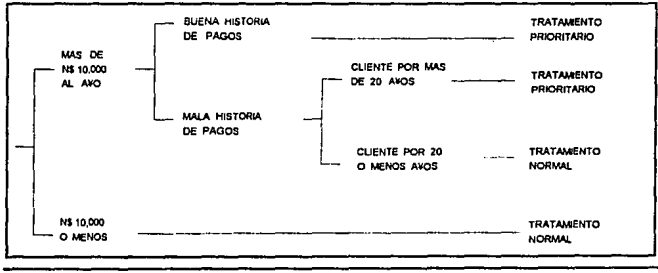


Figura 1.64b Continuación del ejemplo mostrado en la figura anterior.

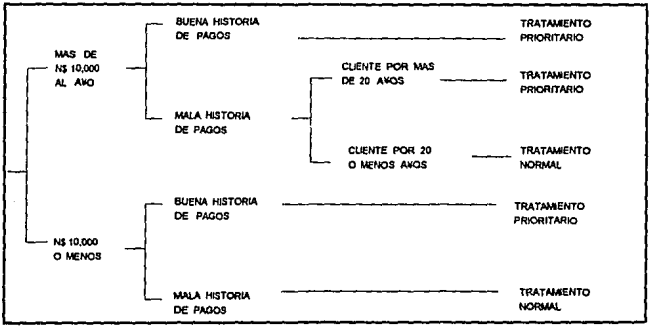
- Propician que el usuario profundice en el exámen del deseño de la lógica de los procesos.
- Fácil de implementar en un computador
- Casa/cad.

Árboles de Decisión



Replanteando la política :

" Los clientes que hagan compras por más de $10,000,000 al año y, además, tengan buena historia de pagos o hayan sido clientes nuestros por más de 20 años recibirán trato prioritario. También recibirán trato prioritario aquellos clientes que aunque compren menos de $10,000,000 al año, tengan buen historial de pagos".



Tablas de Decisión

Expresando la lógica con Tablas de Decisión :

C1: Más de 10,000,000 al año?	S	S	S	S	N	N	N	N
C2: Buen historial de pagos	S	S	N	N	S	S	N	N
C3: Cliente por más de 20 años?	S	N	S	N	S	N	S	N
A1: Tratamiento prioritario	X	X	X		X	X		
A2: Tratamiento normal				X			X	X

TABLAS DE DECISION

CONDICIONES	
ACCIONES	REGLAS

Construyendo la Matríz

1. Encontrar el número total de reglas multiplicando entre si el número de posibilidades de cada condición:

C1: más de $10,000,000	2 posibilidades
C2: historia pagos OK	X 2 posibilidades
C3: más de 20 años	X 2 posibilidades
Total	***8 posibilidades***

2. Crear la matriz con las condiciones y acciones dejando espacio para las reglas.

C1: Más de $10,000,000 al año?								
C2: Buen historial de pagos?								
C3: Cliente con más de 20 años?								
A1: Tratamiento prioritario								
A2: Tratamiento normal								

3. Tomar la última condición y alterar las posibilidades a lo largo del renglón.

C1: Más de $10,000,000 al año?								
C2: Buen historial de pagos?								
C3: Cliente con más de 20 años?	S	N	S	N	S	N	S	N
A1: Tratamiento prioritario								
A2: Tratamiento normal								

4. Observar la frecuencia de repetición del patrón y llenar la condición de arriba.

C1: Más de $10,000,000 al año?								
C2: Buen historial de pagos?								
C3: Cliente con más de 20 años?	S	N	S	N	S	N	S	N
A1: Tratamiento prioritario								
A2: Tratamiento normal								

5. Observar la frecuencia de repetición del patrón (C2) y llenar la condic de arriba.

C1: Más de $10,000,000 al año?		S	S	S	S	N	N	N	N
C2: Buen historial de pagos?		S	S	N	N	S	S	N	N
C3: Cliente con más de 20 años?		S	N	S	N	S	N	S	N
A1: Tratamiento prioritario									
A2: Tratamiento normal									

6. Procedemos a poner una "X" en cada acción que corresponda a la combinación de condiciones.

C1: Más de $10,000,000 añ año?		S	S	S	S	N	N	N	N
C2: Buen historial de pagos?		S	S	N	N	S	S	N	N
C3: Cliente con más de 20 años?		S	N	S	N	S	N	S	N
A1: Tratamiento prioritario		X	X	X		X	X		
A2: Tratamiento normal					X			X	X

7. Eliminar los casos de indiferencia (por ejemplo columnas 7 y 8)

		1	2	3	4	5	6	7	8
C1: Más de $10,000,000 al año?		S	S	S	S	N	N	N	N
C2: Buen historial de pagos?		S	S	N	N	S	S	N	N
C3: Cliente con más de 20 años?		S	N	S	N	S	N	S	N
A1: Tratamiento prioritario		X	X	X		X	X		
A2: Tratamiento noral					X			X	X

7. Eliminar los casos de indiferencia (por ejemplo columnas 7 y 8).

	1	2	3	4	5	6	
C1: Más de $10,000,000 al año?	S	S	S	S	N	N	N
C2: Buen historial de pagos?	S	S	N	N	S	S	N
C3: Cliente con más de 20 años?	S	N	S	N	S	N	S
A1: Tratamiento prioritario	X	X	X		X	X	
A2: Tratamiento normal				X			X

7. Eliminar los casos de indiferencia (por ejemplo columnas 1-2 y 5- 6)

			3	4		
C1: Más de $10,000,000 al año?	S	S	S	N	N	
C2: Buen historial de pagos?	S	N	N	S	N	
C3: Cliente por más de 20 años?	-	S	N	-	-	
A1: Tratamiento prioritario	X	X		X		
A2: Tratamiento normal			X		X	

Construyendo la Matriz- Indiferencia

1. Encontrar un par de reglas tales que:
   - La acción sea la misma
   - Los valores de las condiciones sean los mismos excepto en una y sólo una condición.
2. Reemplace cada pareja con una sola regla usando el simbolo "-" para denotar indiferencia.
3. Repetir para cualquier otra pareja de reglas que cubra las condiciones de (1).

Comparación de las Herramientas

Arboles de decisión: Se recomiendan para verificación lógica o decisiones no muy complicadas que resulten entre 10 y 15 acciones. Los arboles de decisión son útiles para presentar la lógica de una tabla de decisión al usuario.

Tablas de decisión: Son muy útiles para problemas que involucran múltiples condiciones (5 o 6). Las tablas de decisión pueden manejar cualquier número de acciones: un número grande de condiciones puede producir una tabla muy extensa.

Diagrama de acción: Es bueno cuando el problema plantea combinación de acciones con decisiones o "LOOPS".

NORMALIZACION DE LOS DATOS ALMACENADOS

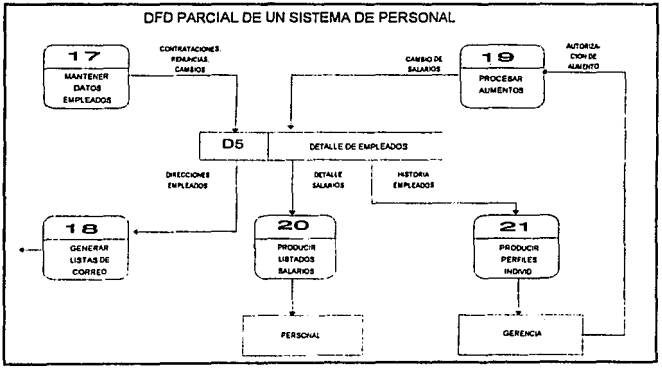


Figura 1.65 Diagrama de Flujo de Datos parcial de un sistema de personal.

Flujos de datos que entran/salen del archivo D5 (DETALLE DE EMPLEADOS, ver figura página anterior)

ESTRUCTURA DE DATOS

FLUJOS ENTRANDO A D5

Empleando-nuevo (17-D5)
- Fecha-Contratación
- Nombre
- Número-empleado
- Dirección
- Puesto
- Salario-Inicial

Renuncias (17-D5)
- Nombre
- Número-empleado

Cambio-dirección (17-D5)
- Nombre
- Número-empleado
- Dirección-anterior
- Dirección-actual

Cambio-salario (19 - D5)
- Nombre
- Número-empleado
- Salario-anterior
- Salario-actual
- Fecha-efectiva

FLUJOS SALIENDO DE D5

Dirección.empleado (D5-18)
- Nombre
- Dirección

Detalle-salario (D5-20)
- Nombre
- Número-empleado
- Salario-actual

Historia-empleado (D5-21)
- Nombre
- Número-empleado
- Fecha-contratación
- Historia-laboral
  - Puesto
  - Fecha-efectiva
- Historia-salarial
  - Salario
  - Fecha efectiva

ESTRUCTURA DE LOS DATOS ALMACENADOS EN D5

```
Data Element
  Nombre
  Número-empleado
  Dirección
  Salario-actual
  Fecha-contratación
  Historia-laboral *
    Puesto
    Fecha-efectiva
  Historia-salarial *
    Salario
    Fecha-efectiva
```

Ejemplo del contenido de D5 : Detalle de Empleados :

NOMBRE	NUM. EMPL.	DIRECCION	SAL. ACT.	FECHA CONT.
J. LOPEZ	1234	AV. 5 NO. 4	18200	011284
L. PEREZ	2365	AV. 8 NO. 3	21250	150683
T. QUIROZ	3456	AV. 7 NO. 4	8500	010185

HISTORIA LABORAL		HISTORIA SALARIAL	
PUESTO	FECHA	SALARIO	FECHA
SUPERVISOR	010886	18,200	011286
PROG. SENIOR	150485	15,250	011285
PROGRAMADOR	011284	14,500	150485
		12,750	011284
GERENTE	011185	21,250	150686
LIDER PROY.	151284	19,000	011185
		16,500	150685
OPERADOR	010784	14,000	151284
CAPTURISTA	150683	9,500	150684
		8,500	150683
MENSAJERO	010185	8,500	010186
		8,250	010185

- Relaciones
- Dominios y atributos
- Llaves
- Formas de normalización
- Dependencia funcional
- Primera forma normal
- Segunda forma normal
- Tercera forma normal

Relaciones

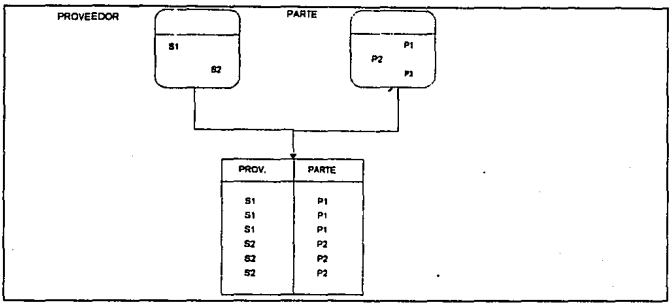
- Una relación esta formada por una serie de conjuntos a los que se les denomina dominio de la relación.

Partes

No.	DESCRIPCION	COLOR	PESO	CIUDAD
P1	DESARMADOR	AZUL	12	COLIMA
P2	RODAJA	PLATA	17	TORREON
P3	TUERCA	NEGRO	10	TEPIC
P4	LLAVE	VERDE	12	MONTERREY
P5	PINZAS	ROJO	12	PUEBLA
P6	DADO	PLATA	17	MORELIA

OTRA DEFINICION DE RELACION

Ejemplo del Producto Cartesiano de dos Dominios para producir una Relación :



Dominios y Atributos

- El dominio de una relación es la colección de todos los valores que puede tomar un elemento que interviene en esa relación, v. gr., el color de una parte.
- Un atributo representa el uso de un dominio dentro de una relación v. gr., la relación "partes" esta definida con cinco atributos (número de parte, descripción, etc.)

Llaves

- Es común que dentro de una relación exista un atributo con valores que sean unicos dentro de esa relación, por ejemplo, el número de parte en la relación "partes".

- No toda relación tendrá un único atributo que sirva como llave, pero siempre habrá una combinación de atributos que tomados en conjunto posean la caraterística de unicidad.

Preguntas que podemos hacer para ver si una llave candidata puede escogerse como llave primaria.

- Si quitamos cualquier elemento de una entidad, ¿sigue siendo única la llave?
- ¿Existe una posible situación en que la llave pudiera ser no única?
- ¿Existe alguna parte de la llave que no este definida?
- De las llaves candidatas restantes cúal usa menos dominios?

EJEMPLO DE TRES RELACIONES

PROVEEDORES

SN	NOMBRE	STATUS	CIUDAD
S1	MARQUEZ	20	VERACRUZ
S2	RODRIGUEZ	10	TAMPICO
S3	SANCHEZ	30	TAMPICO
S4	BERMUDEZ	20	VERACRUZ
S5	JIMENEZ	30	PUEBLA

PARTES

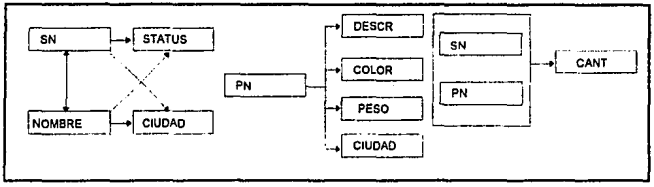
PN	DESCRIPCION	COLOR	PESO	CIUDAD
P1	PINZAS	VERDE	12	MORELIA
P2	RODAJA	AZUL	17	TEPIC
P3	LLAVE	PLATA	17	COLIMA
P4	TUERCA	PLATA	14	VERACRUZ
P5	DADO	ROJO	12	TEPIC
P6	TORNILLO	GRIS	19	VERACRUZ

EMBARQUES

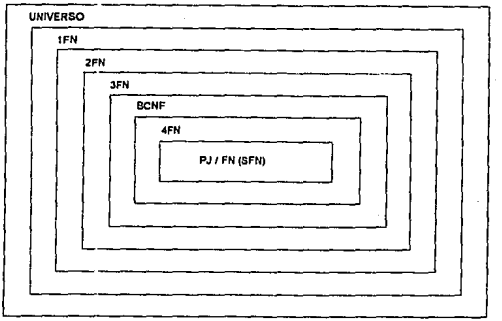
SN	PN	CANT
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P3	200
S4	P4	200
S4	P5	300
S4	P6	400

DEPENDENCIA FUNCIONAL

- Decimos que en una relación **R**, el atributo ***Y*** depende funcionalmente del atributo ***X***, si y solo si, cada valor de ***X*** en **R** tiene asociado precisamente un valor de ***Y***.



FORMAS NORMALES



Primera Forma Normal

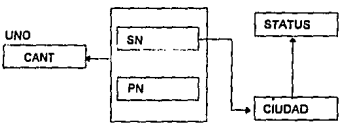
- Una relación **R** esta en su primera forma normal si y solo si todos sus atributos contienen valores atómicos.

Tomemos la relación: UNO (SN, STATUS, CIUDAD, PN, CANT)

SN	STATUS	CIUDAD	PN	CANT
S1	20	VERACRUZ	P1	300
S1	20	VERACRUZ	P2	200
S1	20	VERACRUZ	P3	400
S1	20	VERACRUZ	P4	200
S1	20	VERACRUZ	P5	100
S1	20	VERACRUZ	P6	100
S2	10	COLIMA	P1	300
S2	10	COLIMA	P2	400
S3	10	COLIMA	P3	200
S4	20	VERACRUZ	P4	200
S4	20	VERACRUZ	P5	300
S4	20	VERACRUZ	P6	400

...la relación uno esta en Primera Forma Normal.

Definimos las siguientes dependencias funcionales:



Observe que sucede cuando deseamos:

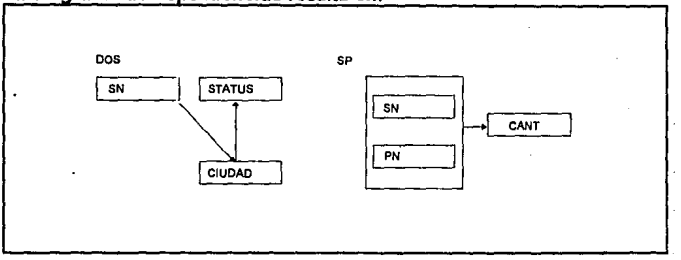
- Insertar S5
- Eliminar el embarque S3/P2
- Cambiar la ciudad de S1 de Veracruz a Tampico

La solución a los problemas anteriores es descomponer la relación "uno" en las siguientes relaciones.

DOS (SN, STATUS, CIUDAD)

SP (SN, PN, CANT)

El diagrama de dependencias resulta en:



Las tablas resultantes serán:

SP

SN	PN	CANT
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P3	200
S4	P4	200
S4	P5	300
S4	P6	400

DOS

SN	STATUS	CIUDAD
S1	20	VERACRUZ
S2	10	COLIMA
S3	10	COLIMA
S4	20	VERACRUZ
S5	30	PUEBLA

Observemos ahora que pasa al:

- Insertar S5
- Eliminar el embarque S3/P2
- Cambiar la ciudad de S1 de Veracruz a Tampico

Segunda Forma Normal

- Decimos que una relación "R" está en la Segunda Forma Normal (2fn) si y sólo si esta en Primera Forma Normal (1fn) y cada atributo que no es llave depende funcionalmente de la llave primaria completa.

Observemos que sucede ahora si:

- Queremos tener registrado que a la ciudad de Monterrey le corresponde un status 50.

- Queremos borrar la entidad para S5 en la relación "dos".

- Deseamos cambiar el status de Veracruz de 20 a 30.

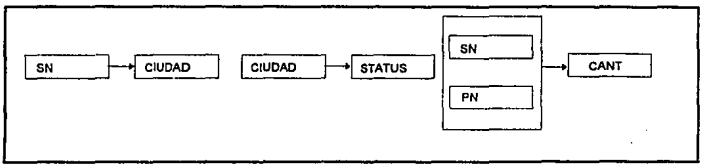
La solución a la situación precedente consiste en descomponer la relación "dos" en las siguientes relaciones:

SC (SN, CIUDAD)
SC (CIUDAD, STATUS)

y manteniendo la relación:

SP (SN, PN, CANT.)

A continuación se muestra el diagrama de dependencia funcional y las tablas para las nuevas relaciones.



SP

SN	CANT
S1	300
S1	200
S1	400
S1	200
S1	100
S1	100
S2	300
S2	400
S3	200
S4	200
S4	300
S4	400

SP

SN	CIUDAD
S1	VERACRUZ
S2	COLIMA
S3	COLIMA
S4	VERACRUZ
S5	PUEBLA

CS

CIUDAD	STATUS
VERACRUZ	20
COLIMA	20
PUEBLA	10

Observemos como estas relaciones solucionan los problemas anteriores:

- Queremos tener registrado que a la ciudad Monterrey le corresponde un status 50.
- Queremos borrar la entidad para S5 en la relación "dos".
- Deseamos cambiar el status de Veracruz de 20 a 30.

Tercera Forma Normal

- Decimos que una relación **R** está en la Tercera Forma Normal (3fn) si y sólo si esta en 2fn y cada atributo que no sea llave depende funcionalmente, en forma directa, de la llave primaria.
- Con la tercera forma normal tenemos una forma simple de representar los datos.

Tercera forma normal (usos) :

- Podemos usar la 3fn como la herramienta básica para estructurar la información de las entidades que nos interesen.
- Se puede usar la 3fn como un modo estandar de comunicarnos con el diseñador físico, ya sea que usemos o no bases de datos. Podemos mostrar el contenido de las entidades a los usuarios interesados en una forma simple.

USO DE MODELOS DE SISTEMAS

Introducción

- Hasta este momento se han visto las herramientas del análisis estructurado.
- Ahora se verá el "cómo" del análisis estructurado, es decir, como trabajar con todas las herramientas para completar la fase de análisis.
- Una vez desarrrollada la descripción física del sistema actual:
  - Transformar esa descripción en un modelo lógico equivalente.
  - Transformar el modelo lógico actual en el nuevo modelo lógico.
  - Trasformar el nuevo modelo lógico en el nuevo modelo físico.

Características de un DFD Lógico y uno Físico

- ***Un DFD físico:***
  - Dependiente de la implementación

- Indica el "cómo"
- Describe la forma de llevar a cabo una política.

- ***Un DFD lógico:***
  - Independiente de la implementación
  - Indica el "qué"
  - Representa una política del usuario.

- ***Caracterisitcas físicas de un DFD:***
  - Nombres de departamentos
  - Nombres de localizaciones
  - Nombres de organizaciones
  - Nombres de personas, archivos, procedimientos, dispositivos, etc.

- No hay una frontera muy clara entre lo físico y lo lógico.

- Los atributos físicos se introducen en la fase inicial del análisis y son eliminados sistemáticamente en fases posteriores.

Cambios al Sistema

- En las dos primeras sub-fases del análisis solamente se considera al sistema acutual. Este sistema eventualmente tendrá que cambiarse y funcionar con un nuevo juego de reglas.
- Esto es lo que distinguirá al nuevo sistema del actual.

Derivando el Documento de Objetivos

- El nuevo DFD junto con el DD y las miniespecificaciones constituyen el modelo lógico del nuevo sistema. En las subsecuentes fases del análisis (y de la implementación) lo que se hace es transformar ese modelo lógico en uno físico.
- El documento de objetivos además de incluir el modelo lógico, deberá incluir alguna información física: frontera hombre/máquina, características operaciones, objetivos de rendimiento.
- Si hay más de una forma de satisfacer los objetivos, se debe dejar la selección al equipo de implementación.

DERIVANDO EL MODELO LOGICO DEL SISTEMA ACTUAL

Introducción

- La primera transformación implica remover toda la información física en el modelo actual.
- Cualquier aspecto físico en el modelo actual caerá en alguna de 4 categorías: ES de tipo político, de trámite, histórica o dependiente de una herramienta.
- La derivación del modelo lógico requiere:
    1. Construir un DFD expandido
    2. Usar normalización
    3. Trabajar bottom-up para eliminar consideraciones históricas y de trámites.

Uso de DFD(s) Expandidos

- Generalmente se trabaja con los DFD(s) de mayor nivel ya que en ellos es donde se encontrarán la mayoría de las consideraciones físicas.
- El proceso consiste en reemplazar un proceso de nivel "N" por su correspondiente de nivel "N+1"
- El proceso se repite hasta conseguir un DFD que minimize los aspectos físicos.
- El DFD resultante es sólo un documento de trabajo que servirá de entrada a la siguiente transformación.

Derivando Equivalentes Lógicos de Archivos

- La siguiente tarea consiste en inspeccionar el conjunto de archivos y aplicarles las 3 formas de normalización.
- Con esto se asegura que cuando se inicie la especificación del nuevo sistema se empezará con estructuras de datos simples y bien diseñadas.
- Siempre es más sencillo modificar un proceso que modificar un archivo.

Reemplazo Lógico con "FUERZA BRUTA".

- Algunas consideraciones físicas se eliminan con el DFD expandido y otras a través de la normalización, pero aún pueden quedar algunas.
- En este esfuerzo se recomienda empezar por las primitivas y preguntarse:

- Este proceso tiene que trabajar así, o solamente sucede que trabaja, así,
- El proceso describe que es una política o como esa política se lleva a cabo?

Inspecciones al DFD Lógico

- Invitar al usuario a que participe de una revisión del modelo lógico actual.
- Para ayudar al usuario a comprender el modelo:
  - Hacer referencias físicas oralmente.
  - Prestar particular atención a los nombres de los nuevos flujos de datos y procesos. Solicitar la opinión del usuario.
  - Hacer que el usuario participe en la transformación del modelo físico al lógico.

DERIVANDO EL MODELO LOGICO DEL NUEVO SISTEMA

Introducción

- Hasta ahora se ha considerado la forma en que trabaja el negocio actualmente, a partir de aqui debemos establecer la forma en que el negocio deberá trabajar.
- Para esto emplearemos como base una porción del modelo lógico actual.

El Dominio del Cambio

- Podría automatizarse parcial o totalmente? o hacerse de otra manera?
- Una vez establecido el dominio del cambio, agrupar todos esos procesos en uno sólo (el area de estudio)
- El DFD resultante mostrará las fronteras del nuevo sistema y sus interfaces.

Particionando el Dominio del Cambio

- Una vez identificada el area del cambio el resto del proceso (aproximadamente la mitad del tiempo del análisis) consiste en proceder top-down.
- Diferencias con respecto a las fases previas del análisis:
  - Es estrictamente top.down
  - El analista trabaja independientemente
  - Hay un nuevo juego de reglas.

Guías para el Particionamiento

1. Particionar para minimizar interfaces
2. Poner atención a los nombres.
3. Respetar los límites razonables del particionamiento.
4. Ser exhaustivo con los flujos de datos.
5. Respetar la regla de conservación de los datos.
6. Usar el modelo lógico actual como guía

7. Usar el diccionario de datos.
8. Estar listos para comenzar de nuevo.

Probando la Nueva Especificación Lógica

- Presentar el modelo nuevo al usuario y revisarlo conjuntamente.
- Pedir al usuario su colaboración para depurarlo.
- Estar dispuesto a ajustar el modelo de acuerdo con los gustos particulares del usuario.
- La meta debe ser hacer sentir al usuario que el sistema esta a su gusto y no que se trata de algo impuesto.

MODELOS FISICOS

Introducción

- La última transformación requiere incorporar información física del modelo, principalmente la frontera hombre/máquina.
- La transformación procede como sigue:
  1. Preparar una opción indicando la frontera hombre/máquina.
  2. Agregar factores dependientes de la implementación
  3. Repetir 1 y 2 hasta contar con varias opciones.
  4. Seleccionar una opción.

Estableciendo Opciones

- Por lo regular se ofrecen 3 opciones yendo de una totalmente automatizada a una minimamente automatizada.
- El usuario será el encargado de seleccionar estre alguna de ellas.
- Esto nuevamente sirve al propósito de hacer sentir al usuario que el sistema es de él.

Selección de Opciones

- Para ayudar al usuario a decidir entre las distintas opciones se debe elaborar una análisis costo/beneficio para cada una de ellas.
- El análisis estructurado no hace ninguna aportación el estudio costo/beneficio.

EMPACANDO LA ESPECIFICACION ESTRUCTURADA

Introducción

- En este punto toda la parte conceptual del análisis ha concluído.
- Resta solamente recolectar los productos del análisis y organizarlos para terminar la especificación estructurada.
- Esta labor involucra:
  - Remarcar las interfaces del sistema
  - Preparar una guía para ayudar a los usuarios a manejar la especificación.
  - Terminar los detalles diferidos.

Terminando los detalles diferidos

- Mensajes de error
- Arranque y terminación del sistema
- Información de control del usuario
- Formatos del usuario
- Conversión
- Rendimiento

Presentando las Interfaces Clave

- Se sugiere re-agrupar y re-dibujar los DFD(s) para mostrar las interfaces en el nivel superior (particularmente la interface hombre/máquina).
- Al hacer esto se obtiene una especificación más legible.

Guía para la Especificación Estructurada

- Con el objeto de ayudar al usuario a manejar e interpretar la especificación es recomendable elaborar un breve guía que le indique:
  - Elementos de la especificación.
  - Papel de los DFD(s)
  - Convenciones de niveles
  - El papel del DD
  - Convenciones del DD
  - Correlación DFD-DD
  - Correlación DFD-MINI especificaciones.

DISEÑO ESTRUCTURADO

Introducción

- El diseño estructurado esta basado en las investigaciones iniciadas por Larry Constantine en 1962.
- El diseño es el proceso iterativo de tomar el modelo lógico del nuevo sistema junto con los objetivos del mismo y producir la especificación de un sistema físico que cumpla con esos objetivos.
- El diseño estructurado es un conjunto de consideraciones generales y técnicas para diseño de programas, que facilitan la codificación, depuración y modificación de estos, a través, de un proceso que reduce la complejidad.
- El diseño estructurado es una técnica para reducir la complejidad, mediante la separación de las funciones de un programa en módulos relativamente independientes..
- Una de las ventajas más importantes del diseño estructurado es la posibilidad de re-utilizar código.
- La técnica de diseño modular no es nueva dentro del campo de la programación, aunque se ha usado con diferentes objetivos y diferentes resultados.

¿ Por qué el diseño estructurado?

- En términos de programación, si se duplica el tamaño de un programa, el tiempo de coficación es más del doble.

- Si lo anterior es cierto entonces haciendo más pequeños los programas, el tiempo de desarrollo se reduciría (si las piezas resultantes son relativamente independientes).
- Los beneficios aumentan si cada pieza realiza sólo una función.
- Existe un número correcto de módulos que reducen la complejidad de un programa.

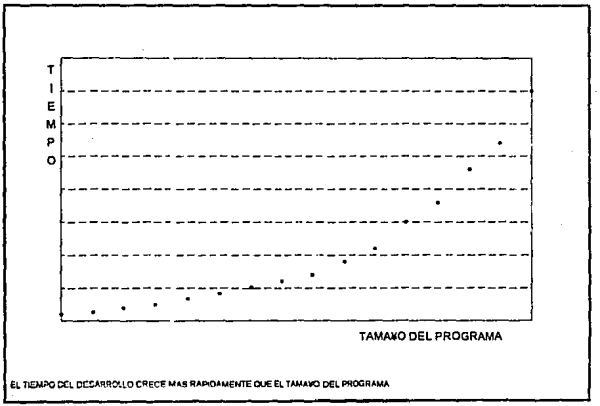


Figura 1.66 En ocasiones el tamaño de los programas incrementa los retrasos en un proyecto.

Consideraciones generales sobre el diseño estructurado:

- La medida primordial para evaluar entre diseños alternativos es la simplicidad.
- La simplicidad se puede conseguir dividiendo al sistema en pequeñas porciones.
- Estas porciones deberán poderse implementar y cambiar con un efecto mínimo sobre las demás porciones del sistema.

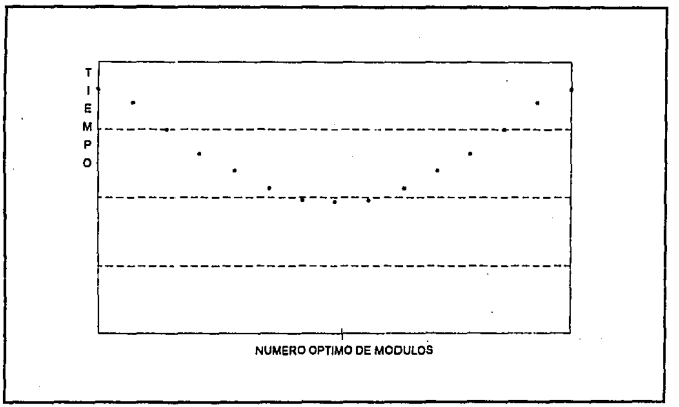
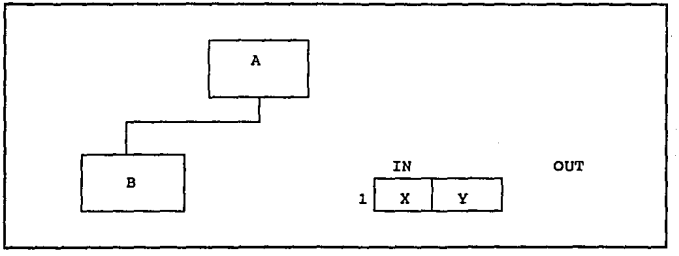


Figura 1.67 El aumento en el número de modulos ocasiona retrasos en la entrega de los sistemas.

- Se deberá poder percibir facilmente "que" y "como" hace su función cada porción.
- También deberá ser fácil cuantificar el impacto de una porción.
- La resolución de problemas es más difícil cuando se tienen que considerar todos los aspectos del problema simultaneamente.
- A las porciones del sistema se les denomina "módulos".
- La herramienta usada en el diseño estructurado es el "diagrama estructurado".



Metas del diseño estructurado :

- Diseñar programas como:
  - Estructuras independientes
  - Módulos que realizan una sola función.
- Las características de los programas resultantes del diseño estructurado serán:
  - Los programas son más simples.

- Los módulos se pueden codificar independientemente.
- El programa se puede entender pieza por pieza.
- Las pruebas se facilitan
- Los efectos colaterales de un cambio se reducen.
- Se reduce la cantidad de errores.

El concepto de módulo:

- Un módulo es un conjunto de instrucciones que pueden ser llamadas a ejecución por medio de un nombre.
- Ejemplos:
  - Programas compilados por separado
  - Un párrafo o sección en cobol
  - Procedimientos en PL/I
  - Sub-rutinas en fortran
  - Macros en ensamblador.
- El objetivo de la modularidad es permitir manejar las piezas de un programa de manera independiente.
- Los módulos serán más independientes si cada uno tiene su propio juego de variables.
- Recomendaciones:
  - Compilar por separado los módulos
  - Invocarlos con un "CALL"

ACOPLAMIENTO Y COHESIVIDAD

Introducción

- Dos medidas para conseguir la independencia entre módulos son: acoplamiento y cohesividad.
- Para lograr máxima independencia:
  - Minimizar acoplamiento
  - Maximizar cohesividad.
- El acoplamiento y la cohesividad miden aspectos opuestos de la misma cosa.

Acoplamiento

- El acoplamiento es una medida de que tan estrecha es una conexión entre 2 módulos.
- Al evaluar alterntivas para dividir un programa resulta útil examinar los tipos de conexiones entre módulos.
- Una conexión es una referencia a una etiqueta o dirección definida en otro lado.
- Al minimizar las conexiones entre módulos también se minimizan las trayectorias a lo largo de las cuales se propagan los errores y los cambios.
- Entre menos y más simples sean las conexiones más fácil será entender los módulos sin hacer referencia a otros.

- La propagación de un error o un cambio provoca un efecto de "cascada" que eleva el mantenimiento del sistema y propicia la aparición de nuevos errores.
- Un alto grado de acoplamiento complica al sistema ya que un módulo es más difícil de entender, cambiar o corregir si está altamente interrelacionado con otros.

Ejemplos de conexiones entre instrucciones.

```
VALIDATE-TRANSACTION.
  MOVE ' YES'  TO VALID-TRANS-FLAG.
  IF STOCK-NUMBER OF TRANS-RECORD IS NOT NUMERIC
      OR DATA-LIMIT OR TRANS-RECORD IS NOT NUMERIC
  THEN
      MOVE CORRESPONDING TRANS-RECORD TO MESSAGE-RECORD
      MOVE NOT-NUMERIC-MSG TO MESSAGE-FIELD
      MOVE MESSAGE-RECORD TO OUTPUT-LINE
      PERFORM LINE-OUT
      MOVE 'NO' TO VALID-TRANS-FLAG
  ELSE
      IF DATE-LIMIT OF TRANS-RECORD IS LESS THAN '70001'
      THEN
          MOVE CORRESPONDING TRANS-RECORD TO MESSAGRE-RECORD
          MOVE DAT-LIMIT-MSG TO MESSAGE-FIELD
          MOVE MESSAGE-REDORD TO QUTPUT-LINE
          PERFORM LINE-OUT
          MOVE 'NO' TO VALID-TRANS-FLAG.
```

Factores que afectan el grado de acoplamiento

El grado de acoplamiento se mide en 3 dimensiones:

1. El tamaño de la conexión.
2. El tipo de conexión.
3. Que es lo que se envía o recibe.

Si la medida en alguna de estas dimensiones es grande, el grado de acoplamiento será alto.

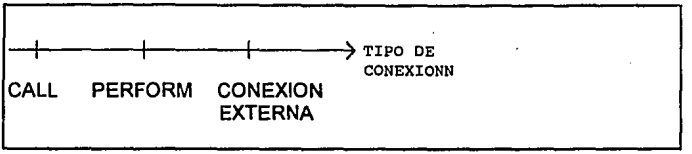
Tamaño de la Conexión

- Entre menos datos se pasen entre módulos menor será el grado de acoplamiento.
- La medida aquí es la cantidad de datos pasada cada vez que se usa la interface.
- No pasar los datos armados dentro de una estructura, pasando los datos individualmente como parametros aumenta la claridad.
- Una técnica comunmente empleada para pasar datos es el uso de una area común de datos. Esto aumenta considerablemente el grado de acoplamiento.
  - En cobol la "linkage section"
  - En PL/I las variables "external"
  - En fortran el postulado "common"
- Un ambiente común de "N" elementos compartidos por "m" módulos resulta en:

$$NM\ (M - 1)$$

Trayectorias a lo largo de las que se pueden propagar los cambios y errores.

Tipo de Conexión



- Si un módulo hace referencia a una variable que esta en otro módulo, entonces el contenido de ese otro módulo deberá tomarse en cuenta al hacer un cambio o corregir un error.
- Un sistema es más simple si sus módulos pueden usarse sin necesidad de conocer su interior.

Problemas con Conexiones Externas

- Este tipo de conexión prolifera a cada nuevo programa "llamador".
- Resulta muy difícil lograr que trabajen los módulos en un ambiente asincrono.
- Al ocurrir un error se deben considerar simultaneamente todos los módulos involucrados.

¿Qués es lo que se Comunica?

Branch directo - external entry (PL/I)
Contiene código - Perform A through B (cobol)

Parametro - Switch, flag, etc.

- Los módulos por lo menos se deberán pasar datos entre si para que formen parte de un mismo sistema.
- Cuando se pasa información de control de un módulo a otro (switch, flag, etc.) se agrega complejidad al sistema. Siempre existirá una estructura alterna que elimine esa complejidad.
- La comunicación híbrida se produce cuando un módulo modifica las instrucciones de otro módulo. Este es el caso más grave (sólo es posible en ensamblador).

Cohesividad

- Un método alterno para reducir el acoplamiento entre módulos es maximizando la relación entre los elementos de un mismo módulo.
- Elemento es una instrucción, segmento o sub-función de un módulo.
- El objetivo es reducir el acoplamiento elevando la ligadura entre los elementos de un módulo.
- La ligadura es la medida de cohesividad de un módulo.
- Escala de cohesividad, de menor a mayor:
  1. Coincidental
  2. Lógica.
  3. Temporal
  4. Por comunicación
  5. Secuencial
  6. Funcional

- Se debe procurar que en cada módulo haya cohesividad funcional.

Ligadura Coincidental

- Cuando nos preguntamos por que un grupo de instrucciones están en un programa y la respuesta es: bueno, tenían que estar en algún lugar, esto es una forma de ver la ligadura coincidental.

  Ejemplo:

  ```
  Move  A to B
  Read  E
  Move C to D
  ```

- Si este módulo fuera llamado por varios programas y uno se cambiara requiriendo mover "A to X" se complicaría la codificación del módulo.

Ligadura Lógica

- Aquí los elementos de un módulo hacen cierta clase de funciones, V. GR., editar datos que incluye altas, bajas y cambios.
- Los módulos se caracterizan por hacer el mismo tipo de función pero sobre diferentes datos.
- Otra característica es que se comparte código entre las funciones del módulo.

- Generalmente, al inicio del módulo hay una estructura "case" que controla el saldo a alguna parte del módulo según la función a desempeñar.
- Comunmente pensamos que usar código compartido ahorra tiempo de codificación. En realidad lo incrementa.
- Es mucho mejor codficar cada función como un módulo separado aunque haya código repetido.

Ligadura Temporal

- Estos módulos se caracterizan porque todos sus elementos se ejecutan en cada llamada.
- Es más simple que la ligadura lógica puesto que no requieren de un "CASE" al inicio.

Ejemplo:

- Validar todos los datos de un registro
- Inicializar variables.
- Cerrar archivos.
- Crear tablas en memoria.

Ligadura por Comunicación

- Los siguientes tipos de ligadura son considerablemente más fuertes que los anteriores.

- Este es el primer tipo donde los elementos de un módulo se refieren a los mismos datos.
- En la ligadura por comunicación se hace referencia a los mismos datos de entrada/salida.

Ejemplo:

- Producción de diversos reportes a partir de los mismos datos de entrada.
- Imprimir y almacenar un registro.

Ligadura Secuencial

- Ocurre cuando la salida de un elemento es usada como entrada para el siguiente elemento.

Ejemplo.

- Leer y editar datos.
- Crear y almacenar registros.
- Sumarizar cifras e imprimirlas.

Ligadura Funcional

- Una función describe la transformación de datos de entrada en datos de salida.
- Si todos los elementos de un módulo contribuyen a un mismo objetivo posiblemente el módulo este ligado funcionalmente.
- Una técnica útil para saber si un módulo esta ligado funciolmente es describir su función y examinar la oración resultante:

  - Si la oración contiene más de un verbo probablemente el módulo desempeñe más de una función.
  - Si el predicado de la oración no contiene un sólo objeto específico, probablemente el módulo tenga ligadura lógica.
  - Palabras como inicializar, limpiar, etc. implican ligadura temporal.

- Los módulos ligados funcionalmente siempre pueden describirse en términos de sus elementos usando una oración.
- Un problema sin resolver es como decidir que tanto hay que dividir las funciones con ligadura funcional.
- Un posible enfoque es cuando el módulo mida de 2 a 4 hojas de codificación.
- Otro enfoque es detenerse cuando el módulo no contenga un subconjunto de elementos que puedan usarse por si solos.

Ejemplo:

- Obtener la raíz cuadrada
- Calcular deducciones
- Calcular seguro social
- Imprimir cheques

Un ejemplo en Mecánica :

- Cada elemento del motor desempeña una función específica.
- Una característica de una unidad funcional es que puede ser facilmente reemplazada por otra que sirva al mismo propósito.
- Además, hay conexiones simples entre las piezas.
- Un módulo esta ligado funcionalmente si contiene todos los elementos, y sólo aquellos elementos necesarios para desarrollar una función.

TECNICAS DEL DISEÑO ESTRUCTURADO

Diagramas de Estructura

- El objetivo del diseño estructurado es construir programas con estructuras de una sola función, compilados por separado, con el menor número de parametros posibles.
- La herramienta del diseño estructurado es el diagrama de estructura.

- El proceso de diseño consiste en construir una estructura que cumpla con las especificaciones y usar los conceptos de acoplamiento y cohesividad para evaluar el diseño paso por paso.

Definir los parámetros requeridos

- Identificar todos los parámetros necesarios para cumplir con las especificaciones.
- El objetivo primordial es obtener la mayor independencia entre módulos.
- Los parámetros son la medida más directa de que tan independientes son los módulos entre si.

Trabajar con la Estructura Completa

- Se recomienda que el equipo de diseño trabaje siempre con el diagrama de estructura completo.
- Aunque el equipo este formado por dos o más personas, es preferible que trabajen juntos con el diagrama completo.
- El diagrama de estructura para la mayoría de los programas puede ser hecho por una sola persona.
- Programa es el conjunto de instrucciones que se ejecutan a través de un comando del sistema operativo (un "exec" en JCL (Job Control Language) o una transacción "on-line").

Diagramas Comunes

- Una de las mejores ayudas para la elaboración del diagrama es saber de antemano como va a quedar.
- Generalmente los diagramas se pueden enmarcar dentro de un par de estructuras comunes:
  - La estructura de entrada/salida.
  - La estructura para transacciones.

Guías para el Diseño

Las guías más importantes :

- Tratar de conseguir la mayor independencia, simplicidad y claridad.
- La independencia es lo más importante de todo ya que es el mecanismo usado para reducir la complejidad.
- Para lograr simplicad no tratar de reducir líneas de codificación. La optimización y la simplicidad se contraponen, debiera ser más importante la simplicidad.
- La claridad se mide en función de que tan fácil es saber porque un programa hace lo que hace.

Elementos de un Módulo Funcional

- Todo módulo debe realizar el trabajo que se le asigna.

- Si existe alguna circunstancia bajo la cual un módulo no pudiera realizar el trabajo que se le asignó, deberá indicar sobre ello al programa llamador.
- Cuando las especificaciones de un programa requieran detalles del funcionamiento interno de un módulo, ese módulo debiera producir un "LOG" reportando las acciones tomadas.

Características de la "Caja Negra"

- Un módulo predecible es aquel en el que dadas entradas identicas, se comporta de la misma manera siempre que es llamado.
- Los módulos que llevan rastro de si mismo, por lo general no son predecibles (en PL/I, uso de variables estáticas).
- El concepto módulos predecibles se conoce también como "caja-negra".
- Con una "caja-negra" el usuario puede entender el funcionamiento del módulo y usarlo sin tener que conocer su interior.

Evitar funciones duplicadas

- Durante el diseño se podría advertir que dos módulos pueden hacer funciones muy similares pero que no son exactamente las mismas.

Implantar cada Primitiva como un Módulo

- El convertir cada primitiva de un DFD es similar a implantar cada función en un módulo.

- De esta forma, si cambia la especificación solo habrá que cambiar un módulo.

Seleccionar la solución más simple

- Evitar la tentación de generalizar una rutina con la idea de que podría ahorrar futuros esfuerzos.
- Cuando se intenta generalizar por lo regular el módulo necesitará más parámetros aumentando el grado de acoplamiento.
- La experiencia ha mostrado que muchas veces no se consigue la generalidad deseada.

Permitir la Flexibilidad

- Flexibilidad es la capacidad de adaptar nuevos requerimientos a un programa.
- Versatilidad es el rango de funciones que un programa puede desempeñar.
- Resulta tentandor tratar de proporcionar flexibilidad para expectativas futuras, escribiendo programas más versátiles.
- La versatilidad aumenta la complejidad haciendo más difícil modificar un programa, decrementando la flexibilidad.
- Las soluciones más simples son las más flexibles.

Parámetros de Error y Fin de Archivo (EOF)

- La mayoría de los módulos necesitan informar al módulo llamador si no pudieron hacer su función. Esto se hace comunmente a través de un parámetro de error.
- Aunque este parámetro puede aparecer en diferentes interfaces deberá tener un significado distinto en cada caso.
- Siempre deberá ser un indicador de un módulo a su llamador inmediato de que ese módulo no pudo realizar su función.
- Siempre que se pueda encontrar una alternativa para estos parámetros se mejorará el acoplamiento entre módulos.

Mejorando los Diagramas de Estructura

- Eliminar el uso de "SWITCHES".
- Extraer los módulos de sus llmadores.
- Checar los módulos que son llamados por más de tres "llamadores".
- Verificar módulos que llaman a más de tres módulos diferentes.
- Revisar módulos que usen más de dos hojas de codificación o que tengan menos de cinco instrucciones.

GUIAS PARA EL DISEÑO. Resumen.

- Al diagrama de estructura obtenido hay que hacerle refinamientos.
- Para reducir el efecto de los cambios se recomienda que la estructura del diseño coincida con la estructura del problema. La forma debe estar subordinada a la función.

- Un sistema resultará más simple cuando el "ambito de efecto" de una decisión este dentro del "ambito de control" del módulo que contiene la decisión.
- El "ambito de control" de un módulo abarca a ese módulo y a todos sus subordinados.
- El "ambito de efecto" de una decisión es el conjunto de módulos que contienen código cuya ejecución depende del resultado de la decisión.
- Para que el ambito de efecto quede dentro del ambito de control hay 2 caminos: mover el elemento de decisión hacía arriba o mover los módulos que estan fuera del ambito de control de tal suerte que caigan dentro del el.
- Minimizar el uso de "flags" de error.
- Tratar de que los módulos de "inicialización" cumplan con el requisito de caja negra.
- Verificar los modulos demasiado grandes (más de 100 instrucciones) o demasidado pequeños (menos de 15 instrucciones)
- Eliminar funciones duplicadas pero no código duplicado.
- Verificar los módulos que son llamados de muchos lugares distintos o aquellos que llaman a muchos otros módulos.
- Tratar de reducir al mínimo la cantidad de parámetros que se pasan de un módulo a otro.
- El equipo de diseño deberá trabajar en forma conjunta y teniendo a la mano el diagrama de estructura completo.

MANTENIMIENTO DE LA ESPECIFICACION ESTRUCTURADA

Metas del Mantenimiento de la Especificación

- Minimizar el costo de procesar cada cambio propuesto. En el costo intervienen:

  - Formulación de la solicitud de cambio
  - Evaluación del impacto
  - Decisión
  - Incorporación del cambio
  - Implantación del cambio.

- Cada cambio deberá ser formulado como un modelo incremental.
- El cambio solicitado deberá tener una interface bien especificada con el modelo del sistema.
- El cambio solicitado deberá tener todas las cualidades de la especificación estructurada, particularmente, deberá ser iterativo.
- Se deberá poder acomodar facilmente el cambio con el sistema en desarrollo.
- La metodología para el análisis del cambio será la misma que la empleada para el análisis del sistema.

El concepto del Módulo Incremental

- Para construir el modelo incremental se debe proceder de la misma forma como se hizo en la fase de análisis, es decir, DFD, DD y Mini-especificaciones.
- Esto deberá mostrarse al usuario para que lo corrija y afine en un proceso iterativo.
- El resultado final será el documento de especificación incremental (DEI).
- El DEI consiste de:
  - Formar de solicitud de cambio
  - DDF(s) que reemplazan a los DFD(s) originales.
  - DD incremental
  - Mini-especificaciones que reemplazan a los procesos afectados.

Procesos para Mantenimiento de la Especificación

- Describir el cambio propuesto elaborando el DEI y marcando la interface con el modelo existente.
- Cuantificación del impacto.
- Incorporación del cambio en la especificación estructurada: reemplazando las partes afectadas del modelo actual con el modelo incremental.
- Implantación del cambio.

El mito de "CONGELAR" los Cambios

- La gerencia de sistemas se siente más segura si al terminar la fase de análisis "congela" todos los cambios que el usuario solicite.
- En algunos lugares se opta por incluir un cómite de control de cambio.
- Es una falacia hablar de congelar los cambios. Lo que se debe de tratar de conseguir es mantener una huella de los cambios.

PRUEBA DE ACEPTACION.

Propósito de la Prueba de Aceptación

- El propósito de la prueba de aceptación es evitar sorpresas cuando un sistema pasa a producción.
- El propósito del análisis es evitar sorpresas en la prueba de aceptación.
- El resultado de la prueba de aceptación es binario: se acepta o se rechaza el sistema.
- La prueba de aceptación no forma parte de un proceso de corrección de errores.
- La prueba de aceptación evalua el trabajo del análisis, no de la implementación.
- La prueba de aceptación se deriva única y exclusivamente del documento de objetivos.
- Existen 5 tipos de pruebas.

  - Pruebas de trayectoria normal
  - Pruebas de trayectorias de excepción.
  - Pruebas de estado transitorio.

- Pruebas de rendimiento
- Pruebas especiales.

Derivación de Pruebas de Trayectoría Normal

- Estas pruebas de derivan del DD.
- Se hace una prueba por cada flujo de datos.
- Cada prueba se hace con un número válido de entradas que se conformen a las definiciones del DD.
- Para cada data element se toman 3 valores: los extremos y un valor medio.
- Para cada flujo de datos se buscan todas las permutaciones de data elements que sean validas.

Derivación de Pruebas de Excepción

- Aqui se trata de generar un conjunto de entradas para probar la forma de manejar errores.
- Se seleccionan valores invalidados para cada data element.
- Estos valores se usan junto con los flujos de datos de la prueba normal.
- Cada prueba de excepción involucrará solamente 1 data element.

Ejemplo:
Valores para monto = nulo,), $5555555555, 555, $abc, abc

Pruebas de Estado Transitorio

- En este tipo de prueba se tratan de introducir secuencias de flujo de datos (altas, cambios, bajas).
- Los flujos de datos para crear estas secuencias se pueden obtener de los creados durante las pruebas normales.

Pruebas de Rendimiento

- Con estas pruebas se mide al sistema en terminos del rendimiento esperado en el documento de objetivos.
- Se hará una prueba por cada factor de rendimiento a evaluar (tiempo de respuesta, capacidad de volumen, cpt, espacio en dasd, etc.)

Pruebas Especiales

- Son todas aquellas pruebas que no estando dentro de las anteriores descritas, dependen de la naturaleza particular del sistema.

Documentación de las Pruebas

- Procedimiento de prueba
- Datos de prueba
- Resultado esperado de la prueba
- Procedimiento para regresar el sistema a su estado inicial.

HEURISTICOS PARA ESTIMACION

Algoritmos VS Heurísticos.

- Son mutuamente excluyentes.
- Un algoritmo conduce a un resultado certero.
- Un heurístico habla de un resultado esperado.
- El desarrollo de un heurístico involucra:
  - Estimados derivados empiricamente.
  - Datos empíricos de productividad
  - Reglas de estimación.

El Estimado derivado Empíricamente

- Para derivar estimados empíricos se necesitan:
  1) Un indicador cuantitativo
  2) Datos empíricos que correlacionen el rendimiento observado con el indicador.
- Se sugiere el número de primitivas en la especificación como indicador.
- Los datos de correlación hay que recopilarlos:
  - Meses/hombre requeridos para desarrollar un sistema, expresado en horas/primitiva.
  - Meses/hombre requeridos por fase.
  - Recursos para pruebas usados por primitiva. etc.

Datos Empíricos de Productividad

- Conservar registros del proyecto tales que permitan producir formulas empíricas que tomen en cuenta variaciones en la productividad debidas a:
  - Duración del proyecto
  - Tamaño del equipo de trabajo
  - Tiempo de máquina y pruebas.
  - Complejidad de las interfaces
  - Lenguaje, etc.

Reglas de Estimación

- Una estimación no es una concesión
- Una estimación no es una negociación.
- Las estimaciones no estan sujetas a contraofertas.
- Estimar no es dividir una duración fija entre sus partes.
- Un reajuste en una fase del proyecto implica ajustes en la misma proporción en el resto de las fases.
- Si se desea un estimado significativo no se debe dar "la respuesta" previamente.
- La proporción entre un estimado optimista y un estimado útil a la planeación suele ser uniforme en un individuo
- Usar comites de estimación y no considerar la estimación del lider del proyecto.

TENDENCIAS EN EL ANALISIS Y DISEÑO DE SISTEMAS.

Introducción

- Con el transcurso del tiempo los costos de las computadoras han ido bajando.
- Eventualmente podremos tener una computadora como la IBM-370 en un solo Chip.
- En los siguientes 10 años la velocidad de cómputo crecerá por lo menos en un factor de 10.
- Se estima que el número de computadoras usadas en aplicaciones cientificas y comerciales crecerá a un ritmo del 25% anual.
- El número de aplicaciones crece actualmente a un ritmo de 45& al año.
- Si se asume que no aumenta la productividad de los programadores, con el rango de crecimiento esperado, en 10 años se requerirán 93 veces más programadores que ahora para satisfacer la demanda de aplicaciones.
- Todo esto indica que la productividad en el desarrollo de aplicaciones tiene que aumentarse al doble.

Estrategias para el Futuro

- El usuario final podrá tener acceso y manipular su información (numerica, textos y gráficas) para resolver el 50% de sus problemas, sin la ayuda de DP.

- 30% de las aplicaciones podrán ser generadas sin programación.
- 70% de las aplicaciones contendrán menos de un 30% de codificaión en lenguajes por procesos ("procedural language")

El cambio en el Papel del Analista

- Para cubrir las necesidades futuras de información se requiere acelerar el análisis de sistemas y/o eliminar la programación.
- En el futuro el propio analista de sistemas creará las aplicaciones.
- El análisis de sistemas no requerirá de hacer la especificación para la programación.
- El analista de sistemas ayudará al usuario a que cree sus propias aplicaciones.

CAPITULO II

ANALISIS

DE LA

PROBLEMATICA

II.1 ANTECEDENTES

II.1.1 ENTORNO DE AUDITORIA CON BANCO NACIONAL DE MÉXICO

El negocio de Banco Nacional de México, S.A. (Banamex) es la intermediación financiera, y consiste en la captación y colocación de recursos.

Estos recursos son:

- Captados a través de ahorradores e inversionistas.
- Colocados concediendo préstamos a los sectores productivos y particulares.

Banamex ofrece una amplia gama de actividades que incluye la captación de recursos dentro y fuera del país, el impulso a las operaciones de aseguramiento y fiduciarias, el respaldo a programas de vivienda, el financiamiento a actividades agropecuarias, comerciales y turísticas, el desarrollo de productos industriales e inmobiliarios, la promoción especializada del comercio exterior y, en general, el apoyo y asesoría a las actividades productivas.

Utilizando modernos sistemas, que combinan la automatización con el servicio personalizado, Banamex atiende a industrias, empresas

comerciales, dependencias del Gobierno y público en general en sus necesidades de inversión y crédito, que le permiten cumplir al Banco con su papel de intermediario financiero.

Hoy en día esta institución cuenta con mas de 125 servicios, figurando entre ellos como uno de los principales las Tarjetas de Crédito, ya que las mismas son uno de los servicios que en la actualidad ha cobrado mas importancia y trascendencia entre sus usuarios, dadas la múltiples opciones y ventajas que les ofrece.

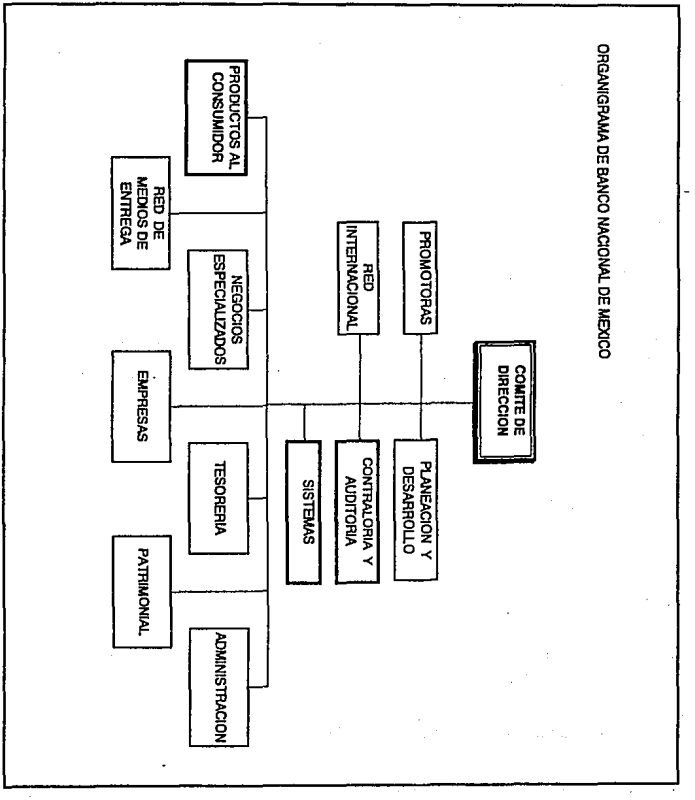
Para entender el entorno de la problemática de nuestra tesis, se describe a continuación un esquema general de la estructura organizacional de Banamex (Figura 2.1.1) destacando las áreas involucradas en el análisis del mismo:

Productos al Consumidor

Comprende la operación de tarjetas de crédito y otros productos, siendo el propósito de éstos:

- Atender al mayor número de clientes con servicios bancarios de la mas alta calidad.
- Lograr altos niveles de rentabilidad en su venta.
- Ser el Banco líder en innovación, tecnología, empaquetamiento y distribución masiva de los mismos.

ORGANIGRAMA DE BANCO NACIONAL DE MEXICO



Funciones principales:

- Medios de Pago.
- Financiamiento para bienes de consumo.
- Captación.
- Financiamientos inmobiliarios.
- Atención a mercados emergentes con productos estandarizados.
- Banca Electrónica.

Sistemas

Su propósito es el de proporcionar a cada una de las áreas del Banco, los sistemas de control de información que le permitan responder a las demandas de servicios bancarios de su clientela, aumentar su productividad y facilitar la administración interna.

Funciones principales:

- Desarrollar sistemas avanzados que apoyen a la Institución.
- Proporcionar a clientes y usuarios productos y servicios a través de los CSIs (Centros de Servicio de Información).

- Dar mantenimiento a los sistemas existentes de acuerdo a las necesidades del Banco dinámicamente en función de la evolución del sistema financiero mexicano.

- Contar con la infraestructura mas avanzada en equipo, redes, software y bases de datos.

Auditoria

Es el área encargada de verificar que las diferentes actividades que conforman la actividad bancaria sean efectuadas tal y como han sido establecidas.

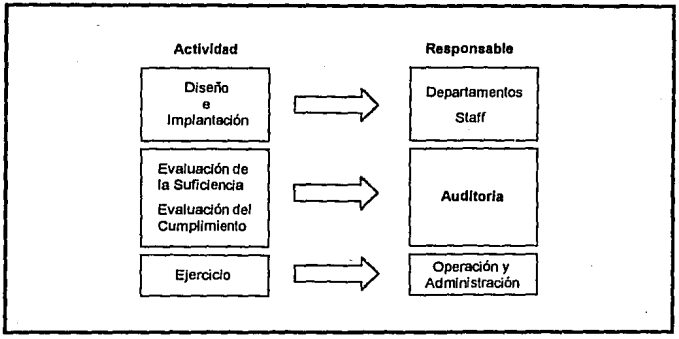


Fig. 2.1.2 Actividades de los departamentos responsables de la información.

Para que Banamex pueda lograr satisfactoriamente sus objetivos, necesita administrar sus recursos, y verificar que las operaciones y procedimientos establecidos se estén llevando a cabo como está estipulado, para lo cual se apoya en el *Control Interno.*

Dentro del Banco, la evaluación del control interno que realiza la División de Auditoría permite dar un diagnóstico sobre la suficiencia y funcionalidad del control incorporado a los sistemas, y conocer el grado de exposición al riesgo operativo derivado de incumplimientos al mismo.

La misión es la de proporcionar información a los Directivos responsables sobre los riesgos presentes, promoviendo que tomen decisiones para evitar su posible materialización. Dicho trabajo es de carácter preventivo.

La Auditoría Interna como tal, es una función profesional e independiente de evaluación, establecida dentro de Banco Nacional de México, para examinar sus actividades como un servicio a la propia organización y se le puede definir de la siguiente manera:

La Auditoria Interna es la evaluación sistemática y objetiva de las operaciones y controles dentro de una organización para determinar si:

- La información financiera y operativa es precisa y confiable.
- Los riesgos en la empresa son identificados y minimizados.
- Se siguen políticas aceptables y procedimientos internos y se cumple con las regulaciones externas.

- Existen estándares de ejecución del trabajo y se vigila su cumplimiento.
- Los recursos son utilizados eficiente y productivamente.
- Los objetivos de la Institución son alcanzados eficazmente.

Todo ello, con el propósito de apoyar a los miembros de la organización en el eficaz cumplimiento de sus responsabilidades.

II.1.1 INTERRELACIÓN DE AUDITORÍA, PRODUCTOS AL CONSUMIDOR Y SISTEMAS

Dentro de la estructura organizacional de la División de Auditoría, existen varios departamentos dedicados a auditar cada una de las áreas que conforman el Banco, el departamento en cuestión para efectos de nuestra tesis es el departamento de Auditoría a Productos al Consumidor.

Se puede decir entonces, que Productos al Consumidor es el área usuaria de Auditoría para llevar a cabo en ese lugar la labor de revisión y ejercicio de la práctica de auditoría en dicha área, entre otros productos de consumo sobresale la operación de las tarjetas de crédito como una de las más importantes.

Tradicionalmente, el personal de Auditoría se rige bajo ciertas normas y principios para realizar su trabajo, sin embargo, en algunas situaciones la labor de investigación y análisis de tarjetas de crédito requiere forzosamente de un proceso automatizado para la obtención de

información, debido a los grandes volúmenes de datos que se manejan.

Es por esta razón, que el departamento de Auditoría se ve en la necesidad de solicitar apoyo al área de Sistemas de Banamex para poder contar con la información específica que le permita realizar un análisis de la situación de las tarjetas de crédito en los siguientes aspectos: ***cartera vencida, cartera preventiva, aperturas recientes, facturación y patrones de fraude.***

Es importante destacar, que el área de Sistemas de Banamex tiene como prioridad dar soporte a las áreas usuarias en el desarrollo y mantenimiento de los sistemas institucionales que soportan la operación del banco (Fig. 2.1.3), por lo que no se cubren como se quisiera todos los requerimientos de la gran cantidad de oficinas del Banco. El Departamento de Auditoría a Productos al Consumidor, recibe del área de Sistemas los datos necesarios de Tarjetas de Crédito para que internamente maneje y obtenga información según sus necesidades específicas.

II.1.2EVOLUCIÓN DE LAS TARJETAS DE CRÉDITO

En Banco Nacional de México, S.A., la Banca de Menudeo como área responsable de un mercado especifico fue creada el 10 de julio de 1987, bajo el nombre de Banca de Autoservicio, con el fin de satisfacer las necesidades financieras del mercado masivo mexicano a través de productos y servicios que desde su diseño contemplen ser elaborados y potencialmente entregados bajo fórmulas de autoservicio.

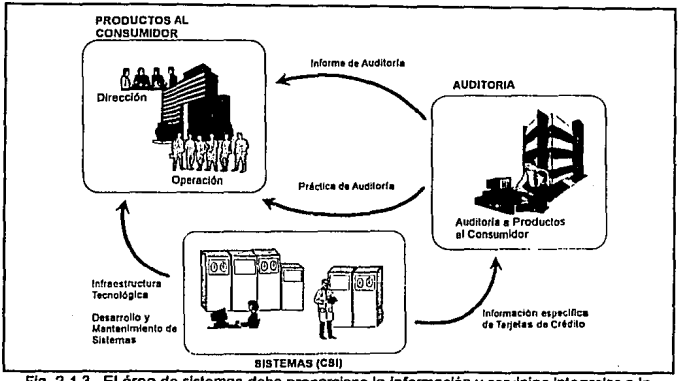


Fig. 2.1.3 El área de sistemas debe proporciona la información y servicios integrales a la institución.

Esta Banca tiene su origen en la División de Tarjetas Banamex, dentro de su proceso de desarrollo natural al igual que la primera tarjeta de crédito bancaria en México, ha evolucionado y permitido a la institución servir en sus necesidades de financiamiento personal a más de 2'000,000 de clientes. La experiencia de operar estos volúmenes de transacciones , la infraestructura tecnológica establecida y el desarrollo de una fuerte y dinámica mercadotecnia, fueron las herramientas para analizar el negocio de la emisión de tarjetas de crédito y plantear una nueva definición que establece como tarea fundamental la creación y el desarrollo de medios de pago, materializables en plástico, papel y electrónica.

Bajo este criterio, las Tarjetas Banamex han diversificado sus mercados, clasificando el origen de los recursos que expresan, en crédito y débito, según se trate de recursos provenientes de una linea de financiamiento o del patrimonio del propio cliente usuario de la tarjeta.

Así las tarjetas de crédito han segmentado su mercado, teniéndose ahora una familia integrada por cuatro grandes lineas de productos que satisfacen las necesidades de uso doméstico y de uso internacional en dos niveles, además de las propias de los ejecutivos de empresas.

La internacionalización de las tarjetas iniciada en 1987, dió lugar a una mayor vinculación con las dos marcas mas importantes del mundo: MasterCard y Visa Internacional. Así, de las tarjetas hasta entonces existentes, sólo Banamex, incorporó a su diseño la normatividad y los logotipos de ambas corporaciones.

En mayo de 1987, se dio al plástico un uso mas pleno de sus capacidades, al convertirse las tarjetas en vehículo de inversión y crédito a la vez.

En su expansión, las tarjetas son hoy en dia, instrumentos que permiten financiar requerimientos de la microindustria, el comercio y el turismo, además del tradicional financiamiento al consumo.

A su vez, la infraestructura tecnológica instalada, ha servido no sólo para optimizar la demanda de operaciones transaccionales ligadas al crédito y al

débito a través de "Cajeros Electrónicos" y otros dispositivos de entrega de efectivo, sino también, para hacer mas eficiente la operación en el punto de venta (comercios), a través del sistema de "Transferencia Electrónica de Fondos y Captura de Datos" desde el negocio afiliado.

La capacidad operativa ha sido comercializada al operarse en Banamex financiamientos otorgados por terceros, entre los que se incluyen instituciones del propio sistema bancario mexicano.

En la medida que el quehacer fundamental de esta área es hacer Banca, las tareas de captación han encontrado también en el plástico el medio idóneo para incorporar nuevos instrumentos de ahorro al portafolio de servicios para el mercado masivo. Un ejemplo de ello es la tarjeta de débito "invermático", producto que inicio su operación en 1988.

Con "Invermático", la Banca de menudeo inicio el desarrollo de un concepto de ahorro masivo, incorporando avances tecnológicos para su manejo; haciendo atractivo y posible el acceso a mercados de economía informal, y otros segmentos que hasta entonces habían estado ausentes en el sistema bancario mexicano.

Atender al mercado masivo en sus necesidades financieras, implica desarrollar productos y servicios que permitan el acceso a los servicios bancarios, a clases socioeconómicas nuevas y diferentes de las que tradicionalmente han servido a la Banca.

En el Banco Nacional de México, la Banca de Menudeo es la resultante de la evolución y el enriquecimiento de un producto -la tarjeta de crédito-, cuyos alcances permitieron incorporar a los servicios bancarios segmentos reconocidos como es el caso del mercado masivo. De ahí que el marco de referencia de esta Banca se sustente en la evolución constante de conceptos y definiciones que, a través del tiempo, se validan y convierten en estrategias y acciones concretas.

Las tarjetas de crédito bancarias en el mundo y también en Banamex, surgieron como una modalidad del crédito personal y, por consiguiente, se aplicaron criterios de concesión equivalentes a los que se manejan hoy en día en la banca tradicional.

Al ubicar a las tarjetas como una modalidad de los prestamos personales, si bien en 1968 imprimió agilidad y dinamismo al quehacer de colocación, también enfrento al Banco a notables retos en el terreno operativo.

Sin embargo, hasta 1981 la modalidad original se mantuvo constante. El potencial de la tarjeta para convertirse también en instrumento de intermediación, no se desarrollo. Para entonces, la dimensión del mercado, la amplia aceptación de la tarjeta como medio de pago, los hábitos de uso de las mismas, el desarrollo de sistemas y otros avances de la tecnología, permitieron visualizar el negocio de emitir y operar tarjetas en un contexto mas amplio.

La División de Tarjetas Banamex redefinió su misión en términos de:

Emitir y operar medios de pago modernos que sirven para transferir fondos de un comprador a un vendedor con independencia del origen de los recursos que se afectan, que pueden ser del propio cliente o provenir de un financiamiento y que, formalmente, se pueden expresar en plástico, papel, fórmulas electrónicas o combinaciones de estas.

Esta definición significa el parteaguas entre una tarea de concesión de crédito personal y el reconocimiento de que el Banco Nacional de México se crean, manejan y operan instrumentos, que sustentados en el concepto de "medios de pago", sirven para realizar intermediación en sus funciones de captación y colocación, con tecnología y mercadotecnia de punta.

En lo que respecta a la historia de las tarjetas de crédito, fué precisamente en enero de 1968 cuando Banco Nacional de México lanzó al mercado la primera tarjeta de crédito bancaria en nuestro país y América Latina: "Bancomático"

Durante su primer año de existencia, el número de tarjetahabientes alcanzó la cifra de 15 mil, en tanto que los negocios afiliados llegaron a 1,500, repartidos en la Ciudad de México, Guadalajara y Acapulco.

A nivel mundial, los primeros registros que se tienen de tarjetas, fué en 1924 en los Estados Unidos, cuando por primera vez se utilizó una tarjeta de crédito. La *General Petroleoum* la introdujo para la adquisición de

combustibles. Desde entonces, numerosos almacenes comerciales, como Sears y expendios de gasolina, emitieron sus propias tarjetas de crédito.

Puede afirmarse que la llamada época del <<dinero de plástico>> comienza en 1949, a raíz de la iniciativa de Frank McNamara, famoso hombre de negocios de Nueva York que ideó un procedimiento que le permitiera comer en algunos de los mejores restaurantes de esa ciudad sin tener que llevar en el bolsillo dinero en efectivo. Así creó una organización que garantizaba el pago de consumos realizados por sus socios, a la cual llamó Dinners Club. Muy pronto se incluyeron grandes almacenes y hoteles entre los establecimientos afiliados. En 1951, debido al gran número de agremiados, fué necesario fabricar tarjetas de cartulina que contenían el nombre y la firma del socio.

Ese mismo año, el Franklin National Bank lanzó la primera tarjeta de crédito bancaria en la historia, propagándose el ejemplo en toda la Unión Americana, por lo que a fines de 1953 existían ya 62 bancos con tarjeta propia, y al término de la década sumaban casi 2000.

En México, el primer intento por emitir tarjetas de crédito se remonta a 1953, cuando se fundó el Club 202, S.A. Su objetivo era expedir tarjetas de identificación que permitieran a funcionarios o empleados de cierta compañía, mediante convenios previos, firmar la cuenta de sus gastos en lugares que aceptaran la garantía de que la institución les pagaría en nombre del cliente.

De esta manera nació la tarjeta de crédito Club 202, que en 1956 se fusionó a Dinners Club, que para entonces tenía más de 400 mil socios y cinco mil negocios afiliados en 800 ciudades en 60 países. Poco mas tarde aparecen en México las tarjetas American Express y Carte Blanche, pero su empleo estaba limitado a un pequeño grupo de personas con un alto poder de compra.

El 6 de Agosto de 1968, algunos banqueros independientes de los Estados Unidos formaron una asociación llamada Interbank Card Association (ICA), para eliminar las autorizaciones, intercambios y liquidaciones de transaccciones efectuadas con sus tarjetas de crédito locales.

En 1968, Banamex es el primer miembro internacional de ICA en México. Poco después se unen los bancos miembros de Eurocard International en Europa, así como los primeros bancos japoneses.

En 1969, ICA adopta la marca Master Card y el logotipo de los dos círculos entrelazados. Para 1977, Bank Americard se convirtió en la tarjeta Visa, conservando sus distintivos de banda azul, blanca y dorada.

Actualmente la tarjeta de crédito Banamex, representa un medio de pago con el cual el usuario puede adquirir bienes y servicios que ofrecen los Negocios Afiliados en la República Mexicana y el extranjero; debiendo presentarla y firmar un pagaré por el valor de su compra. También puede disponer de efectivo en Sucursales y Centros Financieros Banamex así como en las Cajas Permanentes, hasta cierto límite de crédito autorizado, o

bien en fondos previamente depositados.

Si la tarjeta es de carácter internacional su uso se extiende a los Negocios Afiliados a nivel mundial, e incluso disponer de efectivo en cajeros automáticos, en ambos casos afiliados a Visa o Master Card.

Banamex cuenta con varios tipos de tarjetas: Clásica y Oro, Tarjetas de Afinidad, Marca Compartida, Bancos Asociados y Marca Privada.

Las tarjetas de afinidad pueden ser de uso nacional e internacional, con el respaldo de Visa y Master Card, e incluso tienen la opción al servicio de Cuenta Maestra. Se caracterizan por identificar al usuario como miembro de una asociación civil o agrupación que no persiga fines lucrativos, como pueden ser hospitales, universidades, etc. Además, Banamex les otorga a las instituciones una parte de la facturación para ser utilizada para fines altruistas, capacitación o investigaciones, contribuyendo de esta manera a su desarrollo.

Las tarjetas de Marca Compartida son tarjetas que emite Banamex en asociación con negocios comerciales. Integran las ventajas de las tarjetas bancarias y los atributos adicionales que ofrece el negocio coemisor a su clientela frecuente. Además, el hecho de ser una Tarjeta Banamex permite utilizarla como cualquier otra tarjeta bancaria en Negocios Afiliados.

Las de Marca Compartida son Tarjetas de Crédito operadas dentro del sistema de Tarjetas Banamex con las cuales el usuario puede adquirir

bienes y servicios que ofrece la empresa coemisora, la cual cuenta con una marca comercial e imágenes reconocidas. Los pagos, reposiciones y reportes de robo o extravío pueden realizarse en las Sucursales Banamex.

En lo que respecta a las Tarjetas de Marca Privada son exclusivamente para adquirir productos que ofrezca la cadena comercial respectiva.

Por último las Tarjetas de Bancos Asociados, son aquellas que emiten este tipo de bancos y son operadas por Banamex. Con este tipo de tarjeta, el cliente puede adquirir bienes y servicios en Negocios Afiliados a Banamex, así como disponer de efectivo en Sucursales, Centros Financieros y Cajas permanentes, e incluso en oficinas del banco asociado correspondiente. En lo referente a reposiciones, aclaraciones, saldos y pagos, los diferentes Bancos Asociados fungen como intermediarios.

En la actualidad los beneficios generales que ofrecen las Tarjetas Banamex, dependiendo si son nacionales o internacionales, son los de adquirir bienes y servicios en más de 120 mil Negocios Afiliados en la República Mexicana y más de 10 millones en el extranjero. Además, son aceptadas en cerca de 700 sucursales y mil 500 Cajas Permanentes Banamex así como en 260 mil sucursales en el extranjero y mas de 100 mil cajeros automáticos afiliados a Visa y Master Card.

II.2 DETECCION DE LA PROBLEMÁTICA

Las instituciones financieras están usando la tecnología como factor importante para aumentar la eficiencia, tanto en el desarrollo de productos y servicios como para automatizar los procesos que realizan internamente para soportar sus actividades, su limitante está acotada por restricciones que marque la tecnología de punta. La mejor utilización de los sistemas disponibles y su actualización no sólo permiten identificar oportunidades para reducir costos, sino también para aumentar la calidad de la prestación de servicios (sencillez, rapidez y exactitud) y para crear otros. El buen uso de la tecnología facilita el manejo y la disponibilidad de la información, que es vital para los Bancos, así como detectar riesgos y oportunidades y mejorar el control así como la rentabilidad de cada operación.

El aumento de los costos de operación y los volúmenes de información, así como la necesidad de contar con mejores controles administrativos y servicios sin restricción de fronteras e innovación de productos, son una oportunidad de negocio para los Bancos.

MEDIOS DE PAGO EN PLÁSTICO

Las Instituciones Bancarias reflejan un gran volumen de información relativa a clientes de tarjetas de crédito, por lo que se han tenido que enfrentar a grandes cambios en los servicios que se prestan al cliente, los cuales se han ido dando en forma paulatina, siendo éstos:

- Ventas en negocios (centros comerciales, restaurantes, etc.) y disposiciones de efectivo y pagos (sucursales bancarias).

- Cajas permanentes.

- Banco en su casa.

Lo anterior es con el fin de actualizar los servicios de un producto, en relación a las necesidades del tarjeta-habiente.

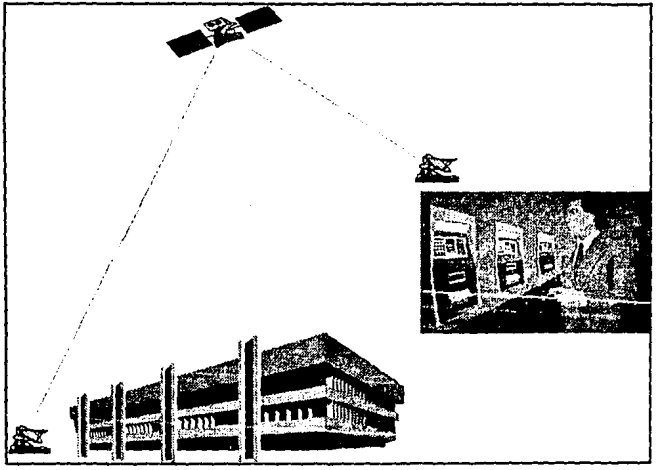


Fig. 2.2.1 Las instituciones financieras utilizan ahora tecnología de punta.

De esta manera, podemos ver que existen equipos para soportar los procesos en lote (emitir estados de cuenta de clientes, avisos, etc.); otros equipos soportan el ambiente transaccional (cajeros permanentes, dispositivos de autorizaciones, autorizaciones de venta, etc.) y redes (de área extendida e internacionales) para ofrecer servicios especiales a los clientes en su oficina o el hogar y para procesar información interna de todas las áreas del Banco. Asimismo, las instituciones bancarias en la actualidad ofrecen una amplia gama de productos y servicios que son soportados con la tecnología más avanzada, como respuesta a las necesidades bancarias y financieras, que han requerido los clientes y las mismas instituciones.

Dentro de los productos bancarios, las tarjetas de crédito se consideran actualmente uno de los más importantes, debido a su aceptación en el mercado así como la rentabilidad que presentan.

Teniendo actualmente en forma global las Instituciones Bancarias los siguientes tipos de tarjetas, que son consideradas como medios de pago en plástico (Fig. 2.2.2):

Tarjetas de débito

Son un instrumento de inversión en moneda nacional, de uso nacional o internacional al amparo de un depósito que se accesa mediante una tarjeta

de plástico, permitiendo al cliente contar con total liquidez, rendimiento y seguridad absoluta.

Tarjetas de crédito

Es un medio en plástico que a través de una línea de crédito revolvente y de acuerdo a su cobertura, es utilizado en la adquisición de bienes y servicios en todos los establecimientos afiliados al sistema de Tarjetas Bancarias en la República Mexicana y en el extranjero (en su caso). Siendo clasificadas según su validez de uso en Nacionales e Internacionales.

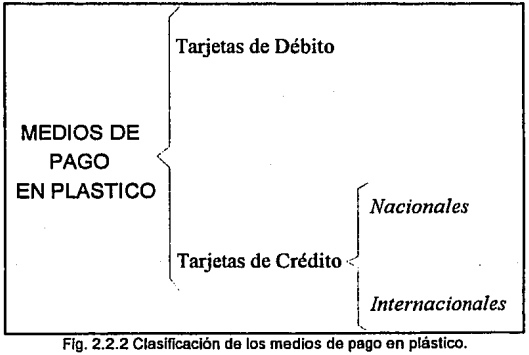


Fig. 2.2.2 Clasificación de los medios de pago en plástico.

CARACTERÍSTICAS OPERATIVAS

Titular

Es la persona física, la cual utiliza el crédito o inversión a cuyo nombre se expide la tarjeta.

Vigencia

La tarjeta se expide hasta por dos años y es prorrogada automáticamente por períodos iguales y sucesivos, en tanto no existan causales para su cancelación por parte del banco, o mientras el usuario no lo solicite.

Como Instrumento de crédito

La tarjeta de crédito, es un instrumento para disponer de una línea de crédito revolvente en cuenta corriente, es decir, que conforme se vayan liquidando en la cuenta los importes cargados en ella, se libera el margen de crédito y es posible seguir efectuando nuevas compras y disposiciones en efectivo dentro del límite de crédito otorgado.

Límite de crédito

El límite de crédito se fija con base en los ingresos y la capacidad de pago comprobable del solicitante. Dentro del límite de crédito quedan comprendidos los importes de los intereses, comisiones, cargos y demás gastos que se originen con motivo del uso de la tarjeta.

Incremento a la línea de crédito otorgada

Los incrementos pueden ser automáticos y generales o a solicitud del usuario. En el segundo caso, es necesario demostrar que se cuenta con la capacidad de pago requerida para el incremento solicitado.

Opciones de pago

El tarjeta-habiente puede liquidar su saldo en cualquiera de las siguientes formas:

1. Pagar la totalidad de su saldo antes de la fecha de corte sin cargo alguno.

2. Pagar la totalidad de su saldo a más tardar en la fecha límite de pago, en cuyo caso pagará una comisión sobre los saldos promedio diarios de las compras y disposiciones del mes anterior.

3. Pagar en abonos sucesivos mensuales no menores al 10% del capital más los intereses y gastos que generen en el mes, excepto si el banco cancela el crédito, en cuyo caso deberá cubrir el importe total del saldo pendiente de pago.

Como instrumento de inversión

En esta modalidad las tarjetas tradicionales, permiten mantener como saldo a favor cualquier cantidad de dinero y ganar intereses.

Esto marca una diferencia significativa con los depósitos a plazo fijo en donde se requiere de un mínimo de inversión. El dinero que integra el saldo a favor del cliente, participa en un fondo de inversión de valores, que permite liquidez y produce rendimientos, de acuerdo con las condiciones del mercado de dinero.

Estos rendimientos se calculan diariamente y los que corresponden a cada cliente, se le abonan mensualmente en su cuenta en la fecha de corte que le corresponda. El límite de crédito autorizado no se modifica bajo la modalidad de inversión, sin embargo, cuando se utiliza como tal, todas las transacciones que se realicen se aplican o debitan contra los recursos que integran el saldo a favor de la cuenta, y agotado éste, contra la línea de crédito otorgada.

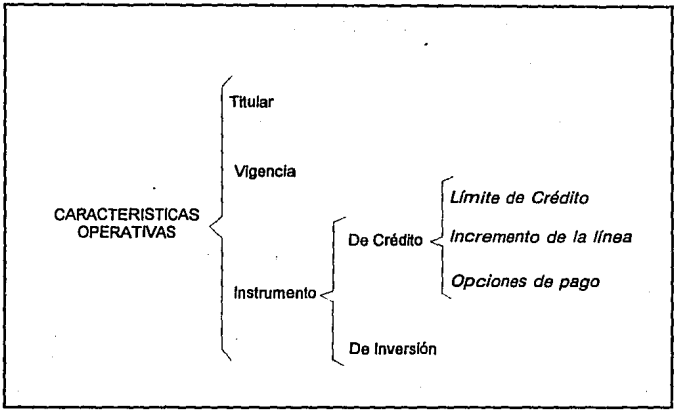


Fig. 2.2.3 Características Operativas de las Tarjetas de Crédito

TARJETAS DE CRÉDITO BANAMEX

Tradicional Nacional

Esta conformado por personas que dependen económicamente de una ocupación fija y cuyo diferencial entre sus ingresos y gastos fijos, lo pueden destinar a la solvencia de crédito o a una forma de inversión mediante el uso de la Tarjeta Tradicional.

Los criterios usados para segmentar el mercado son los siguientes:

a) **Nivel socioeconómico.** Esta formado por empleados, obreros, técnicos, comerciantes y rentistas, que dependen de sus ingresos como fuente de recursos.

b) **Nivel de ingresos.** Su nivel de ingresos total mensual fluctúa entre 2 y 15 veces el salario mínimo.

Tradicional Nacional Plus

Esta conformado por clientes que por la gran importancia de su relación con el banco y por su alto nivel económico son merecedores de una atención preferencial y personalizada por parte de la Institución.

Los criterios usados para segmentar el mercado son los siguientes:

a) **Nivel socioeconómico.** Esta formado por clientes de la Institución con ingresos elevados, con una alta responsabilidad financiera y que poseen un nivel superior de status social.

b) **Nivel de relación con el banco.** Se integra por personas que mantienen relaciones financieras constantes y dinámicas con la Institución Bancaria.

c) **Hábitos de consumo.** Está compuesto por clientes de la Institución que generalmente realizan consumos en negocios exclusivos y especializados de primer nivel.

Tradicional Internacional

Es un sistema que permite realizar transacciones tanto dentro del territorio nacional, así como en más de 160 países del mundo.

Esta conformado por personas que dependen económicamente de una ocupación fija y cuyo diferencial entre sus ingresos y gastos fijos, lo pueden destinar a la solvencia de crédito o a una forma de inversión.

Los criterios usados para segmentar el mercado son los siguientes:

a) **Nivel socioeconómico.** Esta formado por profesionistas, ejecutivos, empresarios y rentistas que perciben ingresos fijos mensuales y cuentan con un patrimonio.

b) **Nivel de ingresos.** Su nivel de ingresos total mensual fluctúa entre 16 y 30 veces el salario mínimo.

c) **Hábitos de consumo.** Lo integran clientes que generalmente realizan consumos en negocios "exclusivos" y especializados dentro y fuera del país.

Tradicional Internacional Plus

Es un sistema que permite realizar transacciones tanto dentro del territorio nacional, así como en más de 160 países del mundo.

Esta conformado por clientes que por la gran importancia de su relación con el banco y por su alto nivel socioeconómico son merecedores de una atención preferencial y personalizada dentro del territorio nacional y el resto de los países del mundo (fig. 2.2.4).

Los criterios usados para segmentar el mercado son los siguientes:

a) **Nivel socioeconómico.** Esta formado por clientes con ingresos elevados, con una alta responsabilidad financiera y poseen un nivel superior de status social.

b) **Nivel de relación con el banco.** Se integra por personas que mantienen relaciones financieras constantes y dinámicas con la Institución Bancaria.

c) **Hábitos de consumo.** Lo integran clientes de la Institución que generalmente realizan consumos en negocios exclusivos y especializados de primer nivel dentro y fuera del país.

Tarjetas Ejecutivas

Es un sistema de financiamiento que permite a las empresas financiar los gastos corporativos de sus ejecutivos, efectuado en los establecimientos afiliados al sistema de tarjetas de crédito.

Siendo conformada por todas aquellas empresas cuyos ejecutivos efectúan gastos de representación y ventas por cuenta de ellas.

Los criterios usados para segmentar el mercado son los siguientes:

a) Es ejecutivo de alto nivel.

b) Efectúa viajes frecuentes por cuenta de la empresa.

c) Desarrolla funciones operativas de constante relación externa a nivel privado y gubernamental.

d) Tiene necesidad de cubrir sus gastos de representación en el momento en que los efectúa.

e) Adquiere bienes y servicios relacionados con su actividad empresarial, tales como: boletos de avión, hospedaje, renta de autos, atención a cliente, etc.

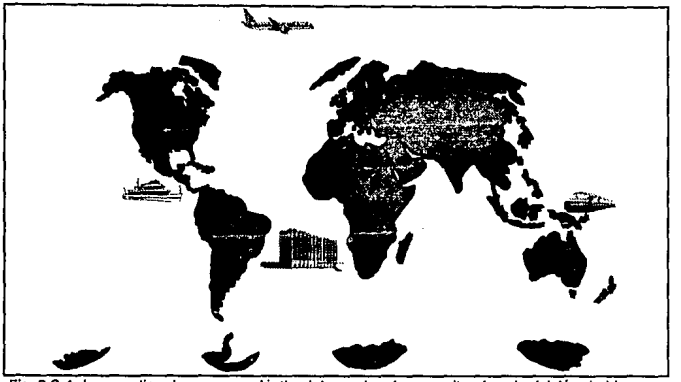


Fig. 2.2.4 Los medios de pago en plástico internacionales permiten la adquisición de bienes y servicios en todo el mundo.

Tarjetas Empresariales

Es un sistema de financiamiento que a través de una línea de crédito revolvente permite a las empresas financiar los gastos corporativos de sus ejecutivos, efectuados en los establecimientos afiliados al Sistema de Tarjetas de Crédito.

Tarjetas de Afinidad

Son tarjetas de crédito revolvente en cuenta corriente, con los mismos servicios y ventajas de las tarjetas nacionales e internacionales, con la diferencia de que tienen un atributo adicional, el de identificar al usuario

como miembro de la asociación o agrupación a la que pertenecen.

Los criterios usados para segmentar el mercado son los siguientes:

Asociaciones y agrupaciones formalmente constituidas, de reconocido prestigio, cuyos miembros tienen actividades comunes y fines específicos tales como: asociaciones civiles, profesionales, instituciones educativas, clubes deportivos, etc.

Tarjetas de Marca Privada

Es un sistema de financiamiento que a través de un crédito revolvente en cuenta corriente se facilita la adquisición de productos que comercializan las empresas de las que se compone una cadena comercial.

Los criterios usados para segmentar el mercado son los siguientes:

Estas tarjetas están dirigidas a personas físicas con ingresos de 5 salarios mínimos mensuales vigentes de la plaza de que se trate, interesadas en obtener una línea de crédito para solventar las compras de los productos que comercializan las empresas (con negociación con la Institución Bancaria).

Tarjetas Patrimoniales

Es un instrumento de crédito revolvente en cuenta corriente, con liquidación automática al término de cada ciclo de corte, que la Banca Patrimonial hace en nombre del cliente con cargo a los recursos generados por la venta de sus valores

Los criterios usados para segmentar el mercado son los siguientes:

Clientes actuales de la Banca Patrimonial que mantienen relaciones constantes, dinámicas y de alta responsabilidad financiera, y que se caracterizan por su sólida posición económica y social.

Cabe mencionar, que esta tarjeta es de uso internacional.

Tarjetas de Bancos Asociados

Son tarjetas emitidas a nombre de personas físicas por los Bancos Asociados y operadas por una Institución Bancaria con infraestructura, con las cuales deben ser aceptadas en el sistema de negocios afiliados de dicha Institución, así como disposición de efectivo y pagos en sus sucursales y cajas permanentes.

Estos bancos tienen sus propias tarjetas nacionales, internacionales, empresariales, afinidad y de marca privada.

Asimismo, las carteras de cobro de las tarjetas mencionadas, corresponden al Banco Asociado respectivo. En consecuencia, estas tarjetas quedan excluidas del Sistema de Desempeño de Tarjetas de Crédito.

Por lo anterior, una Institución Bancaria que actualmente contempla más de 80 tarjetas en el mercado, sin considerar los nuevos productos e innovaciones, viene a representar la generación de grandes volúmenes de información, implicando operar transacciones en un rango de más de 3'000,000 de clientes bajo una infraestructura tecnológica establecida, así como operativa y de servicios.

De la misma forma, implica el desarrollo de una fuerte y dinámica mercadotecnia que defina la creación y el desarrollo de medios de pago, materializables en plástico, papel y electrónica.

Presentando en forma aunada, grandes retos a las Instituciones Bancarias, debido a la tecnología, servicio, operación y mercadotecnia de las mismas; por su naturaleza y aplicación al mercado tienen políticas y parámetros muy independientes entre sí, siendo algunas de ellas:

- Características de apertura y autorización
- Otorgamiento y/o aumento de límite de crédito
- Aplicación de intereses, comisiones, etc.

En la figura 2.2.5 se muestra una tabla en la que se indican las categorías de tarjetas de crédito y en la 2.2.6 las tarjetas de crédito tradicionales.

TARJETAS DE CRÉDITO BANAMEX

- Tradicional Nacional
- Tradicional Nacional Plus
- Tradicional Internacional
- Tradicional Internacional Plus
- Ejecutivas
- Empresariales
- De afinidad
- De Marca Privada
- Patrimoniales
- De Bancos Asociados

Fig. 2.2.5 Categorías de Tarjetas de Crédito en Banamex

Tarjeta	Nivel Socio-económico	Nivel de Ingresos	Relaciones con el banco	Hábitos de consumo
Tradicional Nacional	Empleado, Obreros, Técnicos, Comerciantes y Rentistas	Mensual entre 2-15 salarios mínimos		
Tradicional Nacional Plus	Clientes de la Institución con ingresos elevados, alta responsabilidad financiera y un nivel superior de status social		Constantes y dinámicas	Consumos en negocios exclusivo dentro y fuera del país
Tradicional Internacional	Profesionistas, Ejecutivos, Empresarios y Rentistas que perciben ingresos fijos mensuales y cuentan con un patrimonio	Mensual entre 16-30 salarios mínimos		Consumos en negocios exclusivo dentro y fuera del país
Tradicional Internacional Plus	Elevado con una alta responsabilidad financiera y nivel superior de status social		Relaciones financieras constantes y dinámicas el banco	Consumos en negocios exclusivo dentro y fuera del país

Fig. 2.2.6 Tarjetas de Crédito Tradicionales de Banamex

NECESIDAD DE AUDITORÍA AL TOTAL DE LA INFORMACIÓN

Actualmente, el resultado de las auditorías a los productos bancarios tiene que ser un apoyo de utilidad al directivo, no importando los procesos automatizados en las evaluaciones que tiene que realizar.

Así mismo, se debe considerar que la tecnología es y ha sido un factor determinante en el desarrollo y crecimiento de los bancos, así como enfrentar la competencia, por lo que se debe de tomar la misma actitud, y tener en mente que la tecnología es una herramienta de apoyo para la realización de los trabajos de auditoría.

Consideranto los puntos anteriores, se debe emitir una opinión que apoye al directivo en la toma de decisiones que tenga que efectuar, proporcionándole los resultados precisos para que la institución aumente su productividad y el nivel de riesgo en el que vive, se vea reducido.

Por otra parte, no se debe de olvidar que existen los auditores en informática que tienen su trabajo muy específico, pero que no pueden estar a disposición de todos los auditores tradicionales para apoyarlos en sus evaluaciones. El trabajo de los auditores en informática esta muy bien definido, ya que también tienen que apoyar a los niveles directivos que tienen bajo su responsabilidad el diseño, desarrollo e implementación de sistemas automatizados, así como a las áreas de comunicaciones y teleproceso.

II.2.1 PUNTOS CRÍTICOS

La cartera vencida de la banca mexicana ascendió en 1993 en un 91.5% con respecto a 1992, lo cuál implica para Banamex una cartera vencida de más de 33 mil millones de nuevos pesos. En forma paralela el crecimiento de los riesgos crediticios fueron de un 140 por ciento.

Lo anterior ha sido desde siempre una gran preocupación de los Directivos bancarios Fig. 2.2.7. El incremento incontrolable y acelerado que se ha dado en la cartera vencida, es en gran parte una repercusión de la economía mexicana de esta época.

Estrategias de los Directivos de los Bancos para encontrar la fórmula de controlar el problema de la Cartera Vencida

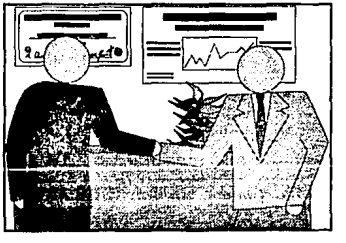


Fig. 2.2.7 Mantener a los Directivos informados de la situación que vive el banco es vital.

Por ello, es necesario mantener informados a los Directivos sobre las operaciones que se encuentren sobre los rangos y políticas fijadas y que están dando como consecuencia una cartera vencida elevada, así como informes sobre la redocumentación de cuentas y créditos que se encuentren en riesgo de originar un quebranto.

CARTERA VENCIDA

Representa una gran responsabilidad canalizar parte del dinero confiado por depositantes en las Instituciones Bancarias, para financiar preferentemente el desarrollo de los sectores mas importantes en la economía del País, así como el cuidado en el otorgamiento y administración de los créditos, en prevención de congelamiento ó perdida de recursos de inversión.

GESTIONES DE COBRANZA

Debido al número tan elevado de deudores que tienen la costumbre de liquidar con atraso sus pagos mensuales, se debe cuidar que las gestiones de cobro se efectúen con la adecuada oportunidad y con la frecuencia necesaria, implicando con ello restringir el uso de la tarjeta al usuario en base a su problemática de cobro, o en su defecto la cancelación de la misma.

Cuentas con mensualidades vencidas

De acuerdo con el número de mensualidades vencidas, los sistemas de información generan los documentos que se indican, con la periodicidad que en cada caso se señala:

- Con una mensualidad:
  - > En la fecha de corte: estado de cuenta y listado; y se le asigna la clave "D" (atraso).
  - > 15 días después del corte: listado y tarjetón de cobro.
- Con dos mensualidades:
  - > En la fecha de corte: estado de cuenta, listado y tarjetón de cobro; y se le asigna la clave "D" (atraso) y además, se cancela el crédito disponible.
- Con tres mensualidades:
  - > En la fecha de corte: estado de cuenta, listado, tarjetón de cobro y carta de cobranza; y se le asigna la clave "P" (problemas de cobro), se cancela y boletina la tarjeta.
- Con cuatro ó más mensualidades:
  - > En la fecha de corte: estado de cuenta, listado, tarjetón de cobro .

Cuentas con sobregiros

De acuerdo con el monto del sobregiro en exceso del límite de crédito, se genera lo siguiente:

- Al presentarse el sobregiro:
  > Si el exceso es de más del 15% y hasta el 35%: listado y tarjetón de cobro; y se le asigna la clave "O" (sobregiro simple).

  > Si el exceso es de más del 35% y hasta 50%: igual que en el caso anterior, incluyendo además carta de cobro para el cliente.

- Quince días después del sobregiro o en la fecha de corte, lo que ocurra primero:
  > Si la cuenta no se regularizó: listado, tarjetón de cobro y carta al cliente; y se le asigna la clave "P" (problemas de cobro), por lo que se cancela y boletina la tarjeta.

  > Si el exceso es mayor al 50%: igual que el caso anterior.

Cuentas con sobregiro y mensualidades vencidas

De acuerdo con el número de mensualidades vencidas, se genera lo siguiente:

- Con una mensualidad:
  1. En la fecha de corte:

     Si el exceso es de más del 15% y hasta el 35%: estado de cuenta, listado y tarjetón de cobro; y se le asigna la clave "D" (atraso).

     Si el exceso es de más de 35%: estado de cuenta, listado y tarjetón de cobro; se le asigna la clave "P" (problemas de cobro), se cancela y boletina la tarjeta.

  2. Quince (15) días después del corte:

     Si la cuenta no se regularizó: se le asigna la clave "P" (problemas de cobro), se cancela y boletina la tarjeta.

- Con dos mensualidades:

  En la fecha de corte, si el exceso sobre el límite de crédito es de más del 15%: estado de cuenta, listado y tarjetón de cobro; y se le asigna la clave "P" (problemas de cobro), se cancela y boletina la tarjeta.

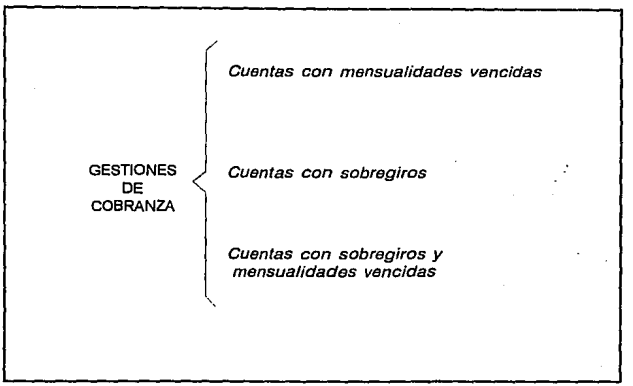


Fig. 2.2.8 Gestiones de cobranza en el área de auditoria

TIPOS DE GESTIONES DE COBRANZA.

Cuando la cartera no fue pagada rigurosamente a su vencimiento y por el conocimiento de su antigüedad, probabilidades o problemas de recuperación (Fig. 2.2.9), la podemos clasificar en:

Cartera vencida transitoria

Son adeudos sobre los que se tiene la seguridad o fundadas esperanzas de que serán liquidados dentro de un mes a su ingreso a cartera vencida y/o sobregiro de 25% sobre el límite de crédito.

Cartera vencida administrativa

Son adeudos con 2 a 3 meses vencidos y/o sobregiro del más del 25% sobre el límite de crédito.

Cartera vencida pre-jurídica

Son adeudos que no son recuperables en forma administrativa y requieren ser cobrados en forma jurídica, por lo que se negocia con el usuario el pago del adeudo o en caso contrario se tramita el registro por vía jurídica.

Cartera vencida jurídica

Esta constituida por el importe de adeudos totales, tanto el pago de los meses vencidos (saldo vencido) como el adeudo vigente; con el cual se entrega documentación al abogado, para ejercer derechos de cobro por vía juridica.

Cartera vencida castigada

Dicha cartera es la correspondiente a la nula recuperación del adeudo, en donde se hayan agotado los recursos legales y se procede a solicitar las reservas correspondientes, con el fin de recuperar en caso de ser aprobado por la Comisión Nacional Bancaria, la recuperación del capital vencido, sin considerar los intereses.

Cartera redocumentada

Esta cartera es resultado de la negociación con el tarjeta-habiente, quién haciendo promesa de pago se redocumenta su adeudo en mensualidades fijas, pero para ello, el tarjeta-habiente debe de presentar bienes hipotecarios con lo que ampare su adeudo y promesa de pago.

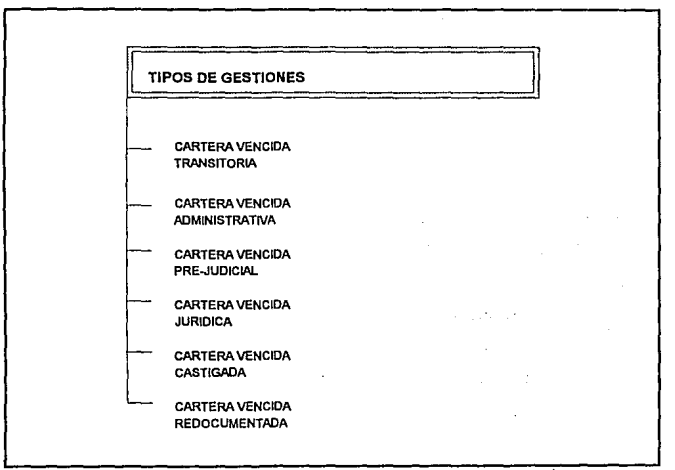


Figura 2.2.9 Tipos de Gestiones de Cobranza

ORIGEN DE LA CARTERA VENCIDA.

Las causas que originan la cartera vencida se genera a través de diversas fuentes, adhesibles; las cuales son presentadas en un otorgamiento de crédito bancario, como son:

1. **Causas fortuitas o situaciones imprevisibles en el estudio de crédito del proyecto de inversión.**

2. **Causas imputables al acreditado, tales como:**

   > Insolvencia económica.

   > Insuficiente conocimiento de su ramo o actividad.

   > Introducción de nuevos productos o servicios, sin el pleno conocimiento del mercado y tendencias.

   > Incursión en negocios desconocidos, riesgosos o especulativos.

   > Negligencia o descuido en la administración de negocios o la actividad de los mismos.

   > Deficiente control de recursos económicos.

   > Excesivo apalancamiento financiero.

> Utilización de créditos en fines distintos a los convenidos.

> Mala fe, Etc.

3. Por causas imputables al operador del crédito, entre otras:

> Desconocimiento o falta de observación a políticas y normas crediticias.

> Deficiente análisis del proyecto de inversión, de los aspectos cualitativos y cuantitativos del cliente y obligados y de su verdadera capacidad de pago.

> Créditos inadecuados a las verdaderas necesidades del cliente.

> Aceptación de garantías riesgosas o de difícil realización; deficientemente evaluadas o aseguradas; mal constituidas o registradas fuera de tiempo.

> Falta de vigilancia en el destino del crédito, administración y marcha del negocio acreditado y del cumplimiento de los términos y condiciones convenidas.

> Insuficiente diligencia en el cobro de créditos.

> No actuar jurídicamente a tiempo, (dentro de los plazos legales), así como falta de documentación requerida; con la consecuente pérdida de las acciones legales en contra de los diferentes obligados.

> Complacencia en la renovación de operaciones, con la pérdida de las garantías originales o aceptando la redocumentación a nombre de otros deudores, con menos posibilidad de pago.

> Etc.

INDEPENDENCIA EN LA OBTENCIÓN DE INFORMACIÓN

Las áreas involucradas en la operación y administración, así como las gestiones de cobranza dependiendo de sus funciones tiene su propia información. Cada una labora en forma independiente, asimismo cada área y departamento o proceso involucrado reporta su propia información gerencial.

Debido a la función del área de Auditoría Interna de una Institución Bancaria, debe obtener información del universo involucrado. Al mismo tiempo, tiene que emitir sus informes de manera independientemente, debiendo ser éstos razonables e imparciales de modo que la información reflejada a los Directivos, auxilie verdaderamente y con precisión en la toma de decisiones.

Por lo anterior, es necesario implementar una herramienta automatizada para el control y análisis del comportamiento de la cartera vencida de tarjetas de crédito, con base en los lineamientos internos de la Institución, con el fin de optimizar y agilizar acciones correctivas y/o preventivas en el otorgamiento y seguimiento de las líneas de crédito.

Del resultado de dicho análisis, en el cual se reflejan desviaciones de control así como la falta de razonabilidad e integridad de la información, resaltan puntos sensibles de la operación que hay que enmendar y/o corregir, en forma inmediata, por ello la necesidad de manipular en forma

total la información, para abarcar en su gran mayoría una revisión cuantitativa y cualitativa, en donde se refleje realmente una evaluación y no sólo una revisión empírica y selectiva de muestras, que no abarca en múltiples circunstancias el mínimo grado de realidad la situación de operación de las áreas involucradas, evitando con ello, la filtración de quebrantos y/o fraudes así como la falta de control dentro de las mismas.

II.2.2 CONSIDERACIONES DE CONTROL INTERNO

La cartera vencida representa el conjunto de créditos y prestamos no liquidados con oportunidad, constituyendo un activo inmovilizado.

Asimismo, la falta de evaluación de los procedimientos que se siguen en los diversos departamentos de una institución bancaria, en funcionalidad, suficiencia de medidas de control interno así como el cumplimiento del objetivo de rentabilidad, implica desatención y riesgos de las siguientes funciones básicas de control interno:

- Prevención de riesgos en los sistemas reglamentados.
- Detección y análisis de las causas que dieron origen a un riesgo materializado (fraudes, robos, abusos de confianza).
- Control estadístico de situaciones especiales (fraudes, abusos de confianza, etc.)
- Promover acciones tendientes a minimizar la posibilidad de su reincidencia.
- Opinar sobre la seguridad y funcionalidad de los sistemas implementados en ciertos departamentos esenciales de operación.

- Proponer mejoras a los sistemas que contribuyan al fortalecimiento del control interno y a la disminución de la exposición al riesgo.

- Generar información suficiente y oportuna sobre la suficiencia y ausencia del control interno incorporado en los diferentes ordenamientos reglamentados.

- Atención especifica de algún requerimiento especial que sea solicitado por algún miembro de dirección sobre su área de responsabilidad.

II.3 IDENTIFICACIÓN Y ANÁLISIS DE LOS REQUERIMIENTOS DEL ÁREA DE AUDITORÍA

Las Instituciones Bancarias, a la vanguardia de los servicios bancarios en el país, fundamentan su acciones en un desarrollado proceso de planeación, esto le permite participar con oportunidad en el dinámico mundo de las finanzas y de los servicios al cliente.

Como respuesta a las políticas institucionales de modernización permanente, se requiere la implementación de un sistema de información gerencial que contribuya a facilitar las labores relacionadas con la toma de decisiones de tipo estratégico.

El propósito de automatizar la obtención de información gerencial en una área de Auditoria, es con el fin de aprovechar integralmente la información de un sistema determinado, para así lograr una mayor eficiencia en la consecución de los objetivos institucionales; permitiendo determinar desviaciones de control en los lineamientos internos.

Asimismo, nos permite obtener parámetros básicos y necesarios para el control y análisis de políticas sobre el comportamiento de cada uno de los productos de las tarjetas de crédito, lo que facilita implementar diversas alternativas tácticas que se encaminen a la agilización de acciones correctivas y/o preventivas en el otorgamiento y seguimiento de las líneas de crédito.

FINALIDAD DE UN SISTEMA AUTOMATIZADO

La información de los sistemas de tarjetas de crédito, registran las operaciones diarias efectuadas en la Banca, tales como disposiciones de efectivo y compras, así como el registro de cálculos monetarios (intereses, capital, saldos mínimos por pagar, etc.).

Y en forma aunada, registra políticas y parámetros de operación, correspondientes a cada uno de los productos de un sistema de tarjetas de crédito, como son límites de crédito otorgados por apertura, límites otorgados por comportamiento, situación de la cuenta, ciclos de corte, etc.

Por lo anterior, la existencia de un sistema automatizado, representa la evaluación interna de información derivada de procesos operativos existentes, dicho sistema sería capaz de proporcionar información necesaria para una eficaz toma de decisiones a nivel estratégico y táctico dentro del desarrollo de las actividades de auditores internos, propiciando un análisis integral.

OBJETIVOS DEL ÁREA DE AUDITORÍA DE TARJETAS DE CRÉDITO

Con base en la necesidad de revisar en forma automatizada la cartera vencida de tarjetas de crédito, se tienen los siguientes objetivos específicos del área:

- Vigilar el comportamiento de la cartera vencida, en forma global así como cada segmento y/o producto de tarjetas de crédito, controlando con ello, excesivos incrementos sorpresivos de la cartera.

- Analizar la cartera vencida, a fin de evaluar el comportamiento y detectar anomalías de control para tomar las medidas correctivas necesarias.

- Elaborar estrategias que permitan mantener dentro de los estándares establecidos los niveles de la cartera vencida.

- Diseñar con base a la información estadística, en coordinación con procesos operativos, las políticas y parámetros para el otorgamiento del crédito.

- Garantizar la correcta aplicación de los procesos en el otorgamiento de crédito, la gestión de cobranza y los proyectos de castigos, a fin de operar con oportunidad, eficiencia y rentabilidad.

- Vigilar el cumplimiento de los despachos jurídicos, a fin de que los clientes y prospectos, tengan una imagen de eficiencia por parte de la Institución.

- Analizar el control y seguimiento de políticas y parámetros establecidos para los productos.

- Vigilar la rentabilidad de los productos, permitiendo proponer estrategias para el incremento de la misma.

- Garantizar que se lleve a cabo las funciones de cobranza, a fin de lograr la optimización del área de Cobranza, lo cuál se viene reflejando en el decremento o incremento de la cartera vencida.

- Por ello, la necesidad implícita de implementar una revisión integral de la cartera vencida, la cuál sea una herramienta de apoyo que refleje una situación general cuantitativa y cualitativa, con indicadores que reflejen las desviaciones de control, razonabilidad e integridad de información, las cuales podrán ser detectadas en forma periódica (monitoreo) e integral, a través de un sistema automatizado de información.

REQUERIMIENTOS DE AUDITORIA A IMPLEMENTAR EN UN SISTEMA AUTOMATIZADO

Los requerimientos de una área de auditoría, por lo que respecta a tarjetas de crédito en una fase de monitoreo, análisis y seguimiento de cartera vencida, para la evaluación de la misma, consiste en:

Análisis de aperturas y manejo de crédito

Reflejar información que contemple la situación cuantitativa y cualitativa de cuentas de apertura reciente en los últimos 12 meses y que presentan mensualidades vencidas, de las cuales se requiere identificar los siguientes parámetros:

- **Calidad de crédito.**
  - > Cuentas de más alto riesgo debido al número de meses vencidos así como saldo vencido,
  - > Cuentas en que no hayan realizado ningún pago desde su inicio de apertura.

- **Asignación de límites de crédito.**
  - > Límites otorgados fuera de las políticas y parámetros de cada una de las tarjetas de crédito.

- **Manejo de cuenta.**
  - > Saturación del crédito otorgado en forma inmediata a su apertura.
  - > Saldo total superior al crédito otorgado.

- **Patrones de fraude.**
  - > Posibilidad de poder identificar cuentas fraudulentas en base a patrones presentados en la operación, considerando la información registrada en el sistema.

Identificar y Controlar sobregiros Morosos

Reflejar información que contemple cuentas de clientes con sobregiros con riesgos materializables, debido a la situación de no recuperar el adeudo y en forma conjunta el sobregiro (saldo deudor superior al crédito), de las cuales se requiere identificar los siguientes puntos:

- **Saldos**
  - > Saldos totales muy superiores al límite de crédito existente.
  - > Sobregiros morosos que no hayan realizado ningún pago.

- **Recuperación**
  - > Importe recuperado en relación al sobregiro.

- **Patrones de Fraude**
  - > Posibilidad de poder identificar cuentas fraudulentas en base a patrones presentados en la operación, considerando la información registrada en el sistema.

Controlar la cartera vencida de tarjetas de crédito

Reflejar información que contemple una visión de la situación de la cartera vencida, en relación a todas sus fases de operación y meses vencidos, así como situaciones críticas para cada una de ellas, donde se requiere identificar el siguiente análisis:

Cartera de aperturas

Status de la cartera de apertura reciente y que presentan mensualidades vencidas, en base a índices de cartera por producto a nivel estatal.

Cartera administrativa

Status de la cartera administrativa (cuentas con 2 a 3 meses vencidos), en base a índices de cartera por producto a nivel nacional (por estados).

Cartera Pre-Jurídica

Status de la cartera pre-jurídica (cuentas con más de 4 ó 5 meses vencidos y la cuenta no refleja la asignación de un abogado respectivo), en base a las acciones de recuperación del área de cobranzas, antes de pasarlo a la cartera de abogados, reflejando la cartera por producto a nivel estatal.

Cartera Jurídica

Status de la cartera jurídica (cuentas con más de 5 meses vencidos y la cuenta refleja la asignación de un abogado respectivo), en base a las acciones de recuperación de los abogados de la cartera por producto a nivel estatal.

Cartera de Castigos

Status de la cartera de Castigos (cuentas consideras en los proyectos de castigos), en base a importe y meses vencidos así como el status de cobranza (irrecuperable del saldo vencido), reflejando la cartera por producto a nivel estatal .

Cartera de redocumentación

Status de la cartera de redocumentación, (cuentas en que se renegocia su adeudo para su pago, dichas cuentas no deben pasar más de 3 meses vencidos), en base a importe y meses vencidos, reflejando la cartera por producto a nivel estatal .

Controlar la cartera preventiva de tarjetas de crédito

Reflejar información que presente cuentas que debido al dolo presentado contra la Institución, presenta adeudos en cartera vencida y en forma fraudulenta continua realizando operaciones de transacción (compras y disposiciones de efectivo) con la apertura de cuentas nuevas, por lo tanto deben ser identificadas y boletinadas a áreas de apertura y operación de transacciones, evitando con ello que se continue el riesgo presentado. En ésta se requiere identificar lo siguiente:

Clientes con 2 ó más tarjetas sobregiradas y/o morosas

Reflejar en forma cuantitativa y cualitativa, cuentas con 2 ó mas tarjetas sobregiradas o en su defecto una cuenta con alto índice de sobregiro y que además presentan otra(s) cuenta(s) en situación normal o de apertura reciente.

Recuperación

Reflejar en forma cuantitativa y cualitativa, la recuperación de los adeudos de dichas cuentas.

Emisión de boletín de cartera

Detallar clientes de alto riesgo e incidencia en la cartera vencida, con el fin de evitar su aceptación en el otorgamiento de cualquier tipo de crédito en la Institución así como transacciones de operación.

Análisis Gerencial de Desempeño de Tarjetas de Crédito

La evaluación de la cartera de tarjetas de crédito en forma automatizada, se pretende realizar en base al comportamiento mensual de los tarjeta-habientes, por lo cual se requiere que dicho proceso refleje un resumen de la situación general que presenta el período analizado, el cuál sólo presente los casos excepcionales y emergentes que requieren rápida visualización y detección para realizar acciones respectivas (Fig. 2.2.10).

Asimismo, se pretende conservar un archivo histórico en el que se conserve mes a mes cuentas y/o situaciones relevantes, con el fin de llevar un comportamiento de los meses analizados así como el seguimiento de los mismos, asimismo se pretende :

- Llevar estadísticas mensuales, permitiendo comparar los períodos analizados, debido a su incremento o decremento de las debilidades detectadas.

- Verificar acciones efectuadas en las desviaciones que se detecten.

- Determinar proyecciones de comportamiento futuro, implicando con ello sensibilizar la situación del comportamiento que presente a la fecha del análisis así como meses posteriores, indicando la probabilidad de la situación, generando toma de decisiones para realizar acciones o en su caso ignorarlas.

En base a lo anterior, se considera que la mayor expectativa de una área de Auditoría en una Institución Bancaria para la implementación de un sistema de Desempeño de Tarjetas de Crédito, sea contar con un medio que brinde información confiable y actualizada para detectar y analizar integralmente las situaciones mencionadas en las páginas anteriores, mediante el acceso directo a información respectiva, sin tener que depender de manera absoluta y funcional de otras áreas.

Asimismo la independencia de fuentes de información, representa una característica propia del área, ya que la información recopilada en forma independiente repercute a su vez en la validación de cifras.

Por lo que se requiere realizar una comparación cuantitativo y cualitativa de índices de carteras, cuentas en las diferentes situaciones de mensualidades vencidas, etc., lo cuál se considera de gran importancia, ya que son un apoyo para la toma de decisiones a nivel estratégico y táctico dentro del desarrollo de actividades de la Dirección del Banco.

FUNCIONES DEL SISTEMA	
Análisis de Aperturas y Manejo de Crédito	Calidad del crédito
	Asignación de límites de crédito
	Manejo de cuenta
	Patrones de fraude
Identificar y Controlar Sobregiros Morosos	Saldos
	Recuperación
	Patrones de fraude
Control de la Cartera Vencida	Clientes sobregiros o morosos
	Recuperación
	Emisión de boletín de cartera
Integridad de la Información	Situación de la cuenta
	Ciclos de Corte
	Asignación de clave de abogado
	Asignación de No. de abogado
	Incongruencia de fechas
Análisis Gerencial de Desempeño	Estadísticas
	Seguimiento
	Proyecciones

Fig. 2.3.1 Funciones del Sistema de Desempeño de Tarjetas de Crédito

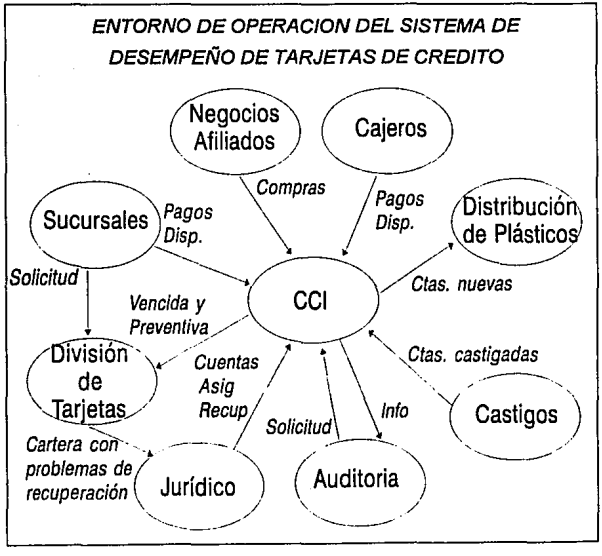
II.4 ANÁLISIS DE LA INFORMACIÓN

Se realizó una evaluación para conocer el funcionamiento y procedimiento de las tarjetas de crédito así como la problemática existente en la obtención de información por parte del usuario, dicha evaluación consistió en:

- Conocer y analizar las necesidades y requerimientos por parte del usuario
- Identificar y analizar el volumen de tarjetas en la cartera así como el número de productos a incluirse en el sistema
- Identificar y analizar las políticas y parámetros de los requerimientos solicitados
- Identificar y analizar las políticas y parámetros de cada uno de los productos
- Identificar y analizar la función de los sistemas que interactuan en relación a los requerimientos del usuario, así como datos que contienen, características y volumen

II.4.1 ENTORNO

A continuación se muestra gráficamente el entorno en el que se desenvuelve el "Sistema de Desempeño de Tarjetas Bancarias", se indican las entidades bancarias con las que interactúa y los vínculos operacionales que existen.



II.4.2 DIAGRAMAS DE FLUJO OPERATIVO ACTUAL

El proceso actual que se realiza para la revisión y análisis de la cartera vencida de tarjetas de crédito, es desarrollado en forma manual, como se describe en las siguientes diagramas de flujo operativo, bajo las entidades de Auditoria, Gerencia del Producto, División de Tarjetas de Crédito, archivo de microfilmación, control de altas y áreas de operación.

Los resultados que se obtienen a través de ésta metodología manual, es representativa de una revisión muy selectiva y aleatoria de cuentas en cartera vencida, lo cual presenta desventajas, como son:

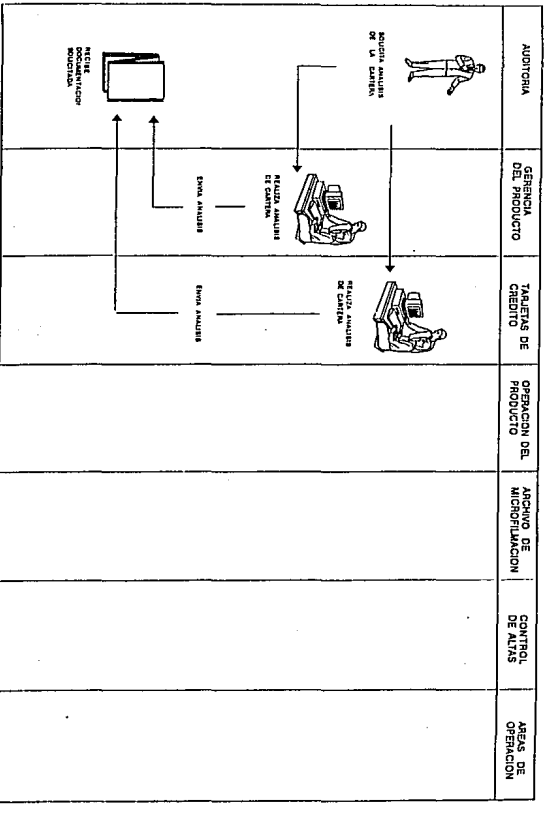
Alcance de revisión	Selectiva-aleatoria
Selección de muestra para análisis	Casos relevantes
Tiempo de respuesta de revisión y análisis	1 - 3 meses

Desventajas:

- Falta de revisión cuantitativa y cualitativa de la información en forma general
- Posible selección errónea
- Tiempos de respuesta largos
- Falta de oportunidad

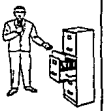
FORMA DE OPERACION ACTUAL DE AUDITORIA

HOJA 1/5



FORMA DE OPERACION ACTUAL DE AUDITORIA

HOJA 2/5

AUDITORIA	GERENCIA DEL PRODUCTO	TARJETAS DE CREDITO	OPERACION DEL PRODUCTO	ARCHIVO DE MICROFILMACION	CONTROL DE ALTAS	AREAS DE OPERACION
 REALIZA ANALISIS DE RIESGO Y CONTROL SOLICITA LISTADOS DE CARTERA A DETALLE			 SELECCIONA LISTADOS Y LOS ENVIA A AUDITORIA			
 REALIZA ANALISIS DE LISTADOS DETECTANDO DESVIACIONES DE CONTROL POR LO QUE SOLICITA DOCUMENTOS FUENTE				 RECIBE SOLICITUD		

FORMA DE OPERACION ACTUAL DE AUDITORIA

HOJA 3/5

AUDITORIA	GERENCIA DEL PRODUCTO	TARJETAS DE CREDITO	OPERACION DEL PRODUCTO	ARCHIVO DE MICROFILMACION	CONTROL DE ALTAS	AREAS DE OPERACION
RECIBE DOCUMENTOS				BUSCA DOCUMENTOS Y ENVIA A AUDITORIA		
ANALIZA DOCUMENTOS, DE LOS CUALES SE APOYA PARA SOLICITAR OTROS LISTADOS Y DOCUMENTOS					RECIBE SOLICITUD	RECIBE SOLICITUD

FORMA DE OPERACION ACTUAL DE AUDITORIA

HOJA 4/5

AUDITORIA	GERENCIA DEL PRODUCTO	TARJETAS DE CREDITO	OPERACION DEL PRODUCTO	ARCHIVO DE MICROFILMACION	CONTROL DE ALTAS	AREAS DE OPERACION
 RECIBE DOCUMENTOS Y LISTADOS  ANALIZA DOCUMENTOS Y LISTADOS					 BUSCA DOCUMENTOS Y ENVIA A AUDITORIA	 BUSCA DOCUMENTOS Y ENVIA A AUDITORIA

FORMA DE OPERACION ACTUAL DE AUDITORIA

HOJA 5/5

AUDITORIA	GERENCIA DEL PRODUCTO	TARJETAS DE CREDITO	OPERACION DEL PRODUCTO	ARCHIVO DE MICROFILMACION	CONTROL DE ALTAS	AREAS DE OPERACION
ANALIZA EN CONJUNTO INFORMACION Y REALIZA INFORME ENTREGA INFORME A AREAS RESPONSABLES	RECIBE INFORME Y REALIZA ACCIONES	RECIBE INFORME Y REALIZA ACCIONES				

II.4.3 IDENTIFICACIÓN DE LOS DATOS E INTERACCIÓN ENTRE SISTEMAS

La identificación de los datos necesarios para cubrir los requerimientos se llevó a cabo mediante la combinación de entrevistas, observación, recopilación de documentos, revisión de manuales escritos sobre políticas, regulaciones y procedimientos de operación, así como la interacción con Ingeniería de Sistemas. Con lo anterior se obtuvo la cantidad de datos necesaria que conforma el universo de la información para la Base de Datos que requerirán los ciclos de procesamiento.

La información que requiere el sistema, en su mayoría ya se encontraba en equipos "mainframe", por lo cual fue necesario identificar los sistemas y su relación con otros para seleccionar la información que es de utilidad para nuestro propósito.

A continuación se muestra mediante tablas el inventario de sistemas, procesos y productos involucrados, así como una relación de listados relativos a tarjetas de crédito.

INVENTARIO DE SISTEMAS Y PRODUCTOS

SISTEMAS

SISTEMA	PROCESO	DESCRIPCIÓN
	PAGOS	REGISTRO DE PAGOS REALIZADOS POR EL TARJETAHABIENTE, A TRAVÉS DE: • CAJEROS PERMANENTES • SUCURSALES
S / 030 TRANSACCIONES	DISPOSICIONES DE EFECTIVO	REGISTRO DE DISPOSICIONES DE EFECTIVO REALIZADOS POR EL TARJETA HABIENTE, A TRAVÉS DE: • CAJEROS PERMANENTES • SUCURSALES
	COMPRAS	COMPRAS REALIZADAS POR EL TARJETA-HABIENTE, A TRAVÉS DE: NEGOCIOS AFILIADOS

SISTEMA	PROCESO	DESCRIPCIÓN
S / 050	ALTAS Y REPOSICIONES DE PLÁSTICO	REGISTRA ALTAS Y REPOSICIONES DE PLÁSTICO, EFECTUADAS POR EL TARJETAHABIENTE A TRAVÉS DE DIVERSOS OPERADORES AUTORIZADOS.
CLIENTE-TARJETAS	ALTA E INCREMENTO DE LIMITE DE CRÉDITO	REGISTRA ALTAS E INCREMENTOS DE LIMITES DE CRÉDITO EFECTUADOS POR EL TARJETA-HABIENTE A TRAVÉS DE OPERADORES AUTORIZADOS.
S/015 ESTRUCTURA	ESTRUCTURA BANCARIA	REGISTRO DE ESTRUCTURA DIVISIONAL Y REGIONAL DE LAS ÁREAS DE OPERACIÓN DE UNA INSTITUCIÓN BANCARIA.
S/080 COBRANZA JURÍDICA	COBRANZA / ABOGADOS	REGISTRO DE ASIGNACIÓN DE CUENTAS, RECUPERACIÓN DE LA CARTERA JURÍDICA, A TRAVÉS DE ABOGADOS BANCARIOS.

SISTEMA	PROCESO	DESCRIPCIÓN
S / 700 REDOCUMENTACION	COBRANZA DE CARTERA REDOCUMENTADA	REGISTRO DE NEGOCIACIÓN DE REDOCUMENTACION DE LA CARTERA VENCIDA, LA CUAL ES EFECTUADA POR EL TARJETAHABIENTE A TRAVÉS DE LA GERENCIA Y DE PERSONAL AUTORIZADO.
S / 610 CASTIGOS	CARTERA EN CASTIGOS	REGISTRO DE LA CARTERA EN CASTIGOS, LA CUAL DEBIDO A LA COBRANZA ES CONSIDERADA IRRECUPERABLE, SIENDO SELECCIONADA A TRAVÉS DE LA GERENCIA Y DE PERSONAL AUTORIZADO.
S / 999 TARJETAS	CLIENTE-SALDOS	REGISTRO DE INFORMACIÓN GENERAL DEL TARJETA-HABIENTE (NOMBRE, DIRECCIÓN, ETC) ASÍ COMO SALDOS DE LAS TARJETAS.

SISTEMA	PROCESO	DESCRIPCIÓN
S / 999 TARJETAS	COBRANZA	REGISTRO DE LA CARTERA VENCIDA (ADMINISTRATIVA Y PREJURÍDICA), LA CUAL SE DETERMINA A TRAVÉS DE POLÍTICAS Y PARÁMETROS DE COBRANZA.
	BOLETÍN	REGISTRO DE INFORMACIÓN DE TARJETAHABIENTES BOLETINADOS A TRAVÉS DE POLÍTICAS Y PARÁMETROS DE COBRANZA, ASÍ COMO SITUACIONES DE ROBO Y EXTRAVIÓ DE LA TARJETA DE CRÉDITO.

PRODUCTOS

PREFIJO	TARJETA	PRODUCTO	LINEA / MERCADO
5290	01	TERRACOTA	L9
4540	69	VISA CLASICA	L3 NACIONAL
5290	91	PLUS	L3
5288	43	MASTERCARD	L6
4552	55	VISA PREMIER	L6 INTERNACIONAL
4540	57	VISA CLASICA	L6
8548	13	CALINDA QUALITY	L7
8813	15	MULTIVISION	L7
5290	17	SUBURBIA PLUS	L7
5290	23	SUBURBIA	L7 PRIVADAS
5290	27	SUBURBIA CLASICA	L7
5290	35	HIGH LIFE	L7
8548	53	SEGUROS AMERICA	L7
5288	4	TRADICIONAL	L2
5206	8	DORADA	L2 AFINIDAD
4901	76	VISA CLASICA	L2

PREFIJO	TARJETA	PRODUCTO	LINEA / MERCADO
5290	88	TURISMO	L2
4901	78	DORADA	L2 AFINIDAD
4937	86	TURISMO	L2

Cabe mencionar, que el inventario de productos solo contempla una parte de toda la gama de productos existentes, ya que los productos reflejados son los utilizados en el sistema automatizado del **Desempeño de Tarjetas de Crédito.**

Lo anterior es debido a los requerimientos de seleccionar las tarjetas de naturaleza de crédito así como las carteras de tarjetas de crédito requeridas por el usuario del sistema.

INVENTARIO DE LISTADOS

- Actividad contable de tarjetas
- Desglose de capital e intereses de la cartera
- Desglose de transmisión
- Sumarización mensual de tarjeta-habientes
- Sumarización mensual general
- Interface de movimientos diarios
- Actualización contable
- Detalle de cálculo para pago de rendimientos
- Control de emisión de plástico
- Auditor de archivo de tarjeta-habientes
- Actividad contable de tarjeta-habientes
- Listados de cartera vencida
- Auditor del archivo de tarjeta-habientes
- Pagos recibidos con una ó mas mensualidades vencidas
- Tarjetones y listado de cartera vencida y sobregiros
- Acuses de recibos

II.5 ALTERNATIVAS DE SOLUCIÓN

Para cubrir los requerimientos descritos anteriormente es necesario desarrollar un sistema que permita la explotación de una base de datos "depurada" en la que sólo exista información relacionada con la medición del desempeño de tarjetas de crédito así como para cubrir las necesidades de auditoría para no interferir con otros procesos.

Con el propósito de ofrecer a la Institución Bancaria una herramienta que le permita garantizar el óptimo apoyo en las labores de medición el desempeño y auditoría de tarjetas de crédito, a continuación se señalan las alternativas para las cuales se deben tener las siguientes consideraciones:

Las determinantes del uso de equipo son institucionales por lo tanto hay que ajustarse en primera instancia a que el origen de la información se genera en un equipo TANDEM VLX del área de tarjetas de crédito.

Los usuarios de auditoría son ajenos al conocimiento de computadoras. Los volúmenes de información son grandes, por lo tanto debemos tomar en cuenta una solución que permita ser lo suficientemente ágil y fácil de manejar.

La información a manejar es de carácter confidencial por lo tanto debemos considerar una solución que brinde la máxima seguridad en cuanto a su manejo como en el control de los resultados que se obtengan.

II.5.1 OPCIONES PROPUESTAS

A continuación se presentan las opciones de solución:

- Desarrollo de un Sistema de Desempeño de Tarjetas de Crédito en el equipo TANDEM.

- Desarrollo de un Sistema de Desempeño de Tarjetas de Crédito en PC transfiriendo información del equipo TANDEM a un SERVIDOR.

Para cada opción se da una breve descripción y se evalúan cuatro aspectos que ayudan a tomar una decisión, tales aspectos son:

- Equipo y Recursos Requeridos
- Responsabilidades
- Ventajas
- Desventajas

A) DESARROLLO DE UN SISTEMA DE DESEMPEÑO DE TARJETAS DE CRÉDITO EN EL EQUIPO TANDEM

Descripción

El equipo TANDEM está orientado a realizar transacciones en línea, es decir trabaja mínimo con dos procesadores en paralelo por si llegara a fallar uno (el denominado principal), puede recibir mantenimiento sin tener

que apagarlo y posee discos en espejo para registrar las transacciones.

En el equipo TANDEM existe un Sistema Integral de control de las transacciones del uso de tarjetas de crédito. Para el desarrollo del "Sistema de Desempeño de Tarjetas de Crédito" sería necesaria la creación de un área de servicios que permitiera apoyar la auditoría del desempeño del uso de tarjetas de crédito en el equipo TANDEM en función de los requerimientos descritos anteriormente.

El sistema estaría disponible para ser compartido por varios usuarios y formaría parte del Sistema Integral de control de las transacciones para el uso de tarjetas de crédito.

El sistema estaría sujeto a las políticas de operación del área de sistemas y se podría disponer de información en línea prácticamente en el momento que se desee.

Equipo y Recursos Necesarios

Se necesitarían siete terminales: dos para desarrollo de mejoras y ajustes, una para mantenimiento y otras cuatro más para los usuarios.

Se necesitarían también compartir completamente los recursos de la máquina tales como espacio en disco, prioridad para el proceso, memoria, así como contar con infraestructura para que el área de tarjetas y la de

auditoría no se encuentren físicamente en el mismo lugar. Además de destinar recursos de monitoreo a las funciones de operación, así como la de compartir y priorizar la emisión de los listados y reportes que se produzcan.

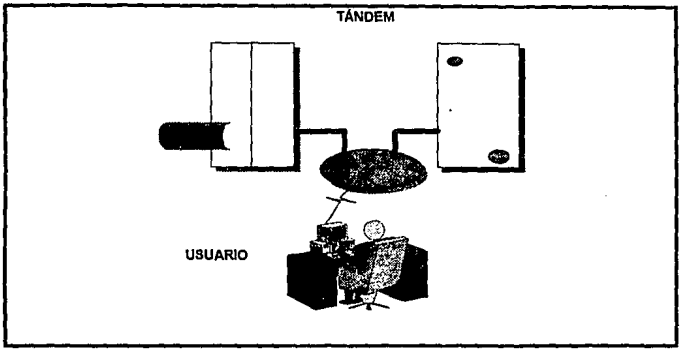


Figura 2.5.1. Transmisión de información de un TANDEM a un PC.

Responsabilidades

- **Del Departamento de Sistemas de Auditoría**

  El área de operación sería la encargada de habilitar las líneas de comunicaciones para la ejecución del "Sistema de Desempeño de Tarjetas de Crédito" en función de un programa de producción previamente establecido, así como de la realización de respaldos

periódicos de información.

La obtención de los productos emitidos estaría controlado por el área de operación de acuerdo a sus calendarios y horarios establecidos.

Las solicitudes para cubrir las nuevas necesidades del sistema que vayan surgiendo en el área de auditoría se harían al departamento encargado del desarrollo de nuevos sistemas al igual que las solicitudes de mantenimiento se canalizarían al área correspondiente.

El cubrir las necesidades de soporte técnico estaría totalmente a cargo del área correspondiente quien proporcionaría los elementos necesarios para prevenir y corregir fallas relativas a equipo y teleproceso.

- **Del Area de Auditoría**

Se hace necesaria la intermediación de personal que domine los aspectos de auditoría con la gente que conozca aspectos relevantes del desarrollo de sistemas.

La actualización de la base de datos del sistema.

El uso y explotación correctos del sistema así como de los productos obtenidos.

Ventajas

La información se encuentra en el mismo medio ambiente de operación que la produce, simplemente habría que tomarla.

La estrategia e infraestructura de teleproceso y transmisión de información estaría sujeta a la ya existente en la institución Bancaria.

Desventajas

Se estaría trabajando con una base de datos muy densa en la que habría información innecesaria y ésta interfiriera con la inherentes a las transacciones en los sistemas de tarjetas de crédito, haciendo lento uno y otro sistema.

Se tendría que adquirir adicionalmente equipo para soportar el almacenaje de información "depurada" o en su caso hacer todavía más lento el sistema si dicha información se obtuviera con herramientas propias del equipo TANDEM tales como SQL y demás utilerías.

Al ajustarse a las políticas de operación como en la emisión de los productos del área de tarjetas de crédito, el sistema corre el riesgo de perder confidencialidad.

No existe independencia de ejecución ya que se estaría sujeto siempre a lo que marcara el área de sistemas de "Tarjetas de Crédito" y no a lo que

determinara según sus necesidades el área de auditoría.

Existiría la necesidad de capacitar a personal de auditoría en la operación del sistema y equipo TANDEM.

B) DESARROLLO DE UN SISTEMA DE DESEMPEÑO DE TARJETAS DE CRÉDITO EN PC TRANSFIRIENDO INFORMACIÓN DEL EQUIPO TANDEM A UN SERVIDOR.

Descripción

Al contar con herramientas de PC podemos pensar en desarrollar el "Sistema de Desempeño de Tarjetas de Crédito" en lenguajes de programación de "alto nivel" o usando "Manejadores de Bases de Datos".

Lenguajes

A través de los años se han desarrollado cada vez lenguajes más poderosos tanto de alto nivel como de nivel intermedio, con lo que las posibilidades de encontrar un lenguaje adecuado para cada aplicación se amplían.

Manejadores de Bases de Datos:

Durante los años 70s se empiezan a desarrollar lenguajes que facilitan la labor de hacer sistemas, uno de los primeros acercamientos es COBOL

que se populariza rápidamente, aunque la primera Base de Datos realmente funcional fue "TOTAL" a finales de los 70s sin embargo la ANSI (American National Standards Institute) a principio de los 80s da una línea de acción para estandarizar estos sistemas creándose las Bases de Datos Relacionales y con estas los Manejadores de Bases de Datos Relacionales, a partir de este momento empiezan a salir al mercado una gran variedad de manejadores de Bases de Datos para distintas máquinas y diversas necesidades de información. Así siguiendo esta línea y con las ventajas que ofrecen los nuevos manejadores, definitivamente esta es la mejor opción para nuestro sistema. La selección del Manejador de Bases de Datos se analizará más adelante.

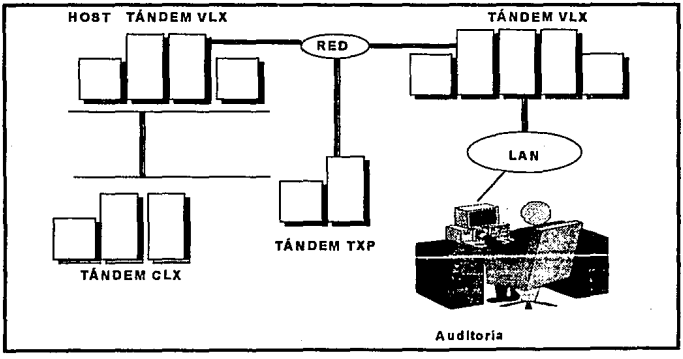


Fig. 2.5.2. Entorno general del hardware y comunicaciones del Banco

Esta opción la podemos dividir en tres incisos dependiendo de la forma en la que se haga la transmisión de la base de datos depurada del equipo TANDEM al SERVIDOR:

Para los tres incisos es válido indicar que de la base de datos correspondiente a los sistemas de tarjetas de crédito a través de un subsistema se obtiene una Base de Datos depurada la cual se tendrá que hacer llegar a un SERVIDOR:

a) El respaldo de la base de datos depurada se hace en un dispositivo de cartucho para posteriormente transferirla a un dispositivo "Drive" de cinta de 8mm (lector de "cassettes") y así hacerla llegar al SERVIDOR.

b) Se transfiere directamente por las utilerías de comunicaciones TCP/IP o X.25AM al dispositivo "Drive" de cinta de 8mm (lector de cassettes) y como en el caso anterior se hace llegar posteriormente al SERVIDOR.

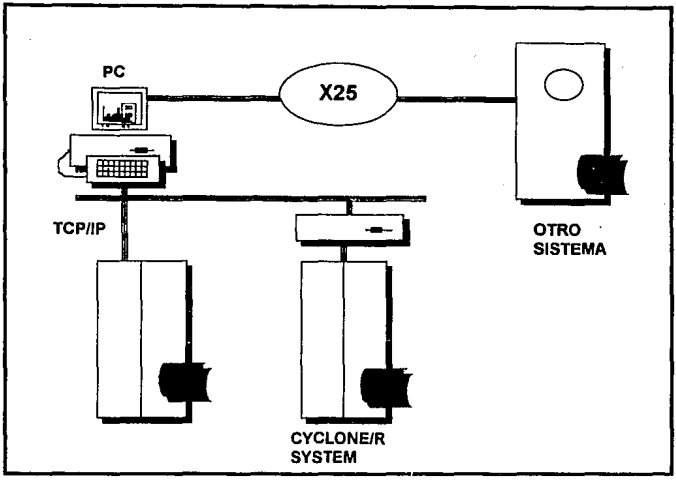


Fig. 2.5.3. Transmisión directa del Host TANDEM al SERVER usando TCP/IP o X.25.

c) La transferencia se hace por medio de la utilería IXF directamente del "Host" TANDEM al SERVER.

La explotación de la información la haría el sistema usando como herramientas al SERVIDOR para el almacenamiento y una PC en la que se desarrollaría un sistema propio que dependería totalmente tanto en su operación, como en su desarrollo y mantenimiento del área de Auditoría y en caso de que fuera necesario un crecimiento en el número de usuarios, podría crecer mediante la instalación de una Red de Area Local que

dependería totalmente del área de Auditoría.

Dicha red se ejecutaría bajo el sistema operativo NetWare 386 diseñado específicamente para el microprocesador 80386 y compatible para 80486 de Intel; tiene la ventaja de estar diseñado de manera abierta y modular.

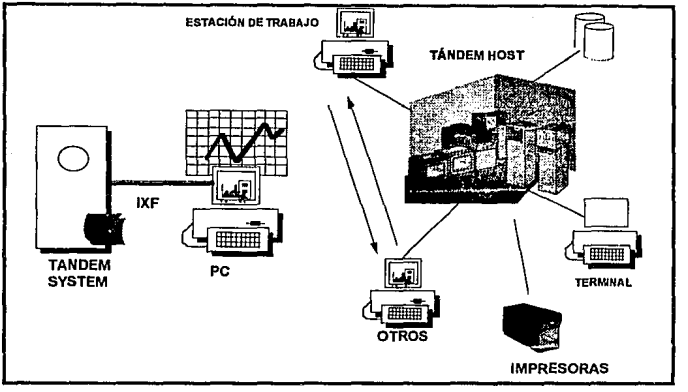


Figura 2.5.4. Transferencia directa del Host TANDEM al server usando IXF.

Equipo y Recursos Requeridos

Para los tres incisos se sabe que se cuenta con la generación de la Base de Datos Depurada obtenida de un equipo TANDEM VLX y a partir de este punto hay diferencias según el medio de transmisión que se utilice, también

se sabe que el SERVIDOR es el mismo para los tres casos:

a) Se requiere de una unidad de cartucho capaz de soportar el almacenamiento de un mínimo de 5 GB. Además de un "Drive" de cinta de 8mm (lector de cassettes) y de la infraestructura de comunicaciones que permita hacer la transferencia entre ambos.

b) Se deberá adquirir la utilería TCP/IP o X25AM así como la unidad "Drive" de cinta de 8mm (lector de cassettes) para poder hacer llegar la información al SERVER.

c) Se deberá contar con la utilería IXF por parte de TANDEM para hacer la transferencia de la Base de Datos Depurada al SERVIDOR.

En esta alternativa de solución la operación, el proceso, emisión de listados y reportes así como la generación de resultados se realizarían completamente por el área de Auditoría.

Responsabilidades

+ Del área de Sistemas de Tarjeta de Crédito

La correcta transmisión de la información.

El soporte técnico derivado de las telecomunicaciones que serían necesario llevar a cabo.

+ Del área de Auditoría.

La ejecución del "Sistema de Desempeño de Tarjetas de Crédito" así como su operación, no pasan por la supervisión de ningún área que no esté involucrada sino sólo por la de Auditoría que es la responsable.

La explotación de los recursos, la emisión de listados y reportes y su generación son responsabilidad absoluta del área de Auditoría.

El área de Auditoría sería responsable de calendarizar e imponer los horarios de servicio, ejecución y operación del sistema según sus necesidades.

La solicitudes para cubrir las nuevas necesidades del sistema que vayan surgiendo se harían directamente en el área de Auditoría, al igual que las necesidades de mantenimiento.

El cubrir las necesidades de soporte técnico estaría totalmente a cargo del área de Auditoría quien proporcionaría los elementos necesarios para prevenir y corregir fallas relativas a equipo y se coordinaría con el área de "Sistemas de Tarjetas de Crédito" para las labores de teleproceso en tal caso.

Se hace necesaria la participación de personal que domine los aspectos de auditoría con la gente que conozca aspectos relevantes del desarrollo de sistemas en PC, con el fin de que realice la actualización de la base

de datos del sistema para el uso y explotación correctos del sistema así como de los productos obtenidos.

Ventajas

La ejecución y operación del sistema la hace directamente personal de Auditoría, teniendo como consecuencia total independencia.

El sistema es más amigable y rápido en su ejecución al ejecutarse en una PC.

Se trabajaría con una base de datos exacta y exclusiva para las labores de Auditoría.

El sistema ofrece absoluta confidencialidad debido a que la emisión de los productos se realiza en Auditoría.

Existe personal capacitado para desarrollar el Sistema en PC.

La estrategia e infraestructura de teleproceso y transmisión de información estaría sujeta a las necesidades de Auditoría.

Desventajas

La información se encuentra en el mismo medio ambiente de operación que la produce.

Se tendría que adquirir adicionalmente equipo para soportar el almacenaje de información depurada en dispositivos de Cartucho o "Drive" de cinta de 8mm.

II.5.3 SELECCIÓN DE LA ALTERNATIVA DE SOLUCIÓN.

CRITERIOS

Para la selección de la mejor opción se han fijado tres criterios.

1. Facilidad de Uso
2. Independencia
3. Rapidez de Implementación

1. Facilidad de uso

Se debe contar con un Sistema que permita ser usado lo más fácilmente posible sin sofisticaciones surgidas de equipos de operación y manejos complicados por ser muy grandes y por lo tanto difíciles de administrar, tomando en cuenta que los usuarios no son personal de "Sistemas".

Es necesario que el sistema solamente reciba la información necesaria para la obtención de los resultados.

2. Independencia

Debido a la confidencialidad que arrojarán los resultados del Sistema se debe depender lo menos posible de una área institucional de "Sistemas".

Se necesita que los resultados que se obtengan del sistema no dependan de procesos externos.

3. Rapidez de implementación

El sistema debe ser desarrollado lo más rápidamente posible.

Es necesario aprovechar los recursos humanos existentes por lo tanto no es deseable que se tengan que impartir cursos complicados y que se lleven mucho tiempo.

EVALUACIÓN

Por la infraestructura de equipo con la que se cuenta y tomando en cuenta que por la naturaleza de la información que se va a trabajar, así como por el tipo de usuarios que producirán la información, nos inclinamos por la opción de desarrollo de un "Sistema de Desempeño de Tarjetas de Crédito" realizado en PC con explotación de información de un SERVIDOR.

La parte a considerar más importante en cuanto a la obtención de la información depurada se refiere a la transmisión de la Base de Datos Depurada, para lo cual consideramos los siguientes aspectos:

La opción de transmisión de información a una unidad de cartucho para posteriormente transmitirla a un "Drive" de cinta de 8mm y después al SERVIDOR, hace un paso innecesario si el equipo TANDEM cuenta con la utilería de transmisión de información TCP/IP o la utilería X25AM. Por lo tanto podemos descartar esta opción.

Por lo que respecta a la opción de trabajar la transmisión directamente a través de la utilería IXF, mediante la cual es posible transmitir la información del equipo TANDEM al SERVIDOR directamente, no es conveniente trabajar en línea debido a que como se verá más adelante es necesario trabajar con información que haya completado una jornada para que cumpla con los requerimientos que señalan los parámetros y políticas de selección de información.

Por lo anterior podemos concluir que la mejor alternativa de solución es la siguiente:

Desarrollar el "Sistema de Desempeño de Tarjetas de Crédito" en una PC desarrollado con un Manejador de Base de Datos explotando la información contenida en un SERVIDOR, la cual fue transmitida previamente de un equipo TANDEM VLX a un dispositivo llamado "Drive" de cinta de 8mm

cuya cinta es un cassette que tiene la posibilidad de almacenar hasta 5 GB. Es muy fácil de manejar y es el más rápido de respaldar.

CAPITULO III

DESCRIPCION Y DESARROLLO DEL SDTAR

III.1.1 CRITERIOS DE EVALUACIÓN

Actualmente la administración de información tiene la mayor importancia. Se puede asegurar que el hecho de que se tenga una organización en la que se siga por norma una buena administración y control en la información es el mejor indicador de que las cosas empiezan bien, sin embargo esto no es todo, ahora el problema es saber cómo hacer que esto se cumpla, en esta sección abordamos precisamente ese "cómo" según las características de solución que se plantearon al principio, es decir, la solución se inclinó por el desarrollo de un sistema con una herramienta de vanguardia, en ambiente personalizado con posibilidad de Red; dicha herramienta es un manejador de Bases de Datos llamado Foxpro para Windows. Sin olvidar por supuesto, otros como Paradox, Visual Basic y Access que están ganando terreno dentro de esta categoría.

En Banamex, la situación está claramente definida, ya que existen estándares institucionales en lo referente al uso de herramientas de desarrollo y aplicaciones.

En esta sección presentamos una evaluación de los principales Manejadores de Bases de Datos que existen actualmente en el mercado. En la sección I.4 "ANALISIS DE HERRAMIENTAS DE SOFTWARE DE BASES DE DATOS PARA PC's", se presentaron algunos manejadores, así como las facilidades que cada uno ofrece al usuario y al desarrollador. Cabe mencionar que la presente evaluación solo intenta mostrar la opinión generalizada que prevalece en revistas especializadas de nuestro medio,

fundamentándose algunas de ellas en resultados de BenchMarks (Pruebas de puntos críticos).

Podemos mencionar por otro lado que, en ausencia de un método unificado de almacenamiento y recuperación de datos críticos, se han generado pilas de texto y gráficos en papel sobre los escritorios. Cuando esa información es arreglada sistemáticamente, dicha información se convierte en una poderosa herramienta de productividad. Al introducir los Manejadores de Bases de Datos (DBMS), se ha permitido que estos se conviertan en "archiveros electrónicos".

Poder de las Bases de Datos

Los manejadores de Bases de Datos deben permitir construir aplicaciones funcionales y hechas a la medida, deben tener también orientación a usuarios finales, ser interactivas, construcción rápida y facilidad de programación.

Las redes de trabajo tienen que llegar a ser estándares en un entorno corporativo y tienen que llegar a ser manejadores de bases de datos multiusuarios con controles de cocurrencia y archivos encadenados. La orientación a futuro es el desarrollo de productos con arquitectura Cliente/Servidor.

Bases de Datos relcionales con arquitectura Archivo/Servidor y multiusuario son el primer paso para desarrollar aplicaciones para usuarios

personalizados pero compartiendo datos. En la arquitectura Archivo/Servidor todas las porciones ejecutables de la aplicación, corren en el "cliente" (PC); el Servidor de la red simplemente almacena archivos de datos compartidos y proporciona servicios de encadenamiento. Cuando un cliente ejecuta un query, el Servidor envía todo el potencial de datos al cliente, el cual descarta de alguna manera los que no necesita. El cliente se encarga de la integridad de datos y valida el chequeo.

En la arquitectura Cliente/Servidor la Base de Datos del Servidor contiene archivos compartidos y se encarga de la integridad de datos y validación de reglas. La aplicación "cliente" contiene el menú formatos, y el código de programa asociado con la interface de usuario. El Server ejecuta querys especificados por el "cliente" y envía solamente la información resultante a través de la red al "cliente", el acceso cliente/Servidor es más rentable y rápido para alto volumen de tráfico de datos, pero más caro. Los sistemas Cliente/Servidor son la mejor opción para aplicaciones de misión crítica, pero los sistemas Archivo/Servidor son una mejor solución en costo efectivo para otras necesidades de bases de datos para los negocios.

El término relacional tiene una definición teórica específica pero en uso común describe un sistema compuesto de archivos separados (o tablas) que juntas conforman una base de datos. Las tablas separadas frecuentemente se relacionan de uno a muchos; esto es, el detalle de los registros de la tabla es almacenado en otra tabla. Dos tablas son ligadas por un campo común. El método relacional de almacenamiento de datos hace eficiente el uso del espacio en disco.

La normalización, que como ya se vió es un proceso de organización de datos de modo que se optimiza el duplicado de los mismos dentro de la base de datos lo que es crucial en el modelo relacional. Es fácil normalizar una pequeña base de datos que contiene un pequeño número de elementos, sin embargo, es un proceso muy complejo para bases de datos grandes. El cómo normalizar una base de datos no es automático y depende de cómo estén relacionados los elementos dato y de la experiencia del administrador de la base de datos. Cuando la base de datos se compone de múltiples tablas los desarrolladores deben tener cuidado de mantener la integridad referencial, esto es, consistencia interna de modo que los registros de una tabla se refieran a registros que existan en otra tabla. De las bases de datos que se están evaluando Advanced Revelation, DataEase, Paradox DOS, Paradox para Windows, R:BASE y Superbase, proporcionan integridad superficial a nivel de máquina.

Muchas de las bases de datos revisadas usan variantes de SQL para acceso y manipulación de datos, este lenguaje fue diseñado para trabajar con bases de datos relacionales, pero SQL no es un lenguaje de programación de propósito general, de modo que todas las bases de datos que proporciona algún otro lenguaje sirven para uso en conjunto o en lugar de SQL. SQL es usado frecuentemente para solicitar datos del Servidor de archivos. Pero no se necesita aprender sintaxis de SQL para usarlo. "Query by example" (QBE) es un método orientado a visualizar especificaciones donde las condiciones se dan con una tabla.

Mucha atención a esfuerzos de desarrollo e investigación se está dando a la optimización de querys, la técnica para analizar los requerimientos de un query y determinar la forma más rápida de ejecución se satisface de la siguiente forma: cuando se usan índices, y cuando la búsqueda en las tablas se hace secuencialmente. La optimización de querys es una tecnología compleja, en parte ciencia y en parte arte, los resultados pueden mejorar o empeorar el desempeño del manejador de la base de datos.

Las técnicas de acceso pueden tener un gran impacto en la rapidez del manejador de bases de datos. Algunas bases de datos hacen que se tenga que esperar hasta que todos los registros especificados sean encontrados antes de mostrarse, otros tales como DBASE, procesan el query hasta que una pantalla llena de datos ha sido encontrada presentándola, entonces se espera hasta que se intente el cambio de pantalla antes de continuar la selección . Otras presentan la pantalla y continúan buscando mientras se puede ver la primera parte del resultado. Para asegurarnos que se midió la misma tarea en cada caso, en las pruebas que se hicieron se definieron búsquedas completas cuando el último registro seleccionado estaba disponible. Pero hay que estar consciente que la aparente rapidez de un query depende de los resultados de la prueba dependiendo de la técnica de retorno del manejador de base de datos. Otro factor que afecta la rapidez es el formato del resultado. Algunos manejadores guardan la información requerida en una tabla de resultados almacenados en disco. Otros crean una fotografía instantánea de los registros en memoria. Otros son discretos como FoxPro, regresan solamente el número de registros de un conjunto de

resultado largo y entonces ejecutan un segundo paso para accesar los registros identificados. Esta técnica de trabajo es una ventaja de FoxPro.

Algunos de los paquetes que fueron probados tales como Access y Paradox para Windows nos permiten manipular directamente archivos con formatos distintos, no se tiene que importar un archivo de un formato nativo. Access implementa la tecnología Open DataBase Connectivity (ODBC) de Microsoft, mientras Paradox para Windows usa la Open DataBase Application Programming Interface (ODAPI). Esas interfaces de programación de aplicaciones ofrecen estándares competitivos en todos aspectos amigables. La capacidad de hacer que esto sea bueno para los desarrolladores intentando integrar datos de múltiples fuentes es usualmente algo que afecta la velocidad. Se creía que todos lo productos probados tenían potencia suficiente para desarrollar más aplicaciones que fueran necesarias, ello varía en el rango en que afecta solamente a los desarrolladores.

La ventaja de las interfaces de usuarios gráficas en el mundo de las bases de datos ha hecho todo más fácil. Los paquetes en Windows permiten construir aplicaciones gráficas con barras de menú, redio buttoms y listas recogidas simplemente por arrastre y punteo, encimando y haciendo "click".

Bases de Datos Multiusuario para Windows

- La mayoría de los usuarios están migrando hacia ambiente Windows, esto es debido a que en el mercado surgen nuevos productos muy poderosos desarrollados en este ambiente:

- Por ejemplo, Access de Microsoft es una herramienta que ofrece una facilidad increible para su uso y está orientada a usuarios finales y es excelente para desarrolladores.

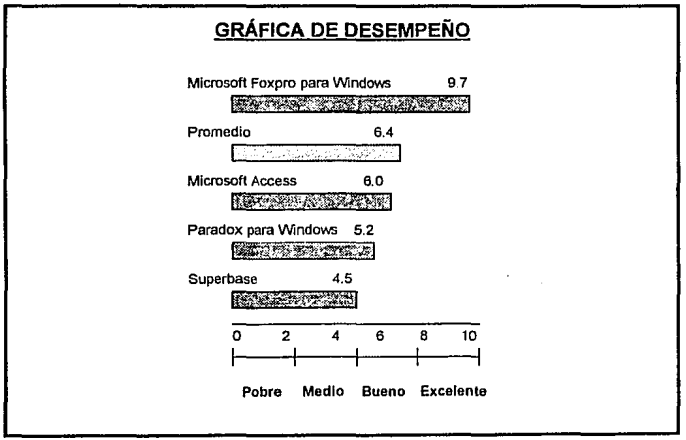
- FoxPro para Windows brinda un poder muy grande en la construcción de pantallas y en la reindexación de archivos. El prestigio que tiene esta versión es ganado desde la versión para DOS.

- Paradox para Windows es una herramienta muy versátil en ambientes de desarrollo orientado a objetos, además de poseer un gran poder en la obtención de información y de contar con herramientas poderosas de manejo de Bases de Datos.

- Para negocios en los que se requieren características de una Base de Datos en Windows con Hardware escalable, existe Superbase.

Desempeño

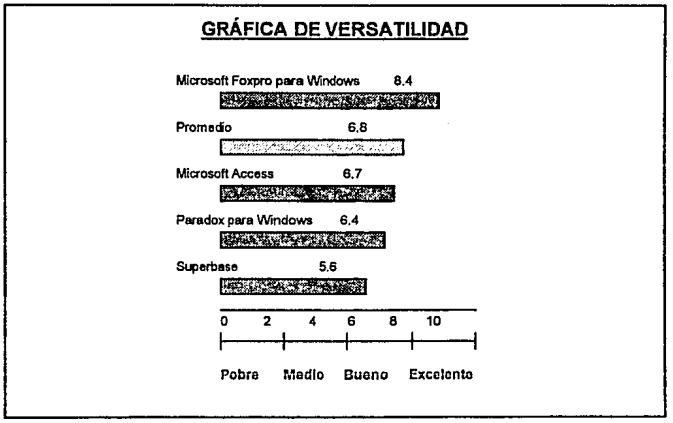
Los manejadores de Bases de Datos que se comparan se pueden dividir en dos distintas categorías de desempeño; FoxPro de Microsoft para Windows y todos los demás manejadores. FoxPro demostró tener consistentemente un desempeño superior (Gráfica 3.1.1) en aplicaciones de usuarios privados y en ambientes multiusuario, tanto en sistemas pequeños y grandes. En otras pruebas tales como búsqueda y selección de registros algunos manejadores son excelentes pero eso no podrá ser significativo para influir en el criterio de selección.



Gráfica 3.1.1 Gráfica de desempeño

Vista general

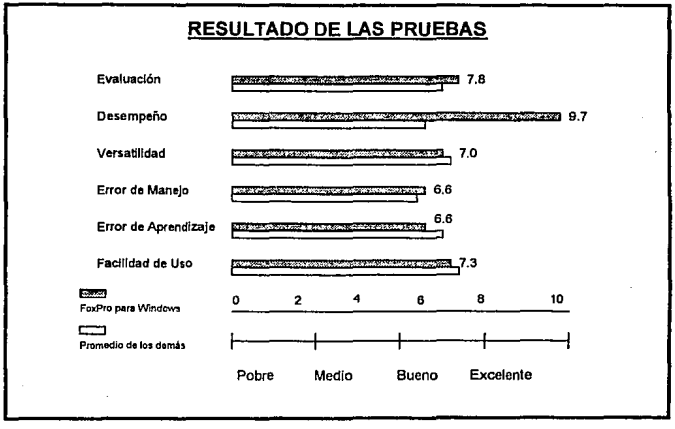
FoxPro tiene un desempeño superior en la mayoría de los indicadores de rendimiento que normalmente se usan para evaluar, demostrando con ello que tiene mayor versatilidad. Paradox es más versátil que Access de Microsoft, pero Access es más poderoso en cuanto desempeño. Superbase tiene equilibrio entre versatilidad y desempeño (Gráfica 3.1.2).



Gráfica 3.1.2 Indicadores de Versatilidad obtenidos

FoxPro de Microsoft para Windows combina una rapidez increible con una cantidad poderosisima de herramientas de desarrollo de aplicaciones que son conocidas desde la versión de FoxPro para DOS.

En la siguiente gráfica se muestran los resultados obtenidos de FoxPro contra el promedio de rendimiento de todos los demás manejadores de Bases de Datos involucrados en la prueba.



Gráfica de comparación entre FoxPro vs. El promedio de los demás

FoxPro para DOS y FoxPro para Windows

Las bases de datos en Windows siempre fueron más lentas significativamente que las de ambiente DOS. FoxPro para Windows resultó ser el manejador más rápido en ese ambiente y demostró que las bases de datos en Windows no tienen porque ser lentas. Todos los paquetes hicieron prácticamente lo mismo, la diferencia está en la rapidez de respuesta y en la flexiblidad del manejo.

FoxPro desde que apareció ha sido el más rápido, sin embargo la nueva versión es todavía más rápida, esto se aprecia significativamente más en una Red. Existen dos versiones de FoxPro 2.5 una para DOS y otra para Windows. Microsoft está desarrollando versiones de FoxPro para Unix y Macintosh (que reemplazará a FoxBASE en ambas plataformas).

Los productos de Fox invadirán el mercado de las bases de datos combinando la rapidez y la compatibilidad con DBASE. Actualmente la compatibilidad con DBASE no es lo más importante, pero sí la rapidez.

FoxPro ha sido siempre uno de los manejadores de base de datos más rápidos. Una razón es que el código es compilado en vez de interpretado. Modificar código fuente para programas, pantallas, reportes, es transparente para el compilador antes de ser ejecutado. Otra razón por la que es rápido es el gran uso de cachés internos que evita accesos innecesarios a disco. EL acceso a disco es minimizado por el uso de índices combinados y comprimidos (archivos .CDX).

La operación SQL SELECT es más rápida por lo bien integrada que está. FoxPro mantiene número en registro en memoria en vez de mantener los registros mismos (los registros pueden retornar rápidamente gracias al formato fijo .DBF). FoxPro asegura que el resultado del query permanezca válido cuando el dato actual es regresado si los registros fueron cambiados entre el regreso y la construcción del resultado. Esos nuevos elementos hacen que la versión 2.5 sea más rápida que sus predecesoras, pero Microsoft está manteniendo los detalles en secreto. Podría ser un nuevo algoritmo para determinar el mínimo necesario de información para leer del Server. Esto reduce significativamente el tráfico en la red; mejora el uso de la información en la memoria: la versión 2.5 puede evitar accesos a disco o red en muchos casos cosa que la versión 2.0 no puede.

La rapidez no es solamente la razón por la que FoxPro es mejor ; es la facilidad de uso en las tareas interactivas. La explotación mediante ventanas puede abrir bases de datos, conjuntos de relaciones e inspeccionar y modificar sus contenidos, simplemente haciendo "click". Cada acción se inicia abriendo una ventana de comando la cual muestra las instrucciones que completan la tarea. Se trabaja en reversa también; la vista en la ventana muestra la configuración de base de datos cambiada cuando se invoca la ventana de comando.

Se pueden construir complejos querys, completados con múltiples tablas afectadas por joins. Los resultados en la operaciones de selección y envío de resultados a una ventana se generan por medio de un SQL SELECT, la impresión del reporte, una base de datos nueva, un cursor SQL o una rutina de graficación también, simplemente haciendo "click".

Posee un generador de aplicaciones automático que instantáneamente genera una aplicación multitabla con querys, actualizaciones y reportes. Pero hacer una mejora significativa a la aplicación puede requerir de mucha programación o construcción de una aplicación con el "FoxPro Power Tools".

El FoxPro Power Tools incluye un constructor de pantallas, Constructor de Menues, Generador de Reportes y convertidor de plataformas (de una versión DOS de FoxPro a Windows) el manejador de proyectos (Project Manager) almacena todas las piezas de una aplicación juntas y genera archivos .APP y archivos .EXE. Los archivos EXE son creados con el "FoxPro Distribution Kit" y puede estar expandido o comprimido. El constructor de pantallas orientado a objetos permite ligar objetos rápidamente tales como textos, campos apretándo botones en una ventana. Se pueden combinar ventanas para formar aplicaciones completas. La versión DOS permite seleccionar objetos a través de un menú. En la versión Windows el proceso es hecho fácilmente por un manejador gráfico de cajas, pero las líneas y cajas que se crean son posicionadas por uno o dos "pixeles" y los campos de datos serán recortados usando letras mayúsculas. Ese problema puede ser fácilmente corregido.

Se especifica un propietario de objeto haciendo doble "click" en el objeto. Las cajas de diálogo permiten seleccionar eventos y un editor permite invocar instrucciones describiendo qué hacer con la ocurrencia de eventos.

El constructor de pantallas genera código fuente de la pantalla FoxPro. El código fuente del constructor de pantallas puede actualizarse y modificarse

o crear un nuevo tipo de objeto. Por ejemplo se puede crear una biblioteca que agregue efectos tridimensionales a objetos. El generador de reportes hace más fácil la creación de reportes. Se especifica un objeto, puede ser un texto, un campo de base de datos o una caja. Entonces se arrastra el objeto y esto apunta al título, detalle o resumen, se puede ver la justificación que se va generando. La versión de Windows proporciona un generador de reportes mediante cajas que maneja objetos haciendo "click" en el ícono, y agrega un objeto "bipmapeado" de modo que puede ser fácilmente agregado al encabezado de la página. Para reportes altamente complejos se deben hacer algo de programación. El usuario define las funciones que serán usadas en los campos de salida para proporcionar funciones avanzadas, tal como un rompiento de página condicional que depende del valor de una variable o cálculos tediosos.

Si se necesita ir más lejos en las tareas de FoxPro existe la "Library Construccion Kit" la cual proporciona todo lo que se necesite para integrar y ensamblar código en la aplicación. Las bibliotecas de FoxPro 2.0 deben ser regeneradas con DOS versión 2.5.

FoxPro para DOS incluye versión para 16-Bit y una versión extendida de 32-Bit para PCs basadas en 386 o superiores. La versión extendida y la versión de Windows permite abrir simultáneamente 225 tablas, esto permite mejorar el desempeño en aplicaciones complejas donde SQL SELECT accesan muchas tablas.

FoxPro no puede procesar directamente datos ajenos o almacenar tablas en diferentes formatos como puede hacer Access. De alguna manera el

APPEND FROM , COPY TO, IMPORT y EXPORT, tiene cláusulas que convierten tablas.

FoxPro ofrece un número de nuevas características para Windows. Eso incluye tareas para las utilerías de Power Tools, objetos Bitmap para el constructor de pantallas y generador de reportes, un "Graficador Mágico" y un revisor de ortografía.

La versión Windows permite especificar fonts y tamaños de font para el constructor de pantallas. El editor de texto de FoxPro para Windows soporta el arrastre y el encimamiento para mover y copiar y puede indentar o desindentar un bloque de texto en una sola operación. Una ventana de barra de estatus ocupa un pequeño espacio de pantalla pero despliega más información de la que ofrece la barra de estatus de la versión para DOS.

Ambas versiones de FoxPro, la 2.5 para DOS o Windows ofrecen un desempeño espectacular aunado a un buen ambiente de desarrollo. Los desarrolladores que usan manejadores de base de datos que están usando otros productos están intentando cambiar a FoxPro.

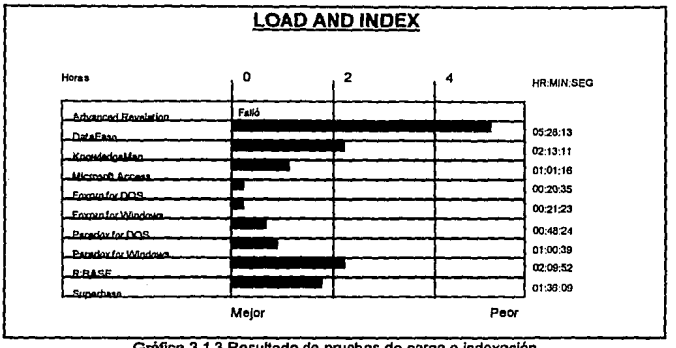
Prueba de Puntos Críticos (Benchmarks)

A continuación se muestran gráficas de pruebas de puntos críticos obtenidas de diferentes manejadores de Bases de Datos en los mismos ambientes de configuración tanto para multiusuario como para usuarios privados. Para usuarios privados se usó un equipo Compaq 386 con 5MB de RAM con un servidor Compaq 486/33 con 32 MB de RAM y 1.6 GB de

almacenamiento en disco duro usando un controlador IDA (Intelligent Drive Array). En ambiente de Red se usó una configuración Token Ring (16-megabit-por-segundo) NetWare 3.11 y 36 estaciones de trabajo con cargas balanceadas simétricamente en cuanto a tráfico, con procesadores 386/SX con 5 MB de RAM. En el Servidor se instaló DOS mientras que en las estaciones de trabajo se instalaron los productos bajo ambiente Windows en la versión 3.1.

Las pruebas de LOAD INDEX (Carga e Indexación) de qué tan rápido se accesa una Base de Datos de cuatro tablas de 100,000 renglones por tabla. Dos tablas contenían dos índices cada una, una tabla contenía cuatro y el resto de las tablas contenía cinco. Cada tabla tenía distintas características de distribución de datos. Una con valores únicos para cada columna, otra con el 10% de valores únicos para cada columna y la tercera con 100 valores distintos para cada columna.

FoxPro para DOS y FoxPro para Windows resultaron ser los que ofrecieron las mejores condiciones de operación, rendimiento y facilidad, completaron la prueba en 21 minutos como se muestra en la gráfica siguiente, el producto que más se le acercó, tardó la mitad del tiempo; fue Paradox para DOS. DataEase fue el producto más lento, que tan solo en la indexación se tardó 5.5 Hrs. Paradox para Windows (Gráfica 3.1.3), necesitó de dos pasos para el proceso de importación, mientras que la versión para DOS tuvo limitaciones para agregar datos a la tabla existente.

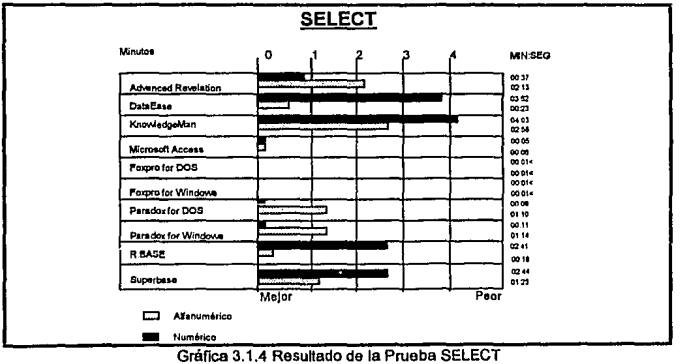


Gráfica 3.1.3 Resultado de pruebas de carga e indexación

Una de las pruebas llevadas a cabo fue la de SELECT, que trata de qué tan rápido se obtienen mediante "querys" paquetes de registros de la Base de Datos donde el 10% de los renglones son extraídos. Se desarrollaron dos tipos de querys, selección numérica usando llave primaria de la tabla y rango secuencial como criterio de búsqueda. La selección alfanumérica se desempeñó exactamente igual que la selección de texto usando llave secundaria indexada.

Tres de los productos probados (Microsoft Access, FoxPro para DOS, y FoxPro para Windows), retornaron una lista de apuntadores en memoria de los datos seleccionados y no el valor actual de las columnas especificadas, esto es más rápido. También las pruebas se desarrollaron de manera que la operación de selección se solicitó instantáneamente, algunos productos requirieron tiempo adicional para el regreso del dato actual si la operación

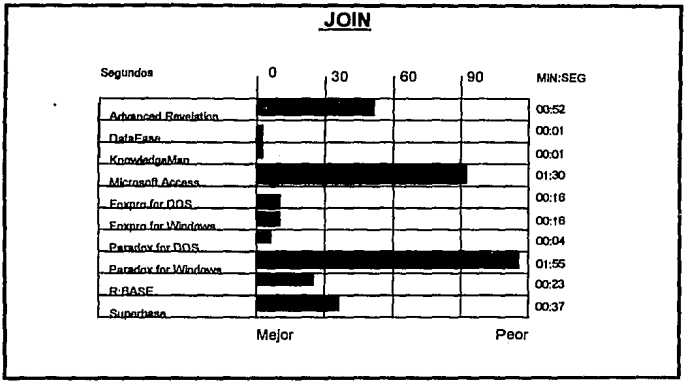
se desarrollaba sobre el conjunto del resultado (Gráfica 3.1.4). Paradox y R:BASE retornaron los datos mientras ejecutaban el query, no requirieron tiempo adicional de acceso a la base de datos. Las dos versiones de Paradox mostraron extremadamente rápido la porción numérica de la selección de prueba, el tiempo de obtención mediante el valor alfanumérico mostró un uso ineficiente de la llave secundaria. Con Paradox, si el conjunto obtenido en el resultado del query es menor a aproximadamente el 8% de la tabla de entrada, el índice secundario ayuda al desempeño. El resultado de las pruebas mostró que el valor alfanumérico retornaba el 10% de la tabla.



Gráfica 3.1.4 Resultado de la Prueba SELECT

Para la prueba JOIN (Intersección) se obtuvieron resultados donde generalmente se procesaban rápidamente. El desempeño dependió del esquema utilizado.

En DataEase, Knowledgeman y Paradox para DOS el JOIN se realizó excepcionalmente (Figura 3.1.4). En contraste, en Paradox para Windows y Access respondieron en tiempos muy lentos, los demás manejadores por los tiempos que tardaron, no se pueden considerar.

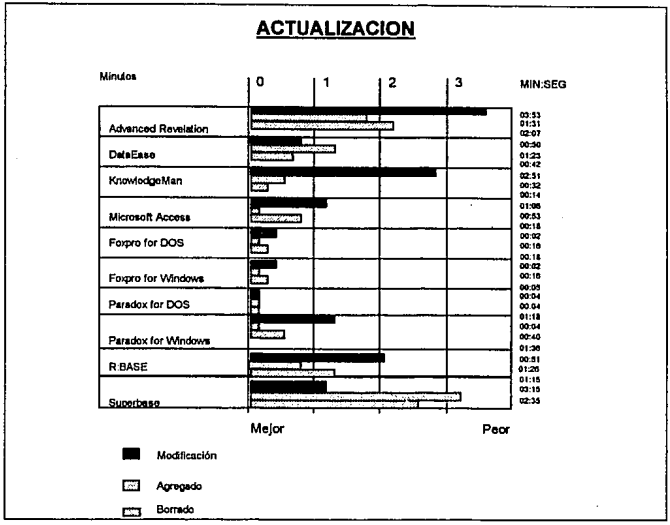


Gráfica 3.1.4 Resultados de la prueba JOIN

Para las pruebas de UPDATE (Actualización) se desarrollaron modificaciones, inserciones y borrado de registros. Primero se salvaron un conjunto de 1000 renglones en una tabla temporal. Después se actualizaron

el rango correspondiente de registros y se cambió el valor de la llave primaria, lo cual forzó a que la base de datos actualizara el índice. Se midió la rapidez cuando se insertaron registros en la tabla, se insertaron cien renglones distribuyéndolos, finalmente se borraron mil registros.

Lo más difícil de las tres operaciones fue la actualización. Advanced Revelation y KnowledgeMan se llevaron un tiempo excepcionalmente largo para esto, debido a que requirió reorganización de índice. Paradox para DOS proporcionó mucha eficiencia en las tres operaciones, complentando cada una en cinco segundos. Ambas versiones de FoxPro también proporcionaron relativamente mucha rapidez en la manipulación de las tareas de modificación (Gráfica 3.1.5).

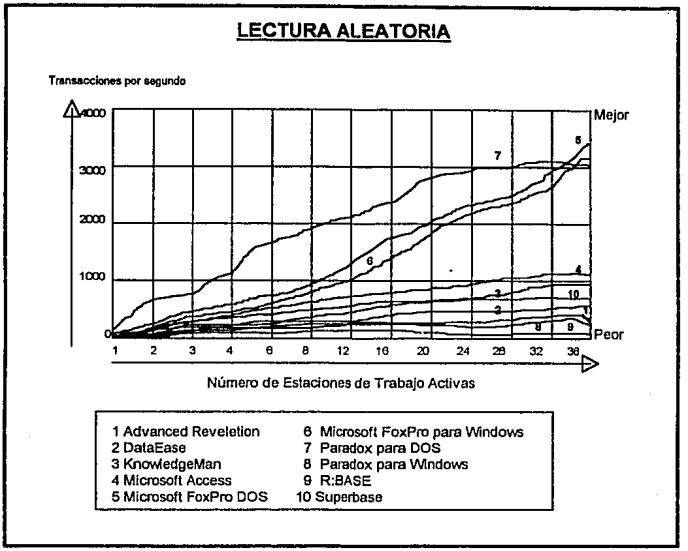


Gráfica 3.1.5 Resultados de la prueba de Actualización

La prueba de RANDOM READ (Lectura Aleatoria) nos da la idea del número máximo de concurrencias que pueden ocurrir en cada paquete de datos manejado. En esta prueba cada estación de trabajo seleccionaba aleatoriamente renglones de la misma tabla y los descargaba directamente en la salida del dispositivo. Los registros eran accesados en modo "browse" para maximizar la cantidad de concurrencias.

La prueba se ejecutó durante 10 horas 10 minutos. Se terminó donde ocurrían puntos de concurrencia, se incrementó el número de estaciones de trabajo gradualmente, agregando estaciones de trabajo en intervalos de 10 minutos hasta 36 o hasta que el Server ya no podía soportar más conexiones. Los puntos de concurrencia fueron calculadas en transacciones por segundo en intervalos de 10 minutos.

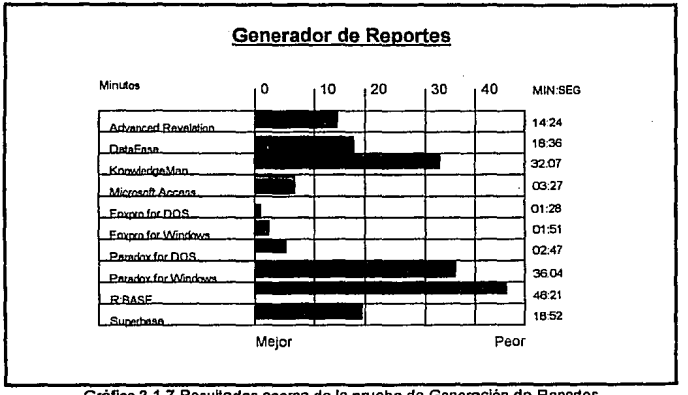
Tres productos (FoxPro para DOS, FoxPro para Windows, y Paradox para DOS) se pueden considerar, ya que cada uno estuvo por encima de las 3000 transacciones por segundo con 36 estaciones de trabajo cargadas. También Paradox para DOS lo logró con 32 estaciones de trabajo, las curvas de puntos de concurrencia para ambas versiones de FoxPro permanecieron esencialmente lineales con 36 estaciones de trabajo. Access también se desempeñó consistentemente en la prueba de Random Read, ejecutando 846 transacciones por segundo con 36 estaciones de trabajo. Paradox para Windows se desempeñó frustrantemente, llegó al límite con 6 estaciones de trabajo y solamente logró ejecutar 60 transacciones por segundo (Gráfica 3.1.6).



Gráfica 3.1.6 Resultados de la prueba de Lectura Aleatoria

Para la prueba de REPORT WRITER (Generación de Reportes) una estación de trabajo imprimió un reporte mientras otras 8 desarrollaban lecturas aleatorias de prueba. Para producir el reporte, el paquete ejecutó un JOIN en dos tablas de 100,000 renglones y generó un resumen de 1000 renglones usando cálculos de fecha, valores máximos y mínimos, subtotales y totales. Los resultados fueron impresos en un archivo ASCII sobre las estaciones de trabajo que generaron el reporte.

De nuevo, ciertos manejadores se desempeñaron mejor que el resto. Ambas versiones de FoxPro, Paradox para DOS, y Access produjeron el reporte considerablemente más rápido que los otros manejadores. Cada versión de FoxPro completó la prueba de bajo de los 2 minutos. KnowledgeMan y Paradox para Windows se llevaron arriba de media hora para completar el reporte y R:BASE fue el más lento, tomando 46 minutos (Gráfica 3.1.7).



Gráfica 3.1.7 Resultados acerca de la prueba de Generación de Reportes

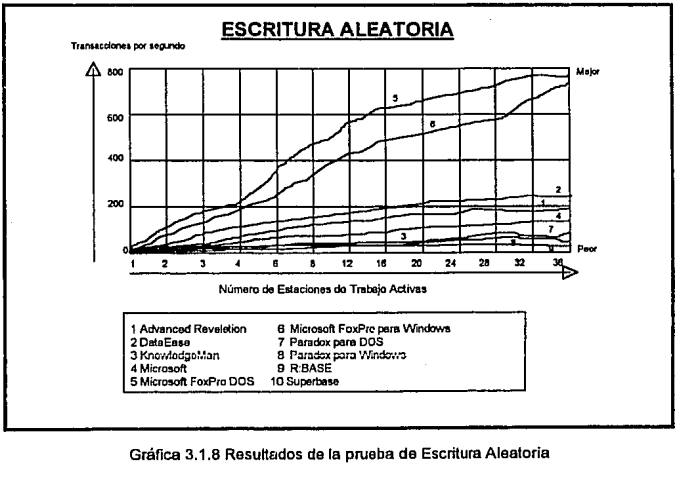
La prueba RANDOM WRITE (Escritura Aleatoria) nos da una idea del número máximo de concurrencia que pueden ocurrir. En esta prueba cada estación de trabajo actualizó aleatoriamente renglones de la misma tabla (Gráfica 3.1.8). Se modificaron valores de un campo no llave inexado de modo que ambos renglones seleccionados y el índice debían ser

actualizados. Los renglones eran accesados en modo de actualización, lo cual permitió compartir datos en la tabla, pero no permitió el acceso a ninguna otra estación de trabajo al renglón que estaba concurrentemente siendo actualizado. Se ejecutó la prueba por 2 horas 10 minutos. Para determinar donde ocurrían los puntos de concurrencia, se incrementó el número de estaciones de trabajo gradualmente, se agregaron estaciones en intervalos de 10 minutos hasta completar 36 estaciones corriendo o hasta que el Servidor no soportara más conexiones. Los puntos de concurrencia se calcularon en transacciones por segundo en intervalos de 10 minutos. El tiempo de proceso no estaba considerado en las repeticiones de la prueba, el cargado de la red produjo que el número de estaciones provocaran una situación de la vida real.

Ambas versiones de FoxPro estuvieron por encima de las demás. Las gráficas de desempeño demuestran que los puntos de concurrencia se nivelaron justamente en 36 estaciones de trabajo con cerca de 770 transacciones por segundo, DataEase resultó ser el tercer lugar en desempeño en esta prueba.

Ambas versiones de Paradox que se usaron en esta prueba resultaron estar en los últimos lugares de desempeño, para obtener un registro con candado de la misma tabla, Paradox para Windows debía adquirir secciones críticas encadenadas sobre el archivo de semáforo. Para requerimientos de concurrencia en la misma tabla, los clientes de Paradox debían accesar el archivo de semáforo en un proceso serial creando cuellos de botella. Paradox DOS llegó al límite con tres estaciones de

trabajo con un total de puntos de concurrencia de 42 transacciones por segundo, mientras Paradox para Windows llegó al límite con sólo 5 transacciones por segundo durante la prueba, Superbase dio una muestra muy pobre en la prueba de escritura aleatoria, porque estuvo obligado a usar una tabla encadenada en vez de usar niveles de renglones encadenados, el producto puede no completar satisfactoriamente la prueba con niveles de renglón encadenados.



Gráfica 3.1.8 Resultados de la prueba de Escritura Aleatoria

CPU	80386 en adelante
Memoria	4MB
Mouse	Requerido
Sistema Operativo	Windows 3.0 o mayor

Figura 3.1.1 Configuración mínima para poder usar la versión 2.5 de FoxPro

MANEJADORES DE BASES DE DATOS
MICROSOFT ACCESS
MICROSOFT FOXPRO FOR WINDOWS
MICROSOFT FOXPRO FOR DOS
ADVANCED REVELATION
DATAEASE
KNOWLEDGEMAN
PARADOX FOR DOS
PARADOX FOR WINDOWS
R:BASE
SUPERBASE

Figura 3.1.2 Relación de productos involucrados en los Benchmark's

NOTA:
En la figura 3.1.1 se muestra la configuración de equipo mínima necesaria para poder usar FoxPro. En la Figura 3.1.2 se muestra mediante una tabla todos los productos involucrados en las pruebas.

III.1.2 Evaluación de datos

La información requerida para el análisis de datos del Sistema de Tarjetas de Crédito, contempla una base de datos principal, la cual es actualizada en forma periodica y constante por cada cliente, como son pagos, disposiciones de efectivo y ventas, las cuales son realizadas en sucursales y/o cajeros automáticos.

Asimismo, presenta procesos internos de operación, los cuales requieren ser procesados en el día de corte que refleje el producto, siendo actualizaciones en saldos del cliente, para la emisión de estados de cuenta, los cuales son: intereses, capital, saldo vencido, meses vencidos, comisiones, gastos de cobranza, etc.

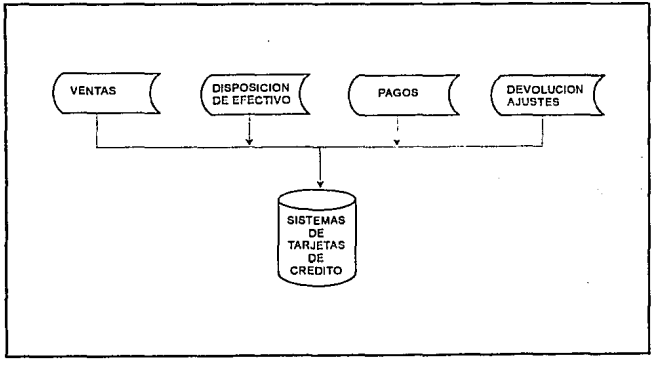


Figura 3.1.3 Alimentación de datos para el Sistema de Tarjetas

En forma aunada, contempla varias líneas de producción, debido al volumen de información y a la interacción de diversos procesos requeridos por la operación de cada producto ó línea en específico, ya que difieren en base a su constante demanda en el mercado así como procesos de corte (actualiza saldos de estado de cuenta).

Dichas líneas de producción son las siguientes:

- **Línea 1** Tarjetas de Débito.
- **Línea 2** Tarjetas de afinidad
- **Línea 3** Tarjetas visa nacional
- **Línea 4** Tarjeta terracota
- **Línea 5** Tarjetas bancos asociados
- **Línea 6** Tarjetas internacionales
- **Línea 7** Tarjetas privadas
- **Línea 8** Tarjetas comercializadoras

Lo anterior implica procesar en forma independiente la base de datos de tarjetas de crédito, en base a la línea y producto de la tarjeta que se requiera para su extracción de información.

CARACTERÍSTICAS DE CAMPOS

Con el objeto de conocer en forma más amplia las características que conlleva el inventario de datos, se analizaron algunos campos esenciales, los cuales se requerirán en la planteación de explotación de información del Sistema de Desempeño de Tarjetas de Crédito.

Número de cuenta

Identifica el número de cliente de la tarjeta, teniendo las siguientes características:

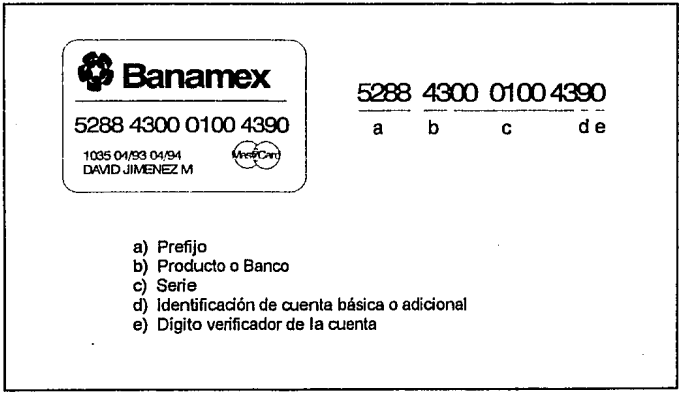


Figura 3.1.3.a Significado del número de cuenta en una Tarjeta de Credito

Tipo: Numérico

Longitud: 16

9999-99-99999999-9-9

a b c d e

a) Prefijo

Identificación de plástico

b) Producto

Se consideran únicamente los correspondientes a tarjetas de crédito.

c) Serie

Número consecutivo

d) Identificación de cuenta básica o adicional

Se consideran para nuestro análisis solamente cuentas básicas, las cuales son identificadas por los números 9,8,7. Ya que en forma inicial de una apertura de cuenta básica, el número inicial es 9, y en caso de reposición de tarjeta de plástico cambia en número a 8, así sucesivamente hasta el 7; lo anterior repercute en el dígito verificador, siendo los dos únicos números que cambian de una cuenta.

e) Dígito verificador de la cuenta

Ciclo de corte

Refleja el día correspondiente al mes en curso, en el que se realiza un cierre de transacciones (ventas y disposiciones en efectivo) por cliente, las cuales se reflejarán en la emisión del estado de cuenta respectivo, asimismo se calcularán importes de intereses, gastos de cobranza, capital, etc.

Asimismo, los diferentes productos difieren en ciclos de corte.

Límite de crédito

Corresponde el monto de crédito autorizado por una Institución Bancaria, asimismo dicho monto se restringe en un límite máximo y un límite mínimo por cada producto, los cuales son modificados en base a los requerimientos del producto y el mercado.

Situación de la cuenta

Identifica la situación del cliente, dado por pagos vencidos, robos, extravíos de tarjeta, sobregiros, etc.

La clasificación de situaciones a darse en una tarjeta de crédito, depende de las características del producto y de la operación, pero en forma global se identifican de la siguiente forma:

Clave de situaciones de una cuenta:

situación	identificación
N	cuenta normal
C	cancelada sin saldo
D	delincuente
E	cancelada con saldo
W	congelada
O	sobregiro
U	robo
L	extraviada
P	problemas de cobro

Cabe mencionar que las situaciones anteriores, son algunas de las que se pueden registrar, pero en base a nuestro análisis de tarjetas de crédito son las básicas del objetivo del Sistema.

Fecha de apertura

Identifica la fecha de inicio de crédito, la cual se encuentra en formato año y mes (AAMM), ejemp. 9402.

Fecha aumento de límite de crédito

Identifica la fecha del último aumento de crédito, el cual se puede dar:

> En forma automática

Se dan en forma periodica a los clientes con características de buen uso de la tarjeta de crédito.

> Por operador

Se dan en sucursal ó centros facultados, donde el cliente determina un ingreso mayor que el registrado en forma inicial, y requiere el aumento de crédito bajo un estudio de análisis económico.

Formato: AAMM (año, mes).

Fecha de boletinado

Identifica la fecha de una cuenta que debido a sus problemas de cobro o en su caso robo ó extravió de la misma, es registrado en el boletín de prevención que se entrega periódicamente a los centros comerciales, tiendas de autoservicio, etc., para la detección y recopilación de las tarjetas que presenta ésta situación anormal.

Formato: AAMM

Fecha último pago

Identifica la fecha del último pago realizado por el cliente, y el cuál es aceptado correctamente, ya que los pagos se realizan por sucursales, cajero permanente y por banco en su casa.

Formato: AAMMDD (año, mes, día)

Fecha de última transacción

Identifica la fecha de última transacción realizada por el cliente, las cuáles pueden ser una venta o una disposición de efectivo.

Cabe mencionar que esta fecha difiere de la fecha de último pago.

Formato: AAMMDD (año, mes, día)

Fecha de turnación al abogado

Identifica la fecha en que fué turnado una cuenta a cobranza jurídica, y asimismo se le asigna un número de abogado.

Formato: AAMM

Fecha de sobregiro

Identifica la fecha del último sobregiro registrado por el cliente, esto es que el saldo deudor sea mayor que el crédito otorgado.

Formato: AAMMDD

Saldo vencido

Registra los importes vencidos de los mínimos a pagar de cada estado de cuenta, por lo que se refleja cada mínimo no pagado en un campo correspondiente al mes no liquidado, hasta 6 ó mas meses vencidos:

Pago vencido mes 1	mensualidad vencida en un mes
Pago vencido mes 2	mensualidad vencida en dos meses
Pago vencido mes 6	mensualidad vencida 6 ó mas meses

Por lo que el **importe total de mensualidades vencidas** es =
(pago vencido 1)+(pago vencido 2)+ . . . .+(pago vencido)

Asimismo, el importe que se requiere para nuestro análisis es el importe total vencido.

Saldo actual

Registra el último saldo de las transacciones (ventas y disposiciones en efectivo) que haya registrado en el último corte, así como los **intereses generados, comisiones, gastos de cobranza** en su caso, etc.; más las transacciones realizadas en los días posteriores al mismo corte.

Saldo actual del extranjero

Registra el último saldo de las transacciones (ventas y disposiciones en efectivo) que haya registrado en el último corte, así como los **intereses generados, comisiones, gastos de cobranza** en su caso, etc.; más las transacciones realizadas en los días posteriores al corte, pero con la diferencia de que se realizaron en el extranjero (uso exclusivo de tarjetas de crédito internacionales).

Cabe mencionar que este importe es registrado en moneda nacional.

Por lo anterior, el saldo total de una cuenta es = a la suma de (saldo actual) + (saldo del extranjero).

Por lo que respecta a nuestro análisis de información, se requiere la extracción de los dos campos en forma independiente, ya que las tarjetas nacionales no pueden registrar transacciones en el extranjero, y en caso de querer validar esta situación, una de las formas de detectarlo sería en el campo de transacciones en el extranjero.

Capital

Registra el último saldo de las transacciones (ventas y disposiciones en efectivo) que haya registrado en el último corte, más las transacciones realizadas en los días posteriores al mismo corte.

Intereses

Registra el cálculo de intereses generados por el período mensual de uso de la tarjeta de crédito, los cuales son:

Intereses de días financiados.

Se calcula sobre el importe del saldo total no pagado, así como los días financiados, hasta el día de corte de la tarjeta.

Intereses moratorios.

Se calcula sobre las mensualidades vencidas.

Comisiones

Registra las comisiones generadas por disposiciones en efectivo, cobro de anualidad de la tarjeta, reposiciones o emisión cíclica por vencimiento de la misma.

Gastos de cobranza

Registra los gastos ocasionados por la cobranza, ésta implica desde la primera mensualidad vencida.

Dichos gastos son por papelería y servicio especial (minicartas que se envían al cliente avisándole el vencimiento de su pago), recordatorios, etc.; los cuales son muy independiente a la cobranza jurídica, ya que estos gastos de cobranza se efectúan hasta la tercera mensualidad vencida.

I.V.A.

Registra el i.v.a. correspondiente a los importes de intereses y comisiones.

Número de mensualidades vencidas

Registra el número de meses vencidos, que el tarjeta-habiente refleja de sus pagos de acreditados.

Sucursal

Identifica el número de sucursal que promovió la apertura de la cuenta, asimismo el responsable de los parámetros de apertura, ya que la sucursal debe de recopilar y determinar un análisis de crédito los clientes prospectos a crédito, como son solvencia económica, residencia en un trabajo, mayoría de edad, etc.

Asimismo, cada sucursal se agrupa en regiones estatales, las cuales son regidas por una responsabilidad superior, que debe estar supervisando la correcta cartera vigente y vencida de su zona respectiva.

Importe recuperado

En base de que los tarjeta-habientes que reflejan más de 3 mensualidades vencidas, quedan marcados como trámite de "cobranza jurídica", los pagos realizados en forma posterior son importes considerados como saldo recuperado del saldo deudor.

Identificación de cuenta boletinada

En caso de ser boletinada la cuenta, debido a problemas de cobro, robo ó extravío de la tarjeta, se identifica el registro del cliente como cuenta boletinada en caso de tener valor de 1 en éste campo.

Número de abogado

Cuando la cuenta es traspasada formalmente a la cartera de cobranza jurídica, debido a la falta de recuperación del saldo, se le asigna un abogado que conlleve los trámites jurídicos para su recuperación, por lo que dicho abogado se encarga de la cobranza en forma directa, y toda recuperación del saldo deudor es entregada a la Institución Bancaria.

Asimismo, un abogado presenta una serie de clientes en cartera vencida, por lo que se presenta un registro de abogados autorizados por una Institución Bancaria, ya que por cada recuperación registrada por parte del abogado, éste manifiesta una comisión respectiva del importe.

TRANSFERENCIA DE INFORMACIÓN

La información mencionada, se registra en las siguientes archivos de la base de datos:

Tarjetas-saldos

Archivo: secuencial

Acceso: indexado

Llaves de acceso: número de cuenta

Campos a extraer: número de cuenta, límite de crédito, situación de la cuenta, fecha de apertura, fecha de último aumento de crédito, fecha de boletinado, fecha último pago, fecha de turnación al abogado, fecha última transacción, fecha de sobregiro, saldo vencido (sumatoria de meses vencidos), saldo actual, saldo del extranjero, capital, intereses, gastos de cobranza, mensualidades vencidas, ciclo de corte, sucursal, nombre del tarjeta-habiente, dirección1 (calle), dirección2 (colonia), dirección3 (estado), código postal, identificación de cuenta boletinada, importe recuperado, número de abogado.

Restricciones:

Solo se extraerán del archivo, los registros que cumplan las siguientes características:

- Una ó más mensualidades vencidas
- Saldo actual + saldo del extranjero sea mayor que el límite de crédito
- Productos de tarjetas de crédito que estén en :
  - Línea 3 Tarjetas visa nacional
  - Línea 4 Tarjeta terracota
  - Línea 6 Tarjetas internacionales
  - Línea 7 Tarjetas privadas
  - Línea 8 Tarjetas comercializadoras

Tarjetas-abogados

Archivo: secuencial
Acceso: indexado
Llaves de acceso: número de abogado
Campos a extraer: número de abogado, nombre del abogado, comisión, zona de cobertura.

Restricciones:

Se extraerán del archivo todos los registros.

Tarjetas-sucursal-región

Archivo: secuencial

Acceso: indexado

Llaves de acceso: número de sucursal

Campos a extraer: número de sucursal, nombre de sucursal, número de región, nombre de región.

Restricciones:

Se extraerán del archivo todos los registros.

Especificaciones generales

De los registros extraídos de los archivos mencionados, se debe excluir los siguientes características:

> Excluir los header (registros de etiqueta)
> Campos filler

Asimismo, los archivos que se generen por cada línea así como cada archivo (saldos,abogados y sucursales), deberán estar grabados en un archivo secuencial, sin ninguna característica de archivo indexado.

III.1.3 SEGURIDAD

Durante algunos años, la computadora ha sido un buen pretexto para realizar publicidad sensacionalista acerca de experiencias en varios sentidos respecto a su uso. Actualmente las empresas especializadas en servicios de asesoría para la seguridad en computación han proliferado debido a que también los riesgos se incrementan.

Si se analiza el asunto con atención, resulta claro que se ha desarrollado un área nueva de preocupación gerencial: el abuso en el manejo de las computadoras o el desastre a causa de robo, fraude, sabotaje o interrupción en las actividades de cómputo.

La conciencia sobre el problema puede surgir temporalmente en caso de desastre o de abuso en los recursos de computación, pero la efectividad como rutina es, cuando mucho, esporádica. Las actitudes más frecuentes son: "estamos satisfechos con la seguridad con nuestra computadora" y " no es probable que eso suceda aquí".

En contraste con los antecedentes sobre la seguridad en computación, generalmente es superficial, existen ciertos factores que han modificado el contexto dentro del cual se usan las computadoras y han aumentado el nivel de seguridad que se requiere:

- Concentración del procesamiento y aplicaciones más grandes y de mayor complejidad, las cuales son parte integral de la Institución
- Dependencia en el personal clave
- Desaparición de los controles tradicionales
- Terrorismo urbano e inestabilidad social
- Mayor conciencia de los proveedores de computadoras

A continuación se tratan a detalle cada uno de estos aspectos.

Concentración del procesamiento y aplicaciones más grandes y de mayor complejidad.

La principal causa del incremento en los riesgos de computación probablemente sea el aumento en la cantidad de aplicaciones que se da a las computadoras y la consecuente concentración de la información y procesamiento. Además, la tendencia creciente hacia la incorporación de sistemas mayores y más complejos que incluyen el procesamiento en línea y en tiempo real, así como el uso frecuente de bases de datos o archivos sofisticados constituye un problema adicional. El uso de los sistemas de bases de datos está cada vez más difundido y gran cantidad de información confidencial se almacena de este modo, por ejemplo, en oficinas de crédito y dependencias gubernamentales.

En contraste con una organización que usa un archivo manual donde la información se diversifica a través de toda la institución, otra, que usa

ampliamente la computación, cuenta con la información y los programas concentrados en las manos de pocas personas. Como consecuencia de ello, la institución computarizada corre el riesgo de sufrir "amnesia corporativa" debido a algún desastre en las computadoras, y de que sobrevenga una suspensión prolongada de procesamiento. La mayor consciencia de este riesgo y algunos desastres publicitados ampliamente genera mayor preocupación por la seguridad en las computadoras.

Dependencia en el personal clave

Además del peligro de algún desastre, existen otras situaciones potencialmente riesgosas de las cuales la más importante es, quizá, depender de individuos clave. Si bien es cierto que la situación existe en todas las funciones de una institución, la relativa novedad de la experiencia con computadoras y la brecha respecto a la comunicación ante los técnicos expertos y la gerencia, trae problemas específicos.

La dependencia en individuos clave, algunos de los cuales poseen un alto nivel de desempeño técnico, con frecuencia pone a la institución en manos de relativamente pocas personas. Los programas de computadora, en especial, se vuelven cada vez más complejos, por lo que una persona provista del conocimiento técnico de la programación y/o del equipo y de "contactos" o debilidades, se encuentra invariablemente en una posición de control única. Este tipo de conocimiento ha conducido a situaciones donde las empresas se han visto expuestas al chantaje o a la extorsión.

La amenaza no solo se restringe a este tipo de abuso. Este personal especializado con frecuencia posee el conocimiento único y no registrado de las modificaciones o el funcionamiento de los programas. La supervisión y el control de su trabajo resulta difícil como lo es el conocimiento de lo que sí funcionaría sin contar con las habilidades del especialista.

Desaparición de los controles tradicionales

La importancia de las habilidades técnicas se fortalece con la desaparición de los controles tradicionales y de las auditorías en muchas instalaciones. La brecha en la comunicación entre el personal técnico, los gerentes de línea y el personal externo, como los auditores antes mencionados, suele causar dificultades para formular las implicaciones prácticas de este desarrollo en los términos comerciales convencionales.

La brecha en la comunicación también se extiende a otros expertos como el personal de seguridad. Los gerentes de seguridad rara vez son expertos en computación, por lo que afrontan dificultades al aplicar sus evaluaciones ya establecidas sobre seguridad y riesgo a la actividad de las computadoras.

Muchas de las nuevas y extensas aplicaciones omiten las auditorías tradicionales y los controles impresos por razones de volumen. Las aplicaciones contienen verificadores automáticos que aseguran la integridad de la información que se procesa. Este gran cambio en el criterio sobre el control de los empleados y las brechas respecto a la comunicación, crean situaciones de seguridad totalmente diferentes.

Huelgas, Terrorismo Urbano e Inestabilidad Social

El nivel actual de riesgo en computación se debe revisar también dentro del contexto de inestabilidad y terrorismo urbanos en muchas partes del mundo. Ha habido ataques físicos a instalaciones en E.U y Europa. Sin embargo, algunas veces se trata de la incursión de personal interno y no de agitadores. Este tipo insidioso de riesgo es, en potencia, la fuente de más alto perjuicio para la institución. Este riesgo, solo, genera una amplia posibilidad de nuevas amenazas ante las cuales hay que responder.

Mayor conciencia de los proveedores

Por último, la investigación y el apoyo por parte de los proveedores se ha incrementado en el área de la seguridad. Hasta hace pocos años este tema no constituía motivo de gran preocupación para los proveedores, pero la conciencia acerca de la exposición a los riesgos los ha obligado a destinar presupuestos considerables para la investigación sobre la seguridad en computación. Como resultado, se dispone de un mayor número de publicaciones de alta calidad para los usuarios, lo que permite mejorar la estructura y el enfoque para la seguridad de las computadoras; asimismo, ha intensificado el interés por reducir en forma progresiva el riesgo causado por un desastre en las computadoras.

ENFOQUES TRADICIONALES SOBRE LA SEGURIDAD EN COMPUTACIÓN

Los progresos descritos anteriormente han dado como resultado una preocupación creciente y una acción destinada a reducir la vulnerabilidad. De manera tradicional, se ha presentado atención a las áreas donde los resultados se pueden ver. De este modo, las principales áreas en que se ha destacado la seguridad, como se menciona anteriormente, han sido:

1. La seguridad física, que incluye la seguridad de acceso y contra incendios.

2. La seguridad de los datos y los archivos.

En consecuencia, actualmente existen muy pocas instituciones donde no se requiera el uso de procedimientos completos de control de acceso, los cuales incluyen personal de seguridad y sistemas de control y tarjetas de acceso. la labor de copiar información de los archivos y ubicarlos en un lugar distante para su almacenamiento se realiza cuidadosamente. Aunque, en efecto estos aspectos son el elementos de un enfoque general de la seguridad en computación, no crean por sí solos un ambiente adecuado. Muchas instituciones confían ciegamente la seguridad de sus computadoras a estos elementos.

Otra característica de esta perspectiva limitada es que resulta raro encontrar un lugar donde se haya llevado a cabo una revisión completa y

exhaustiva de la seguridad. Por lo general, la toma de conciencia acerca de la seguridad comienza con la preocupación de la alta gerencia a causa de alguna falla de seguridad publicitada o de algún incidente menor ocurrido dentro de la institución, o de la visita de algún asesor en seguridad o vendedor de cajas fuertes a prueba de fuego, equipo de extinción de incendios, etc. La consecuencia es un enfoque de la seguridad muy del tipo "sabor del mes": este mes le toca a las cajas de seguridad, el siguiente al nuevo sistema de acceso, luego al equipo de extinción, y así sucesivamente.

CONCEPTO DE SEGURIDAD TOTAL EN COMPUTACIÓN

En estos términos, se requiere un enfoque amplio que abarque cierto número de aspectos relacionados entre sí de manera metódica. Hay dos grandes áreas que se deben incorporar a tal enfoque:

1. Aspectos administrativos
2. Aspectos técnicos y de procesamiento

Los aspectos clave se pueden resumir de la manera siguiente:

Elementos administrativos:

- Política definida sobre seguridad en computación
- Organización y división de las responsabilidades
- Seguridad física y contra incendios

- Políticas hacia el personal
- Seguros

Elementos técnicos y de procedimiento:

- Seguridad de los sistemas (equipo y programación, redes y sistemas terminales)
- Seguridad de las aplicaciones, incluyendo la seguridad de los datos y los archivos
- Estándares de programación y operación de los sistemas
- Función de la auditoría interna y externa
- Plan y simulacro para desastres

La revisión cuidadosa de estas áreas revelará que ninguna de ellas es, por sí sola de importancia exclusiva: en una instalación específica uno puede tener mayor relevancia y, por eso, requerir mayor atención; sin embargo, si se excluye a alguna de estas áreas se dejarán vacíos en el manejo y control de la seguridad. El enfoque desarrollado que abarca en forma metódica todas estas áreas se remite al "Concepto de Seguridad Total en Computación" (figura 3.1.5).

La complejidad creciente y el alcance del uso de la computación ha propiciado que la información se concentre en manos de unas cuantas personas. El punto de vista tradicional otorga mayor atención a los aspectos de seguridad "visibles", como el acceso físico, la extinción de incendios y la seguridad de los archivos. La seguridad efectiva en computación requiere la

revaloración de un amplio número de aspectos descritos del "Concepto de Seguridad Total".

Aspectos a considerar sobre la seguridad física

- Ubicación y construcción del centro de cómputo
- Disposición
- Aire acondicionado
- Suministro de energía
- Riesgo de inundación
- Controles de Acceso (de terceras personas, alarmas contra robos, tarjetas de acceso y gafetes)
- Detección y protección contra incendios
- Mantenimiento

SEGURIDAD DE LOS SISTEMAS

La seguridad de los sistemas se refiere de manera principal a la seguridad del equipo de cómputo, e incluye:

- El equipo
- Los programas de uso general, es decir, se excluyen los programas de aplicación específica
- Las redes, o sea, las líneas y sistemas de comunicación de datos

- Las terminales y los programas generales directamente asociados

La seguridad de los sistemas constituye una parte muy técnica y compleja de la seguridad en la computación. Se requiere un enfoque metódico para identificar las "puertas falsas", definir controles discretos y garantizar que estos se lleven a cabo y se vigilen. Esta revisión debe incluir el equipo y los programas del computador principal, de redes y de terminales.

El seguimiento del desempeño del computador central, las redes y las terminales es una función administrativa y de seguridad importante. Se deben considerar cuidadosamente los recursos disponibles de los programas existentes o sus modificaciones. Además, se requiere seguimiento e informes de todo intento de acceso indebido a los programas o archivos.

SEGURIDAD DE LAS APLICACIONES

El término "Seguridad de las Aplicaciones" abarca tanto a los componentes de la computadora como a los que no lo son, en cada aplicación. Por parte de la computadora comprende datos, programas y archivos que se procesan en el sistema. Los elementos que no son de la computadora incluyen recolección y entrega de datos e información del archivo maestro para el procesamiento, así como el control de dicha información para garantizar que se procese en forma correcta y su distribución lleven al usuario. Las etapas clásicas de cada sistema implican:

- Iniciación manual de los datos
- Conversión de los datos a un formato aceptable por la computadora, es decir, captura de datos
- Procesamiento
- Distribución de los resultados

En muchas instalaciones se presta mucha atención a los controles tanto en los departamentos de usuarios como los de computadoras. A pesar de ello, debido a que los controles los realizar distintas personas en cada área, existen vacíos respecto a los contactos entre las dos áreas. Un buen ejemplo de esto lo constituyen las modificaciones a los datos de los archivos maestros. Se inicia un cambio no autorizado a un límite de crédito. Es difícil para el personal de preparación de datos y para el de cómputo validar la autoridad de las modificaciones en el archivo maestro que se aceptan para procesamiento, bien sea de manera separada o junto con otros datos de crédito. Se necesita, entonces, alguna forma de control sobre estos registros, esta se logra mediante la verificación de secuencia de los documentos y los controles de valor de las remesas, así como las revisiones manuales contra las listas de la computadora por parte del usuario. Al mismo tiempo, se puede verificar la autorización de los distintos documentos.

Los controles de rutina en estas áreas han sido frecuentes y son ahora efectivos en términos generales. La debilidad siempre surge cuando la disciplina no es lo suficientemente rigurosa. Es conveniente la verificación

manual puntual de las impresiones del archivo maestro para neutralizar la posibilidad de errores o cambios no autorizados.

Un gran problema en todos los sistemas de cómputo es el de control de errores. Existen ciertos puntos clave:

- Todos los errores se deben corregir
- Los errores sólo se deben corregir por el personal autorizado
- La división de la responsabilidad se debe mantener cuando se asigne la autoridad para la corrección de errores

En el terreno práctico, todos estos aspectos presentan dificultades. Se requiere un enfoque metódico y muy riguroso.

Otro punto importante de la relación hombre-máquina es la distribución de los datos. Se debe tener precaución y seguridad en la alimentación de la computadora con los datos de entrada, así como también en el procesamiento dentro del departamento de cómputo. Sin embargo, se puede tener menos precaución en la distribución de informes a los usuarios.

La seguridad respecto a la distribución debe cubrir lo siguiente:

1. La responsabilidad e identificación del personal autorizado para el acceso a los informes.
2. El control sobre los resultados tanto para las operaciones válidas como las frustradas.

3. El control sobre las que fueron copias de los informes corregidos.

Controles del usuario

El usuario tiene la responsabilidad primaria de asegurar que los datos recolectados para el procesamiento estén completos y sean precisos; también se debe asegurar de que todos los datos se procesen y se incluyan en los informes que le regresan. En el análisis final no es aceptable culpar a la computadora de las decisiones que se basen en datos imprecisos.

CONTROLES DE PROCESAMIENTO DE LA COMPUTADORA Y SEGURIDAD DE ARCHIVO

Controles de procesamiento de la computadora

Se trata de los controles que se mantienen dentro del departamento de cómputo. Son el reflejo de aquellos que se mantienen en los departamentos de usuarios pero, en algunos aspectos son más minuciosos por ejemplo, los controles detallados de procesos de los programas.

Los controles dentro del departamento de cómputo son de procedimiento y aritméticos; los procedimiento incluyen:

- División de responsabilidad entre captura de datos y operación
- División de la responsabilidad entre operaciones y archivo
- Registro de evidencias que reflejen la transferencia de registros y datos entre las diferentes funciones
- Control sobre la precisión y distribución de los resultados

El principal propósito de estos controles es garantizar el procesamiento completo y preciso de los datos y archivos con el uso de los programas correctos. Los controles aritméticos son los que garantizan el procesamiento completo y preciso de todos los registros de datos, durante y al final del procesamiento.

Seguridad de los archivos

Un elemento importante y tradicional que se debe considerar en el control del procesamiento es la seguridad de los archivos, la cual abarca lo siguiente:

Almacenamiento de las copias de seguridad

De preferencia la ubicación para el almacenamiento de estos archivos debe estar muy distante de la computadora. En las instalaciones de alta seguridad se toman medidas especiales.

Los procedimientos tradicionales rara vez consideran las medidas de seguridad vigentes cuando se transportan discos o cintas a un lugar lejano

y viceversa. Esto representa un área de riesgo real en muchas instalaciones de alta seguridad. En consecuencia, estas actividades no las debe realizar un solo individuo por su cuenta.

Identificación y control de los archivos

Esto incluye la revisión física de las etiquetas y los registros físicos del movimiento de tales archivos, así como la verificación de identificación de los encabezados de los archivos que realiza el programa en forma rutinaria. En algunas partes, esto se realiza de manera eficiente y se deben incorporar ciertas medidas para mejorar los procedimientos.

Precisión de los archivos

Los estándares de la instalación deben incorporar en los programas, controles detallados de principio a fin y conteo de registros. Se debe incorporar el balance detallado registro por registro y/o los conteos binarios. La verificación e igualación de estos conteos aritméticos se deben balancear con los controles manuales de cada proceso que realizan la función de control.

Acceso físico a los archivos

En muchas instalaciones existe el acceso libre al archivo de cintas. En otros lugares el acceso se restringe pero hay poca protección. La cintoteca

representa una gran concentración de datos y archivos por lo que existe un riesgo considerable de abusos.

Los operadores de cómputo no deben de tener acceso rutinario al archivo de cintas, el cual debe estar bajo el control de otras personas quienes, de preferencia, se encuentren adscritas a alguien que no realice funciones de operación. La disciplina de poner los datos y archivos a la disposición de quien los necesite para cada proceso de producción, resulta difícil, especialmente en la planificación de los horarios para el trabajo imprevisto. Sin embargo, muchas instalaciones han llegado a arreglos adecuados y ahora manejan el problema de manera rutinaria.

El alcance de la seguridad de los datos y los archivos de una aplicación incluye tanto el trabajo de una computadora como otras labores. Se necesita considerar en forma cuidadosa la relación de las actividades de cómputo con las que no lo son. Los controles del usuario no se pueden delegar al departamento de cómputo. El control que se mantiene en el centro de cómputo es en general, muy detallado; en especial respecto a los archivos y programas.

La revisión constante de los controles de la aplicación es una parte importante de la función de auditoría interna.

SEGURIDAD Y PRIVACIA

La seguridad y privacia son importantes debido a que mucha gente en diversos lugares tiene acceso a una red de computadoras. La información almacenada en algunas máquinas de la red puede ser de gran valor para una corporación. Esta no debe perderse, o ser dañada. Es importante proteger la información y los programas de fallas de hardware y software, de catástrofes, de criminales, vándalos, incompetentes, y de gente que pudiera hacer mal uso de ella.

Seguridad se refiere a *la protección de los recursos contra daños y a la protección de la información contra accidentes naturales o intencionales de personas no autorizadas, así como a la prevención de modificaciones de información no autorizadas o de su destrucción.*

Privacia se refiere a *los derechos de los individuos y organizaciones para determinar para sí mismos cuando, como, y que información acerca de ellos es transmitida a otros.*

Aunque la tecnología de la privacidad esta íntimamente relacionada a la seguridad, la privacidad va más relacionada con los centros de cómputo y las redes. Para mantener la privacia de la información sobre los individuos, las soluciones necesarias tienen que ver con soluciones de carácter técnico. La sociedad depende actualmente de un uso masivo de redes y bancos de datos, por lo cual se requiere de nuevos controles de tipo social y legal si el grado de privacidad de la información desea ser mantenido.

La información puede protegerse en computadoras tan seguramente como si esta estuviera protegida en una caja fuerte. No obstante, la información en muchos sistemas no se encuentra debidamente protegida debido a la insuficiente atención en el diseño o implementación de los procedimientos de seguridad.

La seguridad es una materia altamente compleja debido a que hay muchos aspectos diferentes para ella. El responsable del diseño de la seguridad de un sistema requiere estar familiarizado con todos los aspectos del sistema ya que ésta puede ser atacada o violada de múltiples y diversas maneras. El diseñador de seguridad algunas veces llega a verse demasiado involucrado con un aspecto del diseño y falla al observar otros caminos de violarlo.

Puntos esenciales sobre seguridad

1. Los usuarios de una red deben ser positivamente *identificables* antes de que ellos la usen. (Figura 3.1.4)
2. Los sistemas y posiblemente también la administración de la red debe estar habilitada para checar que sus acciones son *autorizadas.*

Los usuarios de redes deben ser :
- IDENTIFICABLES

Sus acciones deben ser :
- AUTORIZADAS
- MONITOREADAS

La información, el hardware y el software de estar :
- PROTEGIDOS
- BLOQUEADOS

La información debe ser :
- RECONSTRUIBLE
- AUDITABLE
- A PRUEBA DE FORZADURAS

La transmisión debe ser :
- RECUPERABLE A FALLAS
- PRIVADA

Figura 3.1.4 Puntos esenciales de seguridad en la red

3. Sus acciones deberán ser *monitoreadas* de modo que si alguna de ellas hace algo equivocado pueda ser descubierto.

4. La información, el hardware, y el software deben ser *protegidos* del fuego, robo, u otras formas de destrucción.

5. Dichos elementos deben ser *bloqueados* para prevenir su uso no autorizado.

6. La información deberá poder *reconstruirse* debido a que aún cuando se tomen excelentes precauciones, los accidentes algunas veces ocurren.

7. La información deberá ser *auditable.* Las fallas para auditar un sistema de cómputo adecuadamente ha permitido algunos de los crímenes más grandes del mundo.

8. La red y sistemas deberán ser *a prueba de intentos de violación.* Los programadores ingenuos no deberán poder violar los controles de seguridad.

9. La transmisión deberá ser *recuperable a fallas* de modo que cuando ocurran errores o fallas, los mensajes no se pierdan, ni sean doblemente procesados, o irrecuperables de basura.

10. Las transmisiones deberán ser *privadas* con alguna protección contra indiscreciones a través de criptografía.

Niveles de Protección

La Seguridad puede ser representada por un diagrama de niveles como el mostrado en la figura 3.1.5. El nivel de los controles técnicos esta rodeado por aquellos que involucran la seguridad física. Esta se refiere a seguros en las puertas, guardias, alarmas, y otras que involucran la prevención de accesos no autorizados, e incluye las medidas contra fuego, protección de los archivos donde la información se encuentra almacenada, y demás. No es suficiente tener un buen hardware y software si los discos pueden ser robados o las cintas destruidas por el fuego.

El siguiente nivel es aquél que se refiere a los controles administrativos para asegurar que el sistema esta siendo usado correctamente. El staff de programadores y de procesamiento de datos deben estar controlados de modo que no hagan un mal uso del sistema. Deben implementarse salas de cómputo controladas y con procedimientos para prueba de programas. Los controles administrativos son responsabilidad de la sección de procesamiento de datos hacia los departamentos usuarios, dispersado a través de la red, a los auditores, y a la administración en general.

Los niveles en la figura 3.1.5 no están del todo separados. La seguridad física no es irrelevante cuando se están diseñando las técnicas de acceso al sistema. La cuestión de la seguridad física afecta los métodos de seguridad de la transmisión y del sistema que están siendo empleados. Los procedimientos administrativos están muy relacionados con el diseño del sistema, especialmente con sistemas basados en tiempo real o con nodos remotos.

La capa más exterior de la figura 3.1.5 es la más problemática. Cuando la evolución de las redes alcance su punto máximo, la sociedad será muy diferente.

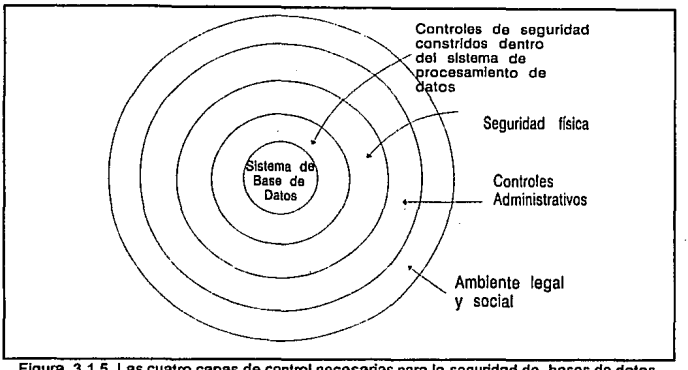


Figura 3.1.5 Las cuatro capas de control necesarias para la seguridad de bases de datos.

Tipos de seguridad

Hay una gran diversidad de tipos de seguridad, la mayoría de ellas relacionadas con los centros de cómputo, independiente de las redes. Catástrofes tales como malversación han resultado muy dramáticas, pero la causa más común de las calamidades de cómputo son el descuido y los accidentes de carácter humano.

Los sistemas distribuidos, si son pobremente diseñados u holgadamente administrados, incrementan la probabilidad de accidentes a mayor descuido. La información, en vez de residir en un centro altamente seguro,

puede estar distribuida entre diferentes localidades con menor protección lo cual incrementa los riesgos de violación o de usos no autorizados.

Con las redes se ha introducido una mayor facilidad para que las personas y máquinas puedan obtener acceso a un centro de cómputo. Es necesario prevenir los accesos y las comunicaciones no autorizadas entre máquinas.

Otro riesgo es que la información transmitida puede ser vista por personas no autorizadas, grabada, desviada, o aún modificada a través de líneas o nodos conmutados.

Aspectos importantes a considerar

1. *Minimizar la Probabilidad de Acontecimientos*

La mayor parte de las precauciones contra fuego deben ser preventivas, y esto es tan importante como los otros aspectos relacionados con seguridad. Se debe estar prevenido contra los desfalcadores desde el principio.

2. *Minimizar el Daño si ello Ocurriera*

Algunos diseñadores de seguridad han cometido el grave error de suponer que sus medidas preventivas siempre funcionarán.

3. *Diseñar un Método de Recuperación de los Daños*

Debe ser posible reconstruir los registros vitales o todos los archivos si éstos llegarán a sufrir algún accidente o ser totalmente dañados o perdidos. Las diversas formas de recuperación en caso de fallas y la prevención de deadlocks son importantes. Conjuntamente con lo anterior, es necesario el cuidado de los usuarios no autorizados a la red y la prevención de errores o piratería de la información transmitida.

Candados de privacia

La cuestión de quién esta autorizado para hacer *que cosas* en una red es muy importante. Antes de cada operación una computadora debe checar que la operación sea autorizada por el usuario solicitante.

Los esquemas de autorización varían desde un forma muy simple hasta una muy compleja. Una de los esquemas más simples requiere que el usuario tenga una clave de acceso (*password)* la cual sólo él debe conocer. Si una clave de acceso es válida para el programa o archivo en cuestión, se le deberá permitir continuar con la operación. El Lenguaje de Descripción de Datos de CODASYL usa los candados de privacia agregados a los datos. La información bloqueada de esta manera no puede ser usada por un programa a menos que el programa suministre un valor que concuerde con la llave de seguridad.

Esquemas de Autorización

Los candados que fueron construidos en los sistemas y redes están relacionados con esquemas de seguridad y tablas indicando quién esta autorizado a realizar que tareas, o que interconexiones están permitidas.

Las tablas de autorización pueden estar relacionadas con :

1. Usuarios individuales.
2. Grupos o categorías de usuarios.
3. Niveles de seguridad (altamente secreto, corporativo confidencial, etc.).
4. Programas de aplicación.
5. Tiempo del día.
6. Identificación de la terminal o ubicación de la misma.
7. Nodo de la red.
8. Tipos de transacción.
9. Combinaciones de las anteriores.

Las restricciones pueden ser ubicadas en las relaciones entre las seis diferentes entidades en una red -los usuarios, la terminal o los dispositivos de entrada/salida que están siendo empleados, los nodos de la red, los programas de aplicación, los conjuntos o elementos de información. Los candados pueden ser implementados en cualquiera de esas relaciones, y se pueden usar alarmas para atraer la atención de cualquier sospecha de violación.

Las relaciones que pueden ser bloqueadas son las siguientes:

1. El usuario puede ser identificado y observado por medio de la terminal, nodo, programa, información, o dispositivo que esta solicitando.

2. Un nodo específico puede estar en una área insegura y por lo tanto puede ser bloqueada en ciertos nodos, programas, información, o dispositivos.

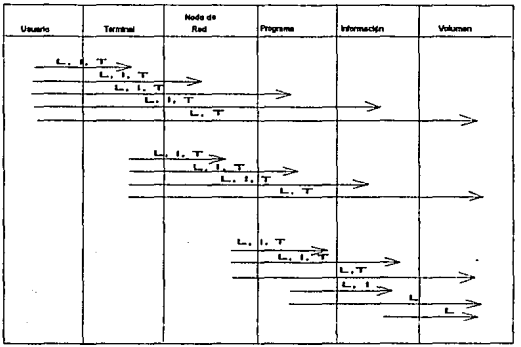
3. Un programa puede ser bloqueado para no accesar determinados archivos o dispositivos.

4. Cierta información puede tener una alta clasificación de seguridad y debe estar prevenida de ser almacenada en dispositivos con una clasificación baja.

Los candados pueden estar basados en niveles de clasificación de seguridad, sobre las entidades individuales o grupos de entidades, o en tiempo. Estas son indicadas con las letras L, I, y T, respectivamente en la figura 3.1.6.

Si se usa la clasificación de niveles seguridad, los tipos de entidades pueden ser asignados a alguna clasificación, tal como CONFIDENCIAL, SECRETA, o cualquier otra previamente definida. Si el nivel de seguridad para un usuario no tiene la clasificación para accesar información SECRETA, él no estará permitido a accesar un nodo definido para un

trabajo SECRETO, o no podrá usar cualquier programa de información, o dispositivo clasificado como SECRETO. La información SECRETA no puede ser transmitida a una terminal no clasificada. Si algún dispositivo no esta clasificado como SECRETO, entonces la información SECRETA no deberá ser grabada en él. Y así sucesivamente. Como se indica en la figura 3.1.6, cualquiera de las relaciones puede estar basada en cualquiera de los niveles de clasificación. Pueden existir tantos niveles como se requieran.



L = *basado en niveles de seguridad (secreto, ultrasecreto, corporativo confidencial)*
I = *basado en aspectos individuales o personas, o grupos de ellos.*
T = *basado en el tiempo del día*

Figura 3.1.6 Relaciones cubiertas por candados, alarmas y tablas de autorización.

Se obtiene una mayor precisión basando las relaciones entre los usuarios y las entidades. El usuario *A*, estará sólo permitido a usar el programa *B*, la

información *C*, los dispositivos *D* y *E*. El programa *X*, podrá accesar solamente la información *Y* y *Z*. Algún archivo determinado o programa estará etiquetado para ser empleado sólo por el usuario que lo creó. Algunos esquemas requieren de grandes tablas de autorización. Para reducir el tamaño de las tablas, los individuos, ítems, o entidades de datos pueden ser arregladas en grupos, y los candados estar basados en agrupaciones.

Finalmente, el sistema puede tener candados de tiempo, basando el acceso a ciertas horas del día. Un intruso *nocturno* no podrá accesar la información aún cuando conozca las claves de acceso y códigos de seguridad. Un nodo en una área de seguridad puede pasar de clasificada a no clasificada en otras horas del día. Si una persona es detectada tratando de usar algún nodo o dispositivo fuera de las horas autorizadas, entonces podría activarse una alarma.

Autentificación de Mensajes

La autentificación de mensajes se refiere a los pasos que se deben tomar para asegurar que un mensaje venga de una fuente legítima o vaya a un destino legítimo. Han existido casos en los cuales información altamente confidencial ha sido impresa en el lugar erróneo. Aún la red telefónica ocasionalmente comunica números equivocados. Es posible que el desvío sea deliberado, o que alguna conexión invalida este siendo usada por un intruso que desea obtener acceso a los archivos.

En un esquema de autentificación de mensajes, el transmisor y el receptor colocan números únicos pseudoaleatorios a los mensajes. Ambos tienen el mismo conjunto de números, ya sea almacenados o los pueden generar. El software de recepción compara el número en el mensaje recibido contra el esperado, y toma alguna acción si no es idéntico.

Criptografia

La manera más segura de tener un grado razonable de que la información transmitida no sea leída, copiada, o sobornada es con el uso de la criptografía. Esto significa que la información es codificada antes de ser transmitida y decodificada después de la transmisión. El proceso de codificación mezcla los bits de modo que una persona que desea tener acceso a la información queda inhabilitada para ponerlos en orden nuevamente.

MÉTODOS DE ENCRIPCIÓN

Los mensajes que se tienen que poner en clave, conocidos como **texto en claro**, se transforman mediante una función que está parametrizada mediante una **clave**. La salida del proceso de puesta en clave, conocido como **texto cifrado** o **criptograma**, es entonces transmitida.

Históricamente, los métodos de cifrado, han sido divididos en dos categorías: ***cifradores de substitución*** (incluyendo los códigos) y ***cifradores de transposición***. Ahora, se estudiará cada uno de ellos a su

vez, como una información previa a la criptografía moderna. Para conocer la historia completa se recomienda ampliamente el libro de Kahn (1967); y para realizar un estudio más teórico, el libro de Kranakis (1986).

Cifradores de substitución

En un ***cifrador de substitución***, cada letra o grupo de letras se substituye por otra letra o grupo de letras para disfrazarlas. El cifrado más antiguo que se conoce es el ***Cifrado de César***, atribuido a Julio César. En este método, *a* se representa por *D, b* se representa por *E, c* se representa por *F,...,*y *z* se representa por *C.* Por ejemplo, *ataquen* se representa por *DWDTXHQ.*

Una sencilla generalización del cifrador de César permite que el alfabeto cifrado se pueda desplazar *k* letras, en lugar de que siempre sean 3. En este caso *k* se convierte en una clave para el método general de alfabetos desplazados circularmente.

Texto en claro : a b c d e f g h i j k l m n o p q r s t u v w x y z
Texto cifrado : Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

A este sistema se le conoce como ***substitución monoalfabética***, en donde la clave está constituida por la cadena de 26 letras, correspondiente al alfabeto completo lo cual da $26! = 4 \times 10^{26}$ combinaciones posibles. Aún con el empleo de un ordenador que pudiera probar, una clave en un microsegundo, el procedimiento para probar todas las claves se llevaría 10^{13} años, aproximadamente.

A primera vista, esto podría parecer un sistema seguro, porque, aún cuando el criptoanalista conozca el sistema general (es decir, la substitución letra por letra), no conoce cuál de las claves está empleándose. El ataque básico aprovecha las probabilidades estadísticas de los lenguajes naturales. En inglés, por ejemplo, la letra más común es la *e*, seguida por las letras *t, o, a, n i*, etc. Las combinaciones más comunes de dos letras, o **digramas**, son : *th, in, er, re y an.* Las combinaciones más comunes de tres letras o **trigramas**, son : *the, ing, and* e *ion.* (Veáse la figura 3.1.14 para consultar más estadísticas sobre el idioma inglés).

Letras		Digramas		Trigramas		Palabras	
E	13.05	TH	3.16	THE	4.72	THE	6.42
T	9.02	IN	1.54	ING	1.42	OF	4.02
O	8.21	ER	1.33	AND	1.13	AND	3.15
A	7.81	RE	1.30	ION	1.00	TO	2.36
N	7.28	AN	1.08	ENT	0.98	A	2.09
I	6.77	HE	1.08	FOR	0.76	IN	1.66
R	6.64	AR	1.02	TIO	0.75	THAT	1.25
S	6.46	EN	1.02	ERE	0.69	IS	1.03
H	5.85	TI	1.02	HER	0.68	I	0.94
D	4.11	TE	0.98	ATE	0.66	IT	0.93
L	3.60	AT	0.88	VER	0.63	FOR	0.77
C	2.93	ON	0.84	TER	0.62	AS	0.76
F	2.88	HA	0.84	THA	0.62	WITH	0.76
U	2.77	OU	0.72	ATI	0.59	WAS	0.72
M	2.62	IT	0.71	HAT	0.55	HIS	0.71
P	2.15	ES	0.69	ERS	0.54	HE	0.71
Y	1.51	ST	0.68	HIS	0.52	BE	0.63
W	1.49	OR	0.68	RES	0.50	NOT	0.61
G	1.39	NT	0.67	ILL	0.47	BY	0.57
B	1.28	HI	0.66	ARE	0.46	BUT	0.56
V	1.00	EA	0.64	CON	0.45	HAVE	0.55
K	0.42	VE	0.64	NCE	0.45	YOU	0.55
X	0.30	CO	0.59	ALL	0.44	WHICH	0.53
J	0.23	DE	0.55	EVE	0.44	ARE	0.50
Q	0.14	RA	0.55	ITH	0.44	ON	0.47
Z	0.09	RO	0.55	TED	0.44	OR	0.45

Figura 3.1.14 Porcentaje de ocurrencias de letras, digramas, trigramas y palabras inglesas.

Para hacer más difícil el trabajo del criptoanalista, es necesario uniformar las frecuencias del texto cifrado, de tal forma que las letras representando a *e, t,* etc., no sobresalgan tan claramente. Una manera de alcanzar este objetivo consiste en introducir múltiples alfabetos de cifrado para utilizarlos en rotación, dando así un resultado, que se conoce como ***cifrado polialfabético***. Como un ejemplo, considérese el ***cifrado Vigenère,*** el cual consiste de una matriz cuadrada que contiene 26 alfabetos de César. El primer renglón, llamado renglón A, es ABCDEFGH...XYZ. El siguiente renglón, llamado renglón B, es BCDEFGHI...YZA. Finalmente, el último renglón, llamado renglón Z, es ZABCDEFG...WXY. Al igual que el cifrado monoalfabético, este crifrado también tiene una clave, la cual es generalmente una palabra o frase corta y fácil de recordar.

El siguiente paso de mayor complejidad para el criptógrafo consiste en utilizar una clave que sea de mayor longitud. Primeramente, se escoje como clave una cadena de bits aleatoria. Después, se convierte el texto en claro en una cadena de bits, por ejemplo, por medio de su representación en ASCII. Por último, se aplica un OR EXLUSIVO, bit por bit, con estas dos cadenas. El texto cifrado resultante, no podrá desbaratarse, debido a que todos los posibles textos en claro son candidatos igualmente probables. El texto cifrado no le proporciona en absoluto ninguna información al criptoanalista. En una muestra suficientemente grande de texto cifrado, cada letra aparecerá con la misma frecuencia, como lo harán todos los digramas y trigramas.

Desafortunadamente este método, conocido como ***clave de una sola vez***, tiene numerosas desventajas en la práctica. Para comenzar, la clave no se puede memorizar. Adicionalmente, la cantidad total de datos que pueden transmitirse queda limitada por la cantidad de clave disponible. La sensibilidad del método ante la pérdida de mensajes, o de mensajes que llegan en un orden incorrecto, viene a ser otro problema importante cuando el transmisor y el receptor se desincronizan.

Los cifradores de substitución no siempre necesitan trabajar con una letra (o bit) a la vez. Por ejemplo, el ***cifrador de Porta*** utiliza una matriz de 26x26, al igual que el cifrador de Vigenère. El texto en claro se codifica con dos caracteres al mismo tiempo; el primer carácter indica un renglón, y el segundo una columna. La pareja de números o letras que se encuentra en una intersección representa un valor puesto en clave. Si se preparan 26 matrices diferentes, los trigramas pueden ponerse en clave como si fueran unidades, utilizando la primera letra de cada trigrama para seleccionar una matriz.

Códigos

A medida que las unidades que se ponen en clave llegan a ser más largas, el cifrado comienza a parecerse a un código. La principal diferencia entre un cifrado y un código es que el primero pone en clave una unidad de tamaño fijo de texto en claro con cada operación efectuada, en tanto que el segundo pone en clave una sóla unidad lingüística de longitud variable, que típicamente viene a ser una palabra o frase. Antes de la llegada de los

ordenadores, los códigos se representaban en dos tipos diferentes: códigos de una parte y códigos de dos partes. En los códigos de una parte, tanto la palabra de texto en claro como el símbolo del código para las palabras *amnesia, amok, among, amorous, amorphus, amortize y ampere* podrían ser 16142, 15144, 16149, 16155, 16160, 16189, 16201 y 16209. En un código de dos partes, estas mismas palabras podrían ser codificadas como 15202, 16902, 40420, 30012, 80032, 76290, 39321 y 10344. Con un código de una parte, tanto la codificación como la decodificación pueden utilizar el mismo libro de código, mientras que en el caso de un código de dos partes se necesitan libros dispuestos en forma diferente para codificar y decodificar. Un código de una parte es mucho más fácil de descifrar que uno de dos partes, dado que el mismo código del símbolo contiene una información aproximada sobre el lugar del libro en el que se encuentra el símbolo en texto en claro. Sin embargo, un código de dos partes, necesita que tanto el transmisor como el receptor transporten cerca del doble de equipaje.

EL desciframiento de un código es igual al desciframiento de un gigantesco cifrador monoalfabético. El símbolo más común en un código es, generalmente, el símbolo para *stop,* que se utiliza para terminar un enunciado. Después, vienen los símbolos para *the, of, and, to, a, in* y *that.* El conocimiento de la estructura de los enunciados en inglés es también muy útil; por ejemplo, la mayoría de los enunciados comienzan con un sujeto, y los sujetos normalmente son de la forma ARTICULO ADJETIVOS SUSTANTIVO.

Los códigos tienen la desventaja de que requieren de grandes libros, los cuales no pueden substituirse tan fácilmente como la clave de un cifrador. Sin embargo, tienen la ventaja de ser generalmente más difíciles de desbaratar que los cifradores. Los códigos y los cifradores pueden combinarse para hacer todavía menos agradable la vida de los criptoanalistas. Por ejemplo, la codificación de un mensaje podría producir una lista de números de cinco dígitos. Estos números podrían concatenarse para formar una secuencia de dígitos que, entonces, se podría poner en clave utilizando un cifrador polialfabético. El cifrado de un mensaje codificado se denomina supercifrado.

Cifradores de transposición

Los cifradores de substitución y los códigos preservan el orden de los símbolos del texto en claro, pero los disfrazan. A diferencia de éstos, los cifradores de transposición, reordenan las letras pero no las disfrazan. En la figura 3.1.15 se describe un cifrador de transposición común, el de tipo columnar. La clave del cifrador es una palabra o frase que no contiene ninguna letra repetida. En este ejemplo, la clave es MEGABUK. El propósito de la clave es el de numerar las columnas, en donde la columna 1 queda bajo la letra de la clave que se encuentra más próxima al comienzo del alfabeto, y así sucesivamente. El texto en claro se escribe horizontalmente, en renglones. El texto cifrado se lee por columnas, comenzando con la columna cuya clave tiene el valor inferior.

Para desbaratar un cifrado de transposición, el criptoanalista deberá primero estar enterado de que se está enfrentando a un cifrador de transposición. Al observar la frecuencia de las letras, *E, T, A, O, I, N,* etc., se puede observar con facilidad si se adaptan al patrón normal de texto en claro. Si es el caso, el cifrador es claramente un cifrador de transposición, porque en dicho cifrado todas las letras se representan a sí mismas.

El siguiente paso consiste en suponer cuál es el número de columnas. En muchos casos una palabra o frase puede llegar a adivinarse a partir del contexto del mensaje. Por ejemplo, supóngase que el criptoanalista sospecha la aparición, en algún lugar del mensaje, de la frase *milliondollars* en el texto en claro. Obsérvese que los digramas *MO, IL, LL, LA, IR* y *OS* se presentan en el texto cifrado, como resultado de la aparición de esta frase. En el texto cifrado las letras *O* sigue a la letra *M* (es decir, se encuentran verticalmente adyacentes en la columna 4), porque están separadas en la frase probable por una distancia igual a la longitud de la clave. Si se hubiera utilizado una clave de longitud siete, los digramas *MD, IO, LL, LL, IA, OR* y *NS* se habrían presentado en su lugar. De hecho, para cada longitud de clave, se produce un juego diferente de digramas en el texto cifrado. Mediante la búsqueda de las diferentes posibilidades, los criptoanalistas pueden frecuentemente determinar la longitud de la clave fácilmente.

El paso siguiente consiste en ordenar las columnas. Cuando el número de columnas, *k,* es pequeño, cada uno de los *k(k -1)* pares de columnas se puede examinar para ver si sus frecuencias de digramas corresponden a

las de un texto cifrado en inglés. Se supone que el par de mayor correspondencia es el que está colocado en la posición correcta. Ahora, cada una de las columnas restantes se prueba tentativamente como la sucesora de este par. La columna predecesora se encuentra de la misma manera. El proceso entero continúa de esta manera hasta que se encuentra un orden probable. Existe la posibilidad de que el texto en claro se reconozca en este momento (por ejemplo, si apareciera *milloin,* se vería con claridad cuál es el error).

```
MEGABUCK      Texto en claro :
74512836            pleasetransferonemilliondollarsto
p l  ease tr        myswissbankaccountsixtwotwo
a n  s feron
e m i l  l ion    Texto cifrado :
d o l l  arst       AFLLSKSOSELAWAITOOSSCTCLNMOMANT
o m ys wiss         ESILYNTWRNNTSOWDPAEDOBUOERIRICXB
b a  nk acco
u n  t s i x t w
o  t w oab cd
```

Fig. 3.1.15 Un cifrador de transposición

Algunos de los cifradores de transposición aceptan un bloque de entrada de longitud fija, y producen un bloque de salida de longitud fija. Estos cifradores pueden describirse claramente al dar solamente una lista indicando el orden en el que deberán salir los caracteres.

III.1.5 REQUERIMIENTOS DE INFORMACIÓN Y EQUIPO

CONSIDERACIONES DE INFORMACIÓN

Los requerimientos de información son los medios necesarios tanto en "tamaño" como de inicialización de datos para el Sistema. La obtención de éstos se realiza con medios automatizados e incluye la creación de cualquier dato necesario para los registros de éstos. La información permite al sistema reflejar con precisión el estado actual de los datos organizativos al principio de la producción.

Aparte de la dimensión, no hay una cuestión que ejerza más profundidad en la planificación del proyecto que la obtención de datos.

Los requerimientos de información pueden sufragarse de cualquiera de las siguientes maneras:

- Eliminación/depuración de los datos actuales
- Reconciliación/balanceo de datos
- Creación de datos nuevos/adicionales
- Conversión de datos que provienen de un medio diferente al del sistema
- En nuestro sistema es conveniente considerar la creación de una estrategia de entrada de datos al sistema.
- El sistema extraerá datos de un sistema a otro.

- La migración constante de información que se requiere debe corresponder al ciclo de corte de cada producto de "Tarjetas de Crédito".

La información que se obtendrá deberá cumplir con las siguientes consideraciones:

- Integridad de datos
- Precisión de datos
- Si se reconcilian regularmente o no
- Llegada balanceada de información
- El volumen de datos
- El procedimiento necesario para datos históricos
- Datos a transferir
- Datos a traducir/convertir
- Superfluos
- A verificar/balancear/reconciliar

Los datos en el sistema que se usen para apoyos internos, como los códigos de control y los números secuenciales en el sistema, constituyen una opción que debe obtenerse íntegramente del equipo TANDEM.

Los datos históricos se necesitarán a nivel resumido. Lo que tenga que cumplir con requisitos legales o fiscales deberá estar detallado.

La fuente de donde provienen los datos contendrá algunos archivos cuyos registros deberán partirse en campos por lo que una vez disponibles en el sistema deberán reconciliarse.

Criterios de transferencia de datos:

Sólo se convertirán datos de una fecha dada.

Considerar estrictamente los campos de eliminación de manera constante, es decir, se sabe qué y cuales se descartarán.

Los datos se obtienen en código ASCII del equipo TANDEM.

No existirá ningún impacto en la utilización en paralelo del "Sistema de Tarjetas de Crédito" y el "Sistema de Desempeño de Tarjetas Bancarias" ya que se trabajará con información generada en tiempos terminales y distintos.

Identificación y depuración de errores

Se refiere a los requisitos de obtención de datos así como los de revisión y de validación para determinar los procedimientos y la determinación de criterios de eliminación adicionales.

Reconciliación de datos.

Es la verificación de los datos obtenidos que están lo suficientemente depurados. Se usa el diccionario de datos para mostrar las relaciones entre datos no transferidos aún y datos transferidos que pudieran contener algún error sobre todo de transmisión.

Los datos no necesitarán intervención manual de corrección.

Consideraciones para la transferencia de información de un equipo TANDEM a un SERVIDOR.

- Los datos que se convertirán.
- La fuente de cada elemento de dato.
- La depuración de datos y la estrategia de depuración.
- Las reglas de conversión, de selección y de eliminación.
- Los procedimientos de reconciliación y clasificación.
- Los recursos necesarios de personal.

La información disponible tendrá la característica que corresponderá al cierre del día anterior, es decir, la medición del desempeño y auditoría del uso de tarjetas de crédito bancarias será conocido con datos al día anterior.

El tamaño de la información a procesar en los "Sistemas de Tarjetas de Crédito" asciende a aproximadamente 4.0 GB; con la obtención de la Base de Datos "Depurada" para el Sistema de "Desempeño de Tarjetas de

Crédito" podemos hablar de un total de 2.0 GB de información aproximadamente.

DESCRIPCIÓN DEL SISTEMA Y REQUERIMIENTOS DE EQUIPO

Existe un sistema de Tarjetas de Crédito que se ejecuta en línea de manera constante en un equipo TANDEM VLX, del que se consignarán sus características más adelante, a manera de respaldo la información pasará a un dispositivo "Drive" de cinta de 8mm con capacidad de almacenar hasta 5.0 GB.

Posteriormente se restaurará la información hacia un SERVIDOR de Red Ethernet bajo un ambiente NetWare-486 de donde se explotará y procesará la información mediante el "Sistema de Desempeño de Tarjetas Bancarias".

El "Sistema de desempeño de Tarjetas Bancarias" será desarrollado en el manejador de Bases de Datos FOXPRO del que se consignaron sus características y del que en una sección previa se indicaron diferencias y criterios de selección con respecto a otros manejadores de Bases de Datos.

CONSIDERACIONES DE HARDWARE

Tandem VLX

Es un Sistema Mainframe ("Propietario") de alto desempeño, diseñado para manejo de Transacciones en Línea y tolerante a fallas, es decir, que se

puede dar mantenimiento al equipo sin que deje de operar, además de ser fácilmente expandible.

Está basado en una arquitectura de procesamiento en paralelo (que tiene desde dos hasta 16 procesadores trabajando en paralelo; que se pueden configurar para trabajar en "espejo") cuya característica es que tiene 32 bits de direccionamiento y hasta 96 megabytes de memoria principal por procesador.

Complementando lo anterior debemos mencionar que TANDEM VLX ofrece una capacidad muy grande de almacenamiento, posee la facultad natural de trabajar con discos "auditores", es decir, discos en espejo además de contener módulos del sistema operativo para manejo de SQL y otras aplicaciones y utilerías, integrados en forma de circuitos, y con facilidad de configurar redes enlazadas con fibra óptica.

Características de un equipo TANDEM VLX NonStop:

- Arquitectura de procesamiento en paralelo
- Capacidad de procesamiento VLSI
- Control dual de almacenamiento
- 32 bits de direccionamiento nativo
- Alta velocidad de "Bus" (40 megabytes por segundo)
- 64 bits de acceso a memoria principal

- Más de 96 megabytes de memoria física por procesador (más de 1.5 GB en un sistema de 16 procesadores)
- Alta velocidad en el hardware de memoria caché
- 100 nanosegundos de acceso a memoria efectiva

Configuración de alta capacidad:

- Alta capacidad de almacenamiento en disco
- Controladores de Entrada/Salida de alta velocidad
- Sistema manejador de Base de Datos relacional distribuida (SQL NonStop)

Las características anteriores del Mainframe TANDEM VLX son generales, a continuación se describe la configuración que genera la "Entrada" para el "Sistema de Desempeño de Tarjetas Bancarias".

Se cuenta con un equipo TANDEM VLX con:

- 16 procesadores
- 96 megabytes de memoria física por procesador lo que da un total de 1.5 GigaBytes tomando en cuenta los 16 procesadores.
- 8 "Drives" de cinta para respaldo.

- Capacidad de almacenamiento en disco de 18 GB.

- El "Bus" tiene una velocidad de transferencia de datos de 40 megabytes por segundo.

"DRIVE" de cinta de 8mm (Lector de Cassettes).

Existen dos modelos de "DRIVE" de cinta de 8mm.

Ambos son compatibles con IBM AT-286 y PS/2 con tarjeta controladora microcanal y con servidores 80386 y 80486.

Consta de una tarjeta controladora SCSI de 16 bits.

Los dos modelos de "Drive" son el I y el II, y existen dos diferencias fundamentales entre ambos:

- 1. Los "Drives" tienen diferentes capacidades (de lectura, escritura)
- 2. Los formatos de lectura-escritura no son intercambiables.

Para aprovechar la máxima capacidad del "Drive" se necesita un cassette "8mm Life Tape"; si se usa un "Drive" de 5.0 GB deberá usarse un cassette de 5.0 GB y si el "Drive" es de 2.2 GB tendrá que usarse un cassette de 2.2 GB, es decir, que los "Drives" están diseñados para un solo formato de escritura, sin embargo se puede hacer lo siguiente:

Un "Drive" de 5.0 GB será capaz de leer información grabada en cassettes de 2.2 GB y 5.0 GB pero sólo podrá escribir información en cassettes de 5.0 GB. El "Drive" de 2.2 GB sólo podrá leer y escribir información en cassettes de 2.2 GB.

Usar "Drives" leyendo o escribiendo cassettes de diferente capacidad genera pérdida de información, por lo que se recomienda usar "Drives" de 5.0 GB leyendo o escribiendo información en cassettes de 5.0 GB y "Drives" de 2.2 GB leyendo o escribiendo cassettes de 2.2 GB.

Carga del "Drive"

- Activar apropiadamente el protector de escritura
- Que esté activado el lector/escritor del cassette
- Que el LED verde del panel del "Drive" esté apagado
- Si es necesario se presiona la cinta de salida para que se abra la puerta de carga

Es importante señalar que al manejar "Bus" con tecnología SCSI, el controlador puede soportar múltiples "Drives".

Cuando se usa más de un "Drive", cada "Drive" excepto el último debe tener dos conectores SCSI. El primer conector se usa para el cable SCSI del previo y el segundo conector se usa para el siguiente "Drive". El último "Drive" hace uso solamente de un conector y éste debe tener un terminador. Cada "Drive" debe tener sólo un ID SCSI de 0 a 6 caracteres.

SERVIDOR

De los muchos factores que se deben considerar al evaluar los sistemas, normalmente el más importante es el tipo de bus utilizado por el sistema. Entre los restantes factores que se deben tener en cuenta al seleccionar y configurar un servidor, están la cantidad de memoria y el tipo de unidades de disco utilizadas.

Potencia, rendimiento y cuellos de botella.

Potencia, rendimiento y cuellos de botella son términos que se utilizan para describir la respuesta de un servidor frente a su uso. El servidor es como una "estación central", y las tarjetas de red como las vías sobre las que se mueven saliendo y entrando los datos. Los datos se mueven de las tarjetas a la memoria del sistema para su procesamiento por el CPU. Potencia es un término que se utiliza para describir el rendimiento combinado de todos los componentes que transfieren datos. Si existen cuellos de botella de los accesos al CPU o de la memoria al disco, esto reduce el rendimiento del servidor.

Aspectos de consideración para el rendimiento de un servidor.

Procesador

El procesador es el corazón de cualquier computadora. A continuación se describen las características de los sistemas 80386, 80486 y Pentium además de los sistemas de multiprocesamiento que utilizan estos chips.

80386 de Intel

Este procesador es el sucesor de la familia de procesadores 8088, 8086 y 80286. Mientras que los procesadores previos eran adecuados para las computadoras de uso personal, el 80386 es el primero de la familia que incorpora capacidades efectivas de multiusuario y multiarea. Aunque el 80286 es capaz de ofrecer multiarea al ejecutar Unix u OS/2, el 80386 ofrece un mejor rendimiento y es más adecuado para tales entornos.

El 80286 posee un puerto de E/S al exterior de 16 bits, mientras que el 80386 posee un puerto de 32 bits completos. El puerto de E/S de 32 bits del 80386 ofrece un punto de conexión para la expansión de memoria de 32 bits de alta velocidad de algunos sistemas ISA (Industry Standard Architecture). En los sistemas más avanzados que utilizan los buses completos MCA (Micro Channel Architecture) y EISA (ISA con posibilidad de expansión), se pueden aprovechar del acceso directo con 32 bits al procesador tanto la memoria como las tarjetas de interface de red.

La velocidad del 80386 es cuatro veces mayor que la del 80286, puede direccionar 4096 MB de memoria, permite la realización real de múltiples tareas y manejo de memoria.

80486 de Intel

Este procesador es compatible con el 80386, por lo que el software del 80386 funcionará sobre el 80486 sin necesidad de cambios. Sin embargo,

el 80486 es un procesador completamente rediseñado que mejora las posibilidades del 80386 y sus características operacionales.

En el chip se han integrado muchas funciones tales como: coprocesador matemático 80387 de Intel y el controlador de memoria caché 80385 de Intel. Las funciones de procesamiento matemático pueden ejecutarse simultáneamente con las funciones normales del 80486.

Los sistemas 80486 son tan rápidos que se han de utilizar técnicas especiales para mover los datos a la velocidad a la que puede tratarlos el procesador. Para mejorar el rendimiento, el 80486 lleva incorporada una memoria caché de 8K para mejorar el rendimiento. El 80386 posee cuatro buffers de escritura que le ayudan a evitar que el bus externo se convierta en cuello de botella. Los buffers almacenan los datos hasta que se libera el bus externo para transferir los datos a la memoria externa. Algunos sistemas 80486 incluyen buffers caché externos para aumentar aún más el rendimiento.

A continuación se mencionan configuraciones básicas de equipos 486 DX4 a 100 MHZ.

HP Vectra VL2 4/100

El Chip DX4 a 100 MHz Vectra Intel tiene 8 MB de RAM (expandible a 64 MB), un disco duro de 340 MB IDE, una caché de 256K y una unidad de disco flexible de 1.44 MB de 3.5 pulgadas.

Micrón PCI/4100

Esta máquina tiene un chip 486 DX4 a 100 MHz con arquitectura PCI, tiene 16 MB de RAM (expandible a 128 MB) disco duro de 540 MB IDE un CD-ROM de velocidad doble, monitor de 15 pulgadas y un generador que incluye un software. Este equipo tiene un chasis estándar con cuatro "slots" ISA y dos PCI.

Poly DX4-100c

Este equipo tiene 16 MB de RAM (expandible a 128), un disco duro de 340 MB IDE, CD-ROM de velocidad doble, monitor de 15 pulgadas.

Sistemas Multiprocesador

Estos sistemas poseen uno o más procesadores, y en algunos casos, buses especializados para mejorar el rendimiento. Estos sistemas poseen utilerías diseñadas para optimizar el rendimiento de cada procesador, del almacenamiento en disco, y la red. La E/S de estos super servidores se evalúa en unos 19 Mbps. Esto sirve de comparación con los 6 Mbps de los sistemas 80386 normales que utilizan componentes de alto rendimiento. La E/S con la red se evalúa en unos 8 Mbps. Usando multiproceso con procesadores 486, Pentium o combinándolos, se tienen posibilidades virtualmente inimaginables.

La idea de multiproceso consiste en incorporar varios procesadores a la computadora para aumentar su potencia de cálculo. Una posibilidad es tener dos procesadores centrales idénticos, sin embargo se puede construir

una configuración tal que involucre varios procesadores de diferentes potencias. La decisión de qué procesador usar en cada segmento de proceso depende más de los procesadores que estén libres en ese instante que de qué procesador sea el más adecuado para dicho proceso. Cada vez que se asignan procesadores a los procesos, se dice que se ha formado una correspondencia entre esos tipos de objetos.

Otra posibilidad para aumentar la eficiencia del sistema es la de asignar procesadores esclavos o periféricos bajo el control del procesador central. En este tipo de proceso, varios procesadores pueden compartir la memoria o tener memoria independiente pero con flujos de instrucciones distintos. Un procesador concreto puede tener el control distribuyendo trabajo entre los otros procesadores a través de mensajes; pero cada uno de ellos, una vez inicializado ejecuta independientemente su programa o segmento de programa que le corresponda.

Pentium

Descripción y características técnicas

El paso de 486 a Pentium es posible mediante una mejora de tarjeta de procesador mediante los chips P24T y eventualmente mediante tarjetas madres de reemplazo. Cada ruta de mejora al Pentium tiene sus aspectos negativos y positivos; sin embargo, la compra de una PC Pentium ofrece mejores resultados.

Uno de los principales problemas en los procesadores es el calentamiento, por lo tanto en las primeras pruebas que se hicieron en los sistemas evaluados se incorporaron disipadores grandes de calor. Con la clase de procesadores Pentium, el enfriamiento del sistema es crítico. Es necesaria una combinación de disipadores de calor, múltiples ventiladores de enfriamiento y patrones de ventilación bien diseñados para disminuir el calentamiento.

Los mejores sistemas de Pentium combinan una vía de acceso de datos amplia con diseño de caché externo y la habilidad para realizar secuencias de lecturas/escrituras muy rápidamente.

La mejora que el Pentium ofrece sobre los CPUs anteriores x86 de Intel radica en la matemática de punto flotante al aumentar su velocidad, aunque pocas aplicaciones de PC para uso general y comercial hacen mucho uso de la matemática de punto flotante, es decir trabajan con cálculos lentos de números enteros. Esto tenderá a cambiar. El haber aumentado en esas proporciones dicha velocidad en el Pentium lo hace cinco veces mayor que la de los 486, pero todavía no está al nivel de la mayoría de los procesadores RISC.

El Pentium tiene una arquitectura PCI (Peripherical component Interconnect) que es un sistema de bus local, el cual se integra bien con el Pentium y ofrece ventajas sobre otros sistemas; soporta la VESA y VL-Bus, con esto demuestra tener mejor desempeño. La ventaja que tiene PCI es

que hace que este procesador soporte una caché externa sincronía (SRAM).

El uso de Pentium también significa una definición nueva del mercado de las estaciones de trabajo. El énfasis en las mejoras de punto flotante resultará ser probablemente mucho más importante como un movimiento estratégico para Intel, permitiendo que los sistemas basados en Pentium entren en el mercado técnico de las estaciones de trabajo que había sido dominado por IBM, Sun Microsystems, Hewlett-Packard, Digital y otros fabricantes de Sistemas RISC. Igualmente, el precio comparativamente más bajo del Pentium probablemente se impondrá sobre las estaciones de trabajo RISC. El elemento clave en esta redefinición es la presentación del Sistema Operativo Windows NT, que funcionará en varias plataformas, desde sistemas RISC tradicionales hasta las computadoras personales. El resultado será que la compatibilidad estará basada en el Sistema Operativo, y no en el Hardware.

Pentium marca el comienzo de Intel en el diseño de semiconductores en tamaño submicrónico. Los 486s se basaban en elementos de 1.0 micrón mientras que las conexiones del Pentium o sus vías para señales, son de 0.8 micrones de ancho. (Un micrón es una millonésima parte de un metro, o 1/15 del grueso de un cabello Humano). Intel usa 5.0 millones de transistores en el Pentium, comparado con los 1.2 millones de transistores del 486.

Los límites de capacidad de computación de los 486, cuando las máquinas se usan como servidores y como máquinas individuales, no se deben a las limitaciones del CPU, sino la E/S y memorias, junto con los cachés los que limitan el rendimiento. Como contraste una PC basada en Pentium sufrirá de esas limitaciones y cuellos de botella, desperdiciando gran parte de su rendimiento potencial.

Las computadoras basadas en Pentium normalmente están consideradas como la opción ideal para servidores de red, algunas máquinas que quieren ser el servidor ideal están mal equipadas para esa función debido a su capacidad de expansión limitada. Por ejemplo, la DECpc 560ST, que se aplica como estación de trabajo o servidor de archivos ofrece sólo cinco cavidades para discos, seis ranuras de expansión EISA y una fuente de alimentación de 256 Watts. Digital vende la línea ST como servidores para redes pequeñas, pero no sería cómodo tener una máquina tan limitada sobre todo por la poca expansión. En comparación, otras máquinas tienen una cantidad suficiente de ranuras para discos y expansión y una fuente de alimentación mejor. El video es otra prueba de adaptabilidad. La mayoría de los administradores de red se preocupan del rendimiento de video o de las especificaciones de los servidores, ya que sus monitores típicamente sólo se usan para la supervisión de la red. Por lo tanto, el rendimiento de video de un servidor no es crítico. Pero para las estaciones de trabajo, la capacidad de video se ha convertido en uno de los elementos más importantes.

Hoy en día el uso de componentes de tamaño en submicrones permite a los diseñadores colocar más de tres millones de transistores en un sólo microcircuito. Así se pueden integrar componentes como por ejemplo: los coprocesadores matemáticos y cachés directamente al CPU -disminuyendo de modo dramático el tiempo de acceso.

Actualmente Pentium tiene una separación entre sus líneas de .8 micrones, lo cual le permite tener 3.1 millones de transistores y se espera que la siguiente generación de Pentium , denominada P54C, cuente con esta tecnología, e incluso se pueda implementar en la plataforma de Laptops. P54C tendrá inicialmente tres velocidades: 50/75 MHz, 60/90 MHz y 33/100 MHz.

Debido a que su línea de ensamblaje es tecnología superescalar (que permite que la información se procese simultáneamente por medio de dos conductos duales -pipeline- es posible ejecutar dos instrucciones simultáneamente en cada ciclo de reloj, después de captar y descifrar una instrucción parcialmente, el procesador Pentium determina si la instrucción se puede ejecutar en paralelo con la instrucción siguiente en línea. Si no detecta ninguna dependencia, las dos instrucciones son enviadas a ejecutarse en conductos paralelos esto lo hace 300 veces más rápido que el primer microprocesador para PCs). Para lograrlo, los conductos dividen la instrucción y luego la envían en cinco etapas. Al pasar de una etapa a la siguiente, se libera al conducto para iniciar otra instrucción. La operación de 64 bits se une con lo anterior para lograr un rendimiento consistentemente

superior, sin importar si un sistema se diseñó para ser un servidor de red o una estación de trabajo de Windows.

El procesador cuenta con dos cachés separados de 8K para código y reescritura de datos, lo cual reduce los conflictos de la memoria caché y aumenta el rendimiento del sistema. Con una memoria caché en microcircuito mismo, tenemos la información esencial muy cerca de la línea de ensamble principal. De este modo, las instrucciones y los datos pueden ser capturados sin pérdida de tiempo. Y en un total del 95% del tiempo, la información está a la mano.

Pentium como la mayoría de los nuevos productos de Intel tienen una disipación de Potencia de 4.0 watts o menor, sin embargo para facilidad de diseño, se añadieron características extras como: el controlador de interrupciones programables avanzado de Intel (APIC). Así, se espera que las PC's y servicios basados en los nuevos chips proliferen fuertemente.

Se duplicó el tamaño del BUS permitiendo que se puedan capturar el doble de información a la vez. Existe inclusive un modo para transferencias de información a alta velocidad -para que la mayor cantidad de información llegue a su destino más rápido. Además, se cuenta con una verificación automática de la integridad de datos para asegurarse que se están moviendo los datos correctos.

El microcircuito posee cierta "inteligencia" ya que cuenta con un pequeño caché conocido como el predictor de ramificación (Branch Target Buffer), el

cual predice cual ramal será escogido por la instrucción. Cuando la predicción es correcta (lo que ocurre más del 90% de las veces) ésta se ejecuta sin demora -aumentando el rendimiento.

Cuenta con piezas especializadas de hardware para acelerar las tres instrucciones más comunes de punto flotante (un multiplicador, un divisor y un sumador). Contando con estas propiedades, la mayoría de las instrucciones de punto flotante pueden ejecutarse en un solo ciclo de reloj. Lo cual brinda hasta cinco veces más que el rendimiento de punto flotante de las computadoras que se basan en un CPU Intel 486 DX2.

Consideraciones en relación a un 486 DX4 y Pentium

Una consideración importante que se puede hacer para la selección de un procesador es, aparte de su rendimiento, su costo, mientras que un Pentium a 100 MHz cuesta casi 1000 dólares un 486 DX4 cuesta 600.

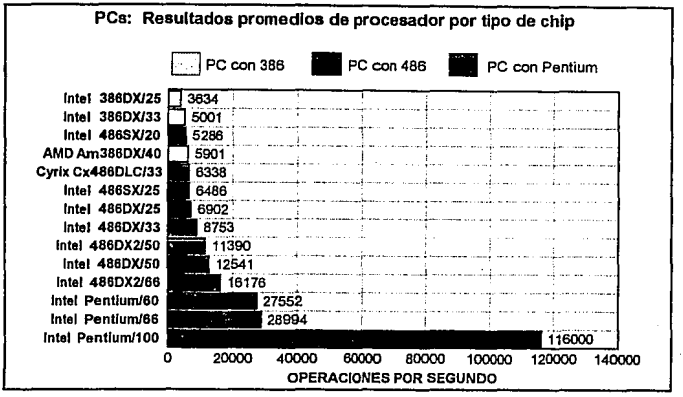


Figura 3.1.16 Gráfica de comparación de rendimiento entre procesadores

TABLA COMPARATIVA ENTRE PROCESADORES

CPU	VEL.RELOJ (MHZ)	VÍA DE DATOS	CACHE	POTENCIA DISIPADA	VOLTAJE	COSTO DOLARES
486SX	25,33	32-bit	8K	3 watts	5 volts	115
486DX	25,33,50	32-bit	8K	8 watts	5 volts	280
486DX2	40,50,66	32-bit	8K	8 watts	5 volts	300
DX4	75,83,100	32-bit	16K	4 watts	3.3 volts	580
Pentium	60,66,90,100	64-bit	16K	16 watts	5 volts	750-1000

Consideraciones en relación a los procesadores INTEL.

Pasando de una 386 a una 486 y después a un Pentium considerando el cambio de procesador como una evolución de sistema no siempre significa una mejor ejecución de PC. El sistema 486 DX4 a 100 MHZ es más rápido que la 486 DX2 a 60 o 66 MHz pero no tan rápido como un Pentium de 60 o 66 MHz y no de diga con uno de 100 MHz que es más rápido que cualquiera que esté disponible en el mercado y 50% más rápido que el de 60. El Chip 486 DX4 ocupa un lugar intermedio entre la 486 DX2 y Pentium. El Chip está basado en 0.6 micrones, de tecnología semiconductora de 3.3 volts, esto significa que la estructura del sistema alrededor del DX4 tendrá problemas con la conservación de la energía. Incluye una caché de 16K y tiene integrado una unidad de punto flotante, para llevar a cabo la ejecución. Mientras que en un CPU Pentium no se podrá teclear rápido, será la velocidad en operaciones tal como reformatear el texto, redibujando la pantalla y checando el deletreo en un 80%: el desempeño gracias a la unidad de punto flotante que ha sido rediseñada del chip Pentium está en el orden de 5 a 6 veces más rápido que un 486 DX2/66 sobre las funciones de punto flotante, tales como modelado de matemáticas complejas que es común en CAD y programas de simulación financiera. Mientras la potencia de punto flotante de Pentium no está al nivel de una estación de trabajo RISC la ventaja sobre un 486 DX2/66 es muy grande.

El Pentium aproximadamente duplica el rendimiento de operaciones matemáticas con enteros de los 486s pero su aprovechamiento real no es muy claro en las aplicaciones.

Un procesador Pentium de 100 MHz que corra en un diseño 486 mejorado, seguiría sufriendo de los problemas de memoria, caché y E/S que afectan a cualquier computadora 486 que corra con un procesador Pentium, pero la velocidad general de procesador de tal chip podría producir una máquina eficaz con relación a su costo. Por lo tanto, podemos decir que no existe en este momento un procesador para PCs que supere a Pentium y considerando que sus características lo hacen muy comercial, es posible que evolucione más rápido y que se consiga fácilmente.

El Bus del servidor

El bus de cualquier computadora es el canal de comunicaciones que se utiliza para transferir datos entre los dispositivos de E/S, la memoria, el almacenamiento en disco y el CPU. El tipo de bus seleccionado es extremadamente importante para el rendimiento y la potencia del servidor.

Bus ISA

El bus ISA (Industry Standard Architecture) ha estado presente desde que apareció la PC de IBM. Se amplió de una entrada de datos de 8 bits a una de 16 bits en 1984 con la aparición de la PC-AT de IBM. Este bus se utiliza en muchas computadoras personales 80386 y en algunos sistemas 80486, pero no es el adecuado para este último.

Es necesario ser previsor cuando se piense configurar un servidor con un bus ISA, ya que existe la posibilidad de que nos quedemos sin

interrupciones libres. Las mayores inconsistencias de los sistemas ISA están entre las velocidades de sus procesadores y su potencia actual. Aunque un sistema 386 puede dar velocidades en el procesador de 16 a 30 MHz, los 8 MHz del bus ISA hacen muy lento el sistema. Esto significa que la transferencia de datos hacia y desde procesador encontrará un cuello de botella en el bus. Si se cuenta con baja velocidad de transferencia de ISA con interfaces poco eficientes, se obtiene como resultado un sistema inaceptable.

El bus ISA admite control de bus, pero no es un verdadero control de bus. El multiprocesamiento tampoco es posible, debido a que existen especificaciones de tiempo estándares.

Bus EISA

El bus EISA fue diseñado por un consorcio de fabricantes de la industria para que admitiera las TARJETAS de expansión ISA existentes y ofreciera una plataforma para el crecimiento futuro.

Para poder trabajar con las TARJETAS ISA, utiliza una velocidad de reloj de 8 MHz, pero el bus ofrece velocidades de acceso a la memoria de hasta 33 Mbps. Paro ello, EISA posee un bus de E/S y otro separado para el procesador; de esta forma, el bus de E/S puede seguir una velocidad de reloj baja para poder trabajar con las TARJETAS ISA, mientras que el bus del procesador trabaja a velocidades superiores. Las máquinas EISA ofrecen E/S en disco a alta velocidad para múltiples usuarios.

EL EISA en un bus de 32 bits completos, de forma que necesita un diseño nuevo para acomodar más pins de los que posee el bus ISA.

A la vez, el conector admitirá tanto las TARJETAS ISA como las EISA. Se utiliza un diseño de conectores en dos filas. La fila superior hace contacto con las TARJETAS ISA, mientras la inferior con las EISA. Aunque EISA mantiene la velocidad de reloj de 8 MHz de ISA para tener compatibilidad, ofrece un método de transferencia rápida de los datos con hasta tres veces la velocidad del bus ISA.

Bus MCA

El bus Micro Channel Architecture fue desarrollado por IBM como respuesta a las dificultades que surgían al combinar procesadores rápidos con los relativamente lentos buses ISA. Si bien MCA no admite las antiguas TARJETAS adecuadas a ISA, ofrece una nueva interface de 32 bits que es más rápida que el ISA, y que encaja mucho mejor con los procesadores 80386 y 80486.

El bus MCA es un diseño de bus simple, que gestiona tanto las transferencias a la memoria y de E/S con multiplexación, lo que permite que diversos procesos puedan compartir el bus simultáneamente. La multiplexación divide el bus en varios canales que pueden ser utilizados por procesos distintos. La comparación en velocidad de este diseño con los sistemas multibus no es favorable, pero en muchos casos es más conveniente. Si se ejecutan en el servidor aplicaciones con un

procesamiento intenso, los superservidores son una mejor opción, debido a sus superiores capacidades de potencia y multiprocesamiento.

Rendimiento de los Buses

Tipo de Bus	Transferencia	Velocidad reloj	Velocidad
Bus AT16 bits	Hasta 1.5 Mbps	N/A	8 MHz
Bus MCA 32 bits	Hasta 5 Mbps	Hasta 40 Mbps	10 MHz
Bus EISA 32 bits	Hasta 33 Mbps	Hasta 32 Mbps	8 MHz

Control de bus (Bus Mastering)

En la mayoría de los casos el CPU posee un control completo sobre el sistema. Sin embargo, puede ser ventajoso el dar un control temporal sobre el sistema a otros dispositivos, de forma que pueda ejecutar tareas específicas. Mientras que se transfieren datos de la interface a la memoria, el CPU es capaz de ejecutar otras tareas debido a que se cuenta con un controlador de acceso a memoria por separado, lo que aumenta el rendimiento.

Es de tomar en cuenta esta serie de consideraciones, de lo contrario correremos riesgos que pueden salirse de nuestro control. Cuando se utiliza una tarjeta con control de bus, un procesador separado de la tarjeta de interfaz puede llevar realizar y llevar a cabo todas la tareas de transferencia

de datos a la memoria, sin necesitar ayuda del CPU principal. De esta forma, el CPU principal no se ve interrumpido para realizar estas tareas, pudiendo realizar mejor las tareas que tenga asignadas.

Es necesario contar con un sistema alterno que evite que dos dispositivos utilicen el mismo bus a la vez, y esto se hace más importante cuando se instalan en el sistema más tarjetas de interfaz con control de bus.

Consideraciones sobre memoria

La memoria debe estar situada en conexiones de bus de 32 bits para un acceso más rápido, tal como lo está en los sistemas MCA y EISA. En los sistemas ISA, normalmente se ofrece una conexión propia de 32 bits separada del bus para expansión de memoria.

Este conector va al puerto de entrada de 32 bits del 80386 para una E/S más rápida en memoria.

Tipos de memoria

A medida que aumenta la velocidad del procesador, la memoria RAM se va perfilando como un cuello de botella para muchos sistemas servidores. Cuando la memoria no puede responder a la continua demanda de flujo de datos del procesador, éste debe esperar uno o más ciclos de reloj. El período de espera se denomina normalmente "estado de espera". Cuando

la memoria es lo suficientemente rápida como para operar al lado del CPU, puede darse la condición de "estado de espera nulo".

Aunque la memoria con estado de espera nulo es la ideal en cualquier sistema, es cara. Un tipo de memoria utilizada a menudo es la DRAM (Dynamic RAM), que no sigue el ritmo de los sistemas 80386 y 80486 más rápidos. Por ello son inevitables los estados de espera, a menos que se utilicen chips de RAM más rápidos.

RAM dinámica (DRAM). Son los más usuales. No son caros y están disponibles en un rango que va de los 80 a 200 nanosegundos.

Necesitan un ciclo de refresco cada 4 milisegundos, no puede seguir a ningún procesador que funcione por encima de los 16 MHz, lo que supone la necesidad de estados de espera.

RAM estática. Los chips SRAM (Static RAM) son mucho más rápidos que los DRAM, y no necesitan refresco continuo. Su diseño incluye circuitos flip-flop, que permanecen en uno u otro estado. Al tener un diseño más complejo, son más caras.

Muchos sistemas poseen una combinación de DRAM y SRAM para poder equilibrar el costo-rendimiento.

Consideraciones sobre discos.

Un cuello de botella de muchos servidores, incluso de los que utilizan un sistema de bus de alto rendimiento, es el acceso al almacenamiento en disco.

Los controladores se evalúan por su tiempo de acceso, que es el tiempo medio que tardan en almacenar o recuperar los datos. Las unidades de un servidor deben tener un tiempo de acceso igual o inferior a 22 milisegundos.

Se dispone de diversos estándares de métodos de codificación e interfaces de disco: Las unidades y controladores ESDI (Enhanced Small-Device Interface) o SCSI (Small Computer Systems Interface). Los sistemas SCSI admiten un "encadenamiento sin problemas" de unidades adicionales en un montaje en el que una unidad se engancha sobre otra. De esta forma, se pueden conectar sin problemas hasta ocho dispositivos. Los SCSI admiten otros tipos de dispositivos, como las unidades de cinta para copias de seguridad, los CD ROM y otras unidades ópticas. Los ESDI admiten hasta un máximo de dos unidades de disco conectadas.

EDSI posee una velocidad de transferencia de datos en un rango de 10 a 15 Mbps, mientras que SCSI es prácticamente ilimitada. Las velocidades actuales son del orden de 15 Mbps.

Los sistemas SCSI sitúan la circuitería del controlador de la unidad en la misma unidad, en lugar de usar un controlador, sólo se necesita una placa de conexión SCSI para conectar hasta 8 dispositivos.

ESDI utiliza un método de transferencia en serie, mientras que SCSI utiliza un método en paralelo más rápido. Debido a que las unidades SCSI poseen sus propios controladores, pueden llevar a cabo una operación de lectura o escritura completa independiente sin intervención del CPU. Durante este tiempo, el CPU puede realizar otros trabajos.

Actualmente SCSI posee un tiempo de acceso medio de 10 a 15 milisegundos y entre más cabezas y platos tenga la unidad, más rápida será.

Estaciones de Trabajo

Las estaciones de trabajo no afectan por lo general al rendimiento del conjunto de la red local, a menos que la estación de trabajo no acepte los datos que se le envían debido a un almacenamiento temporal inadecuado y/o una baja potencia.

Las aplicaciones se ejecutan sobre las estaciones de trabajo a la velocidad y con el rendimiento que viene dado por el diseño hardware de la estación de trabajo. La red sólo se puede ver afectada cuando el usuario acceda a ella. Una estación de trabajo puede hacer lenta una red al transferir grandes archivos.

La potencia de una estación se puede mejorar sustituyéndola por un sistema que utilice componentes más rápidos. Se debería de determinar la carga de trabajo y el volumen de acceso a la red de cada estación de trabajo. Los equipos que acceden a menudo a la red deberían pasar los datos con rapidez de la red a la memoria interna (o viceversa) para evitar cuellos de botella. Algunos sistemas acceden muy poco a la red, por lo que no son candidatos a su sustitución.

Para obtener un buen rendimiento de las estaciones de trabajo a utilizar es necesario como punto de partida un sistema 80386 con un bus ISA de 16 bits como mínimo. Los sistemas que requieran un mayor rendimiento pueden necesitar un procesador 80486.

Para mejorar aún más el rendimiento se recomienda un sistema con un bus MCA o EISA.

Evaluación de tarjetas de red

Servidores y estaciones con alto grado de uso en la red. En estos equipos debemos utilizar las mejores tarjetas que se ajusten a nuestro presupuesto. Esto implica que el servidor o estación sea un equipo potente como un 80386 u 80486 con bus MCA o EISA, que corresponda con la potencia de la tarjeta.

Estaciones con uso medio. Las estaciones de trabajo que generan tráfico de forma ocasional o ráfagas por transmisión de archivos grandes (generalmente archivos gráficos), deben ser 80286, 80386 u 80486 con un bus y una tarjeta de red de 16 bits.

Estaciones con uso bajo. Las estaciones con bajo uso son aquellas que se conectan a la red para transferir algún archivo de modo ocasional. Puede que accedan al servidor menos de 10 veces al día. Debido a los avances en la tecnología de los procesadores, están apareciendo nuevas tarjetas de red que pueden transmitir datos con mayor velocidad y eliminan los cuellos de botella. La mayoría de estas nuevas tarjetas sólo funcionarán en equipos MCA o EISA, debido a la necesidad de alto rendimiento y procesadores más rápidos.

Procesadores incorporados. Algunas tarjetas de red incorporan su propio procesador que realiza las tareas de montaje de paquetes y transfieren directamente los resultados a memoria utilizando un procedimiento de acceso directo a memoria. Se eliminan las transferencias múltiples de memoria, pero puede aparecer un cuello de botella si se utiliza un bus ISA a 8 MHz en el servidor. Además generalmente el procesador de la tarjeta no será tan rápido como el del equipo en que se instala, existiendo un desajuste en el rendimiento.

Control de bus (Bus mastering). Una tarjeta de red instalada en un sistema que soporte "Bus mastering" transfiere directamente los datos

recibidos a la RAM del sistema. La tarjeta de red interrumpe al CPU y gestiona la transferencia de datos pero el CPU ejecuta el protocolo de red.

Método con apuntadores. En este método, los datos son transferidos directamente a un bloque de memoria compartido por la tarjeta de red y el CPU usando un método de control de bus.

La tarjeta de red le envía al CPU un apuntador que indica la posición de los paquetes a procesar. Este método ofrece la mejor respuesta y generalmente sólo se implementa en equipos MCA y EISA. El circuito integrado controlador de red utilizado habitualmente para controlar la transferencia de datos es el 82596 de Intel. Deberíamos evaluar las tarjetas que utilizan este método para instalarlas en el servidor. El tamaño de los paquetes de datos transferidos en la red también resulta importante. Cuando se incrementa dicho tamaño, se puede obtener un gran aumento en el rendimiento. Esto disminuye el trabajo de reorganización de los paquetes y el tráfico en la red, ya que hay menos información de control de paquetes.

Evaluación de topologías de red

Las evaluaciones se harán sobre sistemas ARCNET y Ethernet con cable coaxial, y en sistemas Token Ring con par trenzado. No se considerará una red de gran alcance sino local de cable coaxial.

ARCNET	Medio	Bajo (2.5)	Alta
ETHERNET	Medio	Alto (10)	Media
TOKEN RING	Elevado	Medio (4)	Media

Evaluación de Arcnet

Resulta una opción a considerar debido a que no es muy cara y a que su facilidad de instalación es alta. Además utiliza protocolo de acceso seguro y confiable, utilizando una topología muy flexible que combina las topologías en bus y estrella. Se puede usar una configuración en bus lineal cuando se necesitan tendidos de cable en línea recta y una configuración en estrella para conectar los puestos de trabajo a un punto central.

ARCNET ofrece una topología única que permite más flexibilidad para planificar el esquema de cableado. ARCNET puede utilizarse en combinación con otras topologías de red. Al ser una red de baja velocidad de transferencia y tener limitaciones de distancia no es la mejor opción.

Evaluación de Ethernet

Resulta una muy buena elección debido a que el tráfico de datos se desarrolla enviando un archivo muy grande, cosa que le da ventajas muy grandes sobre ARCNET y TOKEN RING ya que éstas funcionan mejor bajo un tráfico constante de información. Sin embargo tenemos que asegurarnos

de instalar un servidor rápido con un bus MCA o EISA y tener cuidado con el cableado ya que una avería paralizaría toda la red.

Evaluación de Token Ring

No es la más conveniente por ser casi exclusiva de IBM, además transmite a una velocidad de 4 Mbps dependiendo de que el tráfico sea constante y requiere un cableado muy difícil de instalar e inconveniente.

Para una red local TOKEN RING no tiene problemas de distancia. Aunque es caro es fiable y seguro.

CONSIDERACIONES DE SOFTWARE

En Tandem

Para realizar la bajada de la información al "Drive" de 8mm se configura su reconocimiento en el equipo TANDEM VLX como cualquier dispositivo de respaldo ajustando en ambos equipos los protocolos de comunicaciones.

La transferencia se hace de manera síncrona usando las especificaciones X.25 a una velocidad de transferencia de 64 Kilobits por segundo dependiendo del controlador utilizado usando las interfaz eléctrica V24/V28 (RS-232).

Las instrucciones de respaldo son las mismas que en un respaldo normal, con la indicación del direccionamiento del dispositivo a utilizar, en este caso el puerto serial de salida, y con esto la información se transfiere.

En el Servidor

El sistema podrá ejecutarse en WINDOWS en una PC convencional, sin embargo, es necesario considerar que por necesidades propias del área de Auditoría habrá ocasiones en las que será necesario incorporarlo a una red de área local NetWare de Novell.

A continuación se muestran las características para el sistema de archivos de NetWare de Novell:

Usuarios Lógicos soportados por el servidor	1000
Archivos abiertos a la vez por servidor	100000
Transacciones concurrentes	25000
Volúmenes por servidor de archivos	64
Unidades Lógicas de Archivos	2097152
Tamaño máximo de archivo	4 GB
Capacidad de almacenamiento máxima teórica	32 TB
Memoria RAM máxima teórica	4 GB
Buffers caché	Dinámico

El software de desarrollo en el SERVIDOR es como se sabe el manejador de bases de datos FOXPRO del que se consignan sus características en la sección correspondiente.

NetWare requiere un sistema servidor de red, estaciones de trabajo, tarjetas de interfaz y cable de conexión.

El sistema deberá tener un mínimo de 32 MB de RAM aunque es posible aumentar la eficiencia del mismo si disponemos de más memoria, esto debido a los volúmenes de información que se manejarán.

Tomando como referencia lo anterior y aprovechando la infraestructura de equipo que existe en la Institución, se decidió que las características del servidor son las siguientes:

Un aspecto importante a considerares el método de acceso al Bus, que es por donde se transfieren paquetes de datos de la memoria de una estación de trabajo hacia el cable físico de la red.

Cuando los paquetes están listos para ser transmitidos, en el Bus de la red se verifica que sea posible la transmisión.

Existen varios métodos de acceso entre los que se encuentran el "Método de acceso por detección de portadora" (CSMA = Carrier Sense Multiple Access). Un nodo verifica si el cable está siendo utilizado antes de transmitir. También se le denomina método de detección de colisiones. Otro método es el de "Pase de Testigo" y consiste en que un nodo espera hasta disponer de un "testigo", lo que significa que puede transmitir paquetes. Esto asegura que no está transmitiendo ningún otro nodo de la red.

Generalmente, el método de detección de colisiones ofrece un mayor rendimiento en un sistema de cableado, ya que una estación puede acceder al cable de forma simple siempre que lo necesite. Mientras el tráfico de la red no sea muy intenso, es poco probable que se produzcan colisiones, y la red funciona a una velocidad elevada.

Las redes con pase de testigo tienden a ser extremadamente fiables y mantienen una velocidad constante, pero generalmente esta velocidad es menor de la alcanzada en redes que usan detección de colisiones. La velocidad constante se debe al hecho de que en cada momento sólo hay una estación que pueda usar el testigo. Además, una estación no puede

acaparar el uso del testigo, ya que esto impediría que las restantes estaciones accedieran al cable.

Comparación de velocidades de transmisión de redes.

Tipo	Velocidad	Uso
Arcnet	2.5 Mbps	Redes Locales
Token Ring	4-16 Mbps	Redes Locales
Ethernet cable fino	10 Mbps	Redes Locales
Ethernet cable grueso	10 Mbps	Redes Locales Extendidas
Líneas conmutadas	2.4-19.2 Mbps	Enlace remoto monousuario
Conmutación paquetes	menor 64 Kbps	Bajoomedio (WAN)
Fractional T-1	64 Kbps	WAN o enlaces redundantes
T-1	1544 Mbps	Alto en enlaces de WAN
T-3	44184 Mbps	Alto en enlaces de WAN
Fibra óptica	10-100 Mbps	Alto en enlaces de MAN

JUSTIFICACIÓN DE SELECCIÓN DEL PROCESADOR

Los criterios de selección para la elección de un SERVIDOR es un asunto personal y debe reflejar la manera en que se piensa utilizar al sistema. Se debe comenzar con un análisis detallado de las necesidades, requisitos y presupuesto.

Los aspectos importantes a evaluar para seleccionar el SERVIDOR más adecuada son los siguientes:

- El procesador
- La memoria del sistema
- El bus de expansión
- El almacenamiento
- La tecnología de video

Cuatro prototipos de computadoras personales, una máquina ideal, una estación de Microsoft Windows, un sistema de negocios pequeño y un servidor de redes de área local, muestran la diversidad de rendimiento disponible en la plataforma general de la PC. Pequeños cambios de configuración pueden crear diferencias pronunciadas en la funcionalidad del sistema.

Actualmente el procesador 286 es obsoleto; el 386 tiende a lo mismo, mientras que el estándar es el 486 y Pentium . El nivel de CPU mínimo que se debe considerar para las aplicaciones generales de la PC de hoy es un 486/DX2 o DX4 y Pentium.

El Chip Pentium de Intel es actualmente el más rápido y más potente. Disponible en versiones de 60 y 66 MHz, puede procesar a un 75% más de velocidad que un 486 DX2/66. Sin embargo, debido al gran avance de la tecnología existen procesadores 486 a 100 MHz el cual puede compararse en rendimiento con un Pentium/66 MHz y estos a su vez aumentarán su

velocidad siendo preferible seleccionar según las necesidades que se tengan, procesadores Pentium a 100 MHz.

La competencia entre los fabricantes de chips aumenta a medida que Intel introduce varios procesadores nuevos, AMD y Cyrix le siguen los pasos.

Una memoria caché coordina los CPUs rápidos de hoy con los lentos subsistemas de memoria. La caché interno de 8K para datos e instrucciones en la línea 486 de Intel y los 16K en Pentium no son suficientes para las aplicaciones de hoy. Para un servidor, 512K son adecuados, pero 128K son más que suficientes para las aplicaciones de negocios generales.

El bus del sistema que seleccione afectará la capacidad del sistema. ISA será suficiente para la computación general, pero necesitará un bus extendido, como EISA o MCA, para trabajos de mucho tráfico de datos como los que maneja un servidor de archivos. Se puede acelerar el rendimiento del video añadiendo una conexión de bus local.
En función de lo descrito anteriormente se propone la siguiente configuración de equipo. (Figura 3.1.17)

Procesador	Pentium a 100 MHz
Memoria RAM	64 MB
Disco Duro	3 SCSI internos de 2 GB c/u
Estaciones de trabajo	486 DX2/66 MHz con WINDOWS Y DOS
Tarjeta de interfaz	Ethernet con cable coaxial 10 Mbps
Bus	EISA 64 bits
Protocolo de comunicación	CSMA/CD

Figura 3.1.17 Configuración del equipo de la solución propuesta

III.2 DESCRIPCIÓN DEL SISTEMA

En esta sección se describe el "Sistema de Desempeño de Tarjetas de Crédito" en todas las fases de diseño, desde el modelo propuesto tomando en cuenta el modelo operativo anterior analizado en el Capitulo II. Se detalla el diagrama entidad-relación y se describe la normalización, así como el diccionario de datos.

Posteriormente se muestra el Diagrama de Jerarquía de Procesos, explicando los módulos principales que lo conforman. Finalmente se explican detalladamente los Diagramas de Flujo de datos en los niveles que fueron necesarios en el desarrollo, así como las miniespecificaciones correspondientes.

Como se sabe, el objetivo del sistema es el de apoyar mediante la implementación de una herramienta automatizada el control y análisis del comportamiento de la cartera vencida de tarjetas de crédito en base a los lineamientos internos de Banamex, con el fin de optimizar acciones correctivas y/o preventivas en el otorgamiento y seguimiento de las líneas de crédito.

III.2.1 DIAGRAMAS DE FLUJO OPERATIVO CON EL SISTEMA DE DESEMPEÑO DE TARJETAS DE CREDITO

El proceso que se implementará con el sistema de desempeño de tarjetas de crédito como herramienta de apoyo para la revisión y análisis de la cartera vencida de tarjetas de crédito, es reflejado en forma manual, como se describe en las siguientes diagramas de flujo operativo, bajo las entidades de Auditoría, Gerencia del Producto, División de Tarjetas de Crédito, Centro de Servicio Informático CSI, Sistemas, control de altas y áreas de operación.

Los resultados que se obtienen a través de ésta metodología, es característica de una revisión general de la cartera, lo cuál repercute en en grandes beneficios para el análisis y selección de muestras a revisar, como son:

Alcance de revisión	General 100%
Selección de muestra para análisis	Casos relevantes
Tiempo de respuesta de revisión y análisis	5 - 10 días

Ventajas:

- Revisión cuantitativa y cualitativa de la información en forma general
- Tiempos de respuesta cortos
- Oportunidad
- Detección de irregularidades
- Factibilidad de detectar errores de análisis de otras áreas operativas

FORMA DE OPERACION DE AUDITORIA CON EL SISTEMA DE DESEMPEÑO DE TARJETAS DE CREDITO HOJA 1/5

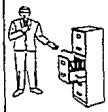
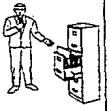
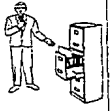
FORMA DE OPERACION DE AUDITORIA CON EL SISTEMA DE DESEMPEÑO DE TARJETAS DE CREDITO HOJA 2/5

FORMA DE OPERACION DE AUDITORIA CON EL SISTEMA DE DESEMPEÑO DE TARJETAS DE CREDITO

HOJA 3/5

AUDITORIA	GERENCIA DEL PRODUCTO	TARJETAS DE CREDITO	CENTRO DE SERVICIO INFORMATICO	ARCHIVO DE MICROFILMACION	CONTROL DE ALTAS	AREAS DE OPERACION
ANALIZA INFORMACION EMITA POR EL SISTEMA Y LA COMPARA VS ANALISIS RECIBIDOS DE GERENCIA Y DIVISION						RECIBE SOLICITUD
DETERMINA NECESIDAD DE OBTENER DIFERENCIAS EN INFORMACION SOLICITANDO INFORMACION ESPECIFICA					RECIBE SOLICITUD	
				RECIBE SOLICITUD		

FORMA DE OPERACION DE AUDITORIA CON EL SISTEMA DE DESEMPEÑO DE TARJETAS DE CREDITO HOJA 4/5

AUDITORIA	GERENCIA DEL PRODUCTO	TARJETAS DE CREDITO	CENTRO DE SERVICIO INFORMATICO	ARCHIVO DE MICROFILMACION	CONTROL DE ALTAS	AREAS DE OPERACION
				 BUSCA DOCUMENTOS Y ENVIA A AUDITORIA	 BUSCA DOCUMENTOS Y ENVIA A AUDITORIA	 BUSCA DOCUMENTOS Y ENVIA A AUDITORIA
 RECIBE DOCUMENTOS Y LISTADOS						
 ANALIZA DOCUMENTOS Y LISTADOS						

FORMA DE OPERACION DE AUDITORIA CON EL SISTEMA DE DESEMPEÑO DE TARJETAS DE CREDITO HOJA 5/5

III.2.2 DIAGRAMA ENTIDAD-RELACION

Objetivo

Mostrar las entidades de datos que surgieron a partir del diseño para apoyar funciones o procesos de Auditoría específicos y determinar las estructuras de datos que necesitarán el apoyo del Sistema de Desempeño de Tarjetas de Crédito.

Resumen

Se realizó utilizando los datos que estaban disponibles y los obtenidos en la fase de análisis de la información y se complementó a partir de entrevistas con el usuario, se determinaron las entidades núcleo de productos al consumidor que sirvieron de base para la creación del modelo entidad-relación. El modelo entidad-relación ayuda a estructurar los datos necesarios para las tareas de análisis, y las relaciones entre las entidades de datos, proporcionan una idea general de las funciones operacionales que el sistema debe desempeñar. Llegado este punto en el modelo entidad-relación se registraron únicamente las entidades y los elementos de datos claves necesarios para el desarrollo de nuestro sistema..

Plan de los recursos de datos

De un inventario general de datos se seleccionaron los datos que se consideran los más importantes. Se indicaron las entidades y sus

relaciones con el área de Auditoría y su entorno. Fue necesario reducir la cantidad de datos para aislar aquellas entidades que no determinan relevante importancia para el sistema.

Definición de las entidades núcleo

Se partió de un diagrama de modelo entidad-relación base surgido a partir de conversaciones hechas con los usuarios. El primer paso consistió en definir y documentar las posibles entidades núcleo o las principales entidades de datos. Estas constituyeron básicamente aquellas áreas .sobre las que deben registrarse datos para que se siga funcionando con éxito. Como regla general, las entidades núcleo pueden determinarse a través de conversaciones con los usuarios ya que suelen mencionarse en forma de sustantivos.

Relación de cardinalidad y del carácter facultativo

Cardinalidad

El número de ocurrencias de una entidad relacionado con ocurrencias específicas de la otra entidad expresado como una o muchas, ejemplo, un cliente puede tener muchas tarjetas, gráficamente en un modelo de datos "muchas" se expresa con la letra n y una con el número 1 como se vió en el capítulo 1.

Carácter facultativo

Carácter facultativo es el criterio que define si una entidad (ocurrencia) debe o no estar relacionada con otra entidad cuando se crea la primera entidad.

Relaciones entre entidades

Se definieron las relaciones de dependencia entre las entidades núcleo en lo que concierne a la cardinalidad de la relación y se determinó si la relación es facultativa u obligatoria, en nuestro caso se señalan únicamente las obligatorias.

Datos de transacciones de entrada salida

Se elaboró un modelo de entidad-relación y se comprobó que aparecen todas las entidades de datos pertinentes mediante su comparación con los documentos de entrada/salida actuales.

La decisión sobre qué tipo de documento utilizar, si de entrada o salida para la comparación es facultativa, aunque sólo debe revisarse un tipo. El uso de documentos de entrada tiene la ventaja de que los elementos de datos de entrada se encuentran normalmente una sola vez o un número reducido de veces, mientras que los elementos de salida se repiten según la información visualizada por el sistema. Por otro lado los datos de salida

tienen la ventaja de ser más pertinentes para nuestro caso ya que los usuarios los reconocen más rápidamente como sus datos.

Esquema global de datos

Aunque las entidades determinadas reflejarán en gran parte el sistema actual, se supone que los datos que el sistema y la institución necesitan son relativamente estables con relación al tiempo, por lo que los datos del Sistema de Tarjetas pueden utilizarse como la fuente principal para definir el esquema global de datos.

El modelo entidad-relación es un esbozo de las entidades que el Sistema de Desempeño de Tarjetas de Crédito necesitó básicamente depurándose y reorganizándose a medida que se fue reuniendo la información.

Si el Sistema de Desempeño de Tarjetas de Crédito está en un área de totalmente nueva, no habrá ningún sistema actual que verifique el modelo entidad-relación, por lo cual fue todavía más importante que el modelo entidad-relación se examinara durante la fase de diseño para asegurar su exactitud. La información derivada del modelo entidad-relación puede utilizarse como entrada en la lista de datos del Sistema.

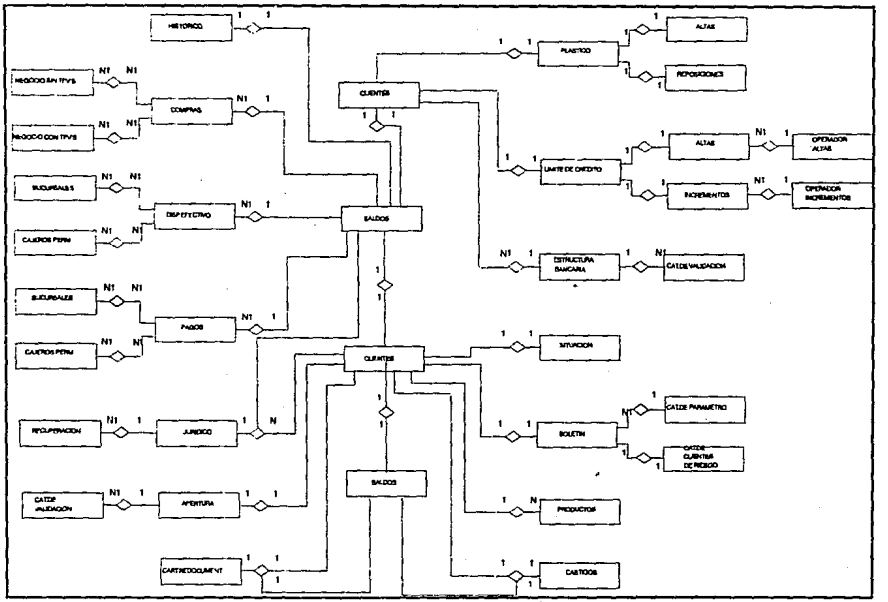


Diagrama Entidad-Relación

III.2.3 NORMALIZACIÓN

Como se sabe, la obtención de datos es un aspecto en el que se debe tener especial cuidado. Es necesario hacer una serie de consideraciones antes de iniciar la normalización. Primero: la información llega de Bases de Datos independientes, es decir de diferentes sistemas automatizados que aunque en sus propios sistemas las Bases de Datos se encuentran normalizadas, el producto final de datos que es necesario para el Sistema de Desempeño de Tarjetas Bancarias entre sí es un nuevo universo de información cuyas estructuras se deben normalizar. Segundo: podemos decir que el nuevo universo de información conforma el inventario de datos a normalizar, ya que la información proporcionada no lo estaba correctamente.

Como resultado de la normalización que a continuación se describe, se obtuvo la Base de Datos depurada.

PRIMERA FORMA NORMAL (1FN)

Cada registro está agrupado, tiene una longitud fija y no contiene grupos de repetición.

Registro de Saldos

Cuenta	Nombre Cliente	Fecha Apertura	Límite Crédito	Ciclo Corte	Meses Vencidos	Saldo Actual	Situación Cuenta	Sucursal	Producto

Registro de Producto

Producto	Nombre Producto	Número Línea	Límite mínimo	Límite máximo	Fecha Límite	Índice Cartera

Registro de Estructura

Sucursal	Nombre Sucursal	Número Area	Nombre Area	Número División	Nombre División	Número Regional	Nombre Regional

Registro de Altas

Número Operador	Cuenta	Límite Crédito	Número Nómina	Número Puesto	Nombre Operador	Nombre Puesto	Fecha Alta

Registro de Jurídico

Cuenta	Número Abogado	Nombre Abogado	Importe Recuperado	Fecha Asignación

Registro de Castigos

Cuenta	Fecha Castigo	Importe Castigos	Importe Interés	Capital deducible	Capital no deducible

Registro de Boletín

Cuenta	Fecha de Boletín	Fecha Boletín PB	Fecha Boletín D	Fecha Sobregiro

Registro de Inclim

Límite Autorizado	Fecha Aumento	Número Operador	Cuenta

SEGUNDA FORMA NORMAL (2FN)

Eliminando todas las relaciones en las que los datos no dependan completamente de la llave primaria del registro.

De la tabla Estructura, los campos Regional, Nombre Regional, División, Nombre División, Área, Nombre de Area, no dependen completamente del campo Sucursal, solamente Nombre Sucursal, por lo tanto, si vinculamos las tablas, nos resultan las siguientes tablas.

Registro de Estructura

Sucursal	Nombre Sucursal	Número Area	Número División	Número Regional

Registro de Nombre de Estructura

Sucursal	Número Area	Nombre Area	Número División	Nombre División	Número Regional	Nombre Regional

De la tabla Jurídico, el campo Nombre de Abogado no depende completamente del campo Número de cuenta depende del Número de Abogado, por lo tanto, si vinculamos , nos resultan las siguientes tablas.

Registro de Jurídico

Cuenta	Número Abogado	Importe Recuperado	Fecha Jurídico

Registro de Nombre Abogados

Número Abogado	Nombre Abogado

De la tabla Altas, el campo Nombre de Puesto no depende completamente del campo Número de Operador sino del Número de Cuenta, por lo tanto, si vinculamos , nos resultan las siguientes tablas.

Registro de Operador

Número Operador	Nombre Operador	Número de Nómina	Número Puesto	Nombre Puesto

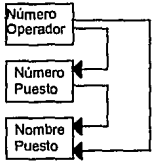
Registro de Puestos

Número Operador	Fecha Alta	Cuenta	Límite	Sucursal

TERCERA FORMA NORMAL (3FN)

Eliminar las relaciones que contengan dependencias transitivas.
De las tablas Estructura y Nombre de Estructura observamos que:

Número de Puesto depende de Número de operador y que Nombre de Puesto depende del Número de Puesto.



Por lo tanto, existe una dependencia transitiva entre Nombre de Puesto y Número de operador, vinculanddo las tablas, como resultante obtenemos:

Registro de Operador

Número Operador	Nombre Operador	Número de Nómina	Número Puesto

Registro de Puestos

Número Operador	Fecha Alta	Cuenta	Límite	Sucursal

Registro de Nombre de Puestos

Número Puesto	Nombre Puesto

III.2.4 DICCIONARIO DE DATOS

Nombre de archivo (DOS): TARSDOS.DBF

Nombre de Archivo (Usuario): SALDOS/CUENTA

Nombre de Archivo (Programas): TARSDOS

Num. Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	1	2	Numérico	0	X	SDO_CVEPRD	Clave o número de producto (banco)
2	3	16	Caracter	0		SDO_NUMCTA	Número de cuenta del tarjeta-habiente
3	19	26	Caracter	0		SDO_NOMCLI	Nombre del cliente
4	25	45	Caracter	0		SDO_DOMCAL	Calle del cliente
5	25	70	Caracter	0		SDO_DOMCOL	Colonia del cliente
6	25	95	Caracter	0		SDO_DOMEDO	Estado del cliente
7	5	120	Caracter	0		SDO_NUMCDP	Código postal
8	7	125	Numérico	0		SDO_TELCAS	Telefono particular
9	7	138	Numérico	0		SDO_TELOFI	Teléfono de oficina
10	1	139	Caracter	0		SDO_CVESIT	Clave de situacion de la cuenta
11	4	143	Numérico	0		SDO_NUMSUC	Número de sucursal o departamento
12	6	144	Numérico	0		SDO_CVEUTR	Clave de última transaccion
13	4	150	Numérico	0		SDO_FECUTR	Fecha de última transaccion
14	10	154	Numérico	0		SDO_FECPAG	Fecha de último pago
15	4	160	Numérico	0		SDO_FECACR	Fecha de último aumento de crédito
16	7	166	Numérico	0		SDO_LIMCRE	Limite de credito
17	4	172	Numérico	0		SDO_FECAPT	Fecha de apertura
18	2	176	Numérico	0		SDO_NUMCC	Dia de ciclo de corte
19	3	178	Numérico	0		SDO_NUMMV	Número de pagos vencidos
20	11	182	Numérico	0		SDO_SDOVDO	Saldo vencido
21	11	197	Numérico	0		SDO_SDOACT	Saldo actual
22	11	212	Numérico	0		SDO_MTOCAP	Importe de capital
23	11	227	Numérico	0		SDO_MTOINT	Importe de intereses
24	11	242	Numérico	0		SDO_MTOCOM	Importe de comisión
25	11	257	Numérico	0		SDO_MTOGTO	Importe de gastos de cobranza
26	11	272	Numérico	0		SDO_MTOIVA	Importe de I.V.A.
27	1	287	Numérico	0		SDO_CVEBOL	Clave de boletin
28	1	288	Numérico	0		SDO_CVECST	Clave de castigo

Longitud de registro: 288

Sistema: Sistema de Desarrollo de Tarjetas Bancarias

Nombre de archivo (DOS): TARALTAS.DBF
Nombre de Archivo (Usuario): ALTAS
Nombre de Archivo (Programas): TARALTAS

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	8	1	Caracter	0	X	ALT_NUMOPE	Numéro de operador
2	6	9	Numérico	0		ALT_FECALT	Fecha de alta de la cuenta
3	16	15	Caracter	0	X	ALT_NUMCTA	Numero de cuenta
4	6	31	Numérico	0		ALT_LIMOTO	Importe de limite de crédito otorgado
5	4	37	Numérico	0		ALT_NUMSUC	Número del depto. encargado de investigar

Longitud de registro: 40

Sistema: Sistema de Desarrollo de Tarjetas Bancarias

Nombre de archivo (DOS): TARINLIM.DBF
Nombre de Archivo (Usuario): INCRE_LIMITE
Nombre de Archivo (Programas): TARINLIM

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	8	1	Caracter	0	X	ILI_NUMOPE	Número de operador
2	6	9	Numérico	0		ILI_FECALT	Fecha de incremento de limite de crédito
3	16	15	Caracter	0	X	ILI_NUMCTA	Numéro de cuenta
4	6	31	Numérico	0		ILI_LIMAUT	Limite de crédito autorizado
5	6	37	Numérico	0		ILI_FECAUM	Fecha de aumento de límite de crédito
6	4	43	Numérico	0		ILI_NUMSUC	Número de sucursal/departamento

Longitud de registro: 49

Nombre de archivo (DOS): TAROPERA.DBF
Nombre de Archivo (Usuario): OPERADOR
Nombre de Archivo (Programas): TOPERA

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	8	1	Caracter	0	X	OPE_NUMOPE	Número de operador
2	26	9	Caracter	0		OPE_NOMOPE	Nombre del operador
3	10	35	Numérico	0		OPE_NUMNOM	Número de nomina del operador
4	5	48	Caracter	0		OPE_NUMPTO	Puesto del operador

Longitud de registro: 48

Nombre de archivo (DOS): TARPTOS.DBF
Nombre de Archivo (Usuario): PUESTO/OPERA
Nombre de Archivo (Programas): TARPTOS

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	5	1	Caracter	0	X	PTO_NUMPTO	Puesto del operador
2	15	6	Caracter	0		PTO_NOMPTO	Nombre del puesto

Longitud de registro: 20

Sistema: Sistema de Descripción de Tipos de Archivos

Nombre de archivo (DOS): TARJURID.DBF
Nombre de Archivo (Usuario): JURIDICO
Nombre de Archivo (Programas): TARJURID

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	16	1	Caracter	0	X	JUR_NUMCTA	Número de cuenta
2	5	17	Numérico	0	X	JUR_NUMABO	Número de abogado
3	15	22	Numérico	0		JUR_IMPREC	Importe recuperado
4	6	37	Numérico	0		JUR_FECASG	Fecha de asignación de abogado

Longitud de registro: 42

Sistema: Sistema de Descripción de Tipos de Archivos

Nombre de archivo (DOS): TARABOGA.DBF
Nombre de Archivo (Usuario): ABOGADOS
Nombre de Archivo (Programas): TARABOGA

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	5	1	Numérico	0	X	ABO_NUMABO	Número de abogado
2	25	6	Caracter	0		ABO_NOMABO	Nombre de abogado

Longitud de registro: 30

Sistema: Sistema de Desempeño de Tarjetas Bancarias

Nombre de archivo (DOS): TARCASTI.DBF
Nombre de Archivo (Usuario): CASTIGOS
Nombre de Archivo (Programas): TARCASTI

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	16	1	Caracter	0	X	CAS_NUMCTA	Número de cuenta
2	6	17	Numérico	0		CAS_FECCAS	Fecha de castigo de la cuenta
3	15	23	Numérico	0		CAS_IMPCAP	Importe de capital castigado
4	15	38	Numérico	0		CAS_IMPINT	Importe de capital castigado
5	15	53	Numérico	0		CAS_MTOCDE	Importe de capital deducible
6	15	68	Numérico	0		CAS_MTOCND	Importe de capital no deducible

Longitud de registro: 82

Sistema: Sistema de Desempeño de Tarjetas Bancarias

Nombre de archivo (DOS): TARBOLET.DBF
Nombre de Archivo (Usuario): BOLETIN
Nombre de Archivo (Programas): TARBOLET

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	16	1	Caracter	0	X	BOL_NUMCTA	Numéro de cuenta
2	6	17	Numérico	0		BOL_FECBOD	Fecha de ultimo boletin delincuente
3	6	23	Numérico	0		BOL_FECBOP	Fecha de ultimo boletin problemas de cobro
4	6	29	Numérico	0		BOL_FECSBG	Fecha de sobregiro

Longitud de registro: 34

Sistema: Sistema de Desempeño de Tarjetas Bancarias

Nombre de archivo (DOS): TARESTRU.DBF
Nombre de Archivo (Usuario): ESTRUCTURA
Nombre de Archivo (Programas): TARESTRU

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	4	1	Numérico	0	X	EST_NUMSUC	Número de sucursal/departamento
2	15	5	Caracter	0		EST_NOMSUC	Nombre de sucursal/departamento
3	4	20	Numérico	0	X	EST_NUMARE	Número de area
4	4	24	Numérico	0		EST_NUMDIV	Número de división
5	4	28	Numérico	0	X	EST_NUMREG	Número de dirección regional

Longitud de registro: 31

Sistema: Sistema de Desempeño de Tarjetas Bancarias

Nombre de archivo (DOS): TARNOMES.DBF
Nombre de Archivo (Usuario): NOM_ESTRUCTURA
Nombre de Archivo (Programas): TARNOMES

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	4	1	Numérico	0	X	NES_NUMARE	Numéro de area
2	15	5	Caracter	0		NES_NOMARE	Nombre del area
3	4	20	Numérico	0	X	NES_NUMDIV	Numéro de división
4	15	24	Caracter	0		NES_NOMDIV	Nombre del división
5	4	39	Numérico	0	X	NES_NUMREG	Numéro de dirección regional
6	15	43	Caracter	0		NES_NOMREG	Nombre de dirección regional

Longitud de registro: 57

Sistema: Sistema de Desempeño de Tarjetas Bancarias

Nombre de archivo (DOS): TUSUARIO
Nombre de Archivo (Usuario): USUARIOS
Nombre de Archivo (Programas): TUSUARIO

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	8	1	Caracter	0	X	USU_LOGIN	Login del usuario
2	35	9	Caracter	0		USU_NOM	Nombre del usuario
3	8	44	Caracter	0		USU_PASS	Password del usuario
4	7	52	Numérico	0		USU_TEL	Teléfono del usuario
5	10	59	Caracter	0		USU_PRIV	Privilegios de acceso al sistema

Longitud de registro: 68

Sistema: Sistema de Desempeño de Tarjetas Bancarias

Nombre de archivo (DOS): TBITACOR.DBF
Nombre de Archivo (Usuario): BITACORA
Nombre de Archivo (Programas): TBITACOR

Num Campo	Long. Campo	Posición Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	8	1	Caracter	0	X	BIT_LOGIN	Login del usuario
2	8	17	Date	0		BIT_FECENT	Fecha de entrada al sistema
3	8	26	Caracter	0		BIT_HRAENT	Hora de entrada
4	8	33	Caracter	0		BIT_HRASAL	Hora de salida

Longitud de registro: 32

Sistema: Sistema de Desempeño de Tarjetas Bancarias

Nombre de archivo (DOS): TARPROD.DBF

Nombre de Archivo (Usuario): PRODUCTO

Nombre de Archivo (Programas): TARPROD

Num Campo	Posición Campo	Long. Campo	Tipo Campo	Puntos Decim.	Indice	Nombre Campo	Descripción
1	2	1	Numerico	0	X	PRD_NUMPRD	Número de producto o servicio
2	20	3	Caracter	0		PRD_NOMPRD	Nombre del producto o servicio
3	2	23	Numerico	0		PRD_NUMLIM	Número del linea>
4	6	25	Numerico	0		PRD_LIMMAX	Límite de crédito máximo
5	6	31	Numerico	0		PRD_LIMMIN	Límite de crédito mínimo
6	6	37	Numerico	0		PRD_FECACL	Fecha de última actualización
7	4	43	Numerico	2		PRD_TASRGO	Indice de cartera vencida

Longuitud de registro: 46

III.2.5 DIAGRAMA JERARQUICO DE PROCESOS

Un sistema está formado por varias actividades o procesos relacionados pertinentemente en los que se indican en forma lógica los pasos para ejecutarlos. En el desarrollo de sistemas de información lo más conveniente es establecer módulos independientes que desde luego interactúan entre sí. Tales módulos para su mayor entendimiento, se representan por medio de diagramas de jerarquía en los que se observan en orden los procedimientos que conforman al sistema.

Estos diagramas de proceso pueden continuar hasta los niveles que sean necesarios para identificar las actividades que forman parte del sistema.

En el diagrama de jerarquía debe incluirse cualquier actividad que genere, modifique o utilice información. Lo común es que se necesiten, de acuerdo con la naturaleza del sistema entre tres y siete niveles.

En esta sección se muestra el diagrama de jerarquía aplicable a nuestro caso en el que fueron necesarios 5 niveles, en el mismo describe al sistema como la colección de actividades y procesos como ayuda para visualizar los niveles que son útiles para la programación.

Módulos principales

Sistema de desempeño. Es el encargado de controlar todo el procesamiento. Llama programas para la administración, procesamiento mensual de la cartera, consultas a la base de datos y utilerías o servicios del sistema

Administración del Sistema. Administra el uso del sistema, manteniendo el control de los usuarios que accesan al sistema, actualizando los parámetros del los productos ó distintos tipos de tarjetas de crédito, además de realizar la conversión inicial de los datos de entrada.

Conversión de Datos. Realiza la conversión de formato de los datos de entrada que se proporcionan mensualmente a través de los cartuchos, dejando los archivos en formato .DBF utilizado por el sistema.

Mantenimiento de Usuarios. Actualiza la información de usuarios de la aplicación. Identificación personal, clave o password, nombre completo de usuario, asignación de atributos o privilegios de acceso dentro del sistema y registro de una bitácora.

Parámetros del producto. Actualiza la información de los productos ó tarjetas bancarias. Tipo de producto, línea a la que pertenece, límites de crédito autorizados e índice de cartera.

Análisis de la cartera. Se encarga del análisis de la cartera de tarjetas de crédito explotando sistemáticamente la información de las bases de datos, para medir el desempeño de las distintas carteras y puntos de control específicos. El resultado de este análisis proporciona información gerencial y a detalle de los distintos tipos de productos, la cual sirve de apoyo para la toma de decisiones en las labores de Auditoria.

Apertura y Manejo de Crédito. Revisa el comportamiento de las cuentas de apertura reciente y manejo de las otra cuentas con mayor tiempo de permanencia, detectando niveles de riesgo.

Sobregiros. Revisa las cuentas sobregiradas, detectando las más relevantes de acuerdo a su saldo además de revisar la recuperación de dicho sobregiro.

Cartera vencida. Clasifica los niveles de cartera de acuerdo a el número de meses vencidos que presentan las cuentas para observar el nivel de riesgo de la cartera así como la revisión de cuentas castigadas y cuentas en redocumentación.

Cartera preventiva. Obtiene información de clientes que representan un riesgo potencial de acuerdo al número de tarjetas que poseen y su validación el boletines de prevención.

Consultas.	En este módulo se llevan a cabo la inspección de información a las bases de datos tomando en consideración las condiciones que establezca el usuario para la obtención de cuentas de tarjetahabientes.
Obtención Individual.	Realiza consultas de cuentas individuales proporcionadas por el usuario
Obtención por Filtro.	Realiza consultas de aquellas cuentas que se obtienen a través de las restricciones de cantidades o condiciones dadas por el usuario.
Utilerías.	Proporciona al usuario herramientas que le permiten mantener la operabilidad y seguridad del sistema.

DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

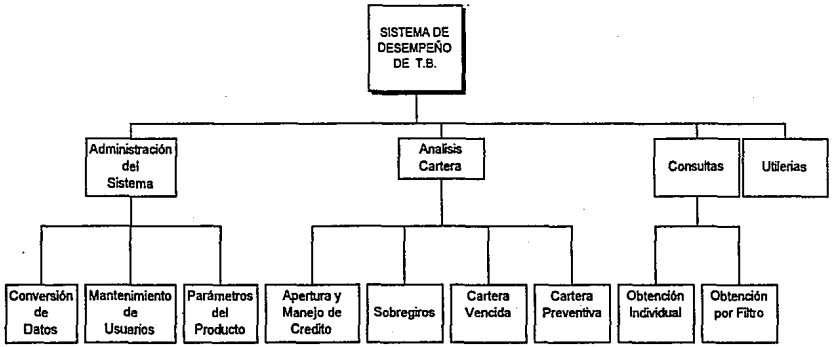


DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

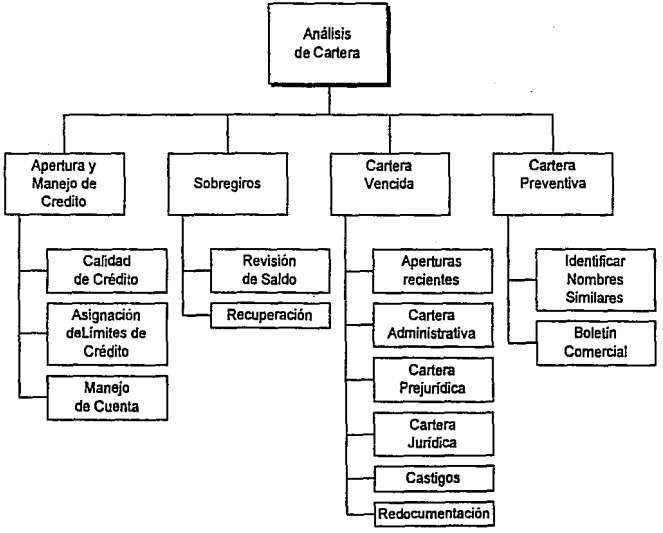


DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

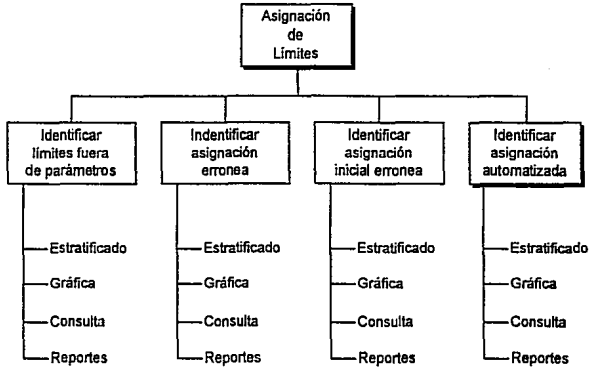


DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

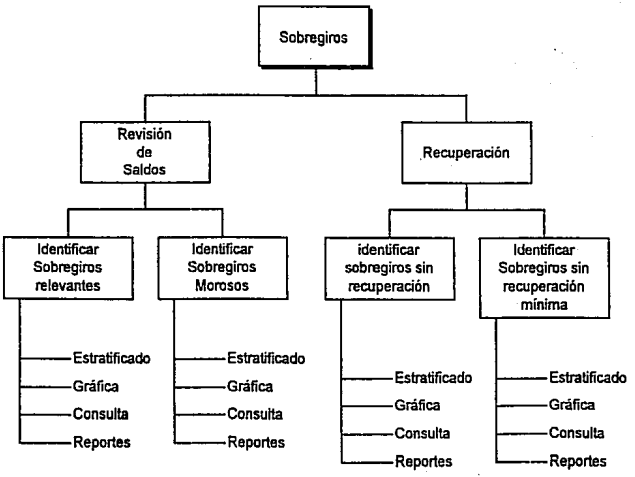


DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

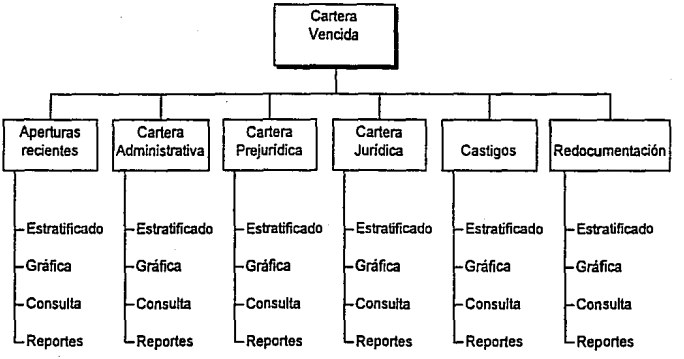


DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

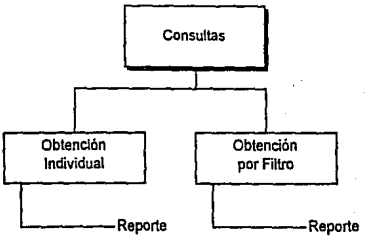


DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS

III.2.6 DIAGRAMA DE FLUJO DE DATOS

Calidad de crédito

Cuentas de más alto riesgo

Se refiere a la identificación en forma cuantitativa y cualitativa de tarjetas de crédito de apertura reciente (últimos 13 meses) y que presentan debilidades de pago, lo cual se proyecta en el número e importe de mensualidades vencidas.

Salidas

Estratificado y/o gráfica por:

- producto
- línea
- general

Información a detalle de la cuenta y el operador.

Cuentas sin pago alguno

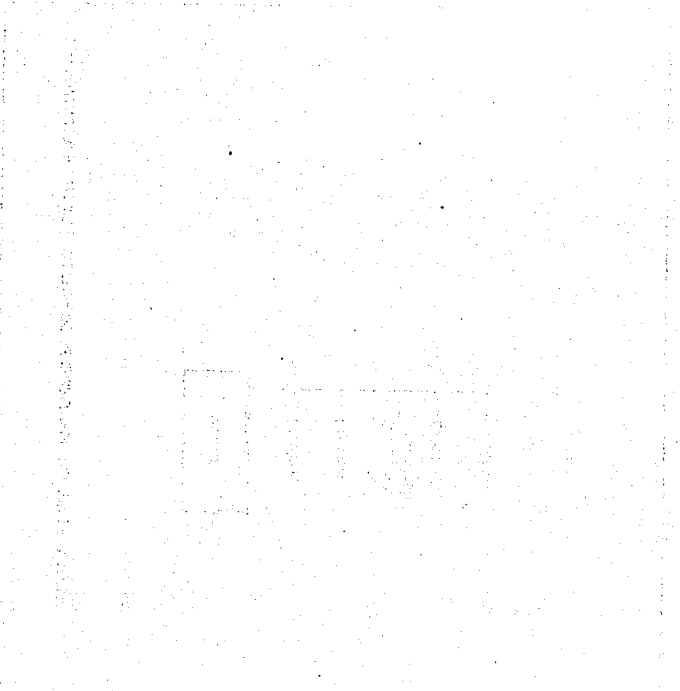
Se refiere a tarjetas de crédito de las cuales no se ha efectuado ningún pago desde su apertura.

Salidas

Estratificado y/o gráfica por:

- producto
- línea y

DIAGRAMA JERARQUICO DEL SISTEMA DE DESEMPEÑO DE TARJETAS BANCARIAS



III.2.6 DIAGRAMA DE FLUJO DE DATOS

Calidad de crédito

Cuentas de más alto riesgo

Se refiere a la identificación en forma cuantitativa y cualitativa de tarjetas de crédito de apertura reciente (últimos 13 meses) y que presentan debilidades de pago, lo cual se proyecta en el número e importe de mensualidades vencidas.

Salidas

Estratificado y/o gráfica por:

- producto
- línea
- general

Información a detalle de la cuenta y el operador.

Cuentas sin pago alguno

Se refiere a tarjetas de crédito de las cuales no se ha efectuado ningún pago desde su apertura.

Salidas

Estratificado y/o gráfica por:

- producto
- línea y

- general

Información a detalle de la cuenta.

Departamento y operador de investigación de crédito

Muestra e identifica las incidencias del operador de investigación de tarjetas de crédito que presenta debilidades de pago en cualquiera de los dos puntos anteriores, así como del departamento de crédito.

Salidas

Estratificado y/o gráfica por:

- operadores
- departamento
- área

Información a detalle de la cuenta, operador y área a la que pertenece.

Límites otorgados fuera de las políticas y parámetros de cada una de las tarjetas de crédito

Se refiere a la identificación en forma cuantitativa y cualitativa de las tarjetas de crédito que se encuentran fuera de los parámetros

Salidas

Estratificado y/o gráfica por:

- producto
- línea

- general
- límite de crédito

Información a detalle de la cuenta y límite otorgado.

Límites de crédito asignados erróneamente

Se refiere a la identificación en forma cuantitativa y cualitativa de las tarjetas de crédito que se encuentran en asignación de límites de crédito en forma errónea por parte del operador.

Salidas

Estratificado y/o gráfica por:

- producto
- línea
- general

Información a detalle de la cuenta, operador y límite otorgado.

Departamento y operador de crédito inicial otorgado

Identificar en forma cualitativa y cuantitativa a los operadores de investigación de crédito que otorgan en forma inicial el crédito fuera de las políticas del producto.

Salidas

Estratificado y/o gráfica por:

- operador
- departamento

- área

Información a detalle de la cuenta y operador.

Límites de crédito asignados por sistema automatizado

Identificar en forma cualitativa y cuantitativa las tarjetas de crédito que se encuentran con incremento del límite de crédito en forma automatizada y la cuenta se encuentra en situación anormal.

Salidas

Estratificado y/o gráfica por:

- producto
- línea
- general
- límite de crédito

Información a detalle de la cuenta, operador y límite otorgado.

Saturación del crédito de aperturas recientes

Identificar en forma cualitativa y cuantitativa tarjetas de crédito de apertura reciente (últimos seis meses) y las cuales presentan saturación de crédito.

Salidas

Estratificado y/o gráfica por:

- producto
- línea
- general

- límite de crédito

Información a detalle de la cuenta y límite otorgado.

Saldo vencido superior al crédito otorgado

Identificar en forma cualitativa y cuantitativa tarjetas de crédito que presenten el saldo vencido superior al límite de crédito.

Salidas

Estratificado y/o gráfica por:

- producto
- línea
- general
- límite de crédito

Información a detalle de la cuenta y límite otorgado.

Saldo total superior al límite de crédito

Identificar en forma cualitativa y cuantitativa aquellas cuentas cuyo saldo total rebasa al límite de crédito otorgado, es decir, aquellas cuentas que presentan sobregiro y el monto del mismo.

Salidas

Estratificado y/o gráfica por:

- producto
- línea
- general

Información a detalle de la cuenta.

Sobregiros morosos

Identificar en forma cualitativa y cuantitativa aquellas cuentas que presentan sobregiro y además no hayan realizado ningún pago desde su apertura. A este tipo de situación se le denomina "sobregiros morosos".

Salidas

Estratificado y/o gráfica por:

- producto
- línea
- general

Información a detalle de la cuenta.

Sobregiros sin recuperación

Obtener en forma cualitativa y cuantitativa la información de aquellas cuentas que no han realizado algún pago desde el momento en que ocurrió el sobregiro.

Salidas

Estratificado y/o gráfica por:

- producto
- línea y
- general

Información a detalle de la cuenta e inclusión en el boletín.

Sobregiros sin recuperación de capital .

Obtener en forma cualitativa y cuantitativa, información de las cuentas que han realizado pagos aplicados al sobregiro pero sin cubrir la cantidad mínima del capital.

Salidas

Estratificado y/o gráficas por:

- línea
- producto
- general

Información a detalle de la cuenta e inclusión en boletín.

Cartera vencida de aperturas recientes

Analizar en forma cualitativa y cuantitativa las cuentas que presentan cualquier número de meses vencidos cuya fecha de apertura está comprendida dentro de los últimos 6 meses.

Salida

Estratificado y/o gráfica por:

- producto
- línea
- general

Información a detalle de la cuenta.

Cartera administrativa

Analizar cualitativa y cuantitativamente aquellas cuentas que presentan hasta tres meses vencidos.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Cartera prejurídica

Analizar cualitativa y cuantitativamente aquellas cuentas que presentan hasta cuatro o cinco meses vencidos.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Cartera prejurídica no actualizada

Analizar cualitativa y cuantitativamente aquellas cuentas que presentan hasta cuatro o cinco meses vencidos pero que su saldo vencido es diferente al saldo total.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Cartera jurídica

Analizar cualitativa y cuantitativamente las cuentas a las que se les ha impuesto un castigo.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Cartera jurídica no actualizada

Analizar cualitativa y cuantitativamente las cuentas que presentan más de o cinco meses vencidos y que además se les ha impuesto un castigo.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Cartera de castigos

Identificar en forma cualitativa y cuantitativa las cuentas a las que se les han impuesto castigos.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Cartera en redocumentación

Identificar cualitativa y cuantitativamente aquellas cuentas cuyo deuda es negociada entre el tarjetahabiente y el Banco y que han llegado a un compromiso para cubrir de alguna forma el adeudo.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Recuperación de cartera jurídica

Obtener información de la recuperación de aquellas cuentas que están en procesos de índole jurídica, pero que no han ingresado a la cartera de castigos.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta y abogados correspondientes.

Clientes con dos o más tarjetas sobregiradas y/o vencidas

Detectar aquellos clientes que presentan patrones de nombres similares para deducir que se trata de una misma persona con dos o más tarjetas de crédito en situaciones de atraso y/o sobregiro.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Emisión de boletín de cartera preventiva para sucursales y departamentos.

Producir información en medios magnéticos y/o listados de aquellos clientes de los que existe la sospecha de que poseen dos o más tarjetas de crédito, a través de la comparación de sus nombres, sin importar la situación en que se encuentran.

Archivo y/o listado del detalle de las cuentas.

Validación de clientes con dos o más tarjetas sobregradas y/o morosas

Obtener la información detallada de aquellos clientes que probablemente poseen dos o más tarjetas y además se encuentran en el boletín de prevención comercial.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta.

Validación de transacciones en boletín comercial

Obtener la información detallada de aquellos clientes que probablemente poseen dos o más tarjetas y además están realizando transacciones a pesar de estar boletinadas.

Salidas

Estratificado y/o gráfica por:

- línea
- producto
- general

Información a detalle de la cuenta y boletín.

III.2.7 MINIESPECIFICACIONES

Cuentas de más alto riesgo

- apertura últimos 13 meses
- mensualidades vencidas mayor a 2
- saldo vencido mayor de la media de la cartera vencida del producto

Cuentas sin pago alguno

- Fecha de último pago igual a cero

Departamento y operador de investigación de crédito

- Número y nombre de operador se relaciona con tarjetas de crédito con debilidades en la calidad de crédito otorgado.

Límites otorgados fuera de las políticas y parámetros de cada una de las tarjetas de crédito

- fecha de apertura menor o igual a la fecha de última actualización de parámetros
- límite de crédito diferente a los rangos autorizados por producto
- fecha de último aumento de crédito igual a cero

Límites de crédito asignados erróneamente

- fecha de último aumento de crédito diferente de cero
- número de cuenta existe en modificaciones por operador y fecha de último aumento en TAR-SALDOS es igual fecha de ultimo pago en TAR-INCLIM

- fecha de último aumento es mayor o igual a fecha de último boletín o fecha de último aumento es mayor a fecha de último boletín o fecha de último pago es igual a cero

Departamento y operador de crédito inicial otorgado

- Número de operador que se relacione con la tarjeta de crédito, la cual presenta límite de crédito otorgado fuera de políticas del producto.

Límites de crédito asignados por sistema automatizado

- fecha de último aumento de crédito diferente de cero
- número de cuenta no existe en modificaciones por operador y fecha de último aumento en TAR-SALDOS es igual fecha de ultimo pago en TAR-INCLIM
- fecha de último aumento es mayor o igual a fecha de último boletín o fecha de último aumento es mayor a fecha de último boletín o fecha de último pago es igual a cero

Saturación del crédito de aperturas recientes

- apertura últimos 13 meses
- el saldo total representa el 80 % o más del límite de crédito

Saldo vencido superior al crédito otorgado

- Saldo vencido mayor al límite de crédito
- identificar rango de %

Saldo total superior al límite de crédito

- clave de castigo igual a cero

- saldo total mayor al límite de crédito arriba de la media
- fecha de último pago diferente de cero

Sobregiros morosos que no hayan realizado ningún pago desde su apertura

- clave de castigo igual a cero
- saldo total mayor al límite de crédito arriba de la media
- fecha de último pago igual a cero

Sobregiros sin recuperación del saldo desde la fecha del sobregiro.

- clave de castigo igual a cero
- saldo total mayor al límite de crédito
- fecha de último pago menor a fecha de sobregiro

Sobregiros sin recuperación mínima del capital .

- clave de castigo igual a cero
- saldo total mayor al límite de crédito
- intereses mayor a capital

Cartera de aperturas recientes

- ciclo de corte diferente a 18
- apertura los últimos 13 meses
- fecha de castigo igual a cero
- sin restricción de meses vencidos
- producto diferente de 30

Cartera administrativa

- ciclo de corte diferente a 18

- apertura no esté en los últimos 13 meses
- fecha de castigo igual a cero
- meses vencidos en 0,1,2,3
- producto diferente de 30

Cartera prejurídica

- ciclo de corte diferente a 18
- apertura no esté en los últimos 13 meses
- fecha de castigo igual a cero
- meses vencidos en 4,5
- producto diferente de 30
- saldo vencido igual a saldo total

Cartera prejurídica no actualizada

- ciclo de corte diferente a 18
- apertura no esté en los últimos 13 meses
- fecha de castigo igual a cero
- meses vencidos en 4,5
- producto diferente de 30
- saldo vencido diferente de saldo total

Cartera jurídica

- ciclo de corte diferente a 18
- fecha de castigo diferente de cero
- producto diferente de 30

Cartera jurídica no actualizada

- ciclo de corte diferente a 18
- fecha de castigo diferente de cero
- producto diferente de 30
- meses vencidos mayor a 5

Cartera de castigos

- fecha de castigo diferente de cero
- producto igual a 30

Cartera en rredocumentación

- banco igual a 30

Recuperación de cartera jurídica

- ciclo de corte igual a 18
- fecha de castigo igual a cero
- producto diferente de 30

Clientes con dos o más tarjetas sobregiradas y/o vencidas

- saldo sobregirado de más de un mes vencido
- nombre igual

Emisión de boletín de cartera preventiva para sucursales y departamentos.

- clientes con dos o más tarjetas

Validación de boletín comercial

- clientes con dos o más tarjetas sobregiradas y/o vencidas

- sobregiro mayor al 15%
- fecha último pago igual a cero

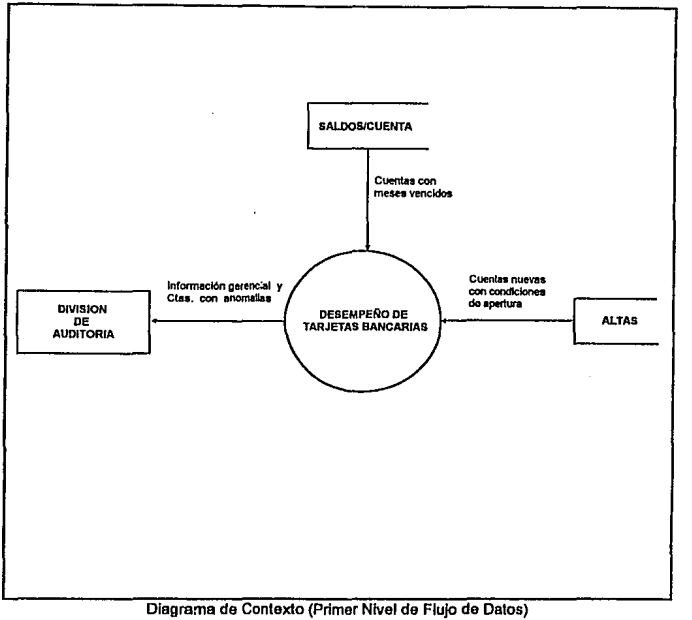


Diagrama de Contexto (Primer Nivel de Flujo de Datos)

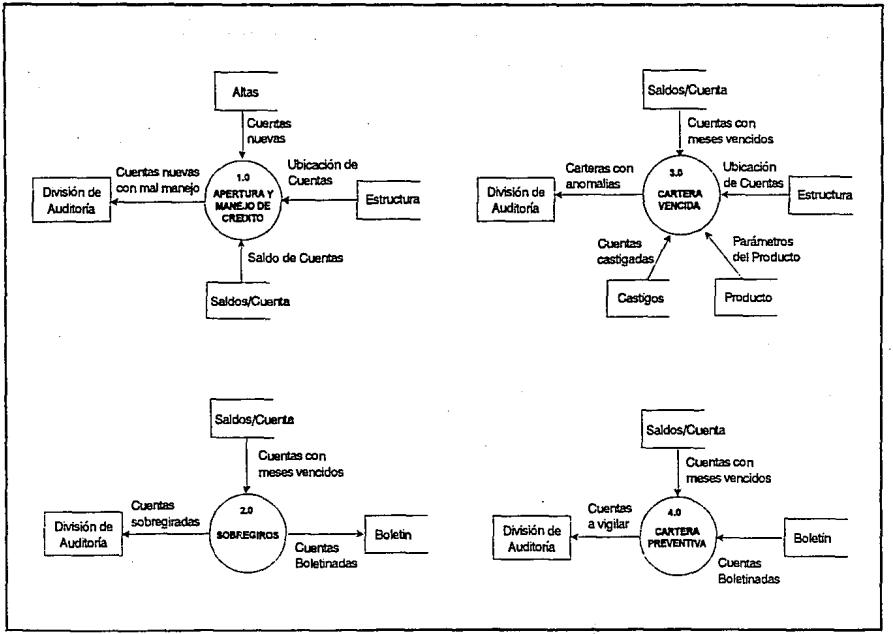


Diagrama de Flujo de Datos de Segundo Nivel

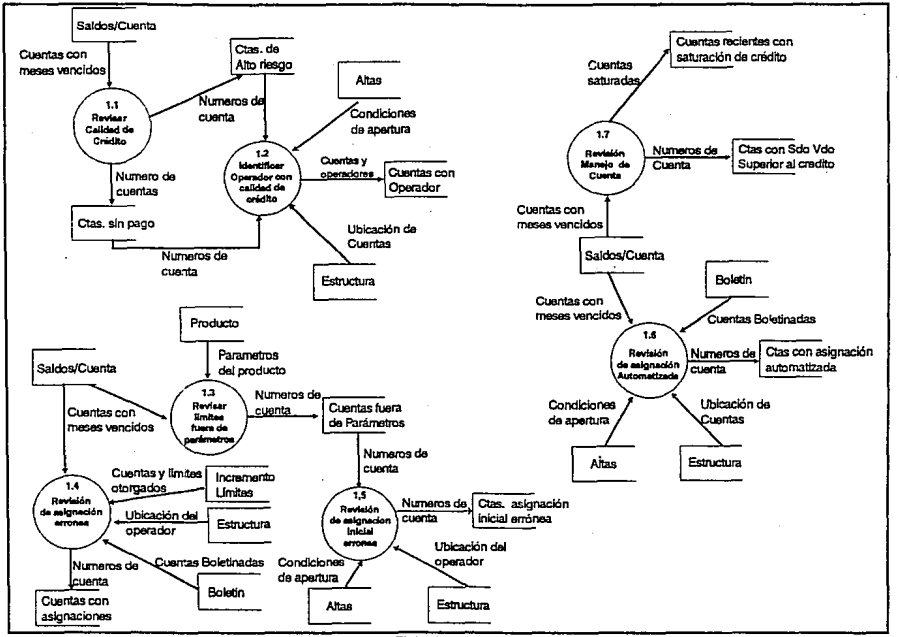


Diagrama de Flujo de Datos de Tercer Nivel

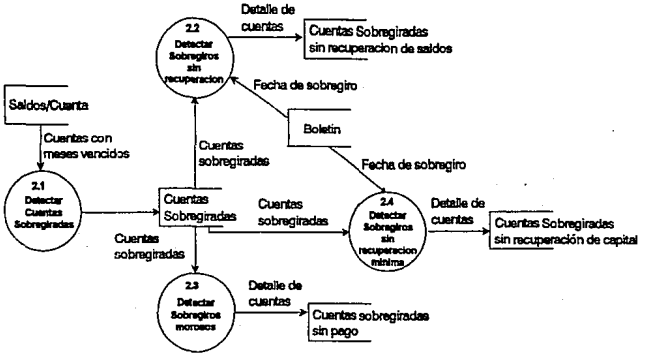


Diagrama de Flujo de Datos de Tercer Nivel

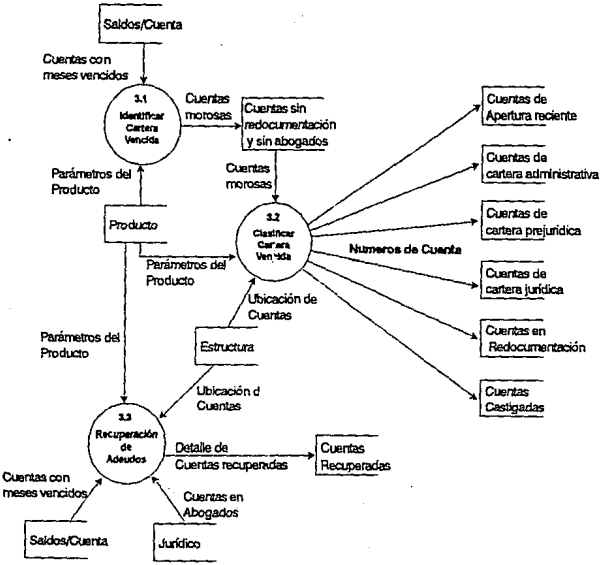


Diagrama de Flujo de Datos de Tercer Nivel

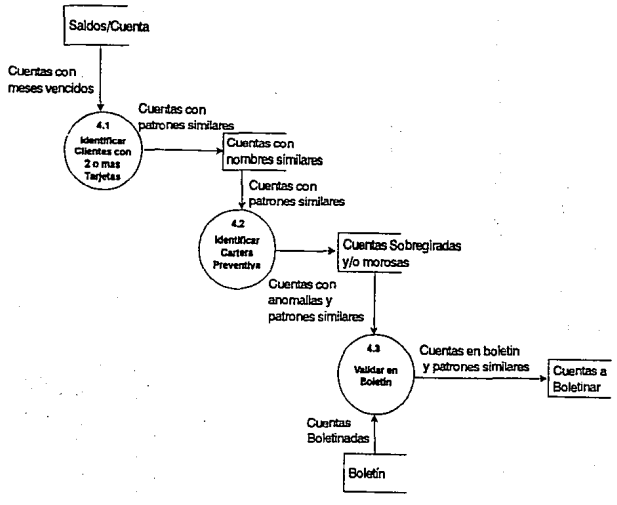


Diagrama de Flujo de Datos de Tercer Nivel

III.3 DESARROLLO DEL SISTEMA

En la fase de desarrollo del "Sistema de Desempeño de Tarjetas Bancarias" (SDTAR), se tomó en cuenta el modelo que recomienda Microsoft para la programación del mismo. (Fig. 3.3.1).

El desarrollo óptimo de una aplicación en FoxPro para Windows se lleva a cabo mediante el Manejador de Proyectos.

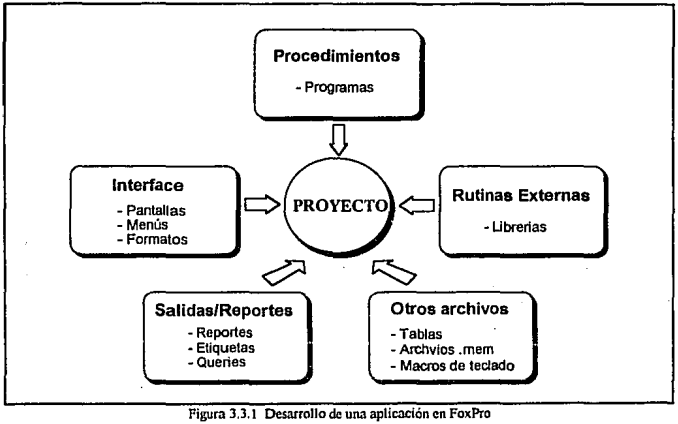


Figura 3.3.1 Desarrollo de una aplicación en FoxPro

Un proyecto es el mecanismo de unificación que reúne todas las piezas de una aplicación en una sola

El manejador de proyectos esta diseñado para automatizar el uso y la actualización de archivos que pertenecen a proyectos específicos así como la construcción de programas de aplicación o programas ejecutables basados en los archivos que conforman un proyecto.

De esta manera, utilizando las herramientas de FoxPro (descritas en el capítulo I. "Manejadores de Bases de datos"), es posible la construcción de las interfaces de usuario, la obtención y presentación de información y la unión de componentes de una aplicación dentro de un archivo .APP o .EXE con la ayuda del Manejador de Proyectos de FoxPro.

Manipulación de Datos

Debido a que el manejador de bases de datos FoxProw es una herramienta integrada a él, FoxPro sabe por sí solo como trabajar con una tabla y sus respectivos índices. Las definiciones de un archivo son almacenadas al inicio de un archivo-tabla. En FoxPro existen dos tipos de archivos índice: los archivos índice estructurales y los no estructurales. Con los archivos índice estructurales FoxPro los abre y mantiene automáticamente, mientras que los no estructurales se deben abrir explícitamente.

El manejador de Bases de datos de FoxPro almacena datos en tablas, las cuales están formadas por registros (renglones) y campos (columnas). Cada tabla es almacenada en un archivo DBF. Una tabla consta de un número de registros de longuitud fija, sin embargo existen campos "Memo" de longutud variable que son almacenados en un archivo por separado.

En FoxPro los registros son agregados en dos pasos. Primero se agrega un registro con *Append* o con *Insert* y después se reemplazan los campos vacios con los datos correspondientes. Cada registro es asignado a un número secuencial de registro único que identifica la posición física del registro dentro de la base. Si se remueven algunos registros de la tabla, lo registros siguientes son movidos hacia arriba, asumiendo números de registro diferentes. Al igual que la inserción de registros, el borrado de ellos se realiza en dos pasos, primero se marcan para su borrado, lo cual consiste en fijar un valor en un campo vacío del registro indicando que deben ser borrados y posteriormente el archivo-tabla es empacado *(Packed)* para que sean removidos físicamente de la tabla todos los registros marcados.

En FoxPro se puede oredenar físicamente una tabla, sin embargo es preferible el uso de índices para especificar el orden lógico de los registros. Cualquier número de índices pueden ser creados, pero el mantenimiento de un número grande de ellos puede afectar al rendimiento.

En lo que respecta al alcance de los comandos aplicados a los registros, en FoxPro algunos comandos relacionados con tablas tienen un alcance implícito. El "default" es todos (*ALL*), mientras que otros se aplican solo para el registro en curso. En la mayoria de los casos se pueden aplicar alcances tales como, *ALL*, *NEXT* x (los siguientes x registros) y *REST* (el resto de la tabla).

En cuanto al manejo de tablas, FoxPro puede operar con tablas múltiples. Se hace referencia a una tabla al seleccionar (*SELECT*) su área de trabajo, permitiendo operar hasta 225 tablas simultáneamente.

Se pueden establecer relaciones uno-a-uno y uno-a-muchos al ligar las tablas sobre valores comunes a ellas. Una liga es hecha desde un campo en una tabla a un índice en la otra tabla. Varias tablas pueden ser ligadas en varias formas, tales com A-B, más A-C, más A-D, A-B-C-D, y casi cualquier otra combinación que no cree ligas circulares.

Para la localización de datos, existen varias maneras de hacerlo:

- *LOCATE* y *SCAN..ENDSCAN* son dos ejemplos de comandos que se mueven a través de una tabla un registro a la vez. Usando esta idea para una búsqueda secuencial es simple y versátil, pero relativamente lento.

- *SEEK* usa un índice para saltar directamente al registro deseado. Este comando es muy rápido, pero requiere que los datos sean indexados y el archivo índice correspondiente esté activo. *SEEK* es apropiado para moverse a un registro específico.

- *SELECT* es un comando de SQL que es relativamente no procedural. Abre tablas según sea necesario y usa cualquier índice disponible. *SELECT* es apropiado para localizar y obtener un conjunto de registros rápidamente.

Proceso de Programación

Aunque en FoxPro no existe una regla especifica para la progración de una aplicación, se tomaron en cuenta para la tesis los siguientes pasos:

- Crear un proyecto nuevo con el "Project Manager".
- Definir la base de datos, incluyendo tablas e índices.
- Crear las pantallas principales con el "Screen Builder". Involucra el conocimiento detallado de los datos que serán capturados y desplegados.
- Crear los menús principales con el "Menu Builder". Requiere identificar los procesos que se pretenden proporcionar.
- Escribir el código para el control de *loop* principal. *Do While Loop.*
- Escribir el código para las rutinas que realizan las operaciones principales de la aplicación. Se refiere a los procedimientos y/o funciones
- Conectar los distintos módulos al *loop*, usando comunmente una estructura DO-CASE
- Perfeccionar la aplicación con rutinas de validación de entrada de datos, diseño de pantallas, colores y tipos letras.
- Optimizar para un mejor rendimiento. Concierne a todo lo que este dentro de un loop. Usar en lo posible comandos de SQL
- Realizar pruebas usando *Trace, Debug y FoxDoc* para identificar problemas.

III.4 IMPLEMENTACIÓN Y EVALUACIÓN

Los conceptos considerados para la implantación del sistema son:

- Planeación de la implantación
- Conversión de datos y
- Prueba de aceptación.

Planeación de la implantación

El plan de implantación es la clave de una implantación exitosa. La implantación debe constituir simplemente, la ejecución del plan.

El plan de implantación debe tener en cuenta:

- El enfoque de la implantación, es decir, directa, paralela, por fases o por etapas. El enfoque más efectivo y práctico para la implantación, debió determinarse teniendo en cuenta, factores como el grado de complejidad del sistema, la magnitud de los cambios organizativos necesarios, la sofisticación técnica de los usuarios del sistema, la dispersión geográfica de los locales de los usuarios y las restricciones de tiempo así como los períodos "pico" de ejecución del sistema. En nuestro caso la implantación fué directa, es decir, no existía un sistema automatizado por lo que las consideraciones de implantación surgieron a la par del inicio del proyecto. Se realizó la

ejecución del "Sistema" en paralelo con los procesos manuales existentes, pero sólo fue para comparación de resultados.

- La planificación de la implantación en la cual se tomaron en cuenta las actividades clave de la implantación que pueda afectar al cumplimiento de los plazos que abarca el plan, como las sesiones de formación y las fechas de entrega e instalación del Hardware/Software.

- La labor de conversión teniendo en cuenta el nivel estimado de esfuerzo y los requerimientos de planificación de cuestiones como el trabajo necesario para la transmisión de datos y la aprobación de los resultados de conversión.

Conversión de datos.

Muchos proyectos de desarrollo de sistemas si no la mayoría requieren cierta labor de conversión de datos. Aunque la labor de conversión se considera que está fuera de la corriente principal de las tareas de un proyecto de implantación, la magnitud de la labor necesaria para planificar y realizar una conversión no debe subestimarse. Si la conversión se mide por el número de programas, por la complejidad de la lógica y por el nivel de esfuerzo necesario, esta puede constituir a menudo un subsistema por derecho propio.

Por lo tanto la labor de la conversión de datos puede considerarse como el desarrollo y la implantación de un subsistema. En muchos casos la conversión de datos puede ser un proyecto de desarrollo individual. En nuestro caso constituye una labor de suma importancia, ya que como se estudió en la fase alternativas de solución, la información para el sistema de desempeño de tarjetas de crédito, se origina en un equipo TANDEM, por lo que es muy importante haber definido la estrategia de conversión de datos de manera independiente ya que ésta se realiza periódicamente.

Prueba de aceptación

La prueba de aceptación se realiza durante la etapa de implantación. La prueba de aceptación confirma que el sistema terminado cumple los requisitos acordados al final de la etapa de análisis.
Se lleva a cabo la prueba de aceptación y se verifican los resultados. Se obtiene del usuario la aprobación del sistema ejecutable. Si la prueba de aceptación se realiza satisfactoriamente, el sistema se acepta y se transfiere a operaciones informáticas.

La prueba de aceptación se ejecuta durante la implantación, bien que la planificación y los criterios de la prueba de aceptación deben definirse antes de terminar el diseño. El área de Auditoría y los elaboradores sólo podrán estar seguros del objetivo que el sistema está intentando cumplir si los criterios de aceptación se han elaborado con antelación y se cumplen cabalmente.

FORMACIÓN

Objetivo

Los objetivos de la fase de formación son:

- Proporcionar a Auditoría una idea general del nuevo sistema que le permita comprender el sistema.
- Enseñar a los usuarios cómo llevar a cabo sus procedimientos de trabajo con el nuevo sistema.

Entradas

Resumen de operación
Diseño
Procedimiento de usuario

Tareas

Personal que debe formarse
Alcance y estrategia de la formación
Material de curso
Formación de los instructores
Clases de formación modelo
Sesiones de formación

Productos parciales

Material del curso
Manual del instructor
Planificación de las sesiones de formación

Manual de la fase completa

Cursos de formación

CONVERSIÓN DE DATOS

Objetivo

Proporcionar mediante la conversión de datos, los medios principales para inicializar los datos para el nuevo sistema a implantar. la conversión de datos se llevó a cabo a través de medios automatizados, incluyó la creación de cualquier dato necesario para los registros de datos nuevos. la conversión permitió al sistema reflejar con precisión el estado actual de los datos organizativos al principio de la producción.

Entradas

Definición de requerimientos
Formatos de archivo del sistema actual

Tareas

Análisis de los requerimientos de conversión de datos y diseño del sistema de conversión
Elaboración del sistema de conversión de datos
Realización de la conversión de datos

Productos Parciales

Plan de conversión de datos
Diseño de conversión de datos
Procedimientos de reconciliación de la conversión de datos
Procedimientos de conversión de datos
Programa de conversión de datos

Manual de la fase completa

Estrategia de conversión de datos

Plan de conversión de datos
Programas y métodos comprobados de conversión de datos
Procedimientos de conversión de datos
Datos convertidos

IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA

Objetivo

Asegurar que la transferencia del entorno de prueba al entorno de producción del sistema de desempeño de tarjetas de crédito se realice de forma controlada y segura.

Confirmar con los usuarios del sistema que el sistema entregado funciona de acuerdo con los requisitos del sistema indicados .

Entradas

Programas comprobados mediante el sistema
Programas comprobados de conversión de datos
Procedimientos de conversión de datos
Procedimientos de usuario
Documentación del sistema
Documentación del programa
Criterios de aceptación del sistema

Tareas

Criterios de aceptación del sistema y estrategia de comprobación
Preparación de instrucciones de explotación del sistema
Establecimiento del entorno de operaciones de producción
Realización de pruebas de aceptación
Implantación del sistema comprobado

Productos Parciales

Cierre de aceptación
Instrucciones de explotación del sistema

Manual de la fase completa

Un sistema ejecutable y documentado
Cierre de transferencia del sistema

Evaluación de la implantación.

A continuación se describe en forma puntual el alcance y la profundidad del esfuerzo de implantación requerido. Se agrupan las actividades de acuerdo a las áreas específicas que conforman los puntos donde actúa el sistema.

La orientación estuvo dirigida a:

- Realizar entrevistas de evaluación del comportamiento del sistema con el personal.

- Integrar información y controles.

- Analizar y calificar la información de acuerdo a las necesidades actuales del banco.

Evaluación de la efectividad

También se muestra la sumarización de la evaluación de la efectividad, así como gráficas que soportan los resultados de acuerdo a las necesidades de la institución así como el grado de satisfacción .

Interpretación de la evaluación

De acuerdo con las gráficas obtenidas, para cumplir con los objetivos y estrategias de la institución, se resumen los resultados dentro del siguiente contexto:

Estructura organizacional

- La gráfica muestra que anteriormente se estaba enfocado al soporte operativo y que actualmente es básico, lo que hace más eficientes las funciones con el nuevo sistema.
- Se observa que la orientación actual va hacia el servicio estratégico.

Recursos Humanos

- En las gráficas se muestra que con una herramienta automatizada el esfuerzo del personal se orienta a actividades menos operativas.

- Se observa además que el grado de motivación desde los niveles operativos hasta los niveles directivos crece.

- Lo que sí fue necesario aumentar la capacitación del personal con respecto a la utilización del sistema.

Tecnología Hardware/Software

- La gráfica muestra que era necesario contar con una herramienta que diera el apoyo suficiente para las labores de auditoría y que previniera las actuales y futuras demandas de manejo de volúmenes de información.

- En lo referente al software se cuenta con el personal adecuado para dar soporte al sistema de desempeño de tarjetas de crédito ya que el software utilizado FOXPRO tiene consideración de uso institucional. (Figuras 3.4.1 y 3.4.2)

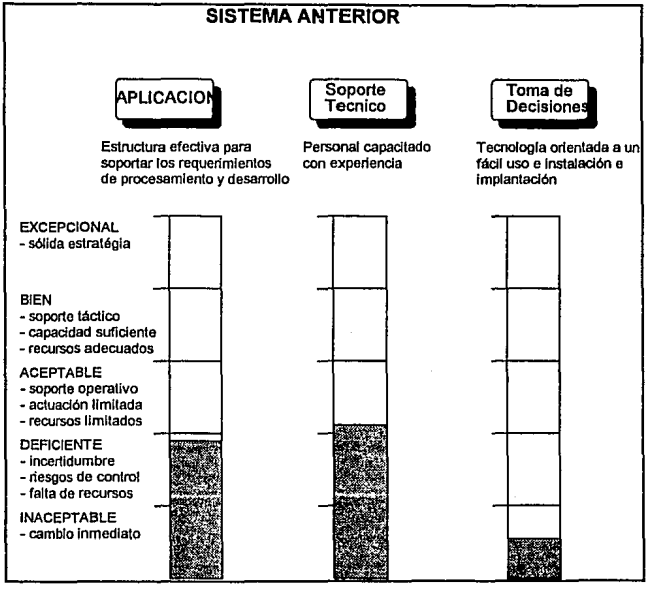


Figura 3.4.1 indicadores de rendimiento del sistema anterior

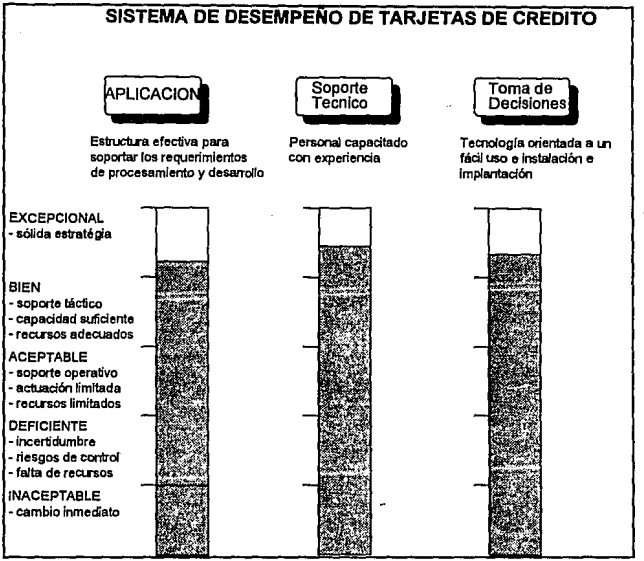


Figura 3.4.2 Indicadores de rendimiento del sistema de desempeño de tarjetas de crédito

Aplicación (Sistema de Desempeño de Tarjetas de Crédito)

- En las gráficas se observa que los procesos administrativos manuales que se llevaban a cabo no eran los adecuados para cumplir con los objetivos de la institución y que con el Sistema de Desempeño de Tarjetas de Crédito se corrige esta situación.

- Se observa que con la estrategia manual anterior, se solicitaba mucho la intervención del área de sistemas para generar información a manera de listados lo que representaba labores tediosas para sistemas y poca confiabilidad y flexibilidad para el usuario.

Soporte a toma de decisiones

- La representación gráfica indica que la información para tomar decisiones generada anteriormente era muy laboriosa, muy tardada y se concentraba en ofrecer resultados inapropiados, sin embargo con el Sistema de Desempeño de Tarjetas de Crédito no sólo se corrige lo anterior sino que aparte de ofrecer una herramienta auxiliar en la toma de decisiones se pueden predecir comportamientos según las tendencias que se obtienen con el mismo.

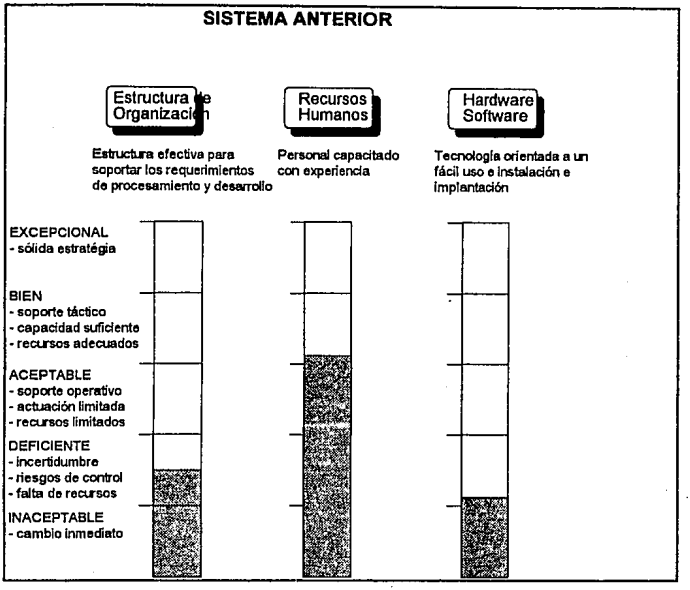


Figura 3.4.3 Indicadores de rendimiento Sistema anterior

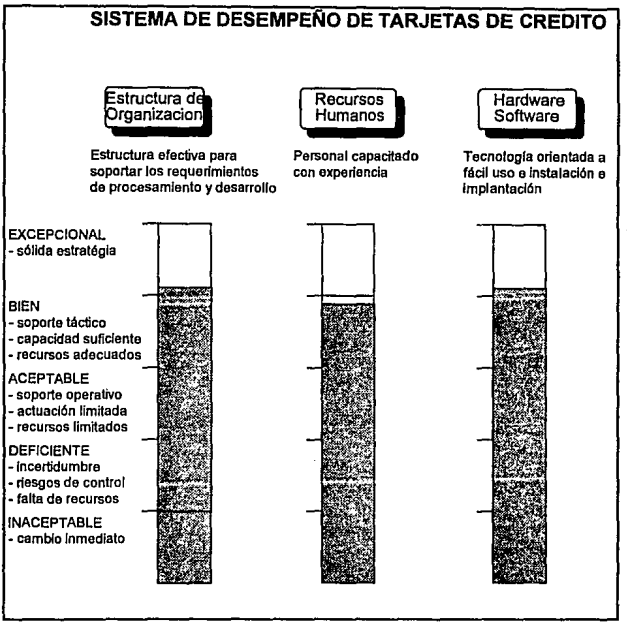


Figura 3.4.4 Indicadores de rendimiento sistema de desempeño de tarjetas de crédito

Resumen de los indicadores de efectividad

La calificación que refleja cada gráfica se determinó a partir de la revisión de los siguientes aspectos:

- Controles e informes revisados
- Forma en que se realizaban y forma en que se realizan actualmente con el sistema las nuevas funciones
- Experiencia y perfil del personal
- Cumplimiento de los objetivos respectivos
- Seguimiento a problemas resueltos
- Tiempo invertido en la obtención de resultados antes y después del sistema automatizado

Los puntos mencionados se revisaron de acuerdo a los criterios del Banco y el grado de satisfacción es el que determinó la calificación.

CONCLUSIONES

La cantidad de soluciones que pueden encontrarse para dominar un problema es muy grande, sin embargo, si aplicamos una metodología, contamos con la mejor ayuda para descartar la mayoría de ellas y enfocar nuestra atención hacia las más factibles que cumplan con los criterios iniciales, para finalmente en base a las fronteras que nos limitan y un juicio pertinente, llegar a la solución más adecuada.

Es importantísimo entender desde un principio que el éxito de cualquier proyecto está fincado sobre el trabajo multidisciplinario en el que hay que interactuar de modo que la participación de todos los involucrados, es fundamental para alcanzar el objetivo. Es digno de mencionar también que la planeación del proyecto es un factor de éxito, es decir, para lograr nuestro propósito tenemos que fijarnos metas a cumplir en el tiempo calculado, de lo contrario perderemos el control de lo que estamos haciendo.

Actualmente no se puede decir que se es experto en el terreno de la computación hablando en términos generales, es necesario considerar que ésta área de la actividad humana es lo suficientemente grande como para asegurar que se puede ser especialista en alguna rama de la misma. La evolución tecnológica de nuestro tiempo camina aceleradamente, por un lado vemos que nuevos productos de Software surgen. Tales productos cada vez se optimizan y en la medida que pueden manejarse con mayor facilidad son más poderosos, pero por su interconectividad con otros

productos y sus requerimientos de Hardware, requieren de personas con mayor grado de preparación y sensibilidad. Por otro lado el Hardware que avanza todavía más aceleradamente que el Software, hace que los profesionales de la computación tengan que ponerse al día en los equipos que las compañías desarrollan. Lo podemos entender como modas que perecen y renacen conforme los investigadores sugieren nuevas soluciones de Software y Hardware, por lo tanto, sabemos que la actualización constante de conocimientos de nuevos productos es vital para estar de acuerdo con los tiempos.

La realización de este trabajo nos dio la oportunidad de aplicar los conocimientos adquiridos a lo largo de la "carrera", nos dio el indicador de dónde nos encontramos y nos hizo ver que es necesario trabajar en equipo para lograr lo que queremos de la mejor forma. Nos ayudó a identificar las mejores alternativas que existen para solucionar un problema de índole profesional y entender que la "carrera" nos dio las herramientas necesarias a usar para salir adelante con excelencia. Nos hizo ver que el camino es muy largo de recorrer y que debemos actualizarnos constantemente en lo nuevo que vaya saliendo.

Con lo expuesto, lo único que podemos hacer es dar las gracias a la Universidad por todo lo que nos ha dado.

BIBLIOGRAFÍA

A. Alatou
Teleinformática y Redes de Computadoras
Mundo Electrónico
Segunda Edición
1991 Alfaomega/Marcombo

Axcess
Introduccion and Communications Utility Program (CUP)
NonStop System
1983 TANDEM COMPUTERS

Computerworld
Año 14 Num. 395
PC Pentium Cliente/Servidor
marzo 7, 1994

Frank & Defler
Guide to Connectivity
Forward by Jim Seymour
1991 PC MAGAZINE

Gio Wiederrhold
Diseño de Bases de Datos
Data Base Design
Traducción de la 2a edición en Inglés
Segunda edición (Primera edición en Español)
1985 Mac Graw Hill International Book Co. U.S.A.

Howard Dickler, Ph. D.
Programmer's Guide to FoxPro 2.5
1993 Sybex

Imagen
Vol. 10 Num. 1
Revista interna Banamex
Enero-Febrero 1993

James A. Senn
Análisis y Diseño de Sistemas de Información
Segunda edición
1992 Mac Graw Hill

L. Morgan, Waite
Introducción al Microprocesador 8086/8088 (16 bit)
1984 Mac Graw Hill

Manual de Auditoria Interna
División de Auditoría
1992 Banamex

Manual de Calidad Administrativa a través de los Sistemas de Control Interno
1991 Price Waterhouse

Manual de Tarjetas Banamex
600. Tarjetas Banamex

Microsoft, Co.
Fox Pro 2.5, Developer's Guide
1993

Orilia
Introducción al Procesamiento de Datos Para los Negocios
Segunda Edición (Primera Edición en Español)
1983 Mac Graw Hill

Pc Magazine
Editor Choice
Mayo 11, 1993

Pc Magazine
Vol.4 Num. 10
La Pc Perfecta
Evaluacion de máquinas Pentium
Octubre 1993

Pc Magazine
Vol. 13 Num. 7
Pentium and DX4 PC
April 12, 1994

Pc Semanal
Vol. 4 Num. 96
Pentium a 90 y 100 MHz
Marzo 14, 1994

Price Waterhouse
Metodología de Gestión de sistemas
Desarrollo de Sistemas
Resumen y Módulo Base de la Implantación de sistemas
1989 Price Waterhouse

Price Waterhouse
System Management Metodology
Strategic Information System Planning (SISP)
Overview/Baseline
1992 Price Waterhouse

Price Waterhouse
Strategic Information System Planning (SISP)
Philosophy and Thecniques
Workbook Version 3
1993 Price Waterhouse

SNAX
Programing and Network Management
NonStop System
1983 TANDEM COMPUTERS

Software Digest
Vol. 10 Num. 7
Ratings Reports, Multiuser Databases for Windows
Julio 1993

Tom Sheldon
Novell Netware 386
Manual de Referencia
Novell Netware 386: *The Complete Reference*
Traducido de la Primera Edición en Inglés
1992 Mac Graw Hill Interamericana de España S.A.

Yourdon/Constantine
Structured D Fundamentals of a Discipline of Computer Program and System Design
1979 Prentice-Hall, Inc

APENDICE A

GLOSARIO

ANSI

(American National Standards Institute)

Instituto Americano de Normas Nacionales.

Organización de afiliados privados sin fines de lucro, fundada en 1918, que coordina el desarrollode normas nacionales voluntarias en Estados Unidos tanto en el sector privado como en el público. Es el miembro internacional de Estados Unidos en Internacional Standards Organization (ISO) (Organización Internacional de Normas) y en la Internacional Electrotechnical Commission (IEC) (Comisión Internacional Electrotécnica).

ARCNET

(Attached Resource Computer NETwork)

Red de Recursos de Computadoras Unidas

Red de área local desarrollada por Datapoint Corporation que interconecta una amplia variedad de computadoras personal y estaciones de trabajo vía cable coaxil, par trenzado o cable de fibras ópticas. Utiliza el método de acceso de pasaje de símbolos a 2'5 Mbps con tipología de estrella distribuida que interconecta hasta 255 computadoras. Las versiones de 20 Mbps fueron introducidas en 1989.

ASCII

(American Standard Code for Information Interchange)

Código Americano Estandard para Intercambio de Información

Pronunciado en inglés "ask-ee". Código binario de datos que se usa en comunicaciones, en la mayor parte de las minicomputadoras y en todas las computadoras personales. ASCII es un código de 7 bits que permite

ASCII file

archivo ASCII

Archivo de datos o de texto que contiene caracteres codificados en ASCII. Los archivos de texto, documentos de procesador de palabras, archivos de lotes y programas fuentes son generalmente archivos ASCII. Sólo los primeros 128 caracteres (0 - 127) dentro de las 256 combinaciones de un byte conforman el estándar ASCII. El resto son utilizados en forma diferente dependiendo del computador.

ASHTON-TATE

Compañía de software fundada en 1980 por Hal Lashlee y George Tate para comercializar dBASE II. Borland la adquirió en 1991.

AT bus

bus o colector tipo AT

Bus de datos de 16 bits usado en la AT. Véase PC bus. Nótese la diferencia con Micro Channel.

attribute

atributo

(1) En administración de base de datos relacionales, campo dentro de un registro.

(2) Para impresoras y pantallas, una característica que cambia la tipografía; por ejemplo, de normal a negrita o a subrayado, o de normal a video inverso.

ATM

(Automatic Teller Machine)

máquina de cajero automático

Terminal bancaria para propósitos especiales que permite a los usuarios hacer depósitos y extracciones. Puede ser una unidad independiente o conectada en línea a un sistema informático central. Los ATM son activados insertando una tarjeta magnética (tarjeta de efectivo o tarjeta de crédito) en la máquina que contiene el número de identificación del usuario.

AUDIT TRAIL

pista de auditoría, intervención de seguimiento

Registro de transacciones en un sistema de información que provee verificación de la actividad del sistema. La pista de auditoría más simple es la transacción misma. Si el salario de una persona es aumentado, la transacción de cambio incluye la fecha, el importe del aumento y el nombre del gerente que la autoriza.

bit

(BInary digiT)

dígito binario

Un dígito simple de un número binario (1 o 0). En el computador, un bit es físicamente una celda de memoria (constituida por transistores o un transistor y un condensador), un punto magnético en un disco o una cinta, o un pulso de alto o bajo voltaje viajando a través de un circuito.

Borland

(Borland International, Inc.) Una compañía líder en software para microcomputadoras. Fundada en 1983 por Philippe Kahn, Borland introdujo el Turbo Pascal y sacó el

lenguaje Pascal fuera de los ámbitos académicos para convertirlo en un producto comercial. Asimismo, sus productos Turbo C y Assembler se han convertido en estándares industriales.

buffer

memoria intermedia, tampón, regulador

Una porción reservada de la memoria que se utiliza para almacenar datos mientras son procesados. En un programa, se crean "buffers" para contener algunos datos de cada uno de los archivos que van a ser leídos o grabados. Un "buffer" puede ser también un pequeño banco de memoria usado para fines especiales.

bus

ducto, colector

Un canal o ruta común entre dispositivos del hardware, ya sea internamente entre componentes del computador o externamente entre estaciones de una red de comunicaciones.

byte

octeto, byte

La unidad común de almacenamiento en computación, desde computadoras personales hasta macrocomputadoras. Se compone de ocho dígitos binarios (bits). Puede agregarse un noveno como bit de paridad, para comprobación de errores.

CCITT

(Consultative Committee for International Telephony and Telegraphy)

Comité Consultativo para telefonía y telegrafía Internacionales.

Una organización internacional de normas de comunicaciones. Es uno de los cuatro órganos de la Unión Internacional de Telecomunicaciones, fundada en 1865, con sede central en Ginebra y compuesta por más de 150 países miembros.

check box

caja de comprobación

Método de mostrar el estado actual de la opción seleccionable visualizando una x o un símbolo de marca de comprobación de una caja pequeña al lado de la opción una vez seleccionada.

client/server

cliente/servidor

(1) En una red de comunicaciones, el cliente es la máquina solicitante y el servidor es la máquina proveedora. Esto implica que existe un software especializado en ambos extremos. Por ejemplo, en un sistema de base de datos para trabajar en red, la interfaz de usuario reside en la estación de trabajo y las funciones de almacenamiento y recuperación residen en el servidor.

(2) Relación petición/suministro entre programas. Se pueden diseñar aplicaciones, ejecutandolas dentro de la misma computadora o en muchas computadoras, en la que un programa (cliente) pide datos al otro programa (el servidor). por ejemplo, en el sistema X de Window, el servidor es el programa que dirige la visualización en la pantalla, y el cliente es la aplicación que pide al servidor visualizar algo.

compiler

compilador

Software que traduce lenguajes de programación de alto nivel, como COBOL y C, en lenguaje máquina. Un compilador habitualmente genera en primer lugar lenguaje ensamblador y a continuación traduce este lenguaje a lenguaje máquina.

CODASYL

(COnference on DAta SYstems Languages)

Conferencia sobre Lenguajes de Sistemas de Datos Una organización dedicada al desarrollo de lenguajes informáticos. Fundada en 1959, está compuesta por individuos e instalaciones que contribuyen con su propio tiempo y esfuerzo. El COBOL es un producto de CODASYL.

data dictionary

diccionario de datos

Base de datos acerca de datos y bases de datos. Contiene el nombre, tipo, rango de valores, fuente y autorización para el acceso a cada elemento de datos en los archivos y bases de datos de la organización. Indica asimismo qué programas de aplicación utilizan dichos datos de tal manera que cuando se contempla un cambio en una estructura de datos, puede generarse una lista de los programas afectados.

data structure

El diseño físico de los datos. Campos de datos, campos memo, campos de longitud fija, campos de longitud variable, registros, documentos de procesamiento de palabra, hojas de cálculo, archivo de datos, archivos e índices de bases de datos son todos ejemplos de estructuras de datos.

database

base de datos

(1) Un conjunto de archivos interrelacionados que es creado y manejado por un sistema de gestión o de administración de bases de datos (DBMS).

(2) Cualquier conjunto de datos almacenado electrónicamente.

database administrator

administrador de base de datos

Una persona responsable del diseño físico y manejo de la base de datos y de la evaluación, selección e implementación del sistema de administración de la base de datos.

database management system

sistema de administración o gestión de bases de datos

Sotware que controla la organización, almacenamiento, recuperación, seguridad e integridad de los datos en una base de datos. Acepta pedidos de datos desde un programa de aplicación y le ordena al sistema operativo transferir los datos apropiados.

deadlock

punto muerto o estacionamiento, bloqueo

DES

(Data Encryption Standard)

estándar de cifrado de datos

Una técnica de cifrado estándar de NIST que embrolla los datos en un código impenetrable para la transmisión en una red pública. Usa un número binario como

clave de cifrado que ofrece más de 72.000.000.000.000.000 de combinaciones. El número, que puede ser elegido al azar para cada transmisión, es usado como un patrón para convertir los bits en ambos extremos de la transmisión.

distributed database

base de datos distribuida

Una base de datos que está físicamente almacenada en dos o más sistemas informáticos. Aunque está geográficamente dispersa, un sistema de base de datos distribuida administra y controla la base de datos completa como una única colección de datos. Si se almacenan datos redundantes en bases de datos separadas debido a requerimientos de rendimiento, las actualizaciones de un conjunto de datos actualizan automáticamente los conjuntos adicionales en el momento oportuno. Una base de datos distribuida implica que la redundancia será administrada y controlada.

DOS

(Disk Operating System)

sistema operativo en disco

(1) La denominación genérica de un sistema operativo.

(2) Un sistema operativo monousuario para las series PC, PS/1 y PS/2 de IBM. Desarrollado por Microsoft, el desarrollo del DOS es compartido con IBM de tanto en tanto. El DOS es a veces denominado PC-DOS, para diferenciarlo del MS-DOS, la versión de Microsoft para las PC no IBM. Ambos productos son prácticamente idénticos y a ambos se los llama DOS.

EISA

(Extended Industry Standard Architecture)

arquitectura estándar industrial extendida

Estándar de bus para PC que extiende la arquitectura del bus de la AT a 32 bits y permite a más de una CPU compartir el bus. EISA fue anunciado a fines de 1988 para competir con Micro Channel de IBM. Las tarjetas de PC y AT existentes, que no pueden enchufarse en el Micro Channel, sí pueden hacerlo en una ranura EISA.

encryption

cifrado, criptografiado, criptografícación

Codificación de datos con propósito de seguridad, convirtiendo el código de datos estándar en un código propio. Los datos cifrados deben de codificarse para ser usados. El cifrado se usa para transmitir documentos por una red, o para codificar texto de modo tal que no pueda ser modificado con un procesador de texto. Véase DES.

entity

entidad

En administración de bases de datos, un registro.

entity relationship model

modelo entidad relación

En una base de datos, una modelo de datos que describe atributos de entidades y relaciones entre ellos.

field

campo

Unidad física de datos que ocupa uno o más bytes. Una colección de campos forma un registro. Un campo también define una unidad de datos en un documento fuente, pantalla o informe. Ejemplos de campos son nombre, dirección, cantidad e importe adeudo.

file

archivo

(1) En administración de datos, una colección de registros relacionados.

(2) En procesamiento de textos, un único documento de texto.

(3) En gráficos por computadora, un conjunto de descriptores de imágenes para una figura, tanto en formato de TV (gráficos de trama) como en formato de líneas o de objetos (gráficos vectoriales).

(4) En programación, el programa fuente y el programa en lenguaje de máquina son almacenados como archivos individuales.

(5) En operaciones de computadora, cualquier colección de datos que es tratada como una sola unidad en un dispositivo periférico.

FoxPro

DBMS compatible con dBASE IV para PCs de Microsoft.

Como una versión mejorada de FoxBASE, FoxPro incluye interfaces de ventanas, SQL y QBE y technología "Rushmore" para consultas rápidas en bases de datos grandes.

Gigabit

gigabit

Un millón de bits. También Gb, Gbit y G-bit. Véase giga y space/time.

hardware

Toda la maquinaria y el equipamiento. Contrástese con software, el cual es un conjunto de instrucciones que le dicen a la computadora qué hacer. También contrástese con data, que son los hechos y cifras que se almacenan en el hardware y son controlados por el software.

host

anfitrión

La computadora central o la computadora controladora en un entorno de procesamiento en tiempo compartido o distribuido.

interactive

interactivo

Un diálogo bilateral entre el usuario y una computadora.

ISA

(Industry Standard Architecture)

arquitectura industrial estándar

Los buses de 8 bits (PC, XT), y de 16 bits (AT) de las primeras series de computadoras personales de IBM. El EISA es una extensión de 32 bits del ISA. Nótese la diferencia con el MicroChannel.

kilobit

kilobit

1.000 o 1024. También se escribe KB, Kb, Kbit o K-bit.

LAN

(1) (Local Area Network) (Red de Area Local) Red de comunicaciones que sirve a usuarios dentro de un área geográficamente limitada.

(2) (LAocal Area Network) (Red de Area Local) Red de computadoras personales dentro de un área geográficamente confinado que se compone de servidores, estaciones de trabajo, sistemas operativos de redes y un enlace de comunicaciones.

login

entrada de identificación, conexión Igual que logon.

mainframe

macrocomputadora

Una computadora grande. A mediados de los años 60, las épocas antiguas de las computadoras, todas las computadoras eran mainframes (literalmente "bastidor principal"), ya que el término se refería al gabinete que contenía la CPU. Aunque mainframe aún significa gabinete principal, usualmente se refiere a un gran sistema de computación y toda la experiencia asociada que va con él.

MAN

(Metropolitan Area Network)

Red de Area Metropolitana

Red de comunicaciones que abarca un área geográfica como una ciudad o un suburbio. Véase LAN y WAN.

memoria auxiliar

Memoria de alta velocidad, utilizada en grandes macrocomputadoras y supercomputadoras. No es directamente direccionable por CPU, y funciona en forma semejante a un disco. Los datos son transferidos desde la memoria auxiliar a la memoria principal por un canal de ancho de banda amplio.

menu

menú

Lista de opciones y comandos disponibles mostrada en la pantalla en un programa interactivo. La selección de una opción del menú se efectúa introduciendo el número o letra que tiene asignada, presionando la tecla de la primera letra de la palabra o resaltando la opción y presionando la tecla "return" o el botón del ratón. Véase Lotus menu y pull-down menu.

Micro Channel

Bus de 32 bits usado en los modelos más avanzados de la serie PS/2 de IBM, la serie RS/6000 y ciertos modelos de 9370. Está diseñado para multiprocesamiento, lo que permite que dos o más CPUs trabajen en paralelo dentro de la computadora al mismo tiempo. Las placas Micro Channel no son intercambiables con las placas de bus de las PC.

micron

micrón

Una millonésima de metro. Aproximadamente 1/25.000 de pulgada. Los pequeños elementos que conforman un transistor en un chip se miden en micrones. Las medidas por debajo del nivel del micrón se hacen en Angstroms; 10.000 Angstroms equivalen a un micrón.

Microsoft

(Microsoft Corporation)

Una compañía de software para microcomputadoras, de primer nivel, fundada en 1975 por Paul Allen y Bill Gates, dos estudiantes universitarios que escribieron el primer intérprete BASIC para el microprocesador 8080 de Intel. Le concedieron licencia a Micro Instrumentation and Telemetry Systems para acompañar el equipo para armar su microcomputadora Altair 8800. A fines de 1976, más de 10000 Altairs fueron vendidas con el BASIC de Microsoft.

multiprocessing

multiprocesamiento

El procesamiento simultáneo con dos o más procesadores en una computadora, o dos o más computadoras que están procesando juntas. Cuando se usan dos o más computadoras, se ligan con un canal de alta velocidad y comparten la carga de trabajo general entre ellas. En caso de que una deje de operar, la otra se hace cargo. En sistemas con tolerancia de fallos, dos o más procesadores se construyen dentro de un mismo gabiente.

multitasking

multitarea

La ejecución de dos o más programas en una computadora al mismo tiempo. La multitarea se controla por el sistema operativo, que carga los programas y los maneja hasta que terminen. El número de programas que pueden ser efectivamente ejecutados depende de la cantidad de memoria disponible, la velocidad de CPU, capacidad y velocidad de los recursos periféricos, así como también de la eficiencia del sistema operativo.

multiuser DOS

DOS multiusuario

(1) Sistema operativo compatible con DOS que permite que muchos terminales no inteligentes se ejecuten desde un simple PC.

(2) DOS multiusuario. Sistema operativo DOS multiusuario de Digital Research, Inc., que permite enlazarse a una CPU 386SX o a máquinas más altas hasta 10 terminales y/o PCs. Reemplaza al Concurrent DOS.

multiuser

multiusuario

Una computadora que es compartida por dos o más usuarios.

normalization

normalización

En gestión de bases de datos relacionales, un proceso que divide los datos en grupos de registros para un procesamiento eficiente. Existen seis etapas. Hacia la tercera etapa (la tercera forma normal), los datos se identifican solamente por el campo clave

del registro. Por ejemplo, la información de ordenación se identifica por el número de orden, información del cliente o por número de cliente.

NetWare

Una familia de sistemas operativos de redes de Novell, Inc., Provo, UT, que se ejecuta en PCs 286 y superiores y soporta DOS, OS/2 y estaciones de trabajo Mac, y una variedad de métodos de acceso de LANs, incluyendo Token Ring, Ethernet y ARCNET. Es el programa de control de redes más usado.

operating system

sistema operativo

Un programa maestro de control que maneja la computadora y actúa como planificador y agente de tránsito. Es el primer programa que se carga (copia) en la memoria de la computadora después de que ésta sea encendida, y el núcleo central ("kernel") del mismo debe estar siempre residente en memoria. El sistema operativo puede ser desarrollado por el fabricante del hardware en el que corre o por una casa independiente de software.

Paradox

Un sistema de administración de bases de datos relacionales para PC, listo para redes, de Borland International. Tiene un único lenguaje de programación, llamado PAL, para el desarrollo de aplicaciones. Muchas instrucciones de PAL son comandos interactivos Paradox, y por esto un usuario de Paradox puede ajustarse a la programación más fácilmente. Paradox es conocido por su facilidad de uso y consulta utilizando un ejemplo, lo cual permite al usuario hacer preguntas complicadas fácilmente.

password

contraseña, palabra de paso

Palabra o código utilizado para identificar a un usuario autorizado; es normalmente provisto por el sistema operativo o DBMS.

pipeline processing

procesamiento por entubamiento, por tubería, por canalización Una categoría de técnicas que proveen procesamiento simultáneo, o paralelo, dentro de la computadora. Esto se refiere al solapamiento de operaciones, llevando los datos o instrucciones a un tubo conceptual, con todas las etapas del tubo en procesamiento simultáneo. Por ejemplo, mientras una instrucción está siendo ejecutada, la computadora está decodificando la próxima instrucción. En procesos vectoriales, varios pasos de una operación de coma flotante pueden ser procesados simultáneamente.

Pentium

CPU de próxima generación de Intel que aparecerá en 1993. De compatibilidad ascendente con el 486, es una CPU de 64 bits con juegos de instrucciones gemelas y memorias caché. Puede contener hasta 3 millones de transistores.

pointer

puntero, apuntador

(1) En gestión de base de datos, una dirección empotrada (incluida) dentro de los datos que especifica la posición de los datos en otro registro o archivo.

(2) En programación, una variable que se utiliza como una referencial al elemento actual de una tabla (matriz o array) o a algún otro objeto, como la fila actual o columna en la pantalla.

(3) Símbolo en pantalla utilizado para identificar las selecciones del menú o la posición actual en la pantalla. Se mueve mediante un ratón u otro dispositivo de tipo apuntador o señalador.

query

consulta

Una interrogación a una base de datos que permite al usuario contar, sumar y listar registros seleccionados contenidos en ella. Nótese la diferencia con report (reporte, informe) que es generalmente una salida impresa más elaborada con encabezamientos y números de página. El informe puede ser también una lista selectiva de elementos; por lo tanto, los dos términos pueden referirse a programas que producen los mismos resultados.

record

registro

Grupo de campos relacionados que se usan para almacenar datos acerca de un tema (registro maestro) o actividad (registro de transacción). Una colección de registros constituye un archivo. Los registros maestros contienen datos permanentes, tales como número de cuenta, y datos variables, tales como saldo adeudado. Los registros de transacciones contienen sólo datos permanentes, tales como cantidad y código de producto.

RISC

(Reduced Instruction Set Computer)

computadora de conjunto de instrucciones reducido

Arquitectura de computadoras que ejecuta un número limitado de instrucciones. El concepto es que la mayoría de los programas usan generalmente unas pocas

instrucciones, y si se acelera la ejecución de esas instrucciones básicas, se mejora el rendimiento.

La arquitectura RISC elimina una capa de carga operativa llamada "microcódigo", que se emplea normalmente para facilitar la agregación de nuevas y complejas instrucciones a una computadora. Las computadoras RISC poseen un pequeño número de instrucciones montadas en los circuitos de nivel inferior, que trabajan a máxima velocidad.

ROM

(Read Only Memory)

Memoria de Solo Lectura

Chip de memoria que almacena permanentemente instrucciones y datos. Sus contenidos se crean en el momento de la fabricación y no se pueden alterar. Se utiliza ampliamente para almacenar rutinas de control en computadoras personales (ROM BIOS) y en controladores de periféricos, también se utiliza en cartuchos conectables para impresoras, video juegos y otros sistemas. Cuando el software se almacena en ROM la actualización a la versión siguiente requiere volver a colocar el chip de la ROM.

server

servidor

En una red, computadora que es compartida por múltiples usuarios. batch proccesing procesamiento por lote o por tanda El procesamiento de un grupo de transacciones de una sola vez. Las transacciones se reúnen y se procesan frente a los archivos maestros (con actualización de los archivos maestros) al final del día o en algún otro período de tiempo.

software

Instrucciones para una computadora. Una serie de instrucciones que realizan una tarea en particular se llama programa o programa de software. Las dos categorías principales son software de sistemas de aplicaciones. El software de sistemas se compone de programas de control, incluyendo el sistema operativo, software de comunicaciones y administrador de bases de datos. El software de aplicaciones es cualquier programa que procesa datos para el usuario (inventario, nómina, hoja de cálculo, procesador de texto, etc.).

SQL

(Structured Query Language)

Lenguaje de Consulta Estructurado

Lenguaje utilizado para interrogar y procesar datos en una base de datos relacional. Desarrollado originalmente por IBM para sus macrocomputadoras, han habido muchas implementaciones creadas para aplicaciones de base de datos en mini y microcomputadoras. Las órdenes (mandatos) de SQL se pueden utilizar para trabajar interactivamente con una base de datos, o pueden incluirse en un lenguaje de programación para servir de interfaz a una base de datos.

structured analysis

análisis estructurado

Técnicas desarrolladas a finales de los años 70 por Yourdon, DeMarco, Gane y Sarson para aplicar un método sistemático al análisis de sistemas. Incluía la utilización de diagramas de flujo de datos y modelo de datos y fomentaba la utilización de notas gráficas independientes de implementación para la documentación.

Tandem

(Tandem Computers Inc.) Fabricante de computadoras tolerantes de fallos. Fue fundada en 1974, para dedicarse al mercado del procesamiento de transacciones en línea (online transaction processing - OLTP), el cual estaba volviéndose dependiente de las computadoras para inventarios, reservas y transferencias financieras. En 1976, Tandem introdujo la primera computadora comercial basada en una arquitectura tolerante de fallos.

La serie NonStop de Tandem está construida en torno a una estructura de procesadores paralelos múltiples, en lugar de procesadores redundantes. Todos los procesadores se emplean para procesar datos, pero si uno falla, el sistema es capaz de distribuir la carga de trabajo a los procesadores restantes. Su arquitectura permite la expansión, aun estando en funcionamiento la computadora. La mayoría de los sistemas se pueden expandir hasta 16 procesadores, o incluso hasta 224 utilizando un enlace de fibra óptica. Mediante su software de red, pueden unirse hasta 4080 procesadores. James G. Treybig es el principal fundador de Tandem, y ha sido presidente de la compañía desde su formación.

token ring network

red de anillo de señales

Red de comunicaciones que emplea la tecnología de paso de señales en forma secuencial. Cada estación de la red recibe la señal y la pasa a la estación ubicada a continuación.

virtual memory

memoria virtual

Una técnica que simula más memoria que la que realmente existe y permita a la computadora ejecutar varios programas simultáneamente, sin importar su tamaño.

WAN

(Wide Area Network)

Red de Area Ancha

Red de comunicaciones que abarca áreas geográficas amplias, como pueden ser estados y países. Véase MAN y LAN.

window

ventana

Area de visión separada en una pantalla de presentación, provista por software. Los sistemas operativos pueden proveer ventanas múltiples en la pantalla, permitiendo al usuario mantener varios programas de aplicación activos y visibles al mismo tiempo.

Windows

Entorno operativo para gráficos de Microsoft que se integra con DOS. Proporciona un entorno de sobremesa similar al Macintosh, en el cual cada aplicación activa se visualiza en una pantalla movible y redimensionable sobre la pantalla.

workstation

estación de trabajo

(1) Micro o minicomputadora para un único usuario, de alto rendimiento, que ha sido especializada para gráficos, diseño asistido por computadora, ingeniería asistida por computadora o aplicaciones científicas.

(2) En una red de área local, una computadora personal que sirve a un único usuario, a diferencia de un servidor de archivos, que sirve a todos los usuarios de la red.

(3) Cualquier terminal o computadora personal.

80486

Comúnmente conocido como el i486 o 486, es un microprocesador multitarea de 32 bits de Intel Corporation, que contiene un coprocesador matemático incluido. Contiene 1.200.000 transistores y funciona a velocidades más altas que el 80386.

APENDICE B

CODIGO FUENTE

DEL SDTAR

```
*  ************************************************************
*  *
*  * 11/04/94          TMENU.MPR          00:23:04
*  *
*  ************************************************************
*   System: Sistema de Desempeño de Tarjetas
*  *
*  * David Jiménez & Adolfo Avila
*  *
*  * Copyright (c) 1994 Banamex - Accival
*  * Insurgentes Sur No. 1863, 2o Piso
*  * City,    Zip
*  *

*  ************************************************************
*  *
*  *                    Setup Code
*  *
*  ************************************************************
```

** Inicializacion de ambiente

```
RELEASE WINDOW
CLEAR READ
CLEAR ALL
CLEAR
MODIFY WINDOW SCREEN TITLE "Sistema de Desempeño de Tarjetas Bancarias"

SET TALK OFF
SET BELL OFF
SET ECHO OFF
SET ESCAPE OFF
SET NOTIFY OFF
SET CONFIRM OFF
SET EXCLUSIVE OFF
SET SAFETY OFF
SET CENTURY ON
SET DELETED ON
SET DATE BRITISH
SET MESSAGE TO " "
SET CLOCK STATUS
```

** Declaración de variables globales

```
PUBLIC drive,pathprg, oldpath, dia_en_curso,ul12meses
PUBLIC offset, privilegios

PUBLIC selecciona, tctas, tsdoact, tsdovdo, login, der, exito
PUBLIC dbfuente, dbdestino, namereport, est_cons, m.done

PUBLIC ARRAY  offset[26,8], arrul12meses[12], arrproducto[1], arrlinea[1], arrdivision[1], arrsucursal[1]
```

** Inicialización de variables

```
dia_en_curso = DATE()
m.done = .F.
login = ""
der   = ""
exito = .F.
```

```
DO media_prod
DO media_sob
ul12meses=""
DO arreglo_meses
DO copia_nombres
DO fill_table IN segurid

s_def = SET("DEFAULT")
s_pat = SET("PATH")

** Pantalla de Presentación del Sistema

SET SYSMENU OFF
DO scrini.spr
CLOSE ALL

SELECT 1
USE tarsdos ALIAS saldos

SELECT 2
USE taraboga ALIAS abogados

SELECT 3
USE taraltas ALIAS altas

SELECT 4
USE tarbolet ALIAS boletin

SELECT 5
USE tarcasti ALIAS castigos

SELECT 6
USE tarestru ALIAS estructura

SELECT 7
USE tarinlim ALIAS incrementos

SELECT 8
USE tarjurid ALIAS juridico

SELECT 9
USE tarnomes ALIAS nomestru

SELECT 10
USE taropera ALIAS operador

SELECT 11
USE tarprod ALIAS producto

SELECT 12
USE tarptos ALIAS puestos

SELECT 13
USE tbitacor ALIAS bitacora

SELECT 14
USE tusuario ALIAS usuario
```

```
* Verificación de la Seguridad
* Pide login y password para establecer derechos

DO scracc.spr
IF !exito
  m.done=.T.
  CLOSE DATA
  CLEAR READ ALL
  CLOSE ALL
  CLEAR READ
  SET SYSMENU TO DEFAULT

  && LAS SIGUIENTES LINEAS DEBEN ESTAR ACTIVAS CUANDO SE GENERE EL EJECUTABLE

  QUIT
  * La variable m.done sirve para hacer un foundation read
  * para mas infor. ver el Users guide del Distribution Kit, pag. 12
ENDIF

*     ************************************************************
*     *
*     *              Cleanup Code & Procedures
*     *
*     ************************************************************
*

** REM ESTAS DOS LINEAS DEBEN ESTAR ACTIVAS SOLO CUANDO SE GENERA EJECUTABLE
** REM FOUNDATION READ
READ VALID m.done
SET MESSAGE TO " "

** Genera un arreglo conteniendo los ultimos 12 meses a partir de la fecha en curso

*!*************************************************************************
*!
*!    Procedure: ARREGLO_MESES
*!
*!    Called by: TMENU.MPR
*!
*!*************************************************************************
PROCEDURE arreglo_meses
PRIVATE i, fecha_anterior, anio, mes

FOR i= 1 TO 12
  fecha_anterior = GOMONTH(dia_en_curso, -i)
  anio = SUBSTR( ALLTRIM(STR(YEAR(fecha_anterior))),3 )
  mes =  ALLTRIM( STR(MONTH(fecha_anterior)) )
  mes = IIF( LEN(mes)=1 , "0"+mes, mes)
  arrul12meses[i] = VAL(anio+mes)
  IF i=1
    ul12meses = anio+mes
  ELSE
    ul12meses = ul12meses + ","+anio+mes
  ENDIF
ENDFOR
RETURN
```

** Genera un arreglo conteniendo el promedio del saldo vencido de cada producto

PROCEDURE media_prod

```
*    ************************************************************
*    *
*    * 11/04/94          TMENU.MPR          00:23:04
*    *
*    ************************************************************
*     System: Sistema de Desempeño de Tarjetas
*    *
*    * David Jiménez & Adolfo Avila
*    *
*    * Copyright (c) 1994 Banamex - Accival
*    * Insurgentes Sur No. 1863, 2o Piso
*    * City,    Zip
*    *

*    ************************************************************
*    *
*    *                  Setup Code
*    *
*    ************************************************************

** Inicializacion de ambiente

RELEASE WINDOW
CLEAR READ
CLEAR ALL
CLEAR
MODIFY WINDOW SCREEN TITLE "Sistema de Desempeño de Tarjetas Bancarias"

SET TALK OFF
SET BELL OFF
SET ECHO OFF
SET ESCAPE OFF
SET NOTIFY OFF
SET CONFIRM OFF
SET EXCLUSIVE OFF
SET SAFETY OFF
SET CENTURY ON
SET DELETED ON
SET DATE BRITISH
SET MESSAGE TO " "
SET CLOCK STATUS


** Declaración de variables globales

PUBLIC drive,pathprg, oldpath, dia_en_curso,ul12meses
PUBLIC offset, privilegios

PUBLIC selecciona, tctas, tsdoact, tsdovdo, login, der, exito
PUBLIC dbfuente, dbdestino, namereport, est_cons, m.done

PUBLIC ARRAY  offset[26,8], arrul12meses[12], arrproducto[1], arrlinea[1], arrdivision[1], arrsucursal[1]


** Inicialización de variables

dia_en_curso = DATE()
m.done = .F.
login = " "
der   = " "
exito = .F.
```

```
DO media_prod
DO media_sob
ul12meses=""
DO arreglo_meses
DO copia_nombres
DO fill_table IN segurid

s_def = SET("DEFAULT")
s_pat = SET("PATH")

** Pantalla de Presentación del Sistema

SET SYSMENU OFF
DO scrini.spr
CLOSE ALL

SELECT 1
USE tarsdos ALIAS saldos

SELECT 2
USE taraboga ALIAS abogados

SELECT 3
USE taraltas ALIAS altas

SELECT 4
USE tarbolet ALIAS boletin

SELECT 5
USE tarcasti ALIAS castigos

SELECT 6
USE tarestru ALIAS estructura

SELECT 7
USE tarinlim ALIAS incrementos

SELECT 8
USE tarjurid ALIAS juridico

SELECT 9
USE tarnomes ALIAS nomestru

SELECT 10
USE taropera ALIAS operador

SELECT 11
USE tarprod ALIAS producto

SELECT 12
USE tarptos ALIAS puestos

SELECT 13
USE tbitacor ALIAS bitacora

SELECT 14
USE tusuario ALIAS usuario
```

```
* Verificación de la Seguridad
* Pide login y password para establecer derechos

DO scracc.spr
IF !exito
  m.done=.T.
  CLOSE DATA
  CLEAR READ ALL
  CLOSE ALL
  CLEAR READ
  SET SYSMENU TO DEFAULT

  && LAS SIGUIENTES LINEAS DEBEN ESTAR ACTIVAS CUANDO SE GENERE EL EJECUTABLE

  QUIT
  * La variable m.done sirve para hacer un foundation read
  * para mas infor. ver el Users guide del Distribution Kit, pag. 12
ENDIF

*     ************************************************************
*     *
*     *              Cleanup Code & Procedures
*     *
*     ************************************************************
*

** REM ESTAS DOS LINEAS DEBEN ESTAR ACTIVAS SOLO CUANDO SE GENERA EJECUTABLE
** REM FOUNDATION READ
READ VALID m.done
SET MESSAGE TO " "

** Genera un arreglo conteniendo los ultimos 12 meses a partir de la fecha en curso

*!*************************************************************************
*!
*!    Procedure: ARREGLO_MESES
*!
*!    Called by: TMENU.MPR
*!
*!*************************************************************************
PROCEDURE arreglo_meses
PRIVATE I, fecha_anterior, anio, mes

FOR I= 1 TO 12
  fecha_anterior = GOMONTH(dia_en_curso, -I)
  anio = SUFSTR( ALLTRIM(STR(YEAR(fecha_anterior))),3 )
  mes =  ALLTRIM( STR(MONTH(fecha_anterior)) )
  mes = IIF( LEN(mes)=1 , "0"+mes, mes)
  arrul12meses[I] = VAL(anio+mes)
  IF I=1
    ul12meses = anio+mes
  ELSE
    ul12meses = ul12meses + ","+anio+mes
  ENDIF
ENDFOR
RETUVN
```

```
** Genera un arreglo conteniendo el promedio del saldo vencido de cada producto

PROCEDURE media_prod
SELECT sdo_cveprd AS producto, AVG(sdo_sdovdo) AS media ;
  FROM  tarsdos ;
  GROUP BY producto ;
  INTO TABLE avgprod.dbf
RETURN


** Genera un arreglo conteniendo el promedio de sobregiro de cada producto

PROCEDURE media_sob

SELECT sdo_cveprd AS producto, AVG(sdo_sdoact - sdo_limcre) AS medsob ;
  FROM tarsdos ;
  WHERE sdo_sdoact > sdo_limcre ;
  GROUP BY producto ;
  INTO TABLE avgsob.dbf

RETURN


**  Genera un arreglo con el numero y nombre de productos o tarjetas

PROCEDURE arreglo_producto
SELECT DISTINCT prd_nomprd, prd_numprd ;
  FROM tarprod ;
  WHERE !EMPTY(prd_numprd);
  INTO ARRAY arrproducto ;
  ORDER BY prd_nomprd
SHOW GETS
RETURN


** Genera un arreglo con el numero y nombre de linea de productos

PROCEDURE arreglo_linea

SELECT DISTINCT "LINEA "+ ALLTRIM(STR(prd_numlin)), prd_numlin ;
  FROM tarprod ;
  WHERE !EMPTY(prd_numlin);
  INTO ARRAY arrlinea ;
  ORDER BY prd_numlin
SHOW GETS
RETURN


** Genera un arreglo con el numero y nombre de sucursales de Banamex

PROCEDURE arreglo_sucursal
SELECT DISTINCT est_nomsuc, est_numsuc ;
  FROM tarestru ;
  WHERE !EMPTY(est_numsuc);
  INTO ARRAY arrsucursal ;
  ORDER BY est_nomsuc
SHOW GETS
RETURN
```

```
** Genera un arreglo con el numero y nombre divisiones de Banamex

PROCEDURE arreglo_division
SELECT DISTINCT nes_nomdiv, nes_numdiv ;
  FROM tamomes ;
  WHERE !EMPTY(nes_numdiv) ;
  INTO ARRAY arrdivision ;
  ORDER BY nes_nomdiv
SHOW GETS
RETURN


** Crea una tabla temporal conteniendo los nombres de cada cliente
** El archivo generado servirá para la deteccion de nombres similares

PROCEDURE copia_nombres

IF !FILE("NOMBRES.DBF")
  SELECT sdo_numcta AS cuenta, sdo_nomcli, flag, ntar ;
    FROM tarsdos ;
    INTO TABLE tmpnom
  USE
ENDIF

RETURN


PROCEDURE _quu00tp5y
DO scr110.spr


PROCEDURE _quu00tpd7
der = SUBSTR(privilegios,9,1)


SET SYSMENU OFF
IF der <> "L"
  do scr120
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON


PROCEDURE _quu00tpla
der = SUBSTR(privilegios,10,1)

SET SYSMENU OFF
IF der <> "L"
  DO scr130.spr
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON


PROCEDURE _quu00tpnl
der = SUBSTR(privilegios,11,1)
```

```
SET SYSMENU OFF
IF der <> "L"
  DO scr140.spr
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON


PROCEDURE _quu00tpsy
DO scr150.spr


PROCEDURE _quu00tpy5
CLOSE DATA
CLEAR READ
SET SYSMENU TO DEFAULT
*KEYBOARD '{ENTER}'
SET DEFAULT TO &s_def
SET PATH TO &s_pat

** ESTAS DOS LINEAS DEBEN ESTAR ACTIVAS SOLO CUANDO SE GENERA EJECUTABLE
m.done = .T.
CLEAR READ ALL

PROCEDURE _quu00tq4q
der = SUBSTR(privilegios,5,1)

SET SYSMENU OFF
IF der <> "L"
  DO scr210.spr
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON


PROCEDURE _quu00tqad
der = SUBSTR(privilegios,6,1)


SET SYSMENU OFF
IF der <> "L"
  DO scr220.spr
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON


PROCEDURE _quu00tqfo
der = SUBSTR(privilegios,7,1)

SET SYSMENU OFF
IF der <> "L"
  DO scr230.spr
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON
```

```
PROCEDURE _quu00tqkv
der = SUBSTR(privilegios,8,1)

SET SYSMENU OFF
IF der <> "L"
  DO scr240.spr
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON


PROCEDURE _quu00tqr9
der = SUBSTR(privilegios,12,1)

SET SYSMENU OFF
IF der <> "L"
  DO scr310.spr
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON

PROCEDURE _quu00tqwk
der = SUBSTR(privilegios,13,1)

SET SYSMENU OFF
IF der <> "L"
  DO scr320.spr
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON

DO scr410.spr

der = SUBSTR(privilegios,14,1)

SET SYSMENU OFF
IF der <> "L"
  DO scr420
ELSE
  DO msgerror.spr WITH 1
ENDIF
SET SYSMENU ON

*: EOF: TMENU.MPR

*    ************************************************************
*    *
*    * 11/04/94        SCR110.SPR        00:39:25
*    *
*    ************************************************************


*    ************************************************************
*    *
*    *        SCR110/Windows Cleanup Code
*    *
*    ************************************************************
```

```
#REGION 1
SET CURSOR ON

*    ************************************************************
*    *
*    * _QUU01EPUO          m.about VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR110,     Record Number:  12
*    * Variable:           m.about
*    * Called By:          VALID Clause
*    * Snippet Number:     1
*    *
*    ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01EPUO
*!
*!     Called by: SCR110.SPR
*!
*!*************************************************************************
FUNCTION _quu01epuo     && m.about VALID
#REGION 1
CLEAR READ
*: EOF: SCR110.SPR

*    ************************************************************
*    *
*    * 11/04/94          SCR130.SPR              00:39:38
*    *
*    ************************************************************



*    ************************************************************
*    *
*    *       SCR130/Windows Setup Code - SECTION 2
*    *
*    ************************************************************

#REGION 1
PUBLIC flag_nuevo, tmpprd
STORE  0 TO flag_nuevo
STORE "" TO tmpprd

m.fecha = {//}
arrproducto = ""
SELECT producto
SCATTER MEMVAR BLANK
SHOW GETS

*    ************************************************************
*    *
*    *           SCR130/Windows Cleanup Code
*    *
*    ************************************************************
```

```
#REGION 1
*|*****************************************************************************
*|
*|    Procedure: LEE_NOMBRE
*|
*|    Called by: _QUU01EYKU()      (function in SCR130.SPR)
*|             : _QUU01F4IB()      (function in SCR140.SPR)
*|
*|*****************************************************************************
PROCEDURE lee_nombre
IF flag_nuevo = 1
  WAIT "Escriba el nombre del producto y <enter> para continuar" WINDOW NOWAIT
ELSE
  WAIT "Edite el nombre o presione <enter> para continuar" WINDOW NOWAIT
ENDIF

@ 2.72,25.6 GET m.prd_nomprd;
  SIZE 1.38,48.4;
  DEFAULT " " ;
  FONT "MS Sans Serif", 8 ;
  STYLE "B" ;
  PICTURE "@!k"
READ
RETURN

**
*|*****************************************************************************
*|
*|    Function: DTON
*|
*|    Called by: _QUU01EYX9()      (function in SCR130.SPR)
*|
*|*****************************************************************************
FUNCTION dton
PARAMETER  dfecha
PRIVATE strfecha

strfecha = ALLTRIM(DTOC(dfecha))
strfecha = ALLTRIM(SUBSTR(strfecha,9)+SUBSTR(strfecha,4,2)+SUBSTR(strfecha,1,2))

RETURN VAL(strfecha)

**
*|*****************************************************************************
*|
*|    Function: NTOD
*|
*|    Called by: _QUU01EYAP()      (function in SCR130.SPR)
*|             : _QUU01EYO0()      (function in SCR130.SPR)
*|             : _QUU01EYX9()      (function in SCR130.SPR)
*|
*|*****************************************************************************
FUNCTION ntod
PARAMETER  nfecha
PRIVATE strfecha

strfecha = ALLTRIM(STR(nfecha))
strfecha = ALLTRIM(SUBSTR(strfecha,5)+"/"+SUBSTR(strfecha,3,2)+"/"+SUBSTR(strfecha,1,2))

RETURN CTOD(strfecha)
```

```
*    ************************************************************
*    *
*    * _QUU01EYAP          m.prd_numprd VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:       Windows
*    * From Screen:         SCR130,     Record Number:  23
*    * Variable:            m.prd_numprd
*    * Called By:           VALID Clause
*    * Snippet Number:      1
*    *
*    ************************************************************
*

*!*************************************************************************
*!
*!      Function: _QUU01EYAP
*!
*!      Called by: SCR130.SPR
*!
*!        Calls: NTOD()          (function  in SCR130.SPR)
*!
*!*************************************************************************
FUNCTION _quu01eyap    && m.prd_numprd VALID
#REGION 1
IF EMPTY(m.prd_numprd)
  IF flag_nuevo = 1 OR flag_nuevo = 2
    _CUROBJ = OBJNUM(m.prd_numprd)
  ELSE
    _CUROBJ = OBJNUM(m.prd_nomprd)
  ENDIF
ELSE
  IF flag_nuevo = 1 OR flag_nuevo = 2
    _CUROBJ = OBJNUM(m.prd_nomprd)
  ELSE
    LOCATE FOR producto.prd_numprd = m.prd_numprd
    IF !FOUND()
      WAIT "Producto Inexistente" WINDOW
      _CUROBJ = OBJNUM(m.prd_nomprd)
    ELSE
      SCATTER MEMVAR
      m.fecha = ntod(m.prd_fecacl)
      SHOW GETS
      _CUROBJ = OBJNUM(m.botprd)
    ENDIF
  ENDIF
ENDIF

*    ************************************************************
*    *
*    * _QUU01EYKU          m.prd_nomprd WHEN
*    *
*    * Function Origin:
*    *
*    * From Platform:       Windows
*    * From Screen:         SCR130,     Record Number:  24
*    * Variable:            m.prd_nomprd
*    * Called By:           WHEN Clause
*    * Snippet Number:      2
```

```
*|*****************************************************************
*|
*|    Function: _QUU01EYKU
*|
*|    Called by: SCR130.SPR
*|
*|       Calls: LEE_NOMBRE       (procedure in SCR130.SPR)
*|            : ARREGLO_PRODUCTO  (procedure in TMENU.MPR)
*|
*|*****************************************************************


FUNCTION _quu01eyku    && m.prd_nomprd WHEN
#REGION 1
IF flag_nuevo = 1 OR flag_nuevo = 2
  DO lee_nombre
  _CUROBJ = OBJNUM(m.prd_numlin)
ELSE
  DO arreglo_producto
  m.prd_nomprd = 0
ENDIF


*    ************************************************
*    *
*    * _QUU01EYO0         m.prd_nomprd VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR130,    Record Number:  24
*    * Variable:           m.prd_nomprd
*    * Called By:          VALID Clause
*    * Snippet Number:     3
*    *
*    ************************************************
*


*|*****************************************************************
*|
*|    Function: _QUU01EYO0
*|
*|    Called by: SCR130.SPR
*|
*|       Calls: NTOD()          (function  in SCR130.SPR)
*|
*|*****************************************************************
FUNCTION _quu01eyo0    && m.prd_nomprd VALID
#REGION 1
m.prd_numprd = arrproducto[m.prd_nomprd,2]
m.prd_nomprd = arrproducto[m.prd_nomprd,1]
SHOW GETS

LOCATE FOR producto.prd_numprd = m.prd_numprd
SCATTER MEMVAR
m.fecha = ntod(m.prd_fecact)
SHOW GETS
_CUROBJ = OBJNUM(m.botprd)

*    ************************************************
```

```
*
* _QUU01EYX9        m.botprd VALID
*
* Function Origin:
*
* From Platform:     Windows
* From Screen:       SCR130,    Record Number:  30
* Variable:          m.botprd
* Called By:         VALID Clause
* Snippet Number:    4
*
*************************************************

*!*************************************************************************
*!
*!     Function: _QUU01EYX9
*!
*!     Called by: SCR130.SPR
*!
*!       Calls: DTON()          (function  in SCR130.SPR)
*!            : NTOD()          (function  in SCR130.SPR)
*!
*!        Uses: TARPROD.DBF          Alias: PRODUCTO
*!
*!     CDX files: TARPROD.CDX
*!
*!  Report Forms: REPPROD.FRX
*!
*!*************************************************************************
FUNCTION _quu01eyx9    && m.botprd VALID
#REGION 1
DO CASE
CASE m.botprd = 1
  DO CASE
  CASE flag_nuevo = 0                                          && Nuevo
    tmpprd = m.prd_numprd
    SCATTER MEMVAR BLANK
    m.fecha = CTOD("")
    SHOW GETS
    SHOW GETS ENABLE
    SHOW GET m.botprd  DISABLE
    SHOW GET m.botprd,1  PROMPT "\<Aceptar"  ENABLE
    SHOW GET m.botprd,2  PROMPT "\<Cancelar" ENABLE
    flag_nuevo = 1
    _CUROBJ = OBJNUM(m.prd_numprd)

  CASE flag_nuevo = 1                                          && Aceptar para Nuevo
    tmpprd = m.prd_numprd
    LOCATE FOR producto.prd_numprd = tmpprd
    IF IFOUND()
      IF IEMPTY(m.prd_nomprd) AND IEMPTY(m.prd_numprd) AND IEMPTY(m.prd_numlin)
        GO BOTTOM
        m.prd_fecacl = dton(m.fecha)
        APPEND BLANK
        GATHER MEMVAR
        SHOW GETS DISABLE
        SHOW GET m.prd_numprd ENABLE
        SHOW GET m.prd_nomprd          ENABLE
        SHOW GET m.botprd ENABLE
        SHOW GET m.botprd,1  PROMPT "\<Nuevo"
```

```
      SHOW GET m.botprd,2  PROMPT "\<Modificar"
      flag_nuevo = 0
    ELSE
      WAIT WINDOW "Datos Incompletos"
    ENDIF
  ELSE
    WAIT WINDOW "El login ya existe !"
    _CUROBJ = OBJNUM(m.botprd)
  ENDIF


 CASE flag_nuevo = 2                                    && Aceptar para Modificar
  tmpprd = m.prd_numprd
  LOCATE FOR producto.prd_numprd = tmpprd
  IF RLOCK()
    m.prd_fecacl = dton(m.fecha)
    GATHER MEMVAR
    UNLOCK
    SHOW GETS DISABLE
    SHOW GET m.prd_numprd ENABLE
    SHOW GET m.prd_nomprd  ENABLE
    SHOW GET m.botprd          ENABLE
    SHOW GET m.botprd,1 PROMPT "\<Nuevo"
    SHOW GET m.botprd,2 PROMPT "\<Modificar"
    flag_nuevo = 0
  ELSE
    WAIT WINDOW "El Registro está siendo Modificado por otro usuario"
    WAIT WINDOW "Proceso Cancelado"
  ENDIF

 CASE flag_nuevo = 3                                    && Aceptar para Borrar
  tmpprd = m.prd_numprd
  LOCATE FOR producto.prd_numprd = tmpprd
  IF RLOCK()
    DELETE
    UNLOCK
    WAIT WINDOW "Producto dado de baja"
    SHOW GET m.prd_numprd  ENABLE
    SHOW GET m.prd_nomprd          ENABLE
    SHOW GET m.botprd              ENABLE
    SHOW GET m.botprd,1  PROMPT "\<Nuevo"
    SHOW GET m.botprd,2  PROMPT "\<Modificar"
    flag_nuevo = 0
    SCATTER MEMVAR BLANK
    m.fecha = {//}
    _CUROBJ = OBJNUM(m.botprd)
  ELSE
    WAIT WINDOW "El Registro está siendo Modificado por otro usuario"
    WAIT WINDOW "Proceso Cancelado"
  ENDIF

 ENDCASE

CASE m.botprd = 2
  DO CASE
  CASE flag_nuevo = 0                                   && Modificar
    IF !EMPTY(m.prd_numprd)
      tmpprd = m.prd_numprd
      SHOW GETS ENABLE
      SHOW GET m.prd_numprd  DISABLE
      SHOW GET m.botprd              DISABLE
```

```
    SHOW GET m.botprd,1  PROMPT "\<Aceptar"  ENABLE
    SHOW GET m.botprd,2  PROMPT "\<Cancelar" ENABLE
    flag_nuevo = 2
    _CUROBJ = OBJNUM(m.prd_nomprd)
  ENDIF

 CASE flag_nuevo = 1                                    && Cancelar para Nuevo
  SHOW GETS DISABLE
  SHOW GET m.prd_numprd    ENABLE
  SHOW GET m.prd_nomprd    ENABLE
  SHOW GET m.botprd        ENABLE
  SHOW GET m.botprd,1      PROMPT "\<Nuevo"
  SHOW GET m.botprd,2  PROMPT "\<Modificar"
  LOCATE FOR producto.prd_numprd = tmpprd
  IF IFOUND()
    WAIT WINDOW "Producto Inexistente"
    _CUROBJ = OBJNUM(m.prd_numprd)
  ELSE
    SCATTER MEMVAR
    m.fecha = ntod(m.prd_fecacl)
    SHOW GETS
  ENDIF
  flag_nuevo = 0


 CASE flag_nuevo = 2                                    && Cancelar para Modificar
  SHOW GETS DISABLE
  SHOW GET m.prd_numprd    ENABLE
  SHOW GET m.prd_nomprd    ENABLE
  SHOW GET m.botprd        ENABLE
  SHOW GET m.botprd,1      PROMPT "\<Nuevo"
  SHOW GET m.botprd,2  PROMPT "\<Modificar"
  LOCATE FOR producto.prd_numprd = tmpprd
  IF IFOUND()
    WAIT WINDOW "Producto Inexistente"
    _CUROBJ = OBJNUM(m.prd_numprd)
  ELSE
    SCATTER MEMVAR
    m.fecha = ntod(m.prd_fecacl)
    SHOW GETS
  ENDIF
  flag_nuevo = 0


 CASE flag_nuevo = 3                                    && Cancelar para Borrar
  SHOW GETS DISABLE
  SHOW GET m.prd_numprd    ENABLE
  SHOW GET m.prd_nomprd    ENABLE
  SHOW GET m.botprd        ENABLE
  SHOW GET m.botprd,1      PROMPT "\<Nuevo"
  SHOW GET m.botprd,2  PROMPT "\<Modificar"
  flag_nuevo = 0
 ENDCASE


CASE m.botprd = 3                                       && Borrar
 tmpprd = m.prd_numprd
 SHOW GETS DISABLE
 SHOW GET m.botprd,1 ENABLE PROMPT "\<Aceptar"
 SHOW GET m.botprd,2 ENABLE PROMPT "\<Cancelar"
 flag_nuevo = 3
```

```
CASE m.botprd = 4
  SELECT * ;
    FROM tarprod ;
    ORDER BY prd_numlin, prd_numprd;
    INTO CURSOR tmpprd
  REPORT FORM repprod PREVIEW

CASE m.botprd = 5

  CLEAR READ
ENDCASE


*     ************************************************************
*     *
*     * _QUU01EZR1          Read Level When
*     *
*     * Function Origin:
*     *
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR130
*     * Called By:          READ Statement
*     * Snippet Number:     5
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!    Function: _QUU01EZR1
*!
*!    Called by: SCR130.SPR
*!
*!**************************************************************************
FUNCTION _quu01ezr1     && Read Level When
*
* When Code from screen: SCR130

#REGION 1
SHOW GETS DISABLE
SHOW GET m.prd_numprd ENABLE
SHOW GET m.prd_nomprd ENABLE
SHOW GET m.botprd ENABLE
_CUROBJ = OBJNUM(m.prd_numprd)
*: EOF: SCR130.SPR

*     ************************************************************
*     *
*     * 11/04/94          SCR140.SPR              00:39:44
*     *
*     ************************************************************


*     ************************************************************
*     *
*     *       SCR140/Windows Setup Code - SECTION 2
*     *
*     ************************************************************
*
```

```
#REGION 1
PUBLIC flag_nuevo, tmplog, tmppas, priv_cartera, usu_priv
STORE  0 TO flag_nuevo
STORE "" TO tmplog

PUBLIC ARRAY arrpriv[14]

DIMENSION arrusuario[1,2]
arrusuario = ""

SELECT usuario
SET ORDER TO TAG usu_login

arrpriv = "L"


*     ************************************************
*     *
*     *            SCR140/Windows Cleanup Code
*     *
*     ************************************************
*

#REGION 1

*     ************************************************
*     *
*     *  SCR140/Windows Supporting Procedures and Functions
*     *
*     ************************************************

#REGION 1
*!*************************************************************************
*!
*!     Procedure: ARREGLO_USUARIO
*!
*!     Called by: _QUU01F4IB()     (function in SCR140.SPR)
*!
*!          Uses: TUSUARIO.DBF          Alias: USUARIO
*!
*!     CDX files: TUSUARIO.CDX
*!
*!*************************************************************************

PROCEDURE arreglo_usuario
SELECT DISTINCT usu_nom, usu_login ;
  FROM tusuario;
  ORDER BY usu_nom;
  WHERE !EMPTY(usu_login);
  INTO ARRAY arrusuario
SHOW GETS
RETURN

*!*************************************************************************
*!
*!     Procedure: LEE_NOMBRE
*!
*!     Called by: _QUU01EYKU()     (function in SCR130.SPR)
*!              : _QUU01F4IB()     (function in SCR140.SPR)
*!
*!*************************************************************************
```

```
PROCEDURE lee_nombre
IF flag_nuevo = 1
  WAIT "Escriba el nombre del usuario y <enter> para continuar" WINDOW NOWAIT
ELSE
  WAIT "Edite el nombre o presione <enter> para continuar" WINDOW NOWAIT
ENDIF

@ 2.7,53 GET m.usu_nom;
  SIZE 1.38,52;
  DEFAULT " " ;
  FONT "MS Sans Serif", 8 ;
  STYLE "B" ;
  PICTURE "@!k"
READ
RETURN

*|*****************************************************************************
*|    Procedure: MAPEA_PRIV
*|
*|    Called by: _QUU01F4BG()     (function  in SCR140.SPR)
*|          : _QUU01F4LQ()     (function  in SCR140.SPR)
*|          : _QUU01F76M()     (function  in SCR140.SPR)
*|
*|*****************************************************************************
PROCEDURE mapea_priv

arrpriv[1] = SUBSTR(m.usu_priv,1,1)
arrpriv[2] = SUBSTR(m.usu_priv,2,1)
arrpriv[3] = SUBSTR(m.usu_priv,3,1)
arrpriv[4] = SUBSTR(m.usu_priv,4,1)
arrpriv[5] = SUBSTR(m.usu_priv,5,1)
arrpriv[6] = SUBSTR(m.usu_priv,6,1)
arrpriv[7] = SUBSTR(m.usu_priv,7,1)
arrpriv[8] = SUBSTR(m.usu_priv,8,1)
arrpriv[9] = SUBSTR(m.usu_priv,9,1)
arrpriv[11] = SUBSTR(m.usu_priv,11,1)
arrpriv[12] = SUBSTR(m.usu_priv,12,1)
arrpriv[13] = SUBSTR(m.usu_priv,13,1)
arrpriv[14] = SUBSTR(m.usu_priv,14,1)

IF arrpriv[1] = "U"
  m.bot_atar = 1
ELSE
  m.bot_atar = 0
  SHOW GET m.chk_cred DISABLE
  SHOW GET m.chk_sob DISABLE
  SHOW GET m.chk_cv DISABLE
  SHOW GET m.chk_cp DISABLE
ENDIF

IF arrpriv[2] = "U"
  m.bot_admin = 1
ELSE
  m.bot_admin = 0
  SHOW GET m.chk_acov DISABLE
  SHOW GET m.chk_aprd DISABLE
  SHOW GET m.chk_ausu DISABLE
ENDIF
```

```
IF arrpriv[3] = "U"
  m.bot_cons = 1
ELSE
  m.bot_cons = 0
  SHOW GET m.chk_cdir DISABLE
  SHOW GET m.chk_cfil DISABLE
ENDIF

IF arrpriv[4] = "U"
  m.bot_util = 1
ELSE
  m.bot_util = 0
  SHOW GET m.chk_ures DISABLE
ENDIF

m.chk_cred = IIF( arrpriv[5] = "L",0,1)
m.chk_sob  = IIF( arrpriv[6] = "L",0,1)
m.chk_cv   = IIF( arrpriv[7] = "L",0,1)
m.chk_cp   = IIF( arrpriv[8] = "L",0,1)
m.chk_acov = IIF( arrpriv[9] = "L",1,0)
m.chk_aprd = IIF( arrpriv[10] = "U",1,0)
m.chk_ausu = IIF( arrpriv[11] = "U",1,0)
m.chk_adir = IIF( arrpriv[12] = "U",1,0)
m.chk_cfil = IIF( arrpriv[13] = "U",1,0)
m.chk_ures = IIF( arrpriv[14] = "U",1,0)

RETURN

*     ************************************************************
*     *
*     * _QUU01F4BG          m.usu_login VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  14
*     * Variable:            m.usu_login
*     * Called By:           VALID Clause
*     * Snippet Number:      1
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!      Function: _QUU01F4BG
*!
*!      Called by: SCR140.SPR
*!
*!        Calls: MAPEA_PRIV        (procedure in SCR140.SPR)
*!
*!*************************************************************************
FUNCTION _quu01f4bg     && m.usu_login VALID
#REGION 1
IF EMPTY(m.usu_login)
  IF flag_nuevo = 1 OR flag_nuevo = 2
    _CUROBJ = OBJNUM(m.usu_login)
  ELSE
    _CUROBJ = OBJNUM(m.usu_nom)
  ENDIF
ELSE
  IF flag_nuevo = 1 OR flag_nuevo = 2
```

```
    _CUROBJ = OBJNUM(m.usu_nom)
  ELSE
    SEEK m.usu_login
    IF !FOUND()
      WAIT "Usuario Inexistente" WINDOW
      _CUROBJ = OBJNUM(m.usu_nom)
    ELSE
      SCATTER MEMVAR
      DO mapea_priv
      SHOW GETS
      _CUROBJ = OBJNUM(m.botusu)
    ENDIF
  ENDIF
ENDIF

*     ************************************************************
*     *
*     * _QUU01F4IB         m.usu_nom WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR140,    Record Number:  15
*     * Variable:           m.usu_nom
*     * Called By:          WHEN Clause
*     * Snippet Number:     2
*     *
*     ************************************************************

*!*************************************************************************
*!
*!      Function: _QUU01F4IB
*!
*!      Called by: SCR140.SPR
*!
*!         Calls: LEE_NOMBRE        (procedure in SCR130.SPR)
*!              : ARREGLO_USUARIO   (procedure in SCR140.SPR)
*!
*!*************************************************************************
FUNCTION _quu01f4ib     && m.usu_nom WHEN
#REGION 1
IF flag_nuevo = 1 OR flag_nuevo = 2
  DO lee_nombre
  _CUROBJ = OBJNUM(m.usu_tel)
ELSE
  m.usu_nom = 1
  DO arreglo_usuario
ENDIF

*     ************************************************************
*     *
*     * _QUU01F4LQ         m.usu_nom VALID
*     *
*     * Function Origin:
*     * From Platform:      Windows
*     * From Screen:        SCR140,    Record Number:  15
*     * Variable:           m.usu_nom
*     * Called By:          VALID Clause
*     * Snippet Number:     3
*     *
*     ************************************************************
```

```
*|*****************************************************************************
*|
*|    Function: _QUU01F4LQ
*|
*|    Called by: SCR140.SPR
*|
*|       Calls: MAPEA_PRIV        (procedure in SCR140.SPR)
*|
*|*****************************************************************************
FUNCTION _quu01f4lq    &&  m.usu_nom VALID
#REGION 1
m.usu_login = arrusuario[m.usu_nom,2]
m.usu_nom = arrusuario[m.usu_nom,1]


SEEK m.usu_login
IF FOUND()
  SCATTER MEMVAR
  DO mapea_priv
  SHOW GETS
  _CUROBJ = OBJNUM(m.botusu)
ELSE
  WAIT WINDOW "Usuario inexistente"
  _CUROBJ = OBJNUM(m.usu_nom)
ENDIF

*    ************************************************************
*    *
*    * _QUU01F4QY          m.pass VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:       Windows
*    * From Screen:         SCR140,     Record Number:  17
*    * Variable:            m.pass
*    * Called By:           VALID Clause
*    * Snippet Number:      4
*    *
*    ************************************************************
*
*|*****************************************************************************
*|
*|    Function: _QUU01F4QY
*|
*|    Called by: SCR140.SPR
*|
*|*****************************************************************************
FUNCTION _quu01f4qy    &&  m.pass VALID
#REGION 1
IF EMPTY(m.pass)
  _CUROBJ = OBJNUM(m.pass)
ELSE
  _CUROBJ = OBJNUM(m.retype)
ENDIF

*    ************************************************************
*    *
*    * _QUU01F4UI          m.retype VALID
*    *
*    * Function Origin:
```

```
*     * From Platform:      Windows
*     * From Screen:        SCR140,    Record Number:  18
*     * Variable:           m.retype
*     * Called By:          VALID Clause
*     * Snippet Number:     5
*     *
*     ************************************************************

*!*************************************************************************
*!
*!      Function: _QUU01F4UI
*!
*!      Called by: SCR140.SPR
*!
*!        Calls: ENCRIPTA          (procedure in SEGURID.PRG)
*!
*!*************************************************************************
FUNCTION _quu01f4ui     && m.retype VALID
#REGION 1
IF EMPTY(m.retype)
  _CUROBJ = OBJNUM(m.retype)
ELSE

  IF ALLTRIM(m.pass) = ALLTRIM(m.retype)
    DO encripta IN segurid WITH m.pass, tmppas
  ELSE
    WAIT "El password no coincide" WINDOW
    _CUROBJ = OBJNUM(m.pass)
  ENDIF
ENDIF

*     ************************************************************
*     *
*     * _QUU01F59R          m.bot_atar VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR140,    Record Number:  19
*     * Variable:           m.bot_atar
*     * Called By:          VALID Clause
*     * Snippet Number:     6
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!      Function: _QUU01F59R
*!
*!      Called by: SCR140.SPR
*!
*!*************************************************************************
FUNCTION _quu01f59r     && m.bot_atar VALID
#REGION 1
IF m.bot_atar = 1
  arrpriv[1] = "U"
  SHOW GET m.chk_cred ENABLE
  SHOW GET m.chk_sob ENABLE
  SHOW GET m.chk_cv ENABLE
  SHOW GET m.chk_cp ENABLE
```

```
ELSE
  m.chk_cred = 0
  m.chk_sob = 0
  m.chk_cv = 0
  m.chk_cp = 0
  SHOW GET m.chk_cred DISABLE
  SHOW GET m.chk_sob DISABLE
  SHOW GET m.chk_cv DISABLE
  SHOW GET m.chk_cp DISABLE
  arrpriv[1] = "L"
  arrpriv[5] = "L"
  arrpriv[6] = "L"
  arrpriv[7] = "L"
  arrpriv[8] = "L"
  SHOW GETS
ENDIF

*     ************************************************************
*     *
*     * _QUU01F5IF          m.bot_admin VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR140,    Record Number:  20
*     * Variable:           m.bot_admin
*     * Called By:          VALID Clause
*     * Snippet Number:     7
*     *
*     ************************************************************


*!*************************************************************************
*!
*!     Function: _QUU01F5IF
*!
*!     Called by: SCR140.SPR
*!
*!*************************************************************************
FUNCTION _quu01f5if    && m.bot_admin VALID
#REGION 1
IF m.bot_admin = 1
  arrpriv[2] = "U"
  SHOW GET m.chk_acov ENABLE
  SHOW GET m.chk_aprd ENABLE
  SHOW GET m.chk_ausu ENABLE

ELSE
  m.chk_acov = 0
  m.chk_aprd = 0
  m.chk_ausu = 0
  SHOW GET m.chk_acov DISABLE
  SHOW GET m.chk_aprd DISABLE
  SHOW GET m.chk_ausu DISABLE
  arrpriv[2] = "L"
  arrpriv[9] = "L"
  arrpriv[10] = "L"
  arrpriv[11] = "L"
  SHOW GETS
ENDIF
```

```
*     ************************************************************
*     *
*     * _QUU01F5TX          m.bot_cons VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  21
*     * Variable:            m.bot_cons
*     * Called By:           VALID Clause
*     * Snippet Number:      8
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!      Function: _QUU01F5TX
*!
*!      Called by: SCR140.SPR
*!
*!**************************************************************************
FUNCTION _quu01f5tx     && m.bot_cons VALID
#REGION 1
IF m.bot_cons = 1
  arrpriv[3] = "U"
  SHOW GET m.chk_cdir ENABLE
  SHOW GET m.chk_cfil ENABLE
ELSE
  m.chk_cdir = 0
  m.chk_cfil = 0
  SHOW GET m.chk_cdir DISABLE
  SHOW GET m.chk_cfil DISABLE
  arrpriv[3] = "L"
  arrpriv[12] = "L"
  arrpriv[13] = "L"
  SHOW GETS
ENDIF

*     ************************************************************
*     *
*     * _QUU01F60E          m.bot_util VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  22
*     * Variable:            m.bot_util
*     * Called By:           VALID Clause
*     * Snippet Number:      9
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!      Function: _QUU01F60E
*!
*!      Called by: SCR140.SPR
*!
*!**************************************************************************
FUNCTION _quu01f60e     && m.bot_util VALID
```

```
#REGION 1
IF m.bot_util = 1
  arrpriv[4] = "U"
  SHOW GET m.chk_ures ENABLE
ELSE
  m.chk_ures = 0
  SHOW GET m.chk_ures DISABLE
  arrpriv[4] = "L"
  arrpriv[14] = "L"
  SHOW GETS
ENDIF

*     ************************************************************
*     *
*     * _QUU01F64N          m.chk_cred VALID
*     *
*     * Function Origin:
*     * From Platform:      Windows
*     * From Screen:        SCR140,    Record Number:  23
*     * Variable:           m.chk_cred
*     * Called By:          VALID Clause
*     * Snippet Number:     10
*     *
*     ************************************************************
*!*************************************************************************
*!
*!    Function: _QUU01F64N
*!
*!    Called by: SCR140.SPR
*!
*!       Calls: SCR141.SPR
*!
*!*************************************************************************

FUNCTION _quu01f64n    && m.chk_cred VALID
#REGION 1
IF m.chk_cred = 1
  DO scr141.spr WITH "Modulo de Credito"
  arrpriv[5] = priv_cartera
ELSE
  arrpriv[5] = "L"
ENDIF

*     ************************************************************
*     *
*     * _QUU01F68A          m.chk_sob VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR140,    Record Number:  24
*     * Variable:           m.chk_sob
*     * Called By:          VALID Clause
*     * Snippet Number:     11
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!    Function: _QUU01F68A
*!
```

```
*!  ,  Called by: SCR140.SPR
*!
*!       Calls: SCR141.SPR
*!
*!*****************************************************************************
FUNCTION _quu01f68a     && m.chk_sob VALID
#REGION 1
IF m.chk_sob = 1

  DO scr141.spr WITH "Modulo de Sobregiros"
  arrpriv[6] = priv_cartera
ELSE

  arrpriv[6] = "L"
ENDIF

*     ************************************************************
*     *
*     * _QUU01F6C1          m.chk_cv VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  25
*     * Variable:           m.chk_cv
*     * Called By:          VALID Clause
*     * Snippet Number:      12
*     *
*     ************************************************************
*
*!*****************************************************************************
*!
*!     Function: _QUU01F6C1
*!
*!     Called by: SCR140.SPR
*!
*!       Calls: SCR141.SPR
*!
*!*****************************************************************************
FUNCTION _quu01f6c1     && m.chk_cv VALID
#REGION 1
IF m.chk_cv = 1

  DO scr141.spr WITH "Modulo de Cartera Vencida"
  arrpriv[7] = priv_cartera
ELSE
  arrpriv[7] = "L"
ENDIF

*     ************************************************************
*     *
*     * _QUU01F6FR          m.chk_cp VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  26
*     * Variable:           m.chk_cp
*     * Called By:          VALID Clause
*     * Snippet Number:      13
*     *
```

```
*|*****************************************************************
*|
*|     Function: _QUU01F6FR
*|
*|     Called by: SCR140.SPR
*|
*|       Calls: SCR141.SPR
*|
*|*****************************************************************
FUNCTION _quu01f6fr     &&  m.chk_cp VALID
#REGION 1
IF m.chk_cp = 1

  DO scr141.spr WITH "Modulo de Cartera Preventiva"
  arrpriv[8] = priv_cartera
ELSE
  arrpriv[8] = "L"
ENDIF

*     ************************************************
*     *
*     * _QUU01F6JG          m.chk_acov VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR140,     Record Number:  27
*     * Variable:           m.chk_acov
*     * Called By:          VALID Clause
*     * Snippet Number:     14
*     *
*     ************************************************
*
*|*****************************************************************
*|
*|     Function: _QUU01F6JG
*|
*|     Called by: SCR140.SPR
*|
*|*****************************************************************
FUNCTION _quu01f6jg     &&  m.chk_acov VALID
#REGION 1
IF m.chk_acov = 1
  arrpriv[9] = "U"
ELSE
  arrpriv[9] = "L"
ENDIF

*     ************************************************
*     *
*     * _QUU01F6O3          m.chk_aprd VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR140,     Record Number:  28
*     * Variable:           m.chk_aprd
*     * Called By:          VALID Clause
*     * Snippet Number:     15
*     *
```

```
*     ************************************************************
*!*************************************************************************
*!
*!     Function: _QUU01F6O3
*!
*!     Called by: SCR140.SPR
*!
*!*************************************************************************
FUNCTION _quu01f6o3     && m.chk_aprd VALID
#REGION 1
IF m.chk_aprd = 1
  arrpriv[10] = "U"
ELSE
  arrpriv[10] = "L"
ENDIF

*     ************************************************************
*     *
*     * _QUU01F6RK          m.chk_ausu VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  29
*     * Variable:            m.chk_ausu
*     * Called By:           VALID Clause
*     * Snippet Number:      16
*     ************************************************************


*!*************************************************************************
*!
*!     Function: _QUU01F6RK
*!
*!     Called by: SCR140.SPR
*!
*!*************************************************************************
FUNCTION _quu01f6rk     && m.chk_ausu VALID
#REGION 1
IF m.chk_ausu = 1
  arrpriv[11] = "U"
ELSE
  arrpriv[11] = "L"
ENDIF

*     ************************************************************
*     *
*     * _QUU01F6WL          m.chk_cdir VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  30
*     * Variable:            m.chk_cdir
*     * Called By:           VALID Clause
*     * Snippet Number:      17
*     *
*     ************************************************************
```

```
*!*************************************************************
*!
*!      Function: _QUU01F6WL
*!
*!      Called by: SCR140.SPR
*!
*!*************************************************************
FUNCTION _quu01f6wl     && m.chk_cdir VALID
#REGION 1
IF m.chk_cdir = 1
  arrpriv[12] = "U"
ELSE
  arrpriv[12] = "L"
ENDIF

*     ************************************************
*     *
*     * _QUU01F702          m.chk_cfil VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  31
*     * Variable:            m.chk_cfil
*     * Called By:           VALID Clause
*     * Snippet Number:      18
*     *
*     ************************************************

*!*************************************************************
*!
*!      Function: _QUU01F702
*!
*!      Called by: SCR140.SPR
*!
*!*************************************************************
FUNCTION _quu01f702     && m.chk_cfil VALID
#REGION 1
IF m.chk_cfil = 1
  arrpriv[13] = "U"
ELSE
  arrpriv[13] = "L"
ENDIF


*     ************************************************
*     *
*     * _QUU01F73J          m.chk_ures VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR140,     Record Number:  32
*     * Variable:            m.chk_ures
*     * Called By:           VALID Clause
*     * Snippet Number:      19
*     *
*     ************************************************
```

```
*|*************************************************************************
*|
*|    Function: _QUU01F73J
*|
*|    Called by: SCR140.SPR
*|
*|*************************************************************************

FUNCTION _quu01f73j    && m.chk_ures VALID
#REGION 1
IF m.chk_ures = 1
  arrpriv[14] = "U"
ELSE
  arrpriv[14] = "L"
ENDIF

*    ************************************************
*    *
*    * _QUU01F78M          m.botusu VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR140,    Record Number:  33
*    * Variable:           m.botusu
*    * Called By:          VALID Clause
*    * Snippet Number:     20
*    *
*    ************************************************
*
*|*************************************************************************
*|
*|    Function: _QUU01F78M
*|
*|    Called by: SCR140.SPR
*|
*|      Calls: MAPEA_PRIV       (procedure in SCR140.SPR)
*|
*|*************************************************************************
FUNCTION _quu01f78m    && m.botusu VALID
#REGION 1
DO CASE
CASE m.botusu = 1
  DO CASE
  CASE flag_nuevo = 0                                          && Nuevo
    tmplog = m.usu_login
    m.usu_priv = "LLLLLLLLLLLLL"
    DO mapea_priv
    SCATTER MEMVAR BLANK
    SHOW GETS
    SHOW GET DISABLE
    SHOW GET m.usu_login ENABLE
    SHOW GET m.usu_nom ENABLE
    SHOW GET m.usu_tel ENABLE
    SHOW GET m.pass ENABLE
    SHOW GET m.retype ENABLE
    SHOW GET m.bot_atar ENABLE
    SHOW GET m.bot_admin ENABLE
    SHOW GET m.bot_cons ENABLE
    SHOW GET m.bot_util ENABLE
    SHOW GET m.botusu DISABLE
```

```
  SHOW GET m.botusu,1  PROMPT "\<Aceptar"  ENABLE
  SHOW GET m.botusu,2  PROMPT "\<Cancelar" ENABLE
  flag_nuevo = 1
  _CUROBJ = OBJNUM(m.usu_login)

CASE flag_nuevo = 1                                    && Aceptar para Nuevo
  tmplog = m.usu_login
  LOCATE FOR usuario.usu_login = tmplog
  IF IFOUND()
    IF !EMPTY(m.usu_nom) AND !EMPTY(m.usu_login) AND !EMPTY(m.usu_tel) AND !EMPTY(tmppas)
      m.usu_pass = tmppas
      m.usu_priv = ""
      FOR i=1 TO 14
        m.usu_priv = m.usu_priv + arrpriv[i]
      NEXT I

      GO BOTTOM
      APPEND BLANK
      GATHER MEMVAR
      SHOW GETS DISABLE
      SHOW GET m.usu_login ENABLE
      SHOW GET m.usu_nom   ENABLE
      SHOW GET m.botusu ENABLE
      SHOW GET m.botusu,1  PROMPT "\<Nuevo"
      SHOW GET m.botusu,2  PROMPT "\<Modificar"
      flag_nuevo = 0
    ELSE
      WAIT WINDOW "Datos Incompletos"
    ENDIF
  ELSE
    WAIT WINDOW "El login ya existe !"
    _CUROBJ = OBJNUM(m.usu_login)
  ENDIF

CASE flag_nuevo = 2                                    && Aceptar para Modificar
  tmplog = m.usu_login
  LOCATE FOR ALLTRIM(usuario.usu_login) = ALLTRIM(tmplog)
  IF !EMPTY(m.usu_nom) AND !EMPTY(m.usu_login) AND !EMPTY(m.usu_tel) AND !EMPTY(tmppas)
    IF RLOCK()
      m.usu_pass = tmppas
      m.usu_priv = ""
      FOR i=1 TO 14
        m.usu_priv = m.usu_priv + arrpriv[i]
      NEXT I
      privilegios = m.usu_priv
      GATHER MEMVAR
      UNLOCK
      SHOW GETS DISABLE
      SHOW GET m.usu_login ENABLE
      SHOW GET m.usu_nom   ENABLE
      SHOW GET m.botusu        ENABLE
      SHOW GET m.botusu,1 PROMPT "\<Nuevo"
      SHOW GET m.botusu,2 PROMPT "\<Modificar"
      flag_nuevo = 0
      LOCATE FOR ALLTRIM(usuario.usu_login) = ALLTRIM(tmplog)
      IF FOUND()
        SCATTER MEMVAR
        DO mapea_priv
        SHOW GETS
        _CUROBJ = OBJNUM(m.botusu)
      ELSE
```

```
        WAIT WINDOW "Usuario Inexistente"
      ENDIF
    ELSE
      WAIT WINDOW "El Registro está siendo Modificado por otro usuario"
      WAIT WINDOW "Proceso Cancelado"
    ENDIF
  ELSE
    WAIT WINDOW "Datos Incompletos"
  ENDIF

 CASE flag_nuevo = 3                                    && Aceptar para Borrar
  tmplog = m.usu_login
  LOCATE FOR ALLTRIM(usuario.usu_login) = ALLTRIM(tmplog)
  IF RLOCK()
    DELETE
    UNLOCK
    WAIT WINDOW "Usuario dada de baja"
    SHOW GET m.usu_login     ENABLE
    SHOW GET m.usu_nom       ENABLE
    SHOW GET m.botusu                ENABLE
    SHOW GET m.botusu,1  PROMPT "\<Nuevo"
    SHOW GET m.botusu,2  PROMPT "\<Modificar"
    flag_nuevo = 0
    _CUROBJ = OBJNUM(m.botusu)
  ELSE
    WAIT WINDOW "El Registro está siendo Modificado por otro usuario"
    WAIT WINDOW "Proceso Cancelado"
  ENDIF
 ENDCASE

CASE m.botusu = 2
  DO CASE
  CASE flag_nuevo = 0                                   && Modificar
    IF !EMPTY(m.usu_login)
      tmplog = m.usu_login
      SHOW GETS ENABLE
      DO mapea_priv
      SHOW GET m.usu_login     DISABLE
      SHOW GET m.botusu                DISABLE
      SHOW GET m.botusu,1  PROMPT "\<Aceptar"  ENABLE
      SHOW GET m.botusu,2  PROMPT "\<Cancelar" ENABLE
      flag_nuevo = 2
      _CUROBJ = OBJNUM(m.usu_nom)
    ENDIF

  CASE flag_nuevo = 1                                   && Cancelar para Nuevo
    SHOW GETS DISABLE
    SHOW GET m.usu_login       ENABLE
    SHOW GET m.usu_nom         ENABLE
    SHOW GET m.botusu                ENABLE
    SHOW GET m.botusu,1        PROMPT "\<Nuevo"
    SHOW GET m.botusu,2  PROMPT "\<Modificar"
    LOCATE FOR ALLTRIM(usuario.usu_login) = ALLTRIM(tmplog)
    IF NOT FOUND()
      WAIT WINDOW "Usuario Inexistente"
      _CUROBJ = OBJNUM(m.botusu)
    ELSE
      SCATTER MEMVAR
      DO mapea_priv
      SHOW GETS
    ENDIF
```

```
    flag_nuevo = 0

  CASE flag_nuevo = 2                                    && Cancelar para Modificar
    SHOW GETS DISABLE
    SHOW GET m.usu_login       ENABLE
    SHOW GET m.usu_nom         ENABLE
    SHOW GET m.botusu                  ENABLE
    SHOW GET m.botusu,1        PROMPT "\<Nuevo"
    SHOW GET m.botusu,2  PROMPT "\<Modificar"
    LOCATE FOR ALLTRIM(usuario.usu_login) = ALLTRIM(tmplog)
    IF IFOUND()
      WAIT WINDOW "Usuario Inexistente"
      _CUROBJ = OBJNUM(m.botusu)
    ELSE
      SCATTER MEMVAR
      DO mapea_priv
      SHOW GETS
    ENDIF
    flag_nuevo = 0

  CASE flag_nuevo = 3                                    && Cancelar para Borrar
    SHOW GETS DISABLE
    SHOW GET m.usu_login       ENABLE
    SHOW GET m.usu_nom         ENABLE
    SHOW GET m.botusu                  ENABLE
    SHOW GET m.botusu,1        PROMPT "\<Nuevo"
    SHOW GET m.botusu,2  PROMPT "\<Modificar"
    flag_nuevo = 0
  ENDCASE

CASE m.botusu = 3                                        && Borrar
  tmplog = m.usu_login
  SHOW GETS DISABLE
  SHOW GET m.botusu,1 ENABLE PROMPT "\<Aceptar"
  SHOW GET m.botusu,2 ENABLE PROMPT "\<Cancelar"
  flag_nuevo = 3

CASE m.botusu = 4

CASE m.botusu = 5

  CLEAR READ
ENDCASE

*     ************************************************************
*     *
*     * _QUU01F82R         Read Level When
*     *
*     * Function Origin:
*     *
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR140
*     * Called By:          READ Statement
*     * Snippet Number:     21
*     *
*     ************************************************************
```

```
*|*****************************************************************
*|
*|    Function: _QUU01F82R
*|
*|    Called by: SCR140.SPR
*|
*|*****************************************************************
FUNCTION _quu01f82r    && Read Level When
*
* When Code from screen: SCR140
#REGION 1
SHOW GETS DISABLE
SHOW GET m.usu_login ENABLE
SHOW GET m.usu_nom ENABLE
SHOW GET m.botusu ENABLE

*    ***********************************************
*    *
*    * _QUU01F82U          Read Level Show
*    *
*    * Function Origin:
*    *
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR140
*    * Called By:          READ Statement
*    * Snippet Number:     22
*    *
*    ***********************************************
*
*|*****************************************************************
*|
*|    Function: _QUU01F82U
*|
*|    Called by: SCR140.SPR
*|
*|*****************************************************************
FUNCTION _quu01f82u    && Read Level Show
PRIVATE currwind
STORE WOUTPUT() TO currwind


* Show Code from screen: SCR140
*
#REGION 1
*tmppas = m.pass
m.pass=""
m.retype=""
IF NOT EMPTY(currwind)
  ACTIVATE WINDOW (currwind) SAME
ENDIF
*: EOF: SCR140.SPR

*    ***********************************************
*    *
*    * 11/04/94          SCR141.SPR          00:41:10
*    *
*    ***********************************************

PARAMETERS modulo
```

```
*     ************************************************************
*     *
*     *       SCR141/Windows Setup Code - SECTION 1
*     *
*     ************************************************************
*


*     ************************************************************
*     *
*     * _QUU01GYGC          m.priv VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR141,    Record Number:  2
*     * Variable:           m.priv
*     * Called By:          VALID Clause
*     * Snippet Number:     1
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!    Function: _QUU01GYGC
*!
*!    Called by: SCR141.SPR
*!
*!**************************************************************************
FUNCTION _quu01gygc    && m.priv VALID
#REGION 1
DO CASE
CASE m.priv = 1
  priv_cartera = "D"
CASE m.priv = 2
  priv_cartera = "Q"
CASE m.priv = 3
  priv_cartera = "B"
ENDCASE
CLEAR READ
*: EOF: SCR141.SPR

*     ************************************************************
*     *
*     * 11/04/94           SCR150.SPR          00:39:58
*     *
*     ************************************************************


*     ************************************************************
*     *
*     *       SCR150/Windows Setup Code - SECTION 2
*     *
*     ************************************************************
*

#REGION 1
SELECT usuario
LOCATE FOR ALLTRIM(login)=ALLTRIM(usuario.usu_login)
m.prim=.T.
flag1=.T.
```

```
*     ************************************************************
*     *
*     * _QUU01FDO0         m.passw WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR150,    Record Number:   9
*     * Variable:           m.passw
*     * Called By:          WHEN Clause
*     * Snippet Number:     1
*     *
*     ************************************************************

*!*************************************************************************
*!
*!     Function: _QUU01FDO0
*!
*!     Called by: SCR150.SPR
*!
*!*************************************************************************
FUNCTION _quu01fdo0    &&  m.passw WHEN
#REGION 1
m.passw=""
SHOW GET m.ok,1 DISABLE
WAIT "Teclee el password actual" WINDOW NOWAIT

*     ************************************************************
*     *
*     * _QUU01FDR9         m.passw VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR150,    Record Number:   9
*     * Variable:           m.passw
*     * Called By:          VALID Clause
*     * Snippet Number:     2
*     *
*     ************************************************************
*!*************************************************************************
*!
*!     Function: _QUU01FDR9
*!
*!     Called by: SCR150.SPR
*!
*!        Calls: ENCRIPTA          (procedure in SEGURID.PRG)
*!
*!*************************************************************************
FUNCTION _quu01fdr9    &&  m.passw VALID
#REGION 1
IF !EMPTY(m.passw)
  passwed=""
  DO encripta IN segurid WITH ALLTRIM(m.passw),passwed
  IF ALLTRIM(passwed)<>ALLTRIM(usuario.usu_pass)
    WAIT "Password incorrecto" WINDOW NOWAIT
    m.passw=""
    _CUROBJ=OBJNUM(m.passw)
  ELSE
```

```
    m.passw=""
    SHOW GET m.passw    DISABLE
    SHOW GET m.passwnvo1 ENABLE
    SHOW GET m.passwnvo2 ENABLE
  ENDIF
ENDIF
*     ************************************************************
*     *
*     * _QUU01FDW6          m.passwnvo1 WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR150,     Record Number:  10
*     * Variable:           m.passwnvo1
*     * Called By:          WHEN Clause
*     * Snippet Number:     3
*     *
*     ************************************************************
*!*************************************************************************
*!
*!    Function: _QUU01FDW6
*!
*!    Called by: SCR150.SPR
*!
*!*************************************************************************
FUNCTION _quu01fdw6    && m.passwnvo1 WHEN
#REGION 1
m.passwnvo1=""

*     ************************************************************
*     * _QUU01FDYG          m.passwnvo1 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR150,     Record Number:  10
*     * Variable:           m.passwnvo1
*     * Called By:          VALID Clause
*     * Snippet Number:     4
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!    Function: _QUU01FDYG
*!
*!    Called by: SCR150.SPR
*!
*!*************************************************************************
FUNCTION _quu01fdyg    && m.passwnvo1 VALID
#REGION 1
IF EMPTY(m.passwnvo1)
  _CUROBJ=OBJNUM(m.passwnvo1)
ENDIF

*     ************************************************************
*     *
*     * _QUU01FE1Y          m.passwnvo2 WHEN
*     *
*     * Function Origin:
```

```
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR150,    Record Number:  11
*     * Variable:           m.passwnvo2
*     * Called By:          WHEN Clause
*     * Snippet Number:     5
*     *
*     ************************************************
*
*!*********************************************************************
*!
*!     Function: _QUU01FE1Y
*!
*!     Called by: SCR150.SPR
*!
*!*********************************************************************
FUNCTION _quu01fe1y    && m.passwnvo2 WHEN
#REGION 1
m.passwnvo2=""

*     ************************************************
*     *
*     * _QUU01FE49          m.passwnvo2 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR150,    Record Number:  11
*     * Variable:           m.passwnvo2
*     * Called By:          VALID Clause
*     * Snippet Number:     6
*     *
*     ************************************************
*
*!*********************************************************************
*!
*!     Function: _QUU01FE49
*!
*!     Called by: SCR150.SPR
*!
*!*********************************************************************

FUNCTION _quu01fe49    && m.passwnvo2 VALID
#REGION 1
IF EMPTY(m.passwnvo2)
  _CUROBJ=OBJNUM(m.passwnvo2)
ELSE
  IF ALLTRIM(m.passwnvo1)=ALLTRIM(m.passwnvo2)
    SHOW GET m.ok,1 ENABLE
    _CUROBJ=OBJNUM(m.ok)
  ELSE
    WAIT "Confirmación inválida, reteclee password nuevo" WINDOW NOWAIT
    SHOW GET m.ok,1 DISABLE
    m.passwnvo1=""
    m.passwnvo2=""
    _CUROBJ=OBJNUM(m.passwnvo1)
  ENDIF
ENDIF
```

```
*     ************************************************
*     *
*     * _QUU01FE94          m.ok VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR150,    Record Number:  12
*     * Variable:            m.ok
*     * Called By:           VALID Clause
*     * Snippet Number:      7
*     *
*     ************************************************
*
*!**************************************************************************
*!
*!     Function: _QUU01FE94
*!
*!     Called by: SCR150.SPR
*!
*!        Calls: ENCRIPTA          (procedure in SEGURID.PRG)
*!
*!**************************************************************************
FUNCTION _quu01fe94     && m.ok VALID
#REGION 1
DO CASE                                                  && Modificar cambio de password
CASE m.ok = 1
  IF flag1
    SHOW GET m.passw ENABLE
    SHOW GET m.ok,1 PROMPT "Aceptar" DISABLE
    flag1=.F.
    _CUROBJ=OBJNUM(m.passw)
  ELSE
    IF EMPTY(m.passwnvo1) OR EMPTY(m.passwnvo2)
      WAIT "Datos Incompletos" WINDOW NOWAIT
      RETURN
    ENDIF
    IF ALLTRIM(m.passwnvo1)=ALLTRIM(m.passwnvo2)
      passwed=""
      DO encripta IN segurid WITH ALLTRIM(m.passwnvo1),passwed
      REPLACE usu_pass WITH passwed
      WAIT "El password ha sido cambiado" WINDOW
      CLEAR READ
    ELSE
      WAIT "Confirmación inválida, reteclee password nuevo" WINDOW
      m.passwnvo1=""
      m.passwnvo2=""
      _CUROBJ=OBJNUM(m.passwnvo1)
    ENDIF
    flag1=.T.
    SHOW GETS DISABLE
    SHOW GET m.ok ENABLE
    SHOW GET m.ok,1 PROMPT "Modificar" ENABLE
  ENDIF
CASE m.ok = 2                            && Terminar
  RELEASE WINDOW hscpa
ENDCASE
```

```
*     ************************************************************
*     *
*     * _QUU01FEJZ          Read Level Activate
*     *
*     * Function Origin:
*     *
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR150
*     * Called By:          READ Statement
*     * Snippet Number:     8
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FEJZ
*!
*!     Called by: SCR150.SPR
*!
*!*************************************************************************
FUNCTION _quu01fejz    && Read Level Activate


* Activate Code from screen: SCR150
*
#REGION 1
IF m.prim
  SHOW GETS DISABLE
  SHOW GET m.ok ENABLE
  m.prim=.F.
ENDIF
*: EOF: SCR150.SPR

*     ************************************************************
*     *
*     * 11/04/94          SCR210.SPR              00:40:02
*     *
*     ************************************************************

*     ************************************************************
*     *
*     *       SCR210/Windows Setup Code - SECTION 2
*     *
*     ************************************************************
*

#REGION 1
PUBLIC selecciona, tctas, tsdoact, tsdovdo
PUBLIC dbfuente, dbdestino, namereport, proceso, est_cons

m.tctas = 0
m.tsdoact = 0
m.tsdovdo = 0
dbfuente = ""
dbdestino = ""
namereport = ""
proceso = ""
est_cons = ""
```

```
DIMENSION arrproducto[1,2], arrlinea[1]
DIMENSION arrcred1[3], arrcred2[4], arrcred3[2], arrsubcred[4]

DEFINE WINDOW mancred FROM 9,42 TO 30,86;
  TITLE "Manejo de credito" ;
  FONT "MS Serif",8 ;
  STYLE "B" ;
  FLOAT;
  GROW;
  NONE;
  NOCLOSE

arrcred1[1]= "CUENTAS DE ALTO RIESGO"
arrcred1[2]= "CUENTAS SIN PAGO"
arrcred1[3]= "OPERADOR/CALIDAD DE CREDITO"

arrcred2[1]= "LIMITES FUERA DE PARAMETROS"
arrcred2[2]= "ASIGNACION ERRONEA"
arrcred2[3]= "ASIGNACION INICIAL ERRONEA"
arrcred2[4]= "ASIGNACION AUTOMATIZADA"

arrcred3[1]= "SATURACION DE APERTURAS"
arrcred3[2]= "SDO. VENCIDO > CRED. OTORGADO"

arrsubcred=""
arrproducto=""
arrlinea=""


*     ************************************************************
*     *
*     *  SCR210/Windows Supporting Procedures and Functions
*     *
*     ************************************************************

#REGION 1
*!*************************************************************************
*!
*!     Procedure: INICIA_GENERAL
*!
*!     Called by: _QUU01FJ89()      (function  in SCR210.SPR)
*!              : _QUU01FJHY()      (function  in SCR210.SPR)
*!              : _QUU01FJQJ()      (function  in SCR210.SPR)
*!              : _QUU01FRX6()      (function  in SCR220.SPR)
*!              : _QUU01FSG0()      (function  in SCR220.SPR)
*!              : _QUU01FZR6()      (function  in SCR230.SPR)
*!              : _QUU01G015()      (function  in SCR230.SPR)
*!              : _QUU01G62N()      (function  in SCR240.SPR)
*!
*!*************************************************************************
PROCEDURE inicia_general
m.producto = ""
m.linea = ""
SHOW GET m.producto DISABLE
SHOW GET m.linea DISABLE
m.clasifica = 3
selecciona=".T."
RETURN
```

```
*|*************************************************************************
*|    Procedure: CCRED01
*|
*|    Called by: _QUU01FJVQ()     (function  in SCR210.SPR)
*|
*|      Calls: CALCULA_TOTAL     (procedure in SCR210.SPR)
*|           : EST_APERTURA      (procedure in SCR210.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|           : AVGPROD.DBF
*|
*|    CDX files: TARSDOS.CDX
*|
*|*************************************************************************
PROCEDURE ccred01              && CUENTAS DE ALTO RIESGO
PARAMETERS origen

origen = IIF(EMPTY(origen), "tqamc01", origen)

IF !FILE("tqamc01.dbf")
  SELECT DISTINCT sdo_numcta AS cuenta;
    FROM tarsdos , avgprcd  ;
    WHERE sdo_fecapt IN (&ul12meses) AND ;
    sdo_nummv > 2 AND ;
    sdo_cveprd = producto AND;
    sdo_sdovdo > media ;
    INTO TABLE tqamc01
ENDIF

DO calcula_total WITH origen
DO est_apertura WITH origen

RETURN

*|*************************************************************************
*|
*|    Procedure: CCRED02
*|
*|    Called by: _QUU01FJVQ()     (function  in SCR210.SPR)
*|
*|      Calls: CALCULA_TOTAL     (procedure in SCR210.SPR)
*|           : EST_APERTURA      (procedure in SCR210.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|
*|    CDX files: TARSDOS.CDX
*|
*|*************************************************************************
PROCEDURE ccred02                              && CUENTAS SIN PAGO ALGUNO
PARAMETER origen

origen = IIF(EMPTY(origen), "tqamc02", origen)

IF !FILE("tqamc02.dbf")
  SELECT DISTINCT sdo_numcta AS cuenta;
    FROM tarsdos ;
    WHERE sdo_fecapt IN (&ul12meses) AND ;
    EMPTY(sdo_fecpag) AND;
    sdo_sdoact > 0;
    INTO TABLE tqamc02
ENDIF
```

```
DO calcula_total WITH origen
DO est_apertura WITH origen

RETURN


*|*****************************************************************************
*|
*|    Procedure: CCRED03
*|
*|    Called by: _QUU01FJVQ()      (function  in SCR210.SPR)
*|
*|       Calls: CALCULA_TOTAL      (procedure in SCR210.SPR)
*|            : EST_OPERADOR      (procedure in SCR210.SPR)
*|
*|       Uses: TARSDOS.DBF         Alias: SALDOS
*|            : AVGPROD.DBF
*|            : TARALTAS.DBF       Alias: ALTAS
*|            : TQAMC01.DBF
*|            : TQAMC02.DBF
*|
*|    CDX files: TARSDOS.CDX
*|            : TARALTAS.CDX
*|
*|*****************************************************************************
PROCEDURE ccred03                          && OPERADORES & CALIDAD CREDITO
PARAMETERS origen

origen = IIF(EMPTY(origen), "tqamc03", origen)

IF !FILE("tqamc01.DBF")
  SELECT DISTINCT sdo_numcta AS cuenta;
    FROM tarsdos , avgprod  ;
    WHERE sdo_fecapt IN (&ul12meses) AND ;
    sdo_nummv > 2 AND ;
    sdo_cveprd = producto AND;
    sdo_sdovdo > media ;
    INTO TABLE tqamc01
ENDIF

IF !FILE("tqamc02.DBF")
  SELECT DISTINCT sdo_numcta AS cuenta;
    FROM tarsdos ;
    WHERE sdo_fecapt IN (&ul12meses) AND ;
    EMPTY(sdo_fecpag) AND;
    sdo_sdoact > 0;
    INTO TABLE tqamc02
ENDIF

IF !FILE("tqamc03.dbf")
  SELECT DISTINCT alt_numope, alt_numcta AS cuenta;
    FROM taraltas, tqamc01, tqamc02;
    WHERE alt_numcta = tqamc01.cuenta  OR ;
    alt_numcta = tqamc02.cuenta ;
    INTO TABLE tqamc03
ENDIF

DO calcula_total WITH origen
DO est_operador WITH origen
RETURN
```

```
*|*************************************************************************
*|
*|    Procedure: ASGLIM01
*|
*|    Called by: _QUU01FJVQ()      (function  in SCR210.SPR)
*|
*|      Calls: CALCULA_TOTAL       (procedure in SCR210.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|         : TARPROD.DBF           Alias: PRODUCTO
*|         : &ORIGEN
*|
*|    CDX files: TARSDOS.CDX
*|         : TARPROD.CDX
*|
*|*************************************************************************
PROCEDURE asglim01                          && LIMITES FUERA DE PARAMETROS
PARAMETERS origen

origen = IIF(EMPTY(origen), "tqamc05", origen)

IF IFILE("tqamc05.dbf")                             && falta porcentaje
  SELECT DISTINCT sdo_numcta AS cuenta;
    FROM tarsdos, tarprod ;
    WHERE sdo_cveprd = prd_numprd AND ;
    sdo_fecapt <= prd_fecacl AND ;
    sdo_limcre > prd_limmax AND ;
    EMPTY(sdo_fecacr) ;
    INTO TABLE tqamc05
ENDIF

DO calcula_total WITH origen

SELECT sdo_limcre, ;
  COUNT(sdo_numcta) AS n_ctas,;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM &origen, tarsdos ;
  WHERE cuenta = sdo_numcta ;
  INTO CURSOR eqamc05

SHOW GETS
BROWSE FIELDS sdo_limcre:10 :p = "999,999,999", ;
  n_ctas:   8 :p = "999,999", ;
  sdo_vdo:  12 :p = "999,999,999", ;
  sdo_act:  12 :p = "999,999,999" ;
  WINDOW mancred
RETURN

*|*************************************************************************
*|    Procedure: ASGLIM02
*|
*|    Called by: _QUU01FJVQ()      (function  in SCR210.SPR)
*|
*|      Calls: CALCULA_TOTAL       (procedure in SCR210.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|         : TARINLIM.DBF          Alias: INCREMENTOS
*|         : TARBOLET.DBF          Alias: BOLETIN
*|         : &ORIGEN
```

```
*|
*|    CDX files: TARSDOS.CDX
*|           : TARINLIM.CDX
*|           : TARBOLET.CDX
*|
*|*************************************************************************
PROCEDURE asglim02                          && ASIGNACION ERRONEA
PARAMETERS origen

origen = IIF(EMPTY(origen), "tqamc04", origen)

IF !FILE("tqamc04.dbf")
  SELECT DISTINCT sdo_numcta AS cuenta, ili_numope ;
    FROM tarsdos, tarinlim, tarbolet ;
    WHERE sdo_numcta = ili_numcta AND ;
    sdo_numcta = bol_numcta AND;
    ( sdo_fecacr >= bol_fecbod OR ;
    sdo_fecacr >= bol_fecbop OR ;
    EMPTY(sdo_fecpag) ) ;
    INTO TABLE tqamc04
ENDIF

DO calcula_total WITH origen

SELECT ili_numope AS operador, ;
  COUNT(sdo_numcta) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act;
  FROM &origen, tarsdos ;
  WHERE cuenta = sdo_numcta AND;
  &selecciona ;
  GROUP BY ili_numope ;
  INTO CURSOR eqamc04

SHOW GETS
BROWSE FIELDS operador  :10 , ;
  n_ctas:  8 :p = "999,999", ;
  sdo_vdo: 12 :p = "999,999,999", ;
  sdo_act: 12 :p = "999,999,999" ;
  WINDOW mancred

RETURN

*|*************************************************************************
*|
*|    Procedure: ASGLIM03
*|
*|    Called by: _QUU01FJVQ()     (function  in SCR210.SPR)
*|
*|       Calls: CALCULA_TOTAL     (procedure in SCR210.SPR)
*|            : EST_OPERADOR     (procedure in SCR210.SPR)
*|
*|        Uses: TARSDOS.DBF         Alias: SALDOS
*|            : TARPROD.DBF        Alias: PRODUCTO
*|            : TQAMC05.DBF
*|            : TARALTAS.DBF       Alias: ALTAS
*|
*|    CDX files: TARSDOS.CDX
*|           : TARPROD.CDX
*|           : TARALTAS.CDX
*|
```

```
*|*************************************************************************
PROCEDURE asglim03                          && ASIGNACION INICIAL ERRONEA
PARAMETERS origen

origen = IIF(EMPTY(origen), "tqamc06", origen)

IF IFILE("tqamc05.dbf")
  SELECT DISTINCT sdo_numcta AS cuenta;
    FROM tarsdos, tarprod ;
    WHERE sdo_cveprd = prd_numprd AND ;
    sdo_fecapt <= prd_fecacl AND ;
    sdo_limcre > prd_limmax AND ;
    EMPTY(sdo_fecacr) ;
    INTO TABLE tqamc05
ENDIF

IF IFILE("tqamc06.dbf")
  SELECT alt_numope, alt_numcta AS cuenta;
    FROM tqamc05, taraltas ;
    WHERE cuenta = alt_numcta ;
    INTO TABLE tqamc06
ENDIF

DO calcula_total WITH origen
DO est_operador WITH origen

RETUVN

*|*************************************************************************
*|
*|    Procedure: ASGLIM04
*|
*|    Called by: _QUU01FJVQ()     (function  in SCR210.SPR)
*|
*|       Calls: CALCULA_TOTAL     (procedure in SCR210.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|           : TARBOLET.DBF         Alias: BOLETIN
*|           : TARINLIM.DBF         Alias: INCREMENTOS
*|           : &ORIGEN
*|
*|    CDX files: TARSDOS.CDX
*|           : TARBOLET.CDX
*|           : TARINLIM.CDX
*|
*|*************************************************************************
PROCEDURE asglim04                          && ASIGNACION AUTOMATIZADA
PARAMETERS origen

origen = IIF(EMPTY(origen), "tqamc09", origen)

IF !EMPTY("tqamc09.DBF")
  SELECT sdo_numcta AS cuenta;
    FROM tarsdos, tarboiet ;
    WHERE  sdo_numcta = bol_numcta AND ;
    sdo_numcta NOT IN (SELECT lil_numcta FROM tarinlim ) AND ;
    (sdo_fecacr >=  bol_fecbod OR sdo_fecacr >= bol_fecbop OR sdo_fecpag =0) AND;
    !EMPTY(sdo_fecacr);
    INTO TABLE tqamc09
ENDIF
```

```
DO calcula_total WITH origen


SELECT sdo_limcre, ;
  COUNT(sdo_numcta) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM tarsdos, &origen ;
  WHERE sdo_numcta = cuenta AND;
  &seleccIona ;
  GROUP BY sdo_limcre ;
  INTO CURSOR eqamc09

SHOW GETS
BROWSE FIELDS sdo_limcre:10 :p = "999,999,999", ;
  n_ctas:   8 :p = "999,999", ;
  sdo_vdo:  12 :p = "999,999,999", ;
  sdo_act:  12 :p = "999,999,999" ;
  WINDOW mancred

RETUVN


*!*************************************************************************
*!
*!    Procedure: MANCTA01
*!
*!    Called by: _QUU01FJVQ()     (function  in SCR210.SPR)
*!
*!       Calls: CALCULA_TOTAL     (procedure in SCR210.SPR)
*!            : EST_APERTURA      (procedure in SCR210.SPR)
*!
*!        Uses: TARSDOS.DBF       Alias: SALDOS
*!
*!    CDX files: TARSDOS.CDX
*!
*!*************************************************************************
PROCEDURE mancta01                          && SATURACION DE CREDITO
PARAMETERS origen

origen = IIF(EMPTY(origen), "tqamc07", origen)

IF !FILE("tqamc07.DBF")
  SELECT sdo_numcta AS cuenta;
    FROM tarsdos ;
    WHERE sdo_fecapt IN (&ul12meses) AND ;
    sdo_sdoact >= (sdo_limcre * 0.8) ;
    INTO TABLE tqamc07
ENDIF

DO calcula_total WITH origen
DO est_apertura WITH origen

RETURN

*!*************************************************************************
*!
*!    Procedure: MANCTA02
*!
*!    Called by: _QUU01FJVQ()     (function  in SCR210.SPR)
*!
```

```
*|      Calls: CALCULA_TOTAL     (procedure in SCR210.SPR)
*|
*|      Uses: TARSDOS.DBF        Alias: SALDOS
*|        : &ORIGEN
*|
*|   CDX files: TARSDOS.CDX
*|
*|*****************************************************************************
PROCEDURE mancta02                          && SALDO VENCIDO SUPERIOR AL LIM. CREDITO
PARAMETERS origen

origen = IIF(EMPTY(origen), "tqamc08", origen)

IF !FILE("tqamc08.DBF")
  SELECT sdo_numcta AS cuenta ;
    FROM tarsdos ;
    WHERE sdo_sdovdo > sdo_limcre ;
    INTO TABLE tqamc08
ENDIF

DO calcula_total WITH origen

SELECT sdo_sdovdo,;
  COUNT(sdo_numcta) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo,;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM tarsdos, &origen ;
  WHERE sdo_numcta = cuenta AND ;
  &selecciona ;
  GROUP BY sdo_sdovdo ;
  INTO CURSOR eqamc08

SHOW GETS
BROWSE FIELDS sdo_sdovdo:10 :p = "999,999,999", ;
  n_ctas:   8 :p = "999,999", ;
  sdo_vdo:  12 :p = "999,999,999", ;
  sdo_act:  12 :p = "999,999,999" ;
  WINDOW mancred

RETURN

*********
PROCEDURE reportes
PARAMETER view_print, dborigen, namereport

DO CASE
CASE namereport = "opecal" OR namereport = "limlerr"

  SELECT DISTINCT * ;
    FROM tarprod, tarsdos, taraltas, &dborigen ;
    WHERE sdo_numcta = cuenta AND ;
    sdo_cveprd = prd_numprd AND ;
    alt_numcta = cuenta AND ;
    &selecciona ;
    INTO CURSOR creport

CASE namereport = "limerr"

  SELECT DISTINCT * ;
    FROM tarprod, tarsdos, tarinlim, &dborigen ;
    WHERE sdo_numcta = cuenta AND ;
```

```
    sdo_cveprd = prd_numprd AND ;
    sdo_numcta = lii_numcta AND ;
    &seleccIona ;
    INTO CURSOR creport

OTHERWISE

  SELECT DISTINCT * ;
    FROM tarprod, tarsdos, &dborigen ;
    WHERE sdo_numcta = cuenta AND ;
    sdo_cveprd = prd_numprd AND ;
    &seleccIona ;
    INTO CURSOR creport

ENDCASE

REPORT FORM &namereport &view_print

RETURN

******
PROCEDURE calcula_total
PARAMETER origen

SELECT COUNT(*) AS tctas, ;
  SUM(sdo_sdovdo) AS tsdovdo, ;
  SUM(sdo_sdoact) AS tsdoact ;
  FROM tarsdos, &origen ;
  WHERE sdo_numcta = cuenta AND ;
  &seleccIona ;
  INTO CURSOR ctotal

SCATTER MEMVAR

RETURN

*!*************************************************************************
*!
*!     Procedure: EST_APERTURA
*!
*!     Called by: CCRED01          (procedure in SCR210.SPR)
*!              : CCRED02          (procedure in SCR210.SPR)
*!              : MANCTA01         (procedure in SCR210.SPR)
*!
*!          Uses: &ORIGEN
*!              : TARSDOS.DBF          Alias: SALDOS
*!
*!     CDX files: TARSDOS.CDX
*!
*!*************************************************************************
PROCEDURE est_apertura
PARAMETERS origen

SELECT sdo_fecapt AS apertura, ;
  COUNT(*) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM  &origen , tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
  &seleccIona ;
  GROUP BY sdo_fecapt ;
```

```
  INTO CURSOR eqamc01

SHOW GETS
BROWSE FIELDS apertura :10 :p = "99/99", ;
  n_ctas:   8 :p = "999,999", ;
  sdo_vdo:  12 :p = "999,999,999", ;
  sdo_act:  12 :p = "999,999,999" ;
  WINDOW mancred

RETUVN

***
*|*************************************************************************
*|
*|    Procedure: EST_OPERADOR
*|
*|    Called by: CCRED03          (procedure in SCR210.SPR)
*|             : ASGLIM03         (procedure in SCR210.SPR)
*|
*|       Uses: &ORIGEN
*|             : TARSDOS.DBF          Alias: SALDOS
*|
*|    CDX files: TARSDOS.CDX
*|
*|*************************************************************************

PROCEDURE est_operador
PARAMETERS origen

SELECT alt_numope AS operador, ;
  COUNT(sdo_numcta) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM &origen, tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
  &selecciona ;
  GROUP BY alt_numope ;
  INTO CURSOR eqamc03

SHOW GETS
BROWSE FIELDS operador:10 , ;
  n_ctas:   8 :p = "999,999", ;
  sdo_vdo:  12 .p = "999,999,999", ;
  sdo_act:  12 :p = "999,999,999" ;
  WINDOW mancred

RETURN

*     ************************************************************
*     *
*     * _QUU01FJ89         m.opcred VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR210,    Record Number:  11
*     * Variable:           m.opcred
*     * Called By:          VALID Clause
*     * Snippet Number:     1
*     *
*     ************************************************************
```

```
*|*****************************************************************************
*|
*|    Function: _QUU01FJ89
*|
*|    Called by: SCR210.SPR
*|
*|       Calls: INICIA_GENERAL    (procedure in SCR210.SPR)
*|
*|*****************************************************************************
FUNCTION _quu01fj89    && m.opcred VALID
#REGION 1
arrsubcred=""
DO CASE
CASE opcred = 1
  =ACOPY(arrcred1,arrsubcred)
CASE opcred = 2
  =ACOPY(arrcred2,arrsubcred)
CASE opcred = 3
  =ACOPY(arrcred3,arrsubcred)
ENDCASE

DIMENSION arrsubcred[ALEN(arrsubcred,1)]

DO inicia_general
SHOW GET m.subcred ENABLE
SHOW GET m.botcred DISABLE
SHOW GET m.botcred,5 ENABLE
SHOW GETS
_CUROBJ = OBJNUM(m.subcred)

*    ************************************************************
*    *
*    * _QUU01FJHY         m.subcred VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR210,    Record Number:  12
*    * Variable:           m.subcred
*    * Called By:          VALID Clause
*    * Snippet Number:     2
*    *
*    ************************************************************
*
*|*****************************************************************************
*|
*|    Function: _QUU01FJHY
*|
*|    Called by: SCR210.SPR
*|
*|       Calls: INICIA_GENERAL    (procedure in SCR210.SPR)
*|
*|*****************************************************************************
FUNCTION _quu01fjhy    && m.subcred VALID
#REGION 1
DO inicia_general
SHOW GET m.clasifica ENABLE

SHOW GET m.botcred DISABLE
SHOW GET m.botcred,1 ENABLE
SHOW GET m.botcred,2 ENABLE
```

```
SHOW GET m.botcred,5 ENABLE

DO CASE
CASE der = "D"
  SHOW GET m.botcred,4 ENABLE
CASE der = "Q"
  SHOW GET m.botcred,3 ENABLE
CASE der = "B"
  SHOW GET m.botcred,3 ENABLE
  SHOW GET m.botcred,4 ENABLE
ENDCASE


_CUROBJ = OBJNUM(m.clasifica)

*     ************************************************************
*     *
*     * _QUU01FJO4          m.clasifica WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR210,     Record Number:  13
*     * Variable:            m.clasifica
*     * Called By:           WHEN Clause
*     * Snippet Number:      3
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!     Function: _QUU01FJO4
*!
*!     Called by: SCR210.SPR
*!
*!**************************************************************************
FUNCTION _quu01fjo4     && m.clasifica WHEN
#REGION 1
selecciona = ".T."



*     ************************************************************
*     *
*     * _QUU01FJQJ          m.clasifica VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR210,     Record Number:  13
*     * Variable:            m.clasifica
*     * Called By:           VALID Clause
*     * Snippet Number:      4
*     *
*     ************************************************************


*!**************************************************************************
*!
*!     Function: _QUU01FJQJ
*!
```

```
*|    Called by: SCR210.SPR
*|
*|       Calls: INICIA_GENERAL    (procedure in SCR210.SPR)
*|
*|*****************************************************************************
FUNCTION _quu01fjqj    && m.clasifica VALID
#REGION 1
DO CASE
CASE m.clasifica = 1
  m.linea=""
  SHOW GET m.linea DISABLE
  SHOW GET m.producto ENABLE
  _CUROBJ = OBJNUM(m.producto)
CASE m.clasifica = 2
  m.producto=""
  SHOW GET m.producto DISABLE
  SHOW GET m.linea ENABLE
  _CUROBJ = OBJNUM(m.linea)
CASE m.clasifica = 3
  DO inicia_general
  _CUROBJ = OBJNUM(m.botcred)
ENDCASE
SHOW GETS

*     ************************************************
*     *
*     * _QUU01FJVQ         m.botcred VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR210,    Record Number:  14
*     * Variable:           m.botcred
*     * Called By:          VALID Clause
*     * Snippet Number:     5
*     *
*     ************************************************
*
*|*****************************************************************************
*|
*|    Function: _QUU01FJVQ
*|
*|    Called by: SCR210.SPR
*|
*|       Calls: CCRED01          (procedure in SCR210.SPR)
*|            : CCRED02          (procedure in SCR210.SPR)
*|            : CCRED03          (procedure in SCR210.SPR)
*|            : ASGLIM01         (procedure in SCR210.SPR)
*|            : ASGLIM02         (procedure in SCR210.SPR)
*|            : ASGLIM03         (procedure in SCR210.SPR)
*|            : ASGLIM04         (procedure in SCR210.SPR)
*|            : MANCTA01          (procedure in SCR210.SPR)
*|            : MANCTA02          (procedure in SCR210.SPR)
*|            : SCR320.SPR
*|            : &PROCESO.PRG
*|            : REPORTES          (procedure in SCR210.SPR)
*|            : SCRPRN.SPR
*|
*|*****************************************************************************
FUNCTION _quu01fjvq    && m.botcred VALID
#REGION 1
```

```
DO CASE
CASE m.botcred = 1                    && ESTRATIFICADO
  m.tctas = 0
  m.tsdovdo = 0
  m.tsdoact = 0
  SHOW GETS

  DO CASE
  CASE opcred =1          && CALIDAD DE CREDITO

    DO CASE
    CASE subcred = 1      && Cuentas de alto riesgo
      DO ccred01
      namereport = "altorsgo"
      dbfuente =  "tqamc01"
      dbdestino = "tcamc01"
      proceso = "ccred01"

    CASE subcred = 2      && Cuentas sin pago
      DO ccred02
      namereport = "sinpago"
      dbfuente =  "tqamc02"
      dbdestino = "tcamc02"
      proceso = "ccred02"

    CASE subcred = 3      && Operador/Calidad de credito
      DO ccred03
      namereport = "opecal"
      dbfuente =  "tqamc03"
      dbdestino = "tcamc03"
      proceso = "ccred03"

    ENDCASE

  CASE opcred =2                      && ASIGNACION DE LIMITES DE CREDITO
    DO CASE
    CASE subcred = 1                  && Limites fuenra de parametros
      DO asglim01
      namereport = "limfpar"
      dbfuente =  "tqamc05"
      dbdestino = "tcamc05"
      proceso = "asglim01"

    CASE subcred = 2                  && Asignacion erronea
      DO asglim02
      namereport = "limerr"
      dbfuente =  "tqamc04"
      dbdestino = "tcamc04"
      proceso = "asglim02"

    CASE subcred = 3                  && Asignacion inicical erronea
      DO asglim03
      namereport = "limierr"
      dbfuente =  "tqamc06"
      dbdestino = "tcamc06"
      proceso = "asglim03"

    CASE subcred = 4      && Asignacion automatizada
      DO asglim04
      namereport = "limaut"
      dbfuente =  "tqamc09"
```

```
      dbdestino = "tcamc09"
      proceso = "asglim04"

    ENDCASE

  CASE opcred =3                        && MANEJO DE CUENTA
    DO CASE
    CASE subcred = 1              && Saturacion de credito
      DO mancta01
      namereport = "sataper"
      dbfuente =  "tqamc07"
      dbdestino = "tcamc07"
      proceso = "mancta01"

    CASE subcred = 2
      DO mancta02     && Saldo vencido > limite de credito
      namereport = "sdossuper"
      dbfuente =  "tqamc08"
      dbdestino = "tcamc08"
      proceso = "mancta02"

    ENDCASE

  ENDCASE
  est_cons = dbfuente

CASE m.botcred = 2                      && GRAFICAS

  DO (_GENGRAPH)

CASE m.botcred = 3                      && CONSULTAS
  m.tctas = 0
  m.tsdovdo = 0
  m.tsdoact = 0
  DO scr320.spr WITH dbfuente, dbdestino, 1
  DO &proceso WITH dbdestino
  est_cons = dbdestino

CASE m.botcred = 4                      && VER

  DO reportes WITH "PREVIEW", est_cons, namereport
  DO scrprn.spr

CASE m.botcred = 5                      && SALIR
  CLEAR READ
ENDCASE

*     ************************************************************
*     *
*     * _QUU01FKCH         m.producto WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR210,     Record Number:  15
*     * Variable:           m.producto
*     * Called By:          WHEN Clause
*     * Snippet Number:     6
*     *
*     ************************************************************
```

```
*|*****************************************************************************
*|
*|     Function: _QUU01FKCH
*|
*|     Called by: SCR210.SPR
*|
*|        Calls: ARREGLO_PRODUCTO   (procedure in TMENU.MPR)
*|
*|*****************************************************************************
FUNCTION _quu01fkch     && m.producto WHEN
#REGION 1
DO arreglo_producto
m.producto = 0

*     ************************************************************
*     * _QUU01FKEV          m.producto VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR210,     Record Number:  15
*     * Variable:            m.producto
*     * Called By:           VALID Clause
*     * Snippet Number:      7
*     *
*     ************************************************************

*|*****************************************************************************
*|
*|     Function: _QUU01FKEV
*|
*|     Called by: SCR210.SPR
*|
*|*****************************************************************************
FUNCTION _quu01fkev     && m.producto VALID
#REGION 1
selecciona = "sdo_cveprd = " + ALLTRIM( STR(arrproducto(m.producto,2)) )
_CUROBJ = OBJNUM(m.botcred)

*     ************************************************************
*     *
*     * _QUU01FKI0          m.linea WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR210,     Record Number:  16
*     * Variable:            m.linea
*     * Called By:           WHEN Clause
*     * Snippet Number:      8
*     *
*     ************************************************************
*|*****************************************************************************
*|
*|     Function: _QUU01FKI0
*|
*|     Called by: SCR210.SPR
*|
*|        Calls: ARREGLO_LINEA     (procedure in TMENU.MPR)
*|
*|*****************************************************************************
```

```
FUNCTION _quu01fkl0     && m.linea WHEN
#REGION 1
DO arreglo_linea
m.linea = 0

*     ************************************************************
*     *
*     * _QUU01FKKE          m.linea VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR210,    Record Number:  16
*     * Variable:            m.linea
*     * Called By:           VALID Clause
*     * Snippet Number:      9

*!*************************************************************************
*!
*!     Function: _QUU01FKKE
*!
*!     Called by: SCR210.SPR
*!
*!        Uses: TARPROD.DBF          Alias: PRODUCTO
*!
*!     CDX files: TARPROD.CDX
*!
*!*************************************************************************
FUNCTION _quu01fkke     && m.linea VALID
#REGION 1
PRIVATE expresion,i, arrprdlin

SELECT prd_numprd ;
  FROM  tarprod ;
  WHERE  prd_numlin = arrlinea(m.linea) ;
  INTO ARRAY arrprdlin

IF ALEN(arrprdlin) = 1
  expresion = "sdo_numprd = " + ALLTRIM( STR(arrprdlin[1]) )
ELSE
  expresion = ALLTRIM( STR(arrprdlin[1]) )
  FOR i=2 TO ALEN(arrprdlin)
    expresion = expresion + "," + ALLTRIM( STR(arrprdlin[i]) )
  ENDFOR
  expresion = "sdo_cveprd IN (" + expresion + ")"
ENDIF

selecciona = expresion
_CUROBJ = OBJNUM(m.botcred)

*     ************************************************************
*     *
*     * _QUU01FKVD          Read Level When
*     *
*     * Function Origin:
*     *
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR210
*     * Called By:           READ Statement
*     * Snippet Number:      10
```

```
*!*************************************************************************
*!
*!      Function: _QUU01FKVD
*!
*!      Called by: SCR210.SPR
*!
*!*************************************************************************
FUNCTION _quu01fkvd     && Read Level When
* When Code from screen: SCR210
*
#REGION 1
der = SUFSTR(privilegios,5,1)
SHOW GET m.botcred DISABLE
SHOW GET m.botcred,5 ENABLE
*: EOF: SCR210.SPR
*     ************************************************************
*     *
*     * 11/04/94          SCR220.SPR          00:40:14
*     *
*     ************************************************************


*     ************************************************************
*     *
*     *       SCR220/Windows Setup Code - SECTION 2
*     *
*     ************************************************************
*

#REGION 1
PUBLIC selecciona

m.tctas = 0
m.tsdoact = 0
m.tsdovdo = 0
dbfuente = ""
dbdestino = ""
namereport = ""
proceso = ""
est_cons = ""
origen = ""


DIMENSION arrproducto[1,2], arrlinea[1]

DEFINE WINDOW sobregiros FROM 9,42 TO 30,86;
  TITLE "Sobregiros" ;
  FONT "MS Serif",8 ;
  STYLE "B" ;
  FLOAT;
  GROW

arrproducto=""
arrlinea=""

m.radsob = 0
m.radsdo = 0
m.radrec = 0


*     ************************************************************
```

```
*    *
*    * SCR220/Windows Supporting Procedures and Functions
*    *
*    ************************************************
*

#REGION 1
*|*****************************************************************************
*|
*|    Procedure: INICIA_GENERAL
*|
*|    Called by: _QUU01FJ89()     (function  in SCR210.SPR)
*|          : _QUU01FJHY()     (function  in SCR210.SPR)
*|          : _QUU01FJQJ()     (function  in SCR210.SPR)
*|          : _QUU01FRX6()     (function  in SCR220.SPR)
*|          : _QUU01FSG0()     (function  in SCR220.SPR)
*|          : _QUU01FZR6()     (function  in SCR230.SPR)
*|          : _QUU01G015()     (function  in SCR230.SPR)
*|          : _QUU01G62N()     (function  in SCR240.SPR)
*|
*|*****************************************************************************
PROCEDURE inicia_general
m.producto = ""
m.linea = ""
SHOW GET m.producto DISABLE
SHOW GET m.linea DISABLE
m.clasifica = 3
selecciona=".T."
RETURN


*|*****************************************************************************
*|
*|    Procedure: SOB21
*|
*|    Called by: _QUU01FSZ7()     (function  in SCR220.SPR)
*|
*|       Calls: CTASSOB          (procedure in SCR220.SPR)
*|          : CALCULA_TOTAL    (procedure in SCR210.SPR)
*|          : ESTRATIFICADO     (procedure in SCR220.SPR)
*|
*|       Uses: TQSOB00.DBF
*|          : TARSDOS.DBF          Alias: SALDOS
*|          : AVGSOB.DBF
*|
*|    CDX files: TARSDOS.CDX
*|
*|*****************************************************************************
PROCEDURE sob21          && SOBREGIROS RELEVANTES
PARAMETER origen
PRIVATE baseorigen

origen = IIF(EMPTY(origen), "tqsob01", origen )
DO ctassob

SELECT DISTINCT cuenta, ((sdo_sdoact - sdo_limcre)/sdo_limcre)*100 AS ps;
  FROM tqsob00, tarsdos, avgsob ;
  WHERE sdo_cveprd = avgsob.producto AND ;
  sdo_numcta = cuenta AND ;
  tqsob00.sobreg > medsob ;
  INTO TABLE tqsob01
```

```
DO calcula_total WITH origen
DO estratificado WITH origen

RETURN

*!*************************************************************************
*!
*!    Procedure: SOB22
*!
*!    Called by: _QUU01FSZ7()      (function  in SCR220.SPR)
*!
*!       Calls: CTASSOB          (procedure in SCR220.SPR)
*!            : CALCULA_TOTAL    (procedure in SCR210.SPR)
*!            : ESTRATIFICADO    (procedure in SCR220.SPR)
*!
*!        Uses: TARSDOS.DBF          Alias: SALDOS
*!            : TQSOB00.DBF
*!            : TARBOLET.DBF         Alias: BOLETIN
*!
*!    CDX files: TARSDOS.CDX
*!            : TARBOLET.CDX
*!
*!*************************************************************************
PROCEDURE sob22                                  && SOBREGIROS SIN RECUPERACION DE SALDO
PARAMETER origen
PRIVATE baseorigen

origen = IIF(EMPTY(origen), "tqsob02", origen)
DO ctassob

SELECT DISTINCT cuenta, ((sdo_sdoact - sdo_limcre)/sdo_limcre)*100 AS ps;
  FROM tarsdos, tqsob00, tarbolet ;
  WHERE sdo_numcta = cuenta AND;
  sdo_numcta = bol_numcta AND;
  sdo_fecpag < bol_fecsbg ;
  INTO TABLE tqsob02

DO calcula_total WITH origen
DO estratificado WITH origen

RETURN

*!*************************************************************************
*!    Procedure: SOB23
*!
*!    Called by: _QUU01FSZ7()      (function  in SCR220.SPR)
*!
*!       Calls: CTASSOB          (procedure in SCR220.SPR)
*!            : CALCULA_TOTAL    (procedure in SCR210.SPR)
*!            : ESTRATIFICADO    (procedure in SCR220.SPR)
*!
*!        Uses: TQSOB00.DBF
*!            : TARSDOS.DBF          Alias: SALDOS
*!            : AVGSOB.DBF
*!
*!    CDX files: TARSDOS.CDX
*!*************************************************************************
PROCEDURE sob23                              && SOBREGIROS MOROSOS
PARAMETER origen
PRIVATE baseorigen
```

```
origen = IIF(EMPTY(origen), "tqsob03", origen)
DO ctassob


SELECT DISTINCT cuenta, ((sdo_sdoact - sdo_limcre)/sdo_limcre)*100 AS ps ;
  FROM tqsob00, tarsdos, avgsob ;
  WHERE sdo_cveprd = avgsob.producto AND ;
  sdo_numcta = cuenta AND ;
  tqsob00.sobreg > medsob AND ;
  sdo_fecpag = 0 ;
  INTO TABLE tqsob03

DO calcula_total WITH origen
DO estratificado WITH origen

RETURN

*|*************************************************************************
*|
*|    Procedure: SOB24
*|
*|    Called by: _QUU01FSZ7()     (function  in SCR220.SPR)
*|
*|       Calls: CTASSOB          (procedure in SCR220.SPR)
*|            : CALCULA_TOTAL    (procedure in SCR210.SPR)
*|            : ESTRATIFICADO    (procedure in SCR220.SPR)
*|
*|       Uses: TARSDOS.DBF         Alias: SALDOS
*|            : TQSOB00.DBF
*|            : TARBOLET.DBF       Alias: BOLETIN
*|
*|    CDX files: TARSDOS.CDX
*|            : TARBOLET.CDX
*|
*|*************************************************************************
PROCEDURE sob24                          && SOBREGIROS SIN RECUPERACION MINIMA
PARAMETER origen
PRIVATE baseorigen

origen = IIF(EMPTY(origen), "tqsob04", origen)
DO ctassob


SELECT DISTINCT cuenta, ((sdo_sdoact - sdo_limcre)/sdo_limcre)*100 AS ps ;
  FROM tarsdos, tqsob00, tarbolet ;
  WHERE sdo_numcta = tqsob00.cuenta AND;
  sdo_numcta = bol_numcta AND;
  sdo_mtoint > sdo_mtocap AND;
  sdo_fecpag >= bol_fecsbg ;
  INTO TABLE tqsob04

DO calcula_total WITH origen
DO estratificado WITH origen

RETURN
```

```
*|*************************************************************************
*|
*|    Procedure: CREACURSOR
*|
*|    Called by: ESTRATIFICADO     (procedure in SCR220.SPR)
*|
*|*************************************************************************
PROCEDURE creacursor
CREATE CURSOR rangossob (sobregiro C(11), n_ctas N(6),sdo_vdo N(9) ,sdo_act N(9))
RETURN

*|*************************************************************************
*|
*|    Procedure: CTASSOB
*|
*|    Called by: SOB21          (procedure in SCR220.SPR)
*|          : SOB22          (procedure in SCR220.SPR)
*|          : SOB23          (procedure in SCR220.SPR)
*|          : SOB24          (procedure in SCR220.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|
*|    CDX files: TARSDOS.CDX
*|
*|*************************************************************************
PROCEDURE ctassob

SELECT DISTINCT sdo_numcta AS cuenta, (sdo_sdoact - sdo_limcre) AS sobreg ;
  FROM tarsdos ;
  WHERE (sdo_sdoact - sdo_limcre) > 0 ;
  INTO TABLE tqsob00

RETURN

*|*************************************************************************
PROCEDURE estratificado
PARAMETER origen

DIMENSION arr010[4], arr1030[4], arr3050[4], arr50100[4], arr100150[4], arrmas150[4]
arr010=0
arr1030=0
arr3050=0
arr50100=0
arr100150=0
arrmas150=0

arr010[1]= "  0% -  10%"
arr1030[1]= " 10% -  30%"
arr3050[1]= " 30% -  50%"
arr50100[1]= " 50% - 100%"
arr100150[1]= "100% - 150%"
arrmas150[1]= "MAYOR 150%"


SELECT "  0% -  10%" AS sobregiro, ;
  COUNT(*) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM  &origen , tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
```

```
  ps > 30 AND ;
  ps <= 50 AND ;
  &selecciona ;
  INTO ARRAY arr010

SELECT " 10% -  30%" AS sobregiro, ;
  COUNT(*) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM  &origen , tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
  ps > 30 AND ;
  ps <= 50 AND ;
  &selecciona ;
  INTO ARRAY arr1030

SELECT " 30% -  50%" AS sobregiro, ;
  COUNT(*) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM  &origen , tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
  ps > 30 AND ;
  ps <= 50 AND ;
  &selecciona ;
  INTO ARRAY arr3050

SELECT " 50% - 100%" AS sobregiro, ;
  COUNT(*) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM  &origen , tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
  ps > 50 AND ;
  ps <= 100 AND;
  &selecciona ;
  INTO ARRAY arr50100

SELECT "100% - 150%" AS sobregiro, ;
  COUNT(*) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM  &origen , tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
  ps > 100 AND ;
  ps <= 150 AND;
  &selecciona ;
  INTO ARRAY arr100150

SELECT "MAYOR 150%" AS sobregiro, ;
  COUNT(*) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM  &origen , tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
  ps > 150 AND ;
  &selecciona ;
  INTO ARRAY arrmas150

DO creacursor
```

```
APPEND BLANK
GATHER MEMVAR FROM arr010
APPEND BLANK
GATHER MEMVAR FROM arr1030
APPEND BLANK
GATHER MEMVAR FROM arr3050
APPEND BLANK
GATHER MEMVAR FROM arr50100
APPEND BLANK
GATHER MEMVAR FROM arr100150
APPEND BLANK
GATHER MEMVAR FROM arrmas150

SHOW GETS
BROWSE FIELDS ;
  sobregiro:10, ;
  n_ctas:    8 :p = "999,999", ;
  sdo_vdo:  12 :p = "999,999,999", ;
  sdo_act:  12 :p = "999,999,999" ;
  WINDOW sobregiros

RETURN

*|*****************************************************************************
PROCEDURE reportes
PARAMETER view_print, dborigen, namereport

SELECT DISTINCT * ;
  FROM tarprod, tarsdos, &dborigen ;
  WHERE sdo_numcta = cuenta AND ;
  sdo_cveprd = prd_numprd AND ;
  &selecciona ;
  INTO CURSOR creport

REPORT FORM &namereport &view_print

RETURN

*|*****************************************************************************
PROCEDURE calcula_total
PARAMETER origen

SELECT COUNT(*) AS tctas, ;
  SUM(sdo_sdovdo) AS tsdovdo, ;
  SUM(sdo_sdoact) AS tsdoact ;
  FROM tarsdos, &origen ;
  WHERE sdo_numcta = cuenta AND ;
  &selecciona ;
  INTO CURSOR ctotal

SCATTER MEMVAR

RETURN

*     ************************************************************
*     *
*     * _QUU01FRX6         m.radsob VALID
*     *
*     * Function Origin:
```

```
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR220,    Record Number:  18
*    * Variable:           m.radsob
*    * Called By:          VALID Clause
*    * Snippet Number:     1
*    *
*    ************************************************

*!*************************************************************************
*!
*!    Function: _QUU01FRX6
*!
*!    Called by: SCR220.SPR
*!
*!       Calls: INICIA_GENERAL    (procedure in SCR210.SPR)
*!
*!*************************************************************************
FUNCTION _quu01frx6    && m.radsob VALID
#REGION 1
DO inicia_general
IF m.radsob = 1
  m.radrec = 0
  SHOW GET m.radrec DISABLE
  SHOW GET m.radsdo ENABLE
  SHOW GETS
  _CUROBJ = OBJNUM(m.radsdo)
ELSE
  m.radsdo = 0
  SHOW GET m.radsdo DISABLE
  SHOW GET m.radrec ENABLE
  SHOW GETS
  _CUROBJ = OBJNUM(m.radrec)
ENDIF


*    ************************************************
*    *
*    * _QUU01FS35          m.radsdo VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR220,    Record Number:  19
*    * Variable:           m.radsdo
*    * Called By:          VALID Clause
*    * Snippet Number:     2
*    *
*    ************************************************

*!*************************************************************************
*!
*!    Function: _QUU01FS35
*!
*!    Called by: SCR220.SPR
*!
*!*************************************************************************
FUNCTION _quu01fs35    && m.radsdo VALID
#REGION 1
SHOW GET m.clasifica ENABLE
```

```
SHOW GET m.botsob DISABLE
SHOW GET m.botsob,1 ENABLE
SHOW GET m.botsob,2 ENABLE
SHOW GET m.botsob,5 ENABLE

DO CASE
CASE der = "D"
  SHOW GET m.botsob,4 ENABLE
CASE der = "Q"
  SHOW GET m.botsob,3 ENABLE
CASE der = "B"
  SHOW GET m.botsob,3 ENABLE
  SHOW GET m.botsob,4 ENABLE
ENDCASE

_CUROBJ = OBJNUM(m.clasifica)

*     ************************************************************
*     *
*     * _QUU01FS8J         m.radrec VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR220,    Record Number:  20
*     * Variable:           m.radrec
*     * Called By:          VALID Clause
*     * Snippet Number:     3
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FS8J
*!
*!     Called by: SCR220.SPR
*!
*!*************************************************************************
FUNCTION _quu01fs8j     && m.radrec VALID
#REGION 1
SHOW GET m.clasifica ENABLE
SHOW GET m.botsob DISABLE
SHOW GET m.botsob,1 ENABLE
SHOW GET m.botsob,2 ENABLE
SHOW GET m.botsob,5 ENABLE

DO CASE
CASE der = "D"
  SHOW GET m.botsob,4 ENABLE
CASE der = "Q"
  SHOW GET m.botsob,3 ENABLE
CASE der = "B"
  SHOW GET m.botsob,3 ENABLE
  SHOW GET m.botsob,4 ENABLE
ENDCASE

_CUROBJ = OBJNUM(m.clasifica)

*     ************************************************************
*     *
*     * _QUU01FSDO         m.clasifica WHEN
```

```
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR220,    Record Number:  21
*     * Variable:          m.clasifica
*     * Called By:          WHEN Clause
*     * Snippet Number:     4
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!     Function: _QUU01FSDO
*!
*!     Called by: SCR220.SPR
*!
*!**************************************************************************
FUNCTION _quu01fsdo     && m.clasifica WHEN
#REGION 1
selecciona = ".T."

*     ************************************************************
*     * _QUU01FSG0          m.clasifica VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR220,    Record Number:  21
*     * Variable:          m.clasifica
*     * Called By:          VALID Clause
*     * Snippet Number:     5
*     *
*     ************************************************************

*!**************************************************************************
*!     Function: _QUU01FSG0
*!
*!     Called by: SCR220.SPR
*!
*!        Calls: INICIA_GENERAL    (procedure in SCR210.SPR)
*!
*!**************************************************************************
FUNCTION _quu01fsg0     && m.clasifica VALID
#REGION 1
DO CASE
CASE m.clasifica = 1
  m.linea=""
  SHOW GET m.linea DISABLE
  SHOW GET m.producto ENABLE
  _CUROBJ = OBJNUM(m.producto)
CASE m.clasifica = 2
  m.producto=""
  SHOW GET m.producto DISABLE
  SHOW GET m.linea ENABLE
  _CUROBJ = OBJNUM(m.linea)
CASE m.clasifica = 3
  DO inicia_general
  _CUROBJ = OBJNUM(m.botsob)
ENDCASE
SHOW GETS
```

```
*     ************************************************************
*     *
*     * _QUU01FSKZ          m.producto WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR220,    Record Number:  22
*     * Variable:            m.producto
*     * Called By:           WHEN Clause
*     * Snippet Number:      6
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FSKZ
*!
*!     Called by: SCR220.SPR
*!
*!        Calls: ARREGLO_PRODUCTO   (procedure in TMENU.MPR)
*!
*!*************************************************************************
FUNCTION _quu01fskz    && m.producto WHEN
#REGION 1
DO arreglo_producto
m.producto = 0

*     ************************************************************
*     *
*     * _QUU01FSNE          m.producto VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR220,    Record Number:  22
*     * Variable:            m.producto
*     * Called By:           VALID Clause
*     * Snippet Number:      7
*     *
*     ************************************************************
*

*!*************************************************************************
*!
*!     Function: _QUU01FSNE
*!
*!     Called by: SCR220.SPR
*!
*!*************************************************************************
FUNCTION _quu01fsne    && m.producto VALID
#REGION 1
selecciona = "sdo_cveprd = " + ALLTRIM( STR(arrproducto(m.producto,2)) )
_CUROBJ = OBJNUM(m.botsob)

*     ************************************************************
*     *
*     * _QUU01FSQR          m.linea WHEN
*     *
*     * Function Origin:
```

```
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR220,    Record Number:  23
*    * Variable:           m.linea
*    * Called By:          WHEN Clause
*    * Snippet Number:     8
*    *
*    ************************************************
*
*!*************************************************************************
*!
*!    Function: _QUU01FSQR
*!
*!    Called by: SCR220.SPR
*!
*!       Calls: ARREGLO_LINEA     (procedure in TMENU.MPR)
*!
*!*************************************************************************
FUNCTION _quu01fsqr    && m.linea WHEN
#REGION 1
DO arreglo_linea
m.linea = 0

*    ************************************************
*    *
*    * _QUU01FST7          m.linea VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR220,    Record Number:  23
*    * Variable:           m.linea
*    * Called By:          VALID Clause
*    * Snippet Number:     9
*    *
*    ************************************************

*!*************************************************************************
*!
*!    Function: _QUU01FST7
*!
*!    Called by: SCR220.SPR
*!
*!       Uses: TARPROD.DBF          Alias: PRODUCTO
*!
*!    CDX files: TARPROD.CDX
*!
*!*************************************************************************
FUNCTION _quu01fst7    && m.linea VALID
#REGION 1
PRIVATE expresion,i, arrprdlin

SELECT prd_numprd ;
  FROM  tarprod ;
  WHERE prd_numlin = arrlinea(m.linea) ;
  INTO ARRAY arrprdlin

IF ALEN(arrprdlin) = 1
  expresion = "sdo_numprd = " + ALLTRIM( STR(arrprdlin[1]) )
ELSE
  expresion = ALLTRIM( STR(arrprdlin[1]) )
```

```
  FOR i=2 TO ALEN(arrprdlin)
    expresion = expresion + "," + ALLTRIM( STR(arrprdlin[i]) )
  ENDFOR
  expresion = "sdo_cveprd IN (" + expresion + ")"
ENDIF

selecciona = expresion
_CUROBJ = OBJNUM(m.botsob)

*     ************************************************************
*     *
*     * _QUU01FSZ7          m.botsob VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR220,    Record Number:  24
*     * Variable:            m.botsob
*     * Called By:           VALID Clause
*     * Snippet Number:      10
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!      Function: _QUU01FSZ7
*!
*!      Called by: SCR220.SPR
*!
*!         Calls: SOB21            (procedure in SCR220.SPR)
*!              : SOB23            (procedure in SCR220.SPR)
*!              : SOB22            (procedure in SCR220.SPR)
*!              : SOB24            (procedure in SCR220.SPR)
*!              : SCR320.SPR
*!              : &PROCESO.PRG
*!              : REPORTES         (procedure in SCR210.SPR)
*!              : SCRPRN.SPR
*!
*!*************************************************************************
FUNCTION _quu01fsz7     && m.botsob VALID
#REGION 1


DO CASE
CASE m.botsob = 1
  m.tctas = 0
  m.tsdovdo = 0
  m.tsdoact = 0
  SHOW GETS

  DO CASE
  CASE m.radsob =1      && REVISION DE SALDOS

    IF m.radsdo = 1                 && SOBREGIROS RELEVANTES
      DO sob21
      namereport = "sobsdo"
      dbfuente = "tqsob01"
      dbdestino = "tcsob01"
      proceso =  "sob21"
    ELSE                                    && SOBREGIROS MOROSOS
```

```
      DO sob23
      namereport = "sobmor"
      dbfuente = "tqsob03"
      dbdestino = "tcsob03"
      proceso =  "sob23"
    ENDIF

  CASE m.radsob =2              && RECUPERACION

    IF m.radrec = 1             &&  SIN RECUPERACION DE SALDO
      DO sob22
      namereport = "sobsrec"
      dbfuente = "tqsob02"
      dbdestino = "tcsob02 "
      proceso =  "sob22"

    ELSE                                    &&  SIN RECUPERACION DE MINIMA
      DO sob24
      namereport = "sobrecm"
      dbfuente = "tqsob04"
      dbdestino = "tcsob04"
      proceso =  "sob24"
    ENDIF
  ENDCASE
  est_cons = dbfuente

CASE m.botsob = 2

  DO (_GENGRAPH)

CASE m.botsob = 3
  m.tctas = 0
  m.tsdovdo = 0
  m.tsdoact = 0
  DO scr320.spr WITH dbfuente, dbdestino, 2
  DO &proceso WITH dbdestino
  est_cons = dbdestino

CASE m.botsob = 4

  DO reportes WITH "PREVIEW", est_cons, namereport
  DO scrpm.spr

CASE m.botsob = 5
  CLEAR READ
ENDCASE

*     ************************************************************
*     *
*     * _QUU01FTC9         Read Level When
*     *
*     * Function Origin:
*     *
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR220
*     * Called By:          READ Statement
*     * Snippet Number:     11
*     *
*     ************************************************************
*
```

```
*!*************************************************************************
*!
*!      Function: _QUU01FTC9
*!
*!      Called by: SCR220.SPR
*!
*!*************************************************************************
FUNCTION _quu01ftc9     && Read Level When
*
* When Code from screen: SCR220
*
#REGION 1
SHOW GET m.botsob DISABLE
SHOW GET m.botsob, 5 ENABLE

*: EOF: SCR220.SPR

*     ************************************************************
*     *
*     * 11/04/94          SCR230.SPR              00:40:24
*     *
*     ************************************************************

#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat

IF SET("TALK") = "ON"
  SET TALK OFF
  m.talkstat = "ON"
ELSE
  m.talkstat = "OFF"
ENDIF
m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

m.rborder = SET("READBORDER")
SET READBORDER ON


*     ************************************************************
*     *
*     *       SCR230/Windows Setup Code - SECTION 2
*     *
*     ************************************************************
#REGION 1
PUBLIC selecciona, norte, noroeste, occidente, centro, sur
PUBLIC seino, seinte, selocc, selcto, selsur
PUBLIC dbfuente, dbdestino, proceso, est_cons, origen, namereport

PUBLIC ARRAY arrest(1)

m.tctas = 0
m.tsdoact = 0
m.tsdovdo = 0
dbfuente = ""
dbdestino = ""
namereport = ""
proceso = ""
est_cons = ""
origen = ""
```

```
arrproducto = ""
arrlinea = ""
m.rad_tpocart = 0

norte = .F.
noroeste = .F.
occidente = .F.
centro = .F.
sur = .F.

*     ************************************************
*     *
*     *  SCR230/Windows Supporting Procedures and Functions
*     *
*     ************************************************
*

#REGION 1
*!*************************************************************************
*!
*!     Procedure: INICIA_GENERAL
*!
*!     Called by: _QUU01FJ89()      (function  in SCR210.SPR)
*!              : _QUU01FJHY()      (function  in SCR210.SPR)
*!              : _QUU01FJQJ()      (function  in SCR210.SPR)
*!              : _QUU01FRX6()      (function  in SCR220.SPR)
*!              : _QUU01FSG0()      (function  in SCR220.SPR)
*!              : _QUU01FZR6()      (function  in SCR230.SPR)
*!              : _QUU01G015()      (function  in SCR230.SPR)
*!              : _QUU01G62N()      (function  in SCR240.SPR)
*!
*!*************************************************************************
PROCEDURE inicia_general

m.producto = ""
m.linea = ""
norte = .F.
noroeste = .F.
occidente = .F.
centro = .F.
sur = .F.
SHOW GET m.producto DISABLE
SHOW GET m.linea DISABLE
SHOW GET m.botno DISABLE
SHOW GET m.botnte DISABLE
SHOW GET m.botocc DISABLE
SHOW GET m.botcto DISABLE
SHOW GET m.botsur DISABLE
m.clasifica = 3
selecciona=".T."

RETUVN


*     ************************************************
*     *
*     *  _QUU01FZC2          m.botsur VALID
*     *
*     *  Function Origin:
*     *
*     *  From Platform:      Windows
```

```
*     * From Screen:       SCR230,    Record Number:  3
*     * Variable:          m.botsur
*     * Called By:         VALID Clause
*     * Snippet Number:    1
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FZC2
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01fzc2    && m.botsur VALID
#REGION 1
IF !sur
  SHOW GET m.botsur,1 PROMPT "SUR"
  sur = .T.
ELSE
  SHOW GET m.botsur,1 PROMPT "..."
  sur = .F.
ENDIF

*     ************************************************
*     *
*     * _QUU01FZFX          m.botno VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR230,    Record Number:  4
*     * Variable:          m.botno
*     * Called By:         VALID Clause
*     * Snippet Number:    2
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FZFX
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01fzfx    && m.botno VALID
#REGION 1
IF !noroeste
  SHOW GET m.botno,1 PROMPT "NORESTE"
  noroeste = .T.
ELSE
  SHOW GET m.botno,1 PROMPT "..."
  noroeste = .F.
ENDIF

*     ************************************************
*     *
*     * _QUU01FZJO          m.botcto VALID
*     *
*     * Function Origin:
*     *
```

```
*     * From Platform:      Windows
*     * From Screen:        SCR230,    Record Number:  5
*     * Variable:           m.botcto
*     * Called By:          VALID Clause
*     * Snippet Number:     3
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FZJO
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01fzjo    && m.botcto VALID
#REGION 1
IF !centro
  SHOW GET m.botcto,1 PROMPT "CENTRO"
  centro = .T.
ELSE
  SHOW GET m.botcto,1 PROMPT "..."
  centro = .F.
ENDIF

*     ************************************************
*     *
*     * _QUU01FZNE          m.botnte VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR230,    Record Number:  6
*     * Variable:           m.botnte
*     * Called By:          VALID Clause
*     * Snippet Number:     4
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FZNE
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01fzne    && m.botnte VALID
#REGION 1
IF !norte
  SHOW GET m.botnte,1 PROMPT "NORTE"
  norte = .T.
ELSE
  SHOW GET m.botnte,1 PROMPT "..."
  norte = .F.
ENDIF

*     ************************************************
*     *
*     * _QUU01FZR6          m.rad_tpocart VALID
*     *
*     * Function Origin:
```

```
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR230,    Record Number:   7
*    * Variable:           m.rad_tpocart
*    * Called By:          VALID Clause
*    * Snippet Number:     5
*    *
*    ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FZR6
*!
*!     Called by: SCR230.SPR
*!
*!        Calls: INICIA_GENERAL    (procedure in SCR210.SPR)
*!
*!*************************************************************************
FUNCTION _quu01fzr6    && m.rad_tpocart VALID
#REGION 1
DO inicia_general
SHOW GET m.clasifica ENABLE
SHOW GET m.botcv ENABLE
SHOW GET m.botno ENABLE
SHOW GET m.botnte ENABLE
SHOW GET m.botocc ENABLE
SHOW GET m.botcto ENABLE
SHOW GET m.botsur ENABLE
_CUROBJ = OBJNUM(m.clasifica)

*    ************************************************************
*    *
*    * _QUU01FZYV          m.clasifica WHEN
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR230,    Record Number:   12
*    * Variable:           m.clasifica
*    * Called By:          WHEN Clause
*    * Snippet Number:     6
*    *
*    ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01FZYV
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01fzyv    && m.clasifica WHEN
#REGION 1
selecciona = ".T."

*    ************************************************************
*    *
*    * _QUU01G015          m.clasifica VALID
*    *
*    * Function Origin:
*    *
```

```
*     * From Platform:      Windows
*     * From Screen:        SCR230,     Record Number:  12
*     * Variable:           m.clasifica
*     * Called By:          VALID Clause
*     * Snippet Number:     7
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01G015
*!
*!     Called by: SCR230.SPR
*!
*!        Calls: INICIA_GENERAL    (procedure in SCR210.SPR)
*!
*!*************************************************************************
FUNCTION _quu01g015     && m.clasifica VALID
#REGION 1
DO CASE
CASE m.clasifica = 1
  m.linea=""
  SHOW GET m.linea DISABLE
  SHOW GET m.producto ENABLE
  _CUROBJ = OBJNUM(m.producto)
CASE m.clasifica = 2
  m.producto=""
  SHOW GET m.producto DISABLE
  SHOW GET m.linea ENABLE
  _CUROBJ = OBJNUM(m.linea)
CASE m.clasifica = 3
  DO inicia_general
  _CUROBJ = OBJNUM(m.botcv)
ENDCASE
SHOW GETS

*     ************************************************************
*     *
*     * _QUU01G06V          m.producto WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR230,     Record Number:  13
*     * Variable:           m.producto
*     * Called By:          WHEN Clause
*     * Snippet Number:     8
*     *
*     ************************************************************
*
*DO arreglo_producto
*m.producto = 0
*!*************************************************************************
*!
*!     Function: _QUU01G06V
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01g06v     && m.producto WHEN
```

```
*     ************************************************************
*     *
*     * _QUU01G096          m.producto VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR230,     Record Number:  13
*     * Variable:            m.producto
*     * Called By:           VALID Clause
*     * Snippet Number:      9
*     *
*     ************************************************************
*

*selecciona = "sdo_cveprd = " + *ALLTRIM( STR(arrproducto(m.producto,2)) )
*_CUROBJ = OBJNUM(m.botcred)
*!*************************************************************************
*!
*!     Function: _QUU01G096
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01g096     &&  m.producto VALID

*     ************************************************************
*     *
*     * _QUU01G0C8          m.linea WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR230,     Record Number:  14
*     * Variable:            m.linea
*     * Called By:           WHEN Clause
*     * Snippet Number:      10
*     *
*     ************************************************************
*
*DO arreglo_linea
*m.linea = 0
*!*************************************************************************
*!
*!     Function: _QUU01G0C8
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01g0c8     &&  m.linea WHEN

*     ************************************************************
*     *
*     * _QUU01G0EK          m.linea VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR230,     Record Number:  14
*     * Variable:            m.linea
*     * Called By:           VALID Clause
```

```
*     * Snippet Number:     11
*     *
*     ************************************************************
*

*PRIVATE expresion,i, arrprdlin

*SELECT prd_numprd ;
*FROM   TARPROD ;
*WHERE  prd_numlin = arrlinea(m.linea) ;
*INTO ARRAY arrprdlin

*IF ALEN(arrprdlin) = 1
*   expresion = "sdo_numprd = " + ALLTRIM( STR(arrprdlin[1]) )
*ELSE
*   expresion = ALLTRIM( STR(arrprdlin[1]) )
*   FOR i=2 to ALEN(arrprdlin)
*      expresion = expresion + "," + ALLTRIM( *STR(arrprdlin[i]) )
*   ENDFOR*
*   expresion = "sdo_cveprd IN (" + expresion + ")"
*ENDIF

*selecciona = expresion
*_CUROBJ = OBJNUM(m.botcred)
*!*************************************************************************
*!
*!     Function: _QUU01G0EK
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01g0ek    &&  m.linea VALID

*     ************************************************************
*     *
*     * _QUU01G0KH          m.botocc VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR230,     Record Number:  17
*     * Variable:           m.botocc
*     * Called By:          VALID Clause
*     * Snippet Number:     12
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01G0KH
*!
*!     Called by: SCR230.SPR
*!
*!*************************************************************************
FUNCTION _quu01g0kh    &&  m.botocc VALID
#REGION 1
IF  !occidente
  SHOW GET m.botocc,1 PROMPT "OCCIDENTE"
  occidente = .T.
ELSE
  SHOW GET m.botocc,1 PROMPT "..."
```

```
  occidente = .F.
ENDIF

*     ************************************************
*     *
*     * _QUU01G0QQ         m.botcv VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR230,    Record Number:  22
*     * Variable:           m.botcv
*     * Called By:          VALID Clause
*     * Snippet Number:     13
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!      Function: _QUU01G0QQ
*!
*!      Called by: SCR230.SPR
*!
*!           Calls: CV321          (procedure in CV.PRG)
*!                : CV322          (procedure in CV.PRG)
*!                : CV323          (procedure in CV.PRG)
*!                : CV324          (procedure in CV.PRG)
*!                : CV325          (procedure in CV.PRG)
*!                : CV326          (procedure in CV.PRG)
*!
*!*************************************************************************
FUNCTION _quu01g0qq     && m.botcv VALID
#REGION 1
DO CASE
CASE m.botcv = 1
  m.tctas = 0
  m.tsdovdo = 0
  m.tsdoact = 0
  SHOW GETS

  selnte = IIF(norte, "est_numdiv = 820", ".T.")
  selno  = IIF(noroeste, "est_numdiv = 810", ".T.")
  selocc = IIF(occidente, "est_numdiv = 830", ".T.")
  selcto = IIF(centro, "est_numdiv = 800", ".T.")
  selsur = IIF(sur, "est_numdiv = 840", ".T.")

  DO CASE
  CASE m.rad_tpocart =1     && Reciente
    namereport = "carape"
    dbfuente =  "tqcv01"
    dbdestino = "tcv01"
    proceso = "cv321"
    est_cons = dbfuente
    DO cv321 IN cv


  CASE m.rad_tpocart =2             && Administrativa
    namereport = "caradm"
    dbfuente =  "tqcv02"
    dbdestino = "tccv02"
```

```
    proceso = "cv322"
    est_cons = dbfuente
    DO cv322 IN cv


  CASE m.rad_tpocart =3    && Prejuridica
    namereport = "carprej"
    dbfuente =  "tqcv03"
    dbdestino = "tccv03"
    proceso = "cv323"
    est_cons = dbfuente
    DO cv323 IN cv


  CASE m.rad_tpocart =4          && Juridica
    namereport = "carjur"
    dbfuente =  "tqcv04"
    dbdestino = "tccv04"
    proceso = "cv324"
    est_cons = dbfuente
    DO cv324 IN cv


  CASE m.rad_tpocart =5    && Castigos
    namereport = "carcas"
    dbfuente =  "tqcv05"
    dbdestino = "tccv05"
    proceso = "cv325"
    est_cons = dbfuente
    DO cv325 IN cv


  CASE m.rad_tpocart =6    && Redocumentacion
    namereport = "carred"
    dbfuente =  "tqcv06"
    dbdestino = "tccv06"
    proceso = "cv326"
    est_cons = dbfuente
    DO cv326 IN cv

  ENDCASE

CASE m.botcv = 2

  CLEAR READ
ENDCASE

*     ************************************************************
*     *
*     * _QUU01G13M         Read Level When
*     *
*     * Function Origin:
*     *
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR230
*     * Called By:          READ Statement
*     * Snippet Number:     14
*     *
*     ************************************************************
```

```
*|*****************************************************************************
*|
*|     Function: _QUU01G13M
*|
*|     Called by: SCR230.SPR
*|
*|*****************************************************************************
FUNCTION _quu01g13m     && Read Level When
*
* When Code from screen: SCR230
*
#REGION 1
SHOW GETS DISABLE
SHOW GET m.rad_tpocart ENABLE
SHOW GET m.botcv, 4 ENABLE
*: EOF: SCR230.SPR

*     ***********************************************************
*     *
*     * 11/04/94          SCR231.SPR              00:41:21
*     *
*     ***********************************************************

PARAMETERS origen

*     ***********************************************************
*     *
*     *       SCR231/Windows Setup Code - SECTION 1
*     *
*     ***********************************************************
*

#REGION 1

PUBLIC long

DIMENSION arrest2[1]

m.etiq1 = "Division"
m.etiq2 = "Direccion"
pantalla = 1


*     ***********************************************************
*     *
*     *     WindowsREAD contains clauses from SCREEN scr231
*     *
*     ***********************************************************
*

READ CYCLE ;
  WHEN _quu01h841() ;
  SHOW _quu01h846() ;
  MODAL

RELEASE WINDOW scr231

#REGION 0
```

```
SET READBORDER &rborder

IF m.talkstat = "ON"
  SET TALK ON
ENDIF
IF m.compstat = "ON"
  SET COMPATIBLE ON
ENDIF



*    ************************************************************
*    *
*    *  SCR231/Windows Supporting Procedures and Functions
*    *
*    ************************************************************
*

PROCEDURE reportes
PARAMETER view_print, dborigen, namereport

SELECT DISTINCT * ;
  FROM tarsdos, &dborigen, tarprod, tarestru, tamomes ;
```

```
SET READBORDER &rborder

IF m.talkstat = "ON"
  SET TALK ON
ENDIF
IF m.compstat = "ON"
  SET COMPATIBLE ON
ENDIF



*     ************************************************************
*     *
*     *  SCR231/Windows Supporting Procedures and Functions
*     *
*     ************************************************************
*

PROCEDURE reportes
PARAMETER view_print, dborigen, namereport

SELECT DISTINCT * ;
  FROM tarsdos, &dborigen, tarprod, tarestru, tarnomes ;
  WHERE sdo_numcta = cuenta AND ;
  sdo_cveprd = prd_numprd AND ;
  sdo_numsuc = est_numsuc AND ;
  est_numdir = nes_numdir AND ;
  est_numdiv = nes_numdiv ;
  INTO CURSOR creport

REPORT FORM &namereport &view_print

RETU N



*     ************************************************************
*     *
*     *  _QUU01H7LN          m.listcv WHEN
*     *
*     *  Function Origin:
*     *
*     *  From Platform:       Windows
*     *  From Screen:         SCR231,     Record Number:   7
*     *  Variable:            m.listcv
*     *  Called By:           WHEN Clause
*     *  Snippet Number:      1
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!     Function: _QUU01H7LN
*!
*!     Called by: SCR231.SPR
*!
*!**************************************************************************
FUNCTION _quu01h7ln     &&  m.listcv WHEN
#REGION 1
m.listcv = 0
```

```
*    ************************************************************
*    *
*    * _QUU01H7OY          m.listcv VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR231,    Record Number:   7
*    * Variable:           m.listcv
*    * Called By:          VALID Clause
*    * Snippet Number:     2
*    *
*    ************************************************************



*!*************************************************************************
*!
*!    Function: _QUU01H7OY
*!
*!    Called by: SCR231.SPR
*!
*!       Uses: &ORIGEN
*!           : TARSDOS.DBF          Alias: SALDOS
*!           : TARESTRU.DBF         Alias: ESTRUCTURA
*!           : TARNOMES.DBF         Alias: NOMESTRU
*!
*!    CDX files: TARSDOS.CDX
*!           : TARESTRU.CDX
*!           : TARNOMES.CDX
*!
*!*************************************************************************
FUNCTION _quu01h7oy    && m.listcv VALID
#REGION 1
DIR = arrest[m.listcv,2]

arrest2 = ""
long = ALEN(arrest,1)
DIMENSION arrest2(long,2)

FOR i=1 TO long
  arrest2[i,1] = arrest[i,1]
  arrest2[i,2] = arrest[i,2]
NEXT i

*=ACOPY(arrest, arrest2)

arrest = ""
SELECT nes_nomdir + " " + ;
  est_nomsuc + " " + ;
  STR(COUNT(*)) + " " + ;
  STR(SUM(sdo_sdovdo)) + " " + ;
  STR(SUM(sdo_sdoact)) ;
  FROM &origen, tarsdos, tarestru, tarnomes ;
  WHERE sdo_numcta = cuenta AND ;
  sdo_numsuc = est_numsuc AND ;
  est_numdir = nes_numdir AND ;
  est_numdiv = nes_numdiv AND ;
  est_numdir = DIR ;
```

```
  GROUP BY est_nomsuc ;
  INTO ARRAY arrest

etiq1 = "Direccion"
etiq2 = "Sucursal"
SHOW GETS

pantalla = 2
SHOW GET m.listcv DISABLE


*     ************************************************************
*     *
*     * _QUU01H7WK          m.boton VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR231,    Record Number:   8
*     * Variable:           m.boton
*     * Called By:          VALID Clause
*     * Snippet Number:     3
*     *
*     ************************************************************


*!*************************************************************************
*!
*!     Function: _QUU01H7WK
*!
*!     Called by: SCR231.SPR
*!
*!        Calls: SCR320.SPR
*!             : &PROCESO.PRG
*!             : REPORTES          (procedure in SCR210.SPR)
*!             : SCRPRN.SPR
*!
*!*************************************************************************
FUNCTION _quu01h7wk     && m.boton VALID
#REGION 1
DO CASE
CASE m.boton = 1

  m.tctas = 0
  m.tsdovdo = 0
  m.tsdoact = 0
  DO scr320.spr WITH dbfuente, dbdestino, 3
  est_cons = dbdestino
  DO &proceso WITH dbdestino IN cv.prg

CASE m.boton = 2

  DO (_GENGRAPH)
CASE m.boton = 3

  DO reportes WITH "PREVIEW", est_cons, namereport
  DO scrprn.spr

CASE m.boton = 4

  IF pantalla = 1
```

```
    CLEAR READ
  ELSE
    pantalla = 1
    etiq1 = "Division"
    etiq2 = "Direccion"
    DIMENSION arrest(long,2)
    arrest = ""
    =ACOPY(arrest2, arrest)
    SHOW GETS
    SHOW GET m.listcv ENABLE
  ENDIF
ENDCASE

*     ************************************************************
*     *
*     * _QUU01H841          Read Level When
*     *
*     * Function Origin:
*     *
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR231
*     * Called By:          READ Statement
*     * Snippet Number:     4
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!     Function: _QUU01H841
*!
*!     Called by: SCR231.SPR
*!
*!**************************************************************************
FUNCTION _quu01h841     && Read Level When
*
* When Code from screen: SCR231
*
#REGION 1

SHOW GET m.boton DISABLE
SHOW GET m.boton,2 ENABLE
SHOW GET m.boton,4 ENABLE

DO CASE
CASE der = "D"
  SHOW GET m.boton,3 ENABLE
CASE der = "Q"
  SHOW GET m.boton,1 ENABLE
CASE der = "B"
  SHOW GET m.boton,1 ENABLE
  SHOW GET m.boton,3 ENABLE
ENDCASE

SHOW GETS

*: EOF: SCR231.SPR
```

```
*    ************************************************************
*    *
*    * 11/04/94          SCR240.SPR              00:40:33
*    *
*    ************************************************************


*    ************************************************************
*    *
*    *       SCR240/Windows Setup Code - SECTION 2
*    *
*    ************************************************************
*

#REGION 1
PUBLIC selecciona

m.tctas = 0
m.tsdoact = 0
m.tsdovdo = 0
dbfuente = ""
dbdestino = ""
namereport = ""
proceso = ""
est_cons = ""
origen = ""

DIMENSION arrproducto[1,2], arrlinea[1]

DEFINE WINDOW carprev FROM 9,42 TO 30,86;
  TITLE "Carteras Preventiva" ;
  FONT "MS Serif",8 ;
  STYLE "B" ;
  FLOAT;
  GROW

arrsubcred=""
arrproducto=""
arrlinea=""

*    ************************************************************
*    *
*    *  SCR240/Windows Supporting Procedures and Functions
*    *
*    ************************************************************

#REGION 1
*|*****************************************************************************
*|
*|    Procedure: INICIA_GENERAL
*|
*|    Called by: _QUU01FJ89()     (function  in SCR210.SPR)
*|             : _QUU01FJHY()     (function  in SCR210.SPR)
*|             : _QUU01FJQJ()     (function  in SCR210.SPR)
*|             : _QUU01FRX6()     (function  in SCR220.SPR)
*|             : _QUU01FSG0()     (function  in SCR220.SPR)
*|             : _QUU01FZR6()     (function  in SCR230.SPR)
*|             : _QUU01G015()     (function  in SCR230.SPR)
*|             : _QUU01G62N()     (function  in SCR240.SPR)
*|
*|*****************************************************************************
```

```
PROCEDUVE inicia_general
m.producto = ""
m.linea = ""
SHOW GET m.producto DISABLE
SHOW GET m.linea DISABLE
m.clasifica = 3
selecciona=".T."
RETUVN


*****************************************************
*!*************************************************************************
*!
*!    Procedure: MATCH_NAME
*!
*!    Called by: _QUU01G7E4()      (function  in SCR240.SPR)
*!
*!       Calls: CHANGE             (procedure in SCR240.SPR)
*!            : GET                (procedure in SCR240.SPR)
*!            : SAME_N()           (function  in SCR240.SPR)
*!            : SAME_SN1()         (function  in SCR240.SPR)
*!            : SAME_SN2()         (function  in SCR240.SPR)
*!
*!        Uses: TMPNOM.DBF
*!            : NOMBRES.DBF        Alias: NOMBRES
*!
*!*************************************************************************
PROCEDURE match_name
PUBLIC sname, ssurn1, ssurn2, pointer
PUBLIC mismo_n, mismo_a1, mismo_a2
PUBLIC fname, fsurn1, fsurn2

PRIVATE COUNT
COUNT = 1

IF !FILE("NOMBRES.DBF")
  USE tmpnom
  DO CHANGE
  SORT TO nombres ON sdo_nomcli
  ERASE tmpnom
ENDIF

SELECT 15
USE nombres ALIAS nombres
REPLACE flag WITH 0 FOR flag = 1

pointer = 1
sname = ""
ssurn1 = ""
ssurn2 = ""
GO TOP


DO WHILE !EOF()
  numr = RECNO()
  DO GET WITH sdo_nomcli, fname, fsurn1, fsurn2
  mismo_n  = IIF(LEN(fname) > LEN(sname), same_n(fname,sname), same_n(sname,fname))
  mismo_a1 = IIF(LEN(fsurn1) > LEN(ssurn1), same_sn1(fsurn1,ssurn1), same_sn1(ssurn1,fsurn1))
  mismo_a2 = IIF(LEN(fsurn2) > LEN(ssurn2), same_sn2(fsurn2,ssurn2), same_sn2(ssurn2,fsurn2))
```

```
  IF (mismo_n AND mismo_s1 AND mismo_s2)
    REPLACE flag WITH COUNT
    GO pointer - 1
    REPLACE flag WITH COUNT
    GO pointer
  ELSE
    COUNT = COUNT + 1
  ENDIF
  sname = fname
  ssum1 = fsum1
  ssum2 = fsum2
  SKIP
  pointer = pointer + 1
ENDDO

*!*************************************************************************
*!
*!      Function: SAME_N
*!
*!      Called by: MATCH_NAME        (procedure in SCR240.SPR)
*!
*!*************************************************************************
FUNCTION same_n

PARAMETERS n1, n2
PRIVATE sameval,posblank,posblank2,fn1,sn1,fn2,sn2

sameval = .F.
posblank = AT(" ",n1)
posblank2 = AT(" ",n2)
fn1 = SUBSTR( n1,1,posblank-1 )
sn1 = SUBSTR( n1,posblank+1,LEN(n1)-posblank )
fn2 = SUBSTR( n2,1,posblank2-1 )
sn2 = SUBSTR( n2,posblank2+1,LEN(n2)-posblank2 )
IF n1 = n2
  sameval = .T.
ELSE
  IF (posblank2 = 0)
    IF (posblank = 0)
      sameval = IIF( n2 = n1, .T., .F. )
    ELSE
      sameval = IIF( n2 = fn1 OR n2 = sn1, .T., .F. )
    ENDIF
  ELSE
    IF (posblank = 0)
      sameval = IIF(n1 = fn2 OR n1 = sn2, .T., .F. )
    ELSE
      sameval = IIF(fn2 = fn1 OR sn2 = fn1 OR fn2 = sn1 OR fn2 = sn1, .T., .F. )
    ENDIF
  ENDIF
ENDIF
RETURN sameval

*!*************************************************************************
*!      Function: SAME_SN1
*!
*!      Called by: MATCH_NAME        (procedure in SCR240.SPR)
*!
*!        Calls: PASS                (procedure in SCR240.SPR)
*!
*!*************************************************************************
```

```
FUNCTION same_sn1

PARAMETERS fsn1, fsn2
PRIVATE sameval

sameval = .F.

IF (fsn1 = fsn2)
  sameval = .T.
ELSE
  IF ( LEN(fsn1)-LEN(fsn2) < 3)
    sameval = pass(fsn1,fsn2)
  ENDIF
ENDIF

RETURN sameval


*|*****************************************************************************
*|
*|     Function: SAME_SN2
*|
*|    Called by: MATCH_NAME       (procedure in SCR240.SPR)
*|
*|       Calls: PASS              (procedure in SCR240.SPR)
*|
*|*****************************************************************************
FUNCTION same_sn2

PARAMETERS ssn1, ssn2
PRIVATE sameval

sameval = .F.
IF (ssn1 = ssn2)
  sameval = .T.
ELSE
  IF (LEN(ssn2) = 1)
    sameval = IIF(ssn2 = SUBSTR(ssn1,1,1), .T., .F.)
  ELSE
    IF ( LEN(ssn1)-LEN(ssn2) < 3 )
      sameval = pass(ssn1, ssn2, .T.)
    ENDIF
  ENDIF
ENDIF

RETURN sameval

*|*****************************************************************************
*|
*|    Procedure: PASS
*|
*|    Called by: SAME_SN1()       (function in SCR240.SPR)
*|          : SAME_SN2()       (function in SCR240.SPR)
*|
*|*****************************************************************************
PROCEDURE pass

PARAMETERS nom1, nom2, ENABLE
PRIVATE passval,ne,R,i,c1,c2,c3,twocar
```

```
passval = .F.
ne = 0
R = LEN(nom2)%2
FOR i = 0 TO (INT(LEN(nom2)/2)-1)
  twocar = SUBSTR(nom2,2*i+1,2)
  IF AT(twocar,nom1) = 0
    ne = ne + 1
  ENDIF
NEXT I

IF R = 1
  IF (RAT(RIGHT(nom2,1),nom1)) = 0
    ne = ne + 1
  ENDIF
ENDIF

passval = IIF( ne <= 1, .T. , .F.)
IF ENABLE = .T.
  IF (LEN(nom2) = 3)
    c1 = SUBSTR(nom2,1,1)
    c2 = SUBSTR(nom2,2,2)
    c3 = SUBSTR(nom2,3,3)
    IF (AT(c1,nom1) != 0 AND AT(c2,nom1)!=0 AND AT(c3,nom1) != 0)
      passval = .T.
    ENDI
```

```
passval = .F.
ne = 0
R = LEN(nom2)%2
FOR I = 0 TO (INT(LEN(nom2)/2)-1)
  twocar = SUBSTR(nom2,2*I+1,2)
  IF AT(twocar,nom1) = 0
    ne = ne + 1
  ENDIF
NEXT I

IF R = 1
  IF (RAT(RIGHT(nom2,1),nom1)) = 0
    ne = ne + 1
  ENDIF
ENDIF

passval = IIF( ne <= 1, .T. , .F.)
IF ENABLE = .T.
  IF (LEN(nom2) = 3)
    c1 = SUBSTR(nom2,1,1)
    c2 = SUBSTR(nom2,2,2)
    c3 = SUBSTR(nom2,3,3)
    IF (AT(c1,nom1) != 0 AND AT(c2,nom1)!=0 AND AT(c3,nom1) != 0)
      passval = .T.
    ENDIF
  ENDIF
ENDIF


RETURN passval


*|*****************************************************************************
*|
*|    Function: DELPOINT
*|
*|    Called by: GET            (procedure in SCR240.SPR)
*|
*|*****************************************************************************
FUNCTION delpoint
PARAMETERS N
PRIVATE pospleft, pospright

pospleft = AT(N,".")
IF pospleft <> 0
  N = STUFF(N, pospleft,1,")
  pospright = RAT(N,".")
  IF pospright <> 0
    N = STUFF(N, pospright,1,")
  ENDIF
ENDIF

RETURN

*|*****************************************************************************
*|    Procedure: GET
*|
*|    Called by: MATCH_NAME      (procedure in SCR240.SPR)
*|
*|      Calls: DELPOINT()      (function  in SCR240.SPR)
*|*****************************************************************************
```

```
PROCEDURE GET

PARAMETERS nomcom, nom, surn1, surn2
PRIVATE ncar,poscoma,posslash,ncarsur

ncar= LEN(nomcom)
poscoma= RAT(",",nomcom)
posslash= AT("/",nomcom)
ncarsur= poscoma - posslash -1
nom = LTRIM(RTRIM(delpoint(SUBSTR(nomcom,poscoma+1,ncar-poscoma))))
surn1 = LTRIM(RTRIM(delpoint(SUBSTR(nomcom,1,posslash-1))))
surn2 = LTRIM(RTRIM(delpoint(SUBSTR(nomcom,posslash+1,ncarsur))))

RETURN

*|*************************************************************************
*|
*|    Procedure: CHANGE
*|
*|    Called by: MATCH_NAME        (procedure in SCR240.SPR)
*|
*|*************************************************************************
PROCEDURE CHANGE
PRIVATE nc, ncnom, poscoma, posslash, ncsurname1
PRIVATE names, surname1, surname2, newname

GO TOP
DO WHILE !EOF()
  ncnom = LEN(sdo_nomcli)
  poscoma = AT(",",sdo_nomcli)
  posslash = RAT("/",sdo_nomcli)

  ncsurname1 = posslash - poscoma - 1
  names = LTRIM(RTRIM(SUBSTR(sdo_nomcli,1,poscoma-1)))
  surname1 = LTRIM(RTRIM(SUBSTR(sdo_nomcli,poscoma+1,ncsurname1)))
  surname2 = LTRIM(RTRIM(SUBSTR(sdo_nomcli,posslash+1)))
  newname = surname1 + "/" + surname2 + "," + names
  REPLACE sdo_nomcli WITH newname
  SKIP
ENDDO
GO TOP

RETURN

*|*************************************************************************
*|
*|    Procedure: CUENTA_TARJETAS
*|
*|    Called by: _QUU01G7E4()      (function  in SCR240.SPR)
*|
*|*************************************************************************
PROCEDURE cuenta_tarjetas
PRIVATE bandera, COUNT, pointer

COUNT = 1
pointer = 1

SELECT nombres
bandera = 0
GO TOP
```

```
DO WHILE !EOF()

  IF  flag = bandera AND !EMPTY(bandera)
    COUNT = COUNT +1
  ELSE
    IF !EMPTY(bandera)
      GO pointer - 1
      REPLACE ntar WITH COUNT
      COUNT = 1
      GO pointer
    ENDIF
  ENDIF
  bandera = flag
  pointer = pointer + 1
  SKIP
ENDDO

RETURN


*!*************************************************************************
*!
*!    Procedure: CP01
*!
*!    Called by: _QUU01G6PU()      (function  in SCR240.SPR)
*!
*!       Calls: CALCULA_TOTAL      (procedure in SCR210.SPR)
*!            : ESTRATIFICADO      (procedure in SCR220.SPR)
*!
*!       Uses: NOMBRES.DBF         Alias: NOMBRES
*!            : TARSDOS.DBF        Alias: SALDOS
*!
*!    CDX files: TARSDOS.CDX
*!
*!*************************************************************************
PROCEDURE cp01
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcp01", origen )

SELECT DISTINCT cuenta, nombres.ntar AS ntarj ;
  FROM nombres, tarsdos ;
  WHERE cuenta = sdo_numcta AND ;
  nombres.flag > 0 AND ;
  (sdo_sdoact - sdo_limcre ) > 0 ;
  INTO TABLE tqcp01

DO calcula_total WITH origen
DO estratificado WITH origen

RETURN

*!*************************************************************************
*!
*!    Procedure: CP02
*!
*!    Called by: _QUU01G6PU()      (function  in SCR240.SPR)
*!
*!       Calls: CALCULA_TOTAL      (procedure in SCR210.SPR)
```

```
*|          : ESTRATIFICADO    (procedure in SCR220.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|           : NOMBRES.DBF          Alias: NOMBRES
*|           : TARBOLET.DBF         Alias: BOLETIN
*|
*|    CDX files: TARSDOS.CDX
*|           : TARBOLET.CDX
*|
*|*****************************************************************************
PROCEDURE cp02
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcp02", origen )

SELECT DISTINCT cuenta, nombres.ntar AS ntarj;
  FROM tarsdos, nombres, tarbolet ;
  WHERE !EMPTY(nombres.flag) AND ;
  sdo_numcta = cuenta AND ;
  sdo_numcta = bol_numcta AND ;
  ((sdo_sdoact - sdo_limcre)/sdo_sdoact) > 0.15 AND ;
  EMPTY(bol_fecbop) ;
  ORDER BY nombres.sdo_nomcli ;
  INTO TABLE tqcp02

DO calcula_total WITH origen
DO estratificado WITH origen

RETURN


*|*****************************************************************************
*|
*|    Procedure: CP03
*|
*|    Called by: _QUU01G6PU()      (function  in SCR240.SPR)
*|
*|      Calls: CALCULA_TOTAL       (procedure in SCR210.SPR)
*|           : ESTRATIFICADO      (procedure in SCR220.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|           : NOMBRES.DBF          Alias: NOMBRES
*|           : TARBOLET.DBF         Alias: BOLETIN
*|
*|    CDX files: TARSDOS.CDX
*|           : TARBOLET.CDX
*|
*|*****************************************************************************
PROCEDURE cp03
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcp03", origen )

SELECT DISTINCT cuenta, nombres.ntar AS ntarj ;
  FROM tarsdos, nombres, tarbolet ;
  WHERE !EMPTY(nombres.flag) AND ;
  sdo_numcta = cuenta AND ;
  sdo_numcta = bol_numcta AND ;
  sdo_cveutr IN (2030,2040) AND ;
  ((sdo_sdoact - sdo_limcre)/sdo_sdoact) > 0.15 AND ;
  (bol_fecbop < sdo_fecutr OR EMPTY(sdo_cvebol)) ;
```

```
  ORDER BY nombres.sdo_nomcli ;
  INTO TABLE tqcp03

DO calcula_total WITH origen
DO estratificado WITH origen

RETURN

*|*****************************************************************
PROCEDURE estratificado
PARAMETERS origen

SELECT DISTINCT ntarj AS n_tarjetas, ;
  COUNT(ntarj) AS n_ctas, ;
  SUM(sdo_sdovdo) AS sdo_vdo, ;
  SUM(sdo_sdoact) AS sdo_act ;
  FROM tarsdos, &origen ;
  WHERE sdo_numcta = cuenta AND;
  &selecciona ;
  GROUP BY ntarj ;
  HAVING !EMPTY(ntarj) ;
  INTO CURSOR eqcp

BROWSE FIELDS ;
  n_tarjetas: 5, ;
  n_ctas:    8 :p = "999,999", ;
  sdo_vdo:  12 :p = "999,999,999", ;
  sdo_act:  12 :p = "999,999,999" ;
  WINDOW carprev

RETURN

*|*****************************************************************
PROCEDURE reportes
PARAMETER view_print, dborigen, namereport

SELECT DISTINCT * ;
  FROM tarprod, tarsdos, &dborigen ;
  WHERE sdo_numcta = cuenta AND ;
  sdo_cveprd = prd_numprd ;
  INTO CURSOR creport

REPORT FORM &namereport &view_print

RETUVN

*|*****************************************************************
PROCEDURE calcula_total
PARAMETER origen

SELECT COUNT(*) AS tctas, ;
  SUM(sdo_sdovdo) AS tsdovdo, ;
  SUM(sdo_sdoact) AS tsdoact ;
  FROM tarsdos, &origen ;
  WHERE sdo_numcta = cuenta AND ;
  &selecciona ;
  INTO CURSOR ctotal

SCATTER MEMVAR

RETURN
```

```
*     ************************************************************
*     *
*     * _QUU01G5Z7          m.clasifica WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR240,    Record Number:   7
*     * Variable:            m.clasifica
*     * Called By:           WHEN Clause
*     * Snippet Number:      1
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!      Function: _QUU01G5Z7
*!
*!     Called by: SCR240.SPR
*!
*!**************************************************************************
FUNCTION _quu01g5z7     && m.clasifica WHEN
#REGION 1
selecciona = ".T."

*     ************************************************************
*     *
*     * _QUU01G62N          m.clasifica VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR240,    Record Number:   7
*     * Variable:            m.clasifica
*     * Called By:           VALID Clause
*     * Snippet Number:      2
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!      Function: _QUU01G62N
*!
*!     Called by: SCR240.SPR
*!
*!        Calls: INICIA_GENERAL     (procedure in SCR210.SPR)
*!
*!**************************************************************************
FUNCTION _quu01g62n     && m.clasifica VALID
#REGION 1
DO CASE
CASE m.clasifica = 1
  m.linea=""
  SHOW GET m.linea DISABLE
  SHOW GET m.producto ENABLE
  _CUROBJ = OBJNUM(m.producto)
```

```
CASE m.clasifica = 2
  m.producto=""
  SHOW GET m.producto DISABLE
  SHOW GET m.linea ENABLE
  _CUROBJ = OBJNUM(m.linea)

CASE m.clasifica = 3
  DO inicia_general
  _CUROBJ = OBJNUM(m.botcp)
ENDCASE
SHOW GETS

*     ************************************************************
*     *
*     * _QUU01G67U         m.producto WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR240,     Record Number:   8
*     * Variable:           m.producto
*     * Called By:          WHEN Clause
*     * Snippet Number:     3
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!    Function: _QUU01G67U
*!
*!    Called by: SCR240.SPR
*!
*!       Calls: ARREGLO_PRODUCTO   (procedure in TMENU.MPR)
*!
*!*************************************************************************
FUNCTION _quu01g67u     && m.producto WHEN
#REGION 1
DO arreglo_producto
m.producto = 0

*     ************************************************************
*     *
*     * _QUU01G6A9         m.producto VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR240,     Record Number:   8
*     * Variable:           m.producto
*     * Called By:          VALID Clause
*     * Snippet Number:     4
*     *
*     ************************************************************

*!*************************************************************************
*!
*!    Function: _QUU01G6A9
*!
*!    Called by: SCR240.SPR
*!
*!*************************************************************************
```

```
FUNCTION _quu01g6a9     && m.producto VALID
#REGION 1
selecciona = "sdo_cveprd = " + ALLTRIM( STR(arrproducto(m.producto,2)) )
_CUROBJ = OBJNUM(m.botcred)

*     ************************************************************
*     *
*     * _QUU01G6DK          m.linea WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR240,    Record Number:   9
*     * Variable:            m.linea
*     * Called By:           WHEN Clause
*     * Snippet Number:      5
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01G6DK
*!
*!     Called by: SCR240.SPR
*!
*!        Calls: ARREGLO_LINEA     (procedure in TMENU.MPR)
*!
*!*************************************************************************
FUNCTION _quu01g6dk     && m.linea WHEN
#REGION 1
DO arreglo_linea
m.linea = 0

*     ************************************************************
*     *
*     * _QUU01G6G0          m.linea VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR240,    Record Number:   9
*     * Variable:            m.linea
*     * Called By:           VALID Clause
*     * Snippet Number:      6
*     *
*     ************************************************************
*


*!*************************************************************************
*!
*!     Function: _QUU01G6G0
*!
*!     Called by: SCR240.SPR
*!
*!        Uses: TARPROD.DBF          Alias: PRODUCTO
*!
*!     CDX files: TARPROD.CDX
*!
*!*************************************************************************
FUNCTION _quu01g6g0     && m.linea VALID
#REGION 1
```

```
PRIVATE expresion,i, arrprdlin

SELECT prd_numprd ;
  FROM  tarprod ;
  WHERE  prd_numlin = arrlinea(m.linea) ;
  INTO ARRAY arrprdlin

IF ALEN(arrprdlin) = 1
  expresion = "sdo_numprd = " + ALLTRIM( STR(arrprdlin[1]) )
ELSE
  expresion = ALLTRIM( STR(arrprdlin[1]) )
  FOR i=2 TO ALEN(arrprdlin)
    expresion = expresion + "," + ALLTRIM( STR(arrprdlin[i]) )
  ENDFOR
  expresion = "sdo_cveprd IN (" + expresion + ")"
ENDIF

selecciona = expresion
_CUROBJ = OBJNUM(m.botcred)

*     ************************************************************
*     *
*     * _QUU01G6PU          m.botcp VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR240,    Record Number:  17
*     * Variable:           m.botcp
*     * Called By:          VALID Clause
*     * Snippet Number:     7
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01G6PU
*!
*!     Called by: SCR240.SPR
*!
*!        Calls: CP01              (procedure in SCR240.SPR)
*!             : CP02              (procedure in SCR240.SPR)
*!             : CP03              (procedure in SCR240.SPR)
*!             : SCR320.SPR
*!             : &PROCESO.PRG
*!             : REPORTES          (procedure in SCR210.SPR)
*!             : SCRPRN.SPR
*!
*!*************************************************************************
FUNCTION _quu01g6pu    && m.botcp VALID
#REGION 1
DO CASE
CASE m.botcp = 1

  m.tctas = 0
  m.tsdovdo = 0
  m.tsdoact = 0
  SHOW GETS
```

```
DO CASE
CASE m.radcp =1     && CLIENTES CON DOS O MAS TARJETAS
  && SOBREGIRADAS

  DO cp01
  namereport = "c2tarsob"
  dbfuente = "tqcp01"
  dbdestino = "tccp01"
  proceso =  "cp01"


CASE m.radcp =2                 && VALIDACION BOLETIN COMERCIAL

  DO cp02
  namereport = "c2bolcom"
  dbfuente = "tqcp02"
  dbdestino = "tccp02"
  proceso =  "cp02"

CASE m.radcp = 3                && VALIDACION BOLETIN COMERCIAL CON TRANDACCIONES

  DO cp03
  namereport = "c2trans"
  dbfuente = "tqcp03"
  dbdestino = "tccp03"
  proceso =  "cp03"


ENDCASE
est_cons = dbfuente

CASE m.botcp = 2

  DO (_GENGRAPH)

CASE m.botcp = 3
  m.tctas = 0
  m.tsdovdo = 0
  m.tsdoact = 0
  DO scr320.spr WITH dbfuente, dbdestino, 4
  DO &proceso WITH dbdestino
  est_cons = dbdestino

CASE m.botcp = 4

  DO reportes WITH "PREVIEW", est_cons, namereport
  DO scrprn.spr

CASE m.botcp = 5
  CLEAR READ
ENDCASE


*     ************************************************
*     *
*     * _QUU01G727         m.radcp VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:     Windows
*     * From Screen:       SCR240    Record Number:  22
```

```
*     * Variable:          m.radcp
*     * Called By:         VALID Clause
*     * Snippet Number:    8
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01G727
*!
*!     Called by: SCR240.SPR
*!
*!*************************************************************************
FUNCTION _quu01g727     && m.radcp VALID
#REGION 1
SHOW GET m.botcp DISABLE
SHOW GET m.botcp,1 ENABLE
SHOW GET m.botcp,2 ENABLE
SHOW GET m.botcp,5 ENABLE

DO CASE
CASE der = "D"
  SHOW GET m.botcp,4 ENABLE
CASE der = "Q"
  SHOW GET m.botcp,3 ENABLE
CASE der = "B"
  SHOW GET m.botcp,3 ENABLE
  SHOW GET m.botcp,4 ENABLE
ENDCASE

*     ************************************************
*     *
*     * _QUU01G7A9          m.botdisk VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR240,    Record Number:  23
*     * Variable:           m.botdisk
*     * Called By:          VALID Clause
*     * Snippet Number:     9
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01G7A9
*!
*!     Called by: SCR240.SPR
*!
*!        Uses: NOMBRES.DBF          Alias: NOMBRES
*!
*!*************************************************************************
FUNCTION _quu01g7a9     && m.botdisk VALID
#REGION 1
WAIT WINDOW "Inserte el diskette en la unidad A:"
SELECT nombres
COPY TO B:\nombres.dbf
```

```
*     ************************************************
*     *
*     * _QUU01G7E4          m.botnom VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR240,    Record Number:  25
*     * Variable:           m.botnom
*     * Called By:          VALID Clause
*     * Snippet Number:     10
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!    Function: _QUU01G7E4
*!
*!    Called by: SCR240.SPR
*!
*!       Calls: MATCH_NAME        (procedure in SCR240.SPR)
*!            : CUENTA_TARJETAS   (procedure in SCR240.SPR)
*!
*!*************************************************************************
FUNCTION _quu01g7e4    && m.botnom VALID
#REGION 1
DO match_name
DO cuenta_tarjetas


*     ************************************************
*     *
*     * _QUU01G7HI          Read Level When
*     *
*     * Function Origin:
*     *
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR240
*     * Called By:          READ Statement
*     * Snippet Number:     11
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!    Function: _QUU01G7HI
*!
*!    Called by: SCR240.SPR
*!
*!*************************************************************************
FUNCTION _quu01g7hi    && Read Level When
*
* When Code from screen: SCR240
*
#REGION 1
SHOW GET m.botcp DISABLE
SHOW GET m.botcp,5 ENABLE

*: EOF: SCR240.SPR
```

```
*    ************************************************************
*    *
*    * 11/04/94          SCR310.SPR              00:40:43
*    *
*    ************************************************************

*    ************************************************************
*    *
*    *       SCR310/Windows Setup Code - SECTION 2
*    *
*    ************************************************************
*

#REGION 1
PUBLIC m.nomproducto, m.nomsuc, m.situacion

SELECT saldos
SCATTER MEMVAR BLANK
m.nomproducto = ""
m.nomsuc = ""
m.situacion = ""

*    ************************************************************
*    *
*    * _QUU01GEI8          m.numcta WHEN
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR310,    Record Number:  25
*    * Variable:           m.numcta
*    * Called By:          WHEN Clause
*    * Snippet Number:     1
*    *
*    ************************************************************


*!*************************************************************************
*!
*!     Function: _QUU01GEI8
*!
*!     Called by: SCR310.SPR
*!
*!*************************************************************************
FUNCTION _quu01gei8     && m.numcta WHEN
#REGION 1
SELECT saldos
SCATTER MEMVAR BLANK
m.nomproducto=""
m.nomsuc=""
m.situacion=""
SHOW GETS

*    ************************************************************
*    *
*    * _QUU01GEM3          m.numcta VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR310,    Record Number:  25
```

```
*     * Variable:          m.numcta
*     * Called By:         VALID Clause
*     * Snippet Number:    2
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!      Function: _QUU01GEM3
*!
*!     Called by: SCR310.SPR
*!
*!         Calls: MSGERROR.SPR
*!
*!**************************************************************************
FUNCTION _quu01gem3     && m.numcta VALID
#REGION 1
IF !EMPTY(m.numcta)

  SELECT saldos
  SET ORDER TO TAG numcta
  SEEK m.numcta
  IF FOUND()
    SCATTER MEMVAR
    SHOW GETS
    SELECT producto
    SET ORDER TO TAG numprd
    SEEK m.sdo_cveprd
    m.nomproducto = IIF(FOUND(),producto.prd_nomprd,"")
    SELECT estructura
    SET ORDER TO TAG numsuc
    SEEK m.sdo_numsuc
    m.nomsuc = IIF(FOUND(),estructura    t_nomsuc,"")
    SHOW GETS
  ELSE
    DO msgerror.spr WITH 2
  ENDIF
  _CUROBJ = OBJNUM(m.botcdir)

ELSE
  _CUROBJ = OBJNUM(m.numcta)

ENDIF


*     ************************************************************
*     *
*     * _QUU01GF4R          m.botcdir VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR310,     Record Number:  46
*     * Variable:           m.botcdir
*     * Called By:          VALID Clause
*     * Snippet Number:     3
*     *
*     ************************************************************
```

```
*|*****************************************************************
*|
*|    Function: _QUU01GF4R
*|
*|    Called by: SCR310.SPR
*|
*|*****************************************************************
FUNCTION _quu01gf4r    && m.botcdir VALID
#REGION 1
DO CASE

CASE m.botcdir = 1

CASE m.botcdir = 2
  CLEAR READ
ENDCASE


*: EOF: SCR310.SPR

*    ************************************************
*    *
*    * 11/04/94          SCR320.SPR          00:40:50
*    *
*    ************************************************

PARAMETERS origen, destino, modulo

*    ************************************************
*    *
*    *       SCR320/Windows Setup Code - SECTION 1
*    *
*    ************************************************
*

#REGION 1


PUBLIC lninom
PUBLIC fec_min, fec_max, lim_min, lim_max, mv_min, mv_max, sdo_min, sdo_max, cambio

arrproducto = ""
arrlinea = ""
arrsucursal = ""
arrdivision = ""
lninom=""
fec_min = CTOD("")
fec_max = CTOD("")
lim_min = 0
lim_max = 0
mv_min = 0
mv_max = 0
sdo_min = 0
sdo_max = 0
and_or1 = "AND"
and_or2 = "AND"
and_or3 = "AND"
op1_signo = ">="
op2_signo = ">="
op3_signo = ">="
op4_signo = ">="
```

```
rad_bus_nom = 0
exp_producto = ".T."
exp_estruc = ".T."
exp_nombre = ".T."
exp_fecha = ".T."
exp_limite = ".T."
exp_meses = ".t."
exp_vencido = ".t."

*     ************************************************************
*     *
*     * _QUU01GKHK          m.rad_cl_producto WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  34
*     * Variable:           m.rad_cl_producto
*     * Called By:          WHEN Clause
*     * Snippet Number:     1
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01GKHK
*!
*!     Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01gkhk    && m.rad_cl_producto WHEN
#REGION 1
SHOW GET m.producto DISABLE
SHOW GET m.linea DISABLE

*     ************************************************************
*     *
*     * _QUU01GKL0          m.rad_cl_producto VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  34
*     * Variable:           m.rad_cl_producto
*     * Called By:          VALID Clause
*     * Snippet Number:     2
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01GKL0
*!
*!     Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01gkl0    && m.rad_cl_producto VALID
#REGION 1
DO CASE
CASE m.rad_cl_producto = 1
  m.linea=""
```

```
  SHOW GET m.linea DISABLE
  SHOW GET m.producto ENABLE
  _CUROBJ = OBJNUM(m.producto)
CASE m.rad_cl_producto = 2
  m.producto=""
  SHOW GET m.producto DISABLE
  SHOW GET m.linea ENABLE
  _CUROBJ = OBJNUM(m.linea)
CASE m.rad_cl_producto = 3
  exp_producto = ".T."
ENDCASE
SHOW GETS

*     ************************************************
*     *
*     * _QUU01GKPU         m.producto WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  35
*     * Variable:           m.producto
*     * Called By:          WHEN Clause
*     * Snippet Number:     3
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01GKPU
*!
*!     Called by: SCR320.SPR
*!
*!        Calls: ARREGLO_PRODUCTO   (procedure in TMENU.MPR)
*!
*!*************************************************************************
FUNCTION _quu01gkpu     && m.producto WHEN
#REGION 1
DO arreglo_producto
m.producto = 0
*     ************************************************
*     *
*     * _QUU01GKS9         m.producto VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  35
*     * Variable:           m.producto
*     * Called By:          VALID Clause
*     * Snippet Number:     4
*     *
*     ************************************************

*!*************************************************************************
*!
*!     Function: _QUU01GKS9
*!
*!     Called by: SCR320.SPR
*!
*!*************************************************************************
```

```
FUNCTION _quu01gks9    && m.producto VALID
#REGION 1
exp_producto = "sdo_cveprd = " + ALLTRIM( STR(arrproducto(m.producto,2)) )
_CUROBJ = OBJNUM(m.rad_ci_estruc)

*    ************************************************************
*    *
*    * _QUU01GKVL         m.linea WHEN
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR320,    Record Number:  36
*    * Variable:           m.linea
*    * Called By:          WHEN Clause
*    * Snippet Number:     5
*    *
*    ************************************************************
*
*!*****************************************************************************
*!
*!     Function: _QUU01GKVL
*!
*!     Called by: SCR320.SPR
*!
*!        Calls: ARREGLO_LINEA     (procedure in TMENU.MPR)
*!
*!*****************************************************************************
FUNCTION _quu01gkvl    && m.linea WHEN
#REGION 1
DO arreglo_linea
m.linea = 0

*    ************************************************************
*    *
*    * _QUU01GKXZ         m.linea VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR320,    Record Number:  36
*    * Variable:           m.linea
*    * Called By:          VALID Clause
*    * Snippet Number:     6
*    *
*    ************************************************************
*


*!*****************************************************************************
*!
*!     Function: _QUU01GKXZ
*!
*!     Called by: SCR320.SPR
*!
*!        Uses: TARPROD.DBF          Alias: PRODUCTO
*!
*!     CDX files: TARPROD.CDX
*!
*!*****************************************************************************
FUNCTION _quu01gkxz    && m.linea VALID
#REGION 1
```

```
PRIVATE expresion,i, arrprdlin

SELECT prd_numprd ;
  FROM  tarprod ;
  WHERE prd_numlin = arrlinea(m.linea,2);
  INTO ARRAY arrprdlin

IF ALEN(arrprdlin) = 1
  expresion = "sdo_numprd = " + ALLTRIM( STR(arrprdlin[1]) )
ELSE
  expresion = ALLTRIM( STR(arrprdlin[1]) )
  FOR i=2 TO ALEN(arrprdlin)
    expresion = expresion + "," + ALLTRIM( STR(arrprdlin[i]) )
  ENDFOR
  expresion = "sdo_cveprd IN (" + expresion + ")"
ENDIF

exp_producto = expresion


*     ************************************************************
*     *
*     * _QUU01GL3P          m.rad_cl_estruc WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  37
*     * Variable:           m.rad_cl_estruc
*     * Called By:          WHEN Clause
*     * Snippet Number:     7
*     *
*     ************************************************************
*
*!**************************************************************************
*!
*!      Function: _QUU01GL3P
*!
*!      Called by: SCR320.SPR
*!
*!**************************************************************************
FUNCTION _quu01gl3p     && m.rad_cl_estruc WHEN
#REGION 1
SHOW GET m.suc DISABLE
SHOW GET m.nomsuc DISABLE
SHOW GET m.division DISABLE

*     ************************************************************
*     *
*     * _QUU01GL6D          m.rad_cl_estruc VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  37
*     * Variable:           m.rad_cl_estruc
*     * Called By:          VALID Clause
*     * Snippet Number:     8
*     *
*     ************************************************************
*
```

```
*|*****************************************************************************
*|
*|     Function: _QUU01GL6D
*|
*|     Called by: SCR320.SPR
*|
*|*****************************************************************************
FUNCTION _quu01gl6d    &&  m.rad_cl_estruc VALID
#REGION 1
DO CASE
CASE m.rad_cl_estruc = 1
  m.division = ""
  SHOW GET m.division DISABLE
  SHOW GET m.suc ENABLE
  SHOW GET m.nomsuc ENABLE
  _CUROBJ = OBJNUM(m.suc)
CASE m.rad_cl_estruc = 2
  m.suc=""
  m.nomsuc=""
  SHOW GET m.suc DISABLE
  SHOW GET m.nomsuc DISABLE
  SHOW GET m.division ENABLE
  _CUROBJ = OBJNUM(m.division)
CASE m.rad_cl_estruc = 3
  exp_estruc = ".T."
ENDCASE
SHOW GETS

*     ************************************************************
*     *
*     * _QUU01GLBN         m.suc VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  38
*     * Variable:           m.suc
*     * Called By:          VALID Clause
*     * Snippet Number:     9
*     *
*     ************************************************************
*
*|*****************************************************************************
*|
*|     Function: _QUU01GLBN
*|
*|     Called by: SCR320.SPR
*|
*|*****************************************************************************
FUNCTION _quu01glbn    &&  m.suc VALID
#REGION 1
IF EMPTY(m.suc)
  _CUROBJ = OBJNUM(m.nomsuc)
ELSE
  SELECT nomestru
  SET ORDER TO TAG numreg OF tarnomes.cdx
  SEEK m.suc
  IF FOUND()
    m.nomsuc = nomestru.nomsuc
    SHOW GETS
  ELSE
```

```
    WAIT "SUCURSAL INEXISTENTE" WINDOW
  ENDIF
ENDIF

*     ************************************************
*     *
*     * _QUU01GLFZ          m.nomsuc WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR320,    Record Number:  39
*     * Variable:            m.nomsuc
*     * Called By:           WHEN Clause
*     * Snippet Number:      10
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01GLFZ
*!
*!     Called by: SCR320.SPR
*!
*!        Calls: ARREGLO_SUCURSAL   (procedure in TMENU.MPR)
*!
*!*************************************************************************
FUNCTION _quu01glfz     && m.nomsuc WHEN
#REGION 1
DO arreglo_sucursal
m.nomsuc = 0

*     ************************************************
*     *
*     * _QUU01GLIC          m.nomsuc VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR320,    Record Number:  39
*     * Variable:            m.nomsuc
*     * Called By:           VALID Clause
*     * Snippet Number:      11
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01GLIC
*!
*!     Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01glic     && m.nomsuc VALID
#REGION 1
m.suc = arrsucursal(m.nomsuc,2)
SHOW GETS
exp_estruc = "sdo_numsuc = " + ALLTRIM( STR(arrsucursal(m.nomsuc,2)) )
```

```
*     ************************************************************
*     *
*     * _QUU01GLM1          m.division WHEN
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  40
*     * Variable:           m.division
*     * Called By:          WHEN Clause
*     * Snippet Number:     12
*     *
*     ************************************************************
*
*|*****************************************************************************
*|
*|     Function: _QUU01GLM1
*|
*|     Called by: SCR320.SPR
*|
*|        Calls: ARREGLO_DIVISION  (procedure in TMENU.MPR)
*|
*|*****************************************************************************
FUNCTION _quu01glm1     && m.division WHEN
#REGION 1
DO arreglo_division
m.division = 0

*     ************************************************************
*     *
*     * _QUU01GLOF          m.division VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  40
*     * Variable:           m.division
*     * Called By:          VALID Clause
*     * Snippet Number:     13
*     *
*     ************************************************************
*

*|*****************************************************************************
*|
*|     Function: _QUU01GLOF
*|
*|     Called by: SCR320.SPR
*|
*|*****************************************************************************
FUNCTION _quu01glof     && m.division VALID
#REGION 1
exp_estruc = "est_numdiv = " + ALLTRIM( STR(arrdivision(m.division,2)) )


*     ************************************************************
*     *
*     * _QUU01GLRK          m.rad_bus_nom WHEN
*     *
*     * Function Origin:
*     *
```

```
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  41
*     * Variable:           m.rad_bus_nom
*     * Called By:          WHEN Clause
*     * Snippet Number:     14
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!    Function: _QUU01GLRK
*!
*!    Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01glrk    && m.rad_bus_nom WHEN
#REGION 1
m.ininom= ""

*     ************************************************
*     *
*     * _QUU01GLTT          m.rad_bus_nom VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  41
*     * Variable:           m.rad_bus_nom
*     * Called By:          VALID Clause
*     * Snippet Number:     15
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!    Function: _QUU01GLTT
*!
*!    Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01gltt    && m.rad_bus_nom VALID
#REGION 1
_CUROBJ = OBJNUM(m.ininom)

*     ************************************************
*     *
*     * _QUU01GLXE          m.ininom VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  42
*     * Variable:           m.ininom
*     * Called By:          VALID Clause
*     * Snippet Number:     16
*     *
*     ************************************************
```

```
*!*************************************************************************
*!
*!      Function: _QUU01GLXE
*!
*!      Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01glxe     && m.lninom VALID
#REGION 1
IF !EMPTY(m.lninom)
  IF rad_bus_nom = 2
    exp_nombre = "sdo_nomcli LIKE "+ ALLTRIM(m.lninom) + "%"
  ELSE
    exp_nombre = "sdo_nomcli LIKE "+ "'%" + ALLTRIM(m.lninom) + "%'
  ENDIF
ELSE
  exp_nombre = .T.
ENDIF


*     ************************************************
*     *
*     * _QUU01GM1K         op1 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,     Record Number:  43
*     * Variable:           op1
*     * Called By:          VALID Clause
*     * Snippet Number:     17
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!      Function: _QUU01GM1K
*!
*!      Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01gm1k     && op1 VALID
#REGION 1
IF op1_signo = "="
  SHOW GET m.op1,1 PROMPT ">="
  op1_signo = ">="
  SHOW GET m.fec_max ENABLE
ELSE
  SHOW GET m.op1,1 PROMPT "="
  op1_signo = "="
  m.fec_max = CTOD('')
  SHOW GET m.fec_max DISABLE
  SHOW GETS
ENDIF
_CUROBJ = OBJNUM(m.fec_min)
```

```
*     ************************************************************
*     *
*     * _QUU01GM7M          m.oplog1 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  46
*     * Variable:           m.oplog1
*     * Called By:          VALID Clause
*     * Snippet Number:     18
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01GM7M
*!
*!     Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01gm7m    && m.oplog1 VALID
#REGION 1
IF and_or1 = "OR"
  SHOW GET m.oplog1,1 PROMPT "AND"
  and_or1 = "AND"
ELSE
  SHOW GET m.oplog1,1 PROMPT "OR"
  and_or1 = "OR"
ENDIF

*     ************************************************************
*     *
*     * _QUU01GMBD          op2 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  47
*     * Variable:           op2
*     * Called By:          VALID Clause
*     * Snippet Number:     19
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!     Function: _QUU01GMBD
*!
*!     Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01gmbd    && op2 VALID
#REGION 1
IF op2_signo = "="
  SHOW GET m.op2,1 PROMPT ">="
  op2_signo = ">="
  SHOW GET m.lim_max ENABLE
ELSE
  SHOW GET m.op2,1 PROMPT "="
  op2_signo = "="
```

```
  m.lim_max = 0
  SHOW GET m.lim_max DISABLE
  SHOW GETS
ENDIF
_CUROBJ = OBJNUM(m.lim_min)


*     ************************************************************
*     *
*     * _QUU01GMI9          m.oplog2 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR320,    Record Number:  50
*     * Variable:            m.oplog2
*     * Called By:           VALID Clause
*     * Snippet Number:      20
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!      Function: _QUU01GMI9
*!
*!     Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01gmi9    && m.oplog2 VALID
#REGION 1
IF and_or2 = "OR"
  SHOW GET m.oplog2,1 PROMPT "AND"
  and_or2 = "AND"
ELSE
  SHOW GET m.oplog2,1 PROMPT "OR"
  and_or2 = "OR"
ENDIF

*     ************************************************************
*     *
*     * _QUU01GMLZ          op3 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR320,    Record Number:  51
*     * Variable:            op3
*     * Called By:           VALID Clause
*     * Snippet Number:      21
*     *
*     ************************************************************
*
*!*************************************************************************
*!
*!      Function: _QUU01GMLZ
*!
*!     Called by: SCR320.SPR
*!
*!*************************************************************************
```

```
FUNCTION _quu01gmlz    && op3 VALID
#REGION 1
IF op3_signo = "="
  SHOW GET m.op3,1 PROMPT ">="
  op3_signo = ">="
  SHOW GET m.mv_max ENABLE
ELSE
  SHOW GET m.op3,1 PROMPT "="
  op3_signo = "="
  m.mv_max = 0
  SHOW GET m.mv_max DISABLE
  SHOW GETS
ENDIF
_CUROBJ = OBJNUM(m.mv_min)



*     ************************************************
*     *
*     * _QUU01GMRX          m.oplog3 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  54
*     * Variable:           m.oplog3
*     * Called By:          VALID Clause
*     * Snippet Number:     22
*     *
*     ************************************************
*
*!**************************************************************************
*!
*!      Function: _QUU01GMRX
*!
*!     Called by: SCR320.SPR
*!
*!**************************************************************************
FUNCTION _quu01gmrx    && m.oplog3 VALID
#REGION 1
IF and_or3 = "OR"
  SHOW GET m.oplog3,1 PROMPT "AND"
  and_or3 = "AND"
ELSE
  SHOW GET m.oplog3,1 PROMPT "OR"
  and_or3 = "OR"
ENDIF

*     ************************************************
*     *
*     * _QUU01GMVL          op4 VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR320,    Record Number:  55
*     * Variable:           op4
*     * Called By:          VALID Clause
*     * Snippet Number:     23
*     *
*     ************************************************
```

```
*
*!*************************************************************************
*!
*!      Function: _QUU01GMVL
*!
*!      Called by: SCR320.SPR
*!
*!*************************************************************************
FUNCTION _quu01gmvl     && op4 VALID
#REGION 1
IF op4_signo = "="
  SHOW GET m.op4,1 PROMPT ">="
  op4_signo = ">="
  SHOW GET m.sdo_max ENABLE
ELSE
  SHOW GET m.op4,1 PROMPT "="
  op4_signo = "="
  m.sdo_max = 0
  SHOW GET m.sdo_max DISABLE
  SHOW GETS
ENDIF
_CUROBJ = OBJNUM(m.sdo_min)

*     ************************************************
*     *
*     * _QUU01GN1M          m.bot_cons VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:       Windows
*     * From Screen:         SCR320,     Record Number:  58
*     * Variable:            m.bot_cons
*     * Called By:           VALID Clause
*     * Snippet Number:      24
*     *
*     ************************************************
*
*!*************************************************************************
*!
*!      Function: _QUU01GN1M
*!
*!      Called by: SCR320.SPR
*!
*!         Uses: TARSDOS.DBF          Alias: SALDOS
*!             : &ORIGEN
*!
*!      CDX files: TARSDOS.CDX
*!
*!*************************************************************************
FUNCTION _quu01gn1m     && m.bot_cons VALID
#REGION 1
DO CASE
CASE bot_cons = 1

  IF EMPTY(fec_min) AND EMPTY(m.fec_max)
    exp_fecha = ".T."
  ELSE
    exp_fecha = IIF(op1_signo = "=", "sdo_fecapt = m.fec_min", ;
      IIF(EMPTY(m.fec_max), "sdo_fecapt > m.fec_min" , ;
      "sdo_fecapt >= m.fec_min AND sdo_fecapt <= m.fec_max"))
  ENDIF
```

```
IF EMPTY(m.lim_min) AND EMPTY(m.lim_max)
  exp_limite = ".T."
ELSE
  exp_limite = IIF(op2_signo = "=", "sdo_limcre = m.lim_min", ;
    IIF(m.lim_max = 0, "sdo_limcre > m.lim_min" , ;
    "sdo_limcre >= m.lim_min AND sdo_limcre <= m.lim_max"))
ENDIF


IF EMPTY(m.mv_min) AND EMPTY(m.mv_max)
  exp_meses = ".T."
ELSE
  exp_meses = IIF(op3_signo = "=", "sdo_nummv = m.mv_min", ;
    IIF(m.mv_max = 0, "sdo_nummv > m.mv_min" , ;
    "sdo_nummv >= m.mv_min AND sdo_nummv <= m.mv_max"))
ENDIF


IF EMPTY(m.sdo_min) AND EMPTY(m.sdo_max)
  exp_vencido = ".T."
ELSE
  exp_vencido= IIF(op4_signo = "=", "sdo_sdovdo = m.sdo_min", ;
    IIF(m.sdo_max = 0, "sdo_sdovdo > m.sdo_min" , ;
    "sdo_sdovdo >= m.sdo_min AND sdo_sdovdo <= m.sdo_max"))
ENDIF


IF EMPTY(origen)
  SELECT sdo_numcta, sdo_nomcli, sdo_limcre, sdo_fecapt, sdo_nummv, sdo_sdovdo ;
    FROM tarsdos ;
    WHERE &exp_producto AND &exp_estruc AND &exp_nombre AND ;
    &exp_fecha &and_or1 &exp_limite &and_or2 &exp_meses &and_or3 &exp_vencido
ELSE
  DO CASE
  CASE modulo = 1 OR modulo = 3                    && APERTURA Y MANEJO DE CREDITO
    && CARTERA VENCIDA

    SELECT cuenta;
      FROM &origen, tarsdos ;
      WHERE cuenta = sdo_numcta AND ;
      &exp_producto AND ;
      &exp_estruc AND ;
      &exp_nombre AND ;
      &exp_fecha &and_or1 ;
      &exp_limite &and_or2 ;
      &exp_meses &and_or3 ;
      &exp_vencido ;
      INTO TABLE &destino

  CASE modulo = 2                                  && SOBREGIROS

    SELECT cuenta,ps ;
      FROM &origen, tarsdos ;
      WHERE cuenta = sdo_numcta AND ;
      &exp_producto AND ;
      &exp_estruc AND ;
      &exp_nombre AND ;
```

```
        &exp_fecha &and_or1 ;
        &exp_limite &and_or2 ;
        &exp_meses &and_or3 ;
        &exp_vencido ;
        INTO TABLE &destino

    CASE modulo = 4                          && CARTERA PREVENTIVA

      SELECT cuenta, ntarj;
        FROM &origen, tarsdos ;
        WHERE cuenta = sdo_numcta AND ;
        &exp_producto AND ;
        &exp_estruc AND ;
        &exp_nombre AND ;
        &exp_fecha &and_or1 ;
        &exp_limite &and_or2 ;
        &exp_meses &and_or3 ;
        &exp_vencido ;
        INTO TABLE &destino

    ENDCASE

  ENDIF
  CLEAR READ

CASE bot_cons = 2
  CLEAR READ

CASE bot_cons = 3

  CLEAR READ
ENDCASE

*: EOF: SCR320.SPR

*    ************************************************************
*    *
*    * 11/04/94        SCR410.SPR              00:41:01
*    *
*    ************************************************************


*    ************************************************************
*    *
*    *       SCR410/Windows Setup Code - SECTION 1
*    *
*    ************************************************************
*

#REGION 1

DIMENSION  arrinddisp[14,2], arrindsel[14,2]

PUBLIC flag_empty

arrindsel= ""

arrinddisp[1,1] = "Saldos/Cuenta"
arrinddisp[2,1] = "Altas"
arrinddisp[3,1] = "Incre_limite"
arrinddisp[4,1] = "Operador"
```

```
arrinddisp[5,1] = "Puesto/opera"
arrinddisp[6,1] = "Juridico"
arrinddisp[7,1] = "Abogados"
arrinddisp[8,1] = "Castgos"
arrinddisp[9,1] = "Boletin"
arrinddisp[10,1] = "Estructura"
arrinddisp[11,1] = "Nom_estructura"
arrinddisp[12,1] = "Usuarios"
arrinddisp[13,1] = "Bitacora"
arrinddisp[14,1] = "Producto"

FOR i=1 TO 14
  arrinddisp[i,2] = i
ENDFOR


*     ************************************************
*     *
*     *          SCR410/Windows Cleanup Code
*     *
*     ************************************************

*     ************************************************
*     *
*     *  SCR410/Windows Supporting Procedures and Functions
*     *
*     ************************************************
*

#REGION 1
*|*****************************************************************************
*|
*|    Procedure: SET_UNSET_ALL
*|
*|    Called by: _QUU01GSSC()      (function  in SCR410.SPR)
*|          : MOVING          (procedure in SCR410.SPR)
*|
*|*****************************************************************************
PROCEDURE set_unset_all
PARAMETERS arreglo, flag_set

PRIVATE i, long
long = ALEN(arreglo,1)
DIMENSION arreglo(long,2)

IF flag_set
  FOR i = 1 TO long
    IF SUBSTR(arreglo[i,1],1,1) <> "*"
      arreglo[i,1] = "*"+arreglo[i,1]
    ENDIF
  ENDFOR
ELSE
  FOR i = 1 TO long
    IF SUBSTR(arreglo[i,1],1,1) = "*"
      arreglo[i,1] = SUBSTR(arreglo[i,1],2,LEN(arreglo[i,1]))
    ENDIF
  ENDFOR
ENDIF
RETURN

*****
```

```
*|*****************************************************************
*|
*|    Procedure: MOVING
*|
*|    Called by: _QUU01GSJK()     (function in SCR410.SPR)
*|
*|      Calls: SET_UNSET_ALL     (procedure in SCR410.SPR)
*|
*|*****************************************************************
PROCEDURE moving
PARAMETERS boton,arr1,arr2


PRIVATE m.i, m.lugares, longuitud, m.cont
m.i     = 0
m.lugares = ALEN(arr1,1)

DIMENSION arr1(m.lugares,2)
DIMENSION arr2(m.lugares,2)

flag_empty=.F.
DO CASE
CASE boton = 1
  arr2 = ""
  DO set_unset_all WITH arr1,.T.
CASE boton = 2
  DO set_unset_all WITH arr1,.F.
  arr2 = ""
CASE boton = 3
  SET EXACT OFF
  flag_empty = .F.
  IF ASCAN(arr1,"**") != 0
    m.cont = 0
    FOR m.i = 1 TO m.lugares
      IF SUBSTR(arr1[m.i,1],1,1) = "**"
        m.longitud = LEN(arr1[m.i,1])
        m.cont = m.cont +1
        arr2[m.cont,1] = SUBSTR(arr1[m.i,1],2,m.longitud)
        arr2[m.cont,2] = arr1[m.i,2]
      ENDIF
    ENDFOR
    DIMENSION arr2(m.cont,2)
  ELSE
    flag_empty = .T.
  ENDIF
  SET EXACT ON
ENDCASE
SHOW GETS

*     ************************************************
*     * _QUU01GSDV         m.inddisp VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR410,    Record Number:   5
*     * Variable:           m.inddisp
*     * Called By:          VALID Clause
*     * Snippet Number:     1
*     *
*     ************************************************
```

```
*
*|*****************************************************************************
*|
*|    Function: _QUU01GSDV
*|
*|    Called by: SCR410.SPR
*|
*|*****************************************************************************
FUNCTION _quu01gsdv    && m.inddisp VALID
#REGION 1
IF SUBSTR(arrinddisp[m.inddisp,1],1,1) <> "*"
  * SELECCIONANDO EL AREA
  *
  arrinddisp[m.inddisp,1] = "*"+arrinddisp[m.inddisp,1]
ELSE
  *
  * DESACTIVANDO LAS AREAS
  *
  m.longitud = LEN(arrinddisp[m.inddisp,1])
  arrinddisp[m.inddisp,1] = SUBSTR(arrinddisp[m.inddisp,1],2,m.longitud)
ENDIF

SHOW GETS


*    ************************************************
*    *
*    * _QUU01GSJK          m.mover2 VALID
*    *
*    * Function Origin:
*    *
*    * From Platform:      Windows
*    * From Screen:        SCR410,    Record Number:   6
*    * Variable:           m.mover2
*    * Called By:          VALID Clause
*    * Snippet Number:     2
*    *
*    ************************************************
*
*|*****************************************************************************
*|
*|    Function: _QUU01GSJK
*|
*|    Called by: SCR410.SPR
*|
*|       Calls: MOVING            (procedure in SCR410.SPR)
*|
*|*****************************************************************************
FUNCTION _quu01gsjk    && m.mover2 VALID
#REGION 1
DO moving WITH m.mover2,arrinddisp, arrindsel
IF flag_empty
  WAIT "Error, archivos no seleccionados" WINDOW
  _CUROBJ = OBJNUM(m.inddisp)
ELSE
  IF mover2 = 3
    _CUROBJ = OBJNUM(m.boton)
  ELSE
    _CUROBJ = OBJNUM(m.inddisp)
  ENDIF
ENDIF
```

```
*     ************************************************************
*     *
*     * _QUU01GSOF          m.indsel VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR410,    Record Number:  7
*     * Variable:           m.indsel
*     * Called By:          VALID Clause
*     * Snippet Number:     3
*     *
*     ************************************************************
*


*!*************************************************************************
*!
*!      Function: _QUU01GSOF
*!
*!      Called by: SCR410.SPR
*!
*!*************************************************************************
FUNCTION _quu01gsof     && m.indsel VALID

*     ************************************************************
*     *
*     * _QUU01GSSC          m.boton VALID
*     *
*     * Function Origin:
*     *
*     * From Platform:      Windows
*     * From Screen:        SCR410,    Record Number:  9
*     * Variable:           m.boton
*     * Called By:          VALID Clause
*     * Snippet Number:     4
*     *
*     ************************************************************
*

*!*************************************************************************
FUNCTION _quu01gssc     && m.boton VALID
#REGION 1
DO CASE
CASE m.boton = 1

  IF EMPTY(arrindsel)
    WAIT "Error, no ha seleccionado los archivos" WINDOW
    _CUROBJ = OBJNUM(m.inddisp)
  ELSE
    CLOSE DATABASES
    SET EXCL ON
    SET SAFETY OFF

    WAIT "Reindexando..." WINDOW NOWAIT

    FOR i= 1 TO ALEN(arrindsel,1)

      SELECT 1
      DO CASE
```

```
CASE arrindsel[i,2] = 1

  USE tarsdos
  INDEX ON sdo_cveprd TAG numprd
  INDEX ON sdo_numcta TAG numcta

CASE arrindsel[1,2] = 2

  USE taraltas
  INDEX ON alt_numope TAG numope
  INDEX ON alt_numcta TAG numcta

CASE arrindsel[1,2] = 3

  USE tarinlim
  INDEX ON lil_numope TAG numope
  INDEX ON lil_numcta TAG numcta


CASE arrindsel[1,2] = 4

  USE taropera
  INDEX ON ope_numope TAG numope

CASE arrindsel[1,2] = 5

  USE tarptos
  INDEX ON pto_nompto TAG pto_nompto

CASE arrindsel[1,2] = 6

  USE tarjurid
  INDEX ON jur_numcta TAG numcta
  INDEX ON jur_numabo TAG numabo

CASE arrindsel[1,2] = 7

  USE taraboga
  INDEX ON abo_numabo TAG numabo

CASE arrindsel[1,2] = 8

  USE tarcasti
  INDEX ON cas_numcta TAG numcta

CASE arrindsel[1,2] = 9

  USE tarbolet
  INDEX ON bol_numcta TAG numcta

CASE arrindsel[1,2] = 10

  USE tarestru
  INDEX ON est_numsuc TAG numsuc
  INDEX ON est_numare TAG numare
  INDEX ON est_numreg TAG numreg

CASE arrindsel[1,2] = 11

  USE tarnomes
  INDEX ON nes_numare TAG numare
```

```
        INDEX ON nes_numdiv TAG numdiv
        INDEX ON nes_numreg TAG numreg

      CASE arrindsel[1,2] = 12

        USE tusuario
        INDEX ON usu_login TAG usu_login

      CASE arrindsel[1,2] = 13

        USE tbitacor
        INDEX ON bit_login TAG bit_login

      CASE arrindsel[1,2] = 14

        USE tarprod
        INDEX ON prd_numprd TAG numprd

      ENDCASE
    ENDFOR

    USE
    SET SAFETY ON
    WAIT "Proceso terminado" WINDOW NOWAIT
    DO set_unset_all WITH arrinddisp,.F.
    arrindsel = ""
    SHOW GETS
  ENDIF

CASE m.boton = 2
  CLEAR READ
ENDCASE

*: EOF: SCR410.SPR

*!*************************************************************************
*!
*: Procedure file: C:\SDTAR\PRGS\SEGURID.PRG
*:       System: Sistema de Desempeño de Tarjetas
*:       Author: David Jimenez & Adolfo Avila
*:    Copyright (c) 1994, Banamex - Accival
*: Last modified: 06/04/94 at 15:10:16
*:
*: Procs & Fncts: FILL_TABLE
*:              : ENCRIPTA
*:              : DESENCRIPTA
*:
*:    Documented 00:43:48                  FoxDoc version 3.00a
*!*************************************************************************



* ASIGNA VALORES ALEATORIOS A LA TABLA OFFSET,LOS CUALES REPRESENTAN
* LOS CARACTERES DE SUSTITUCION PARA CADA LETRA DEL ALFABETO, CON LA
* FINALIDAD DE ENCRIPTAR UNA CADENA DE CARACTERES. EL VALOR DE SUSTI
* TUCION DE UNA LETRA VARIA DE ACUERDO A SU POSICION (1..6) DENTRO DE
* LA CADENA DE CARACTERES.

*!*************************************************************************
*!
*!    Procedure: FILL_TABLE
```

```
*|
*|    Called by: TMENU.MPR
*|
*|*************************************************************************
PROCEDURE fill_table
EXTERNAL ARRAY offset

offset[1,1] = CHR(      78      )  &&CARACTER EQUIVALENTE PARA LA LETRA  A
offset[1,2] = CHR(      159     )
offset[1,3] = CHR(      46      )
offset[1,4] = CHR(      142     )
offset[1,5] = CHR(      178     )
offset[1,6] = CHR(      78      )
offset[1,7] = CHR(      201     )
offset[1,8] = CHR(      194     )


offset[2,1] = CHR(      79      )  &&CARACTER EQUIVALENTE PARA LA LETRA  B
offset[2,2] = CHR(      197     )
offset[2,3] = CHR(      96      )
offset[2,4] = CHR(      231     )
offset[2,5] = CHR(      84      )
offset[2,6] = CHR(      77      )
offset[2,7] = CHR(      167     )
offset[2,8] = CHR(      82      )

offset[3,1] = CHR(      191     )       &&CARACTER EQUIVALENTE PARA LA LETRA  C
offset[3,2] = CHR(      65      )
offset[3,3] = CHR(      149     )
offset[3,4] = CHR(      89      )
offset[3,5] = CHR(      116     )
offset[3,6] = CHR(      114     )
offset[3,7] = CHR(      164     )
offset[3,8] = CHR(      242     )

offset[4,1] = CHR(      99      )       &&CARACTER EQUIVALENTE PARA LA LETRA  D
offset[4,2] = CHR(      98      )
offset[4,3] = CHR(      94      )
offset[4,4] = CHR(      98      )
offset[4,5] = CHR(      162     )
offset[4,6] = CHR(      204     )
offset[4,7] = CHR(      35      )
offset[4,8] = CHR(      234     )

offset[5,1] = CHR(      155     )       &&CARACTER EQUIVALENTE PARA LA LETRA  E
offset[5,2] = CHR(      212     )
offset[5,3] = CHR(      136     )
offset[5,4] = CHR(      202     )
offset[5,5] = CHR(      137     )
offset[5,6] = CHR(      252     )
offset[5,7] = CHR(      207     )
offset[5,8] = CHR(      231     )

offset[6,1] = CHR(      97      )       &&CARACTER EQUIVALENTE PARA LA LETRA  F
offset[6,2] = CHR(      161     )
offset[6,3] = CHR(      236     )
offset[6,4] = CHR(      197     )
offset[6,5] = CHR(      99      )
offset[6,6] = CHR(      214     )
offset[6,7] = CHR(      225     )
offset[6,8] = CHR(      44      )
```

```
offset[7,1] = CHR(  156  )      &&CARACTER EQUIVALENTE PARA LA LETRA  G
offset[7,2] = CHR(  214  )
offset[7,3] = CHR(  240  )
offset[7,4] = CHR(  94   )
offset[7,5] = CHR(  198  )
offset[7,6] = CHR(  89   )
offset[7,7] = CHR(  88   )
offset[7,8] = CHR(  154  )

offset[8,1] = CHR(  142  )      &&CARACTER EQUIVALENTE PARA LA LETRA  H
offset[8,2] = CHR(  147  )
offset[8,3] = CHR(  202  )
offset[8,4] = CHR(  111  )
offset[8,5] = CHR(  113  )
offset[8,6] = CHR(  236  )
offset[8,7] = CHR(  79   )
offset[8,8] = CHR(  47   )

offset[9,1] = CHR(  166  )      &&CARACTER EQUIVALENTE PARA LA LETRA  I
offset[9,2] = CHR(  44   )
offset[9,3] = CHR(  99   )
offset[9,4] = CHR(  85   )
offset[9,5] = CHR(  59   )
offset[9,6] = CHR(  35   )
offset[9,7] = CHR(  93   )
offset[9,8] = CHR(  39   )

offset[10,1] = CHR(  73  )      &&CARACTER EQUIVALENTE PARA LA LETRA  J
offset[10,2] = CHR(  251 )
offset[10,3] = CHR(  218 )
offset[10,4] = CHR(  141 )
offset[10,5] = CHR(  174 )
offset[10,6] = CHR(  98  )
offset[10,7] = CHR(  120 )
offset[10,8] = CHR(  45  )

offset[11,1] = CHR(  86  )      &&CARACTER EQUIVALENTE PARA LA LETRA  K
offset[11,2] = CHR(  233 )
offset[11,3] = CHR(  121 )
offset[11,4] = CHR(  237 )
offset[11,5] = CHR(  252 )
offset[11,6] = CHR(  201 )
offset[11,7] = CHR(  138 )
offset[11,8] = CHR(  107 )

offset[12,1] = CHR(  117 )      &&CARACTER EQUIVALENTE PARA LA LETRA  L
offset[12,2] = CHR(  196 )
offset[12,3] = CHR(  135 )
offset[12,4] = CHR(  35  )
offset[12,5] = CHR(  65  )
offset[12,6] = CHR(  111 )
offset[12,7] = CHR(  155 )
offset[12,8] = CHR(  112 )

offset[13,1] = CHR(  35  )      &&CARACTER EQUIVALENTE PARA LA LETRA  M
offset[13,2] = CHR(  112 )
offset[13,3] = CHR(  252 )
offset[13,4] = CHR(  241 )
offset[13,5] = CHR(  78  )
offset[13,6] = CHR(  38  )
```

```
offset[13,7] = CHR(  55  )
offset[13,8] = CHR(  159 )

offset[14,1] = CHR(  154 )    &&CARACTER EQUIVALENTE PARA LA LETRA  N
offset[14,2] = CHR(  118 )
offset[14,3] = CHR(  146 )
offset[14,4] = CHR(  238 )
offset[14,5] = CHR(  229 )
offset[14,6] = CHR(  234 )
offset[14,7] = CHR(  98  )
offset[14,8] = CHR(  190 )

offset[15,1] = CHR(  177 )    &&CARACTER EQUIVALENTE PARA LA LETRA  O
offset[15,2] = CHR(  250 )
offset[15,3] = CHR(  168 )
offset[15,4] = CHR(  56  )
offset[15,5] = CHR(  132 )
offset[15,6] = CHR(  229 )
offset[15,7] = CHR(  90  )
offset[15,8] = CHR(  160 )

offset[16,1] = CHR(  150 )    &&CARACTER EQUIVALENTE PARA LA LETRA  P
offset[16,2] = CHR(  135 )
offset[16,3] = CHR(  176 )
offset[16,4] = CHR(  105 )
offset[16,5] = CHR(  44  )
offset[16,6] = CHR(  246 )
offset[16,7] = CHR(  149 )
offset[16,8] = CHR(  36  )

offset[17,1] = CHR(  225 )    &&CARACTER EQUIVALENTE PARA LA LETRA  Q
offset[17,2] = CHR(  179 )
offset[17,3] = CHR(  76  )
offset[17,4] = CHR(  134 )
offset[17,5] = CHR(  36  )
offset[17,6] = CHR(  240 )
offset[17,7] = CHR(  75  )
offset[17,8] = CHR(  223 )

offset[18,1] = CHR(  112 )    &&CARACTER EQUIVALENTE PARA LA LETRA  R
offset[18,2] = CHR(  129 )
offset[18,3] = CHR(  109 )
offset[18,4] = CHR(  68  )
offset[18,5] = CHR(  191 )
offset[18,6] = CHR(  103 )
offset[18,7] = CHR(  254 )
offset[18,8] = CHR(  188 )

offset[19,1] = CHR(  72  )    &&CARACTER EQUIVALENTE PARA LA LETRA  S
offset[19,2] = CHR(  109 )
offset[19,3] = CHR(  187 )
offset[19,4] = CHR(  254 )
offset[19,5] = CHR(  81  )
offset[19,6] = CHR(  213 )
offset[19,7] = CHR(  186 )
offset[19,8] = CHR(  223 )

offset[20,1] = CHR(  114 )    &&CARACTER EQUIVALENTE PARA LA LETRA  T
offset[20,2] = CHR(  135 )
offset[20,3] = CHR(  151 )
offset[20,4] = CHR(  65  )
```

```
offset[20,5] = CHR(     133     )
offset[20,6] = CHR(     129     )
offset[20,7] = CHR(     120     )
offset[20,8] = CHR(     113     )

offset[21,1] = CHR(     138     )     &&CARACTER EQUIVALENTE PARA LA LETRA  U
offset[21,2] = CHR(     136     )
offset[21,3] = CHR(     107     )
offset[21,4] = CHR(     39      )
offset[21,5] = CHR(     202     )
offset[21,6] = CHR(     195     )
offset[21,7] = CHR(     153     )
offset[21,8] = CHR(     75      )

offset[22,1] = CHR(     131     )     &&CARACTER EQUYVALENTE PARA LA LETRA  V
offset[22,2] = CHR(     236     )
offset[22,3] = CHR(     86      )
offset[22,4] = CHR(     95      )
offset[22,5] = CHR(     92      )
offset[22,6] = CHR(     128     )
offset[22,7] = CHR(     63      )
offset[22,8] = CHR(     222     )

offset[23,1] = CHR(     220     )     &&CARACTER EQUIVALENTE PARA LA LETRA  W
offset[23,2] = CHR(     128     )
offset[23,3] = CHR(     191     )
offset[23,4] = CHR(     239     )
offset[23,5] = CHR(     41      )
offset[23,6] = CHR(     162     )
offset[23,7] = CHR(     74      )
offset[23,8] = CHR(     170     )

offset[24,1] = CHR(     45      )     &&CARACTER EQUIVALENTE PARA LA LETRA  X
offset[24,2] = CHR(     90      )
offset[24,3] = CHR(     152     )
offset[24,4] = CHR(     204     )
offset[24,5] = CHR(     90
```

```
* ENCRIPTA  Encripta una cadena de caracteres
* PARAMETROS cadena:  Cadena a encriptar
*         cad_enc:  Cadena encriptada
*
* NOTA: Sustituye cada caracter por el que le
*      le corresponde segun la tabla offset.

*|*****************************************************************
*|
*|     Procedure: ENCRIPTA
*|
*|     Called by: _QUU01EL50()       (function  in SCRACC.SPR)
*|              : _QUU01F4UI()       (function  in SCR140.SPR)
*|              : _QUU01FDR8()       (function  in SCR150.SPR)
*|              : _QUU01FE94()       (function  in SCR150.SPR)
*|
*|*****************************************************************
PROCEDURE  encripta
PARAMETERS cadena, cad_enc
PRIVATE long_cadena, char_nuevo, pos, x

cad_enc = ""
long_cadena = LEN(ALLTRIM(cadena))
FOR pos=1 TO long_cadena
  x = ASC(SUBS(cadena, pos, 1)) - 64
  char_nuevo = offset[x,pos]
  cad_enc = cad_enc + char_nuevo
ENDFOR
RETURN cad_enc

* DESENCRIPTA  Desencripta una cadena
* PARAMETROS   cad_enc: cadena a desencriptar
*          cadena:  cadena desencriptada
*
* NOTA: Busca cada caracter encriptado en la
*      la tabla offset y devuelve el caracter
*      original.

*|*****************************************************************
*|
*|     Procedure: DESENCRIPTA
*|
*|*****************************************************************
PROCEDURE desencripta
PARAMETERS cad_enc, cadena
PRIVATE long_cadena, char, pos

cadena=""
long_cadena = LEN(ALLTRIM(cadena))
FOR pos   = 1 TO long
  elemento = ASCAN(offset, ASC(SUBS(cad_enc,pos,1)), 1+(pos-1)*26 ,26 )
  char = CHR( ASUBSCRIPT(offset,elemento,1) + 64 )
  cadena = cadena + char
NEXT
RETURN cadena

*: EOF: SEGURID.PRG
```

```
*!*************************************************************************
*!
*! Procedure file: C:\SDTAR\PRGS\CV.PRG
*!       System: Sistema de Desempeño de Tarjetas
*!       Author: David Jimenez & Adolfo Avila
*!    Copyright (c) 1994, Banamex - Accival
*! Last modified: 11/03/94 at 22:43:18
*!
*! Procs & Fncts: CV31
*!              : CV321
*!              : CV322
*!              : CV323
*!              : CV324
*!              : CV325
*!              : CV326
*!              : CV33
*!
*!    Documented 00:44:40                         FoxDoc version 3.00a
*!*************************************************************************
*!*************************************************************************
*!
*!    Procedure: CV31
*!
*!    Called by: CV321          (procedure in CV.PRG)
*!             : CV322          (procedure in CV.PRG)
*!             : CV323          (procedure in CV.PRG)
*!             : CV325          (procedure in CV.PRG)
*!             : CV326          (procedure in CV.PRG)
*!             : CV33           (procedure in CV.PRG)
*!
*!       Uses: TARSDOS.DBF          Alias: SALDOS
*!
*!    CDX files: TARSDOS.CDX
*!
*!*************************************************************************
PROCEDURE cv31                                  && cuentas morosas

SELECT DISTINCT sdo_numcta AS ctas;
  FROM tarsdos ;
  WHERE sdo_cveprd <> 30 AND;
  sdo_numcc <> 18 AND ;
  sdo_cvecst = 0 ;
  INTO TABLE tqcv00

RETUVN

*!*************************************************************************
*!
*!    Procedure: CV321
*!
*!    Called by: _QUU01G0QQ()     (function  in SCR230.SPR)
*!
*!       Calls: CV31              (procedure in CV.PRG)
*!            : ESTRATIFICADO     (procedure in SCR220.SPR)
*!
*!       Uses: TARSDOS.DBF          Alias: SALDOS
*!            : TQCV00.DBF
*!
*!    CDX files: TARSDOS.CDX
*!
*!*************************************************************************
```

```
PROCEDURE cv321                 && cartera apertura reciente
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcv01", origen )
DO cv31

SELECT DISTINCT sdo_numcta AS cuenta;
  FROM tarsdos, tqcv00 ;
  WHERE sdo_numcta = ctas AND ;
  sdo_fecapt IN (&ul12meses) ;
  INTO TABLE tqcv01

*DO calcula_total WITH origen
DO estratificado WITH origen

RETUVN


*!*************************************************************************
*!
*!    Procedure: CV322
*!
*!    Called by: _QUU01G0QQ()     (function  in SCR230.SPR)
*!
*!       Calls: CV31              (procedure in CV.PRG)
*!            : ESTRATIFICADO     (procedure in SCR220.SPR)
*!
*!        Uses: TARSDOS.DBF       Alias: SALDOS
*!            : TQCV00.DBF
*!
*!    CDX files: TARSDOS.CDX
*!
*!*************************************************************************
PROCEDURE cv322                                 && cartera administrativa
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcv02", origen )
DO cv31

SELECT DISTINCT sdo_numcta AS cuenta;
  FROM tarsdos, tqcv00 ;
  WHERE sdo_numcta = ctas AND ;
  sdo_fecapt NOT IN (&ul12meses) AND ;
  sdo_nummv IN (1,2,3) ;
  INTO TABLE tqcv02

*DO calcula_total WITH origen
DO estratificado WITH origen

RETURN


*!*************************************************************************
*!
*!    Procedure: CV323
*!
*!    Called by: _QUU01G0QQ()     (function  in SCR230.SPR)
*!
*!       Calls: CV31              (procedure in CV.PRG)
*!            : ESTRATIFICADO     (procedure in SCR220.SPR)
*!
```

```
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|         : TQCV00.DBF
*|
*|    CDX files: TARSDOS.CDX
*|
*|*************************************************************************
PROCEDURE cv323                                  && cartera prejuridica
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcv03", origen )
DO cv31

SELECT DISTINCT sdo_numcta AS cuenta;
  FROM tarsdos, tqcv00 ;
  WHERE sdo_numcta = ctas AND ;
  sdo_fecapt NOT IN (&ul12meses) AND ;
  sdo_nummv IN (4,5) AND ;
  sdo_sdovdo = sdo_sdoact ;
  INTO TABLE tqcv03

*DO calcula_total WITH origen
DO estratificado WITH origen

RETURN

***
*|*************************************************************************
*|
*|    Procedure: CV324
*|
*|    Called by: _QUU01G0QQ()     (function  in SCR230.SPR)
*|
*|      Calls: ESTRATIFICADO     (procedure in SCR220.SPR)
*|
*|       Uses: TARSDOS.DBF          Alias: SALDOS
*|
*|    CDX files: TARSDOS.CDX
*|
*|*************************************************************************
PROCEDURE cv324                                  && cartera juridica
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcv04", origen )

SELECT DISTINCT sdo_numcta AS cuenta;
  FROM tarsdos ;
  WHERE sdo_cveprd <> 30 AND;
  sdo_numcc = 18 AND ;
  sdo_cvecst = 0 ;
  INTO TABLE tqcv04

*DO calcula_total WITH origen
DO estratificado WITH origen

RETUVN
```

```
*|*****************************************************************
*|
*|    Procedure: CV325
*|
*|    Called by: _QUU01G0QQ()      (function  in SCR230.SPR)
*|
*|       Calls: CV31               (procedure in CV.PRG)
*|            : ESTRATIFICADO     (procedure in SCR220.SPR)
*|
*|       Uses: TARSDOS.DBF         Alias: SALDOS
*|            : TARCASTI.DBF       Alias: CASTIGOS
*|
*|    CDX files: TARSDOS.CDX
*|            : TARCASTI.CDX
*|
*|*****************************************************************
PROCEDURE cv325                                 && cartera en castigos
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcv05", origen )
DO cv31


SELECT DISTINCT sdo_numcta AS cuenta;
  FROM tarsdos, tarcasti ;
  WHERE sdo_numcta = cas_numcta ;
  INTO TABLE tqcv05

*DO calcula_total WITH origen
DO estratificado WITH origen

RETURN


***
*|*****************************************************************
*|
*|    Procedure: CV326
*|
*|    Called by: _QUU01G0QQ()      (function  in SCR230.SPR)
*|
*|       Calls: CV31               (procedure in CV.PRG)
*|            : ESTRATIFICADO     (procedure in SCR220.SPR)
*|
*|       Uses: TARSDOS.DBF         Alias: SALDOS
*|
*|    CDX files: TARSDOS.CDX
*|
*|*****************************************************************
PROCEDURE cv326                                 && cartera en redocumentacion
PARAMETER origen

origen = IIF(EMPTY(origen), "tqcv06", origen )
DO cv31

SELECT DISTINCT sdo_numcta AS cuenta ;
  FROM tarsdos ;
  WHERE sdo_cveprd = 30 ;
  INTO TABLE tqcv06
```

```
*DO calcula_total WITH origen
DO estratificado WITH origen

RETURN

*!*************************************************************************
*!
*!    Procedure: CV33
*!
*!       Calls: CV31            (procedure in CV.PRG)
*!            : ESTRATIFICADO   (procedure in SCR220.SPR)
*!
*!        Uses: TARSDOS.DBF          Alias: SALDOS
*!            : TARJURID.DBF         Alias: JURIDICO
*!
*!    CDX files: TARSDOS.CDX
*!            : TARJURID.CDX
*!
*!*************************************************************************
PROCEDURE cv33                                  && cuentas recuperadas
PARAMETER origen

origen = IIF(EMPTY(origen), "tqsob01", origen )
DO cv31

SELECT DISTINCT sdo_numcta AS cuenta;
  FROM tarsdos, tarjurid ;
  WHERE sdo_numcta = jur_numcta AND ;
  sdo_cveprd <> 30 AND ;
  sdo_numcc = 18 AND ;
  sdo_cvecst = 0 ;
  INTO TABLE tqcv09

*DO calcula_total WITH origen
DO estratificado WITH origen

RETURN

*!*************************************************************************
*!
*!    Procedure: ESTRATIFICADO
*!
*!    Called by: SOB21          (procedure in SCR220.SPR)
*!             : SOB22          (procedure in SCR220.SPR)
*!             : SOB23          (procedure in SCR220.SPR)
*!             : SOB24          (procedure in SCR220.SPR)
*!             : CV321          (procedure in CV.PRG)
*!             : CV322          (procedure in CV.PRG)
*!             : CV323          (procedure in CV.PRG)
*!             : CV324          (procedure in CV.PRG)
*!             : CV325          (procedure in CV.PRG)
*!             : CV326          (procedure in CV.PRG)
*!             : CV33           (procedure in CV.PRG)
*!             : CP01           (procedure in SCR240.SPR)
*!             : CP02           (procedure in SCR240.SPR)
*!             : CP03           (procedure in SCR240.SPR)
*!
*!       Calls: CREACURSOR      (procedure in SCR220.SPR)
*!            : SCR231.SPR
*!
*!        Uses: &ORIGEN
```

```
*|          : TARSDOS.DBF        Alias: SALDOS
*|          : TARESTRU.DBF       Alias: ESTRUCTURA
*|          : TARNOMES.DBF       Alias: NOMESTRU
*|
*|    CDX files: TARSDOS.CDX
*|          : TARESTRU.CDX
*|          : TARNOMES.CDX
*|
*|*************************************************************************
PROCEDURE estratificado
PARAMETER origen

arrest = ""
SELECT nes_nomdiv + " " + ;
  nes_nomdir + " " + ;
  STR(COUNT(*)) + " " + ;
  STR(SUM(sdo_sdovdo)) + " " + ;
  STR(SUM(sdo_sdoact)), est_numdir ;
  FROM &origen, tarsdos, tarestru, tarnomes ;
  WHERE sdo_numcta = cuenta AND ;
  sdo_numsuc = est_numsuc AND ;
  est_numdir = nes_numdir AND ;
  est_numdiv = nes_numdiv AND ;
  (&selno OR &selnte OR &selocc OR &selcto OR &selsur );
  GROUP BY nes_nomdir ;
  INTO ARRAY arrest

DO scr231.spr WITH origen
RETURN

*|*************************************************************************
*|
*|    Procedure: CALCULA_TOTAL
*|
*|    Called by: CCRED01          (procedure in SCR210.SPR)
*|          : CCRED02          (procedure in SCR210.SPR)
*|          : CCRED03          (procedure in SCR210.SPR)
*|          : ASGLIM01         (procedure in SCR210.SPR)
*|          : ASGLIM02         (procedure in SCR210.SPR)
*|          : ASGLIM03         (procedure in SCR210.SPR)
*|          : ASGLIM04         (procedure in SCR210.SPR)
*|          : MANCTA01          (procedure in SCR210.SPR)
*|          : MANCTA02          (procedure in SCR210.SPR)
*|          : SOB21           (procedure in SCR220.SPR)
*|          : SOB22           (procedure in SCR220.SPR)
*|          : SOB23           (procedure in SCR220.SPR)
*|          : SOB24           (procedure in SCR220.SPR)
*|          : CP01           (procedure in SCR240.SPR)
*|          : CP02           (procedure in SCR240.SPR)
*|          : CP03           (procedure in SCR240.SPR)
*|
*|       Uses: TARSDOS.DBF        Alias: SALDOS
*|          : &ORIGEN
*|
*|    CDX files: TARSDOS.CDX
*|
*|*************************************************************************
PROCEDURE calcula_total
PARAMETER origen
```

```
SELECT COUNT(*) AS tctas, ;
  SUM(sdo_sdovdo) AS tsdovdo, ;
  SUM(sdo_sdoact) AS tsdoact ;
  FROM tarsdos, &origen ;
  WHERE sdo_numcta = cuenta AND ;
  &selecciona ;
  INTO CURSOR ctotal

SCATTER MEMVAR

RETURN
*: EOF: CV.PRG
```

APENDICE C

MANUAL DE USUARIO

DEL SDTAR

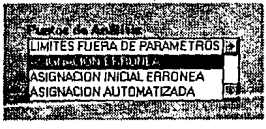
Sistema de Desempeño de Tarjetas Bancarias

El Sistema de Desempeño de Tarjetas Bancarias fue desarrollado en un ambiente gráfico como el de Windows, por lo tanto la operación del sistema se basa en controles que encontramos en todas la aplicaciones típicas de Windows.

Controles utilizados en el Sistema

1. Listas

Seleccione un elemento de los que hay al recorrer la lista. El elemento elegido aparece iluminado. Puede desplazarse con los botones de flechas ubicados a la derecha de las listas.



2. Listas despegables

Despliegue la lista seleccionandola y después presione la tecla de flecha descendente. Presione la flecha descendente para seleccionar el elemento que desea de la lista o presione sobre el elemento que eligió.

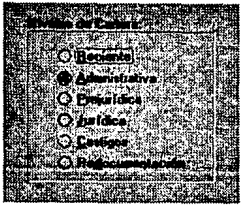
También puede seleccionar un elemento de la lista escribiendo directamente el número de elemento a la izquierda de la lista.

3. Cuadros de Verificación

Selecciona múltuples opciones de un grupo de cuadros de verificación. Los cuadros de revisión contienen una X cuando han sido seleccionados. Para quitar la marca de un cuadro de revisión seleccionelo por segunda vez.

4. Boton de opción

Selecciona una opción de un grupo de botones (usted puede seleccionar sólo un botón de opción de cada grupo). Los botones de opción son redondos y tienen el centro negro cuando han sido elegidos. Para remover una selección, elija una opción diferente en el mismo grupo.

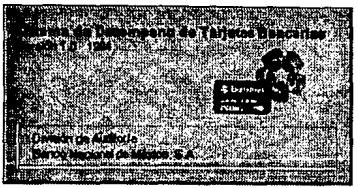


5. Botones

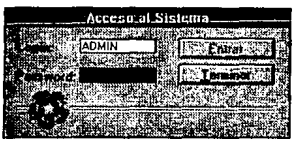
Con los botones se ejecutan comandos o también dan la posibilidad de abrir un recuadro adicional de diálogo con más alternativas. Seleccione el botón deseado y presione <Enter> para ejecutar cierta acción. El botón seleccionado aparece con negritas.

INICIO DEL SISTEMA

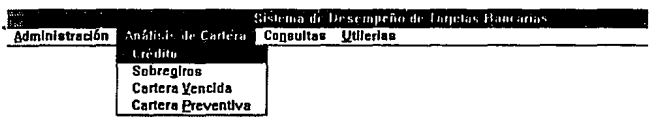
Para arrancar el sistema seleccione desde Windows el ícono correspondiente a la aplicación, una vez ejecutado podrá observar la siguiente pantalla de inicio:



Posteriormente digite su login y password para tener acceso al sistema.



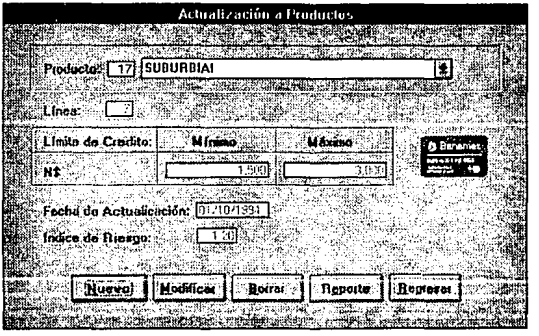
Una vez que ha sido identificado como usuario de la aplicación usted prodrá ver el siguiente menú general de opciones.



En la opción de Administración del menú se podrá dar mantenimiento general al sistema como en la actualización de productos, usuarios, cambio de password, conversión de archivos y salida del sistema.

ACTUALIZACION DE PRODUCTOS

Para la actualización de productos, seleccione la opción correspondiente del menú de Administración.

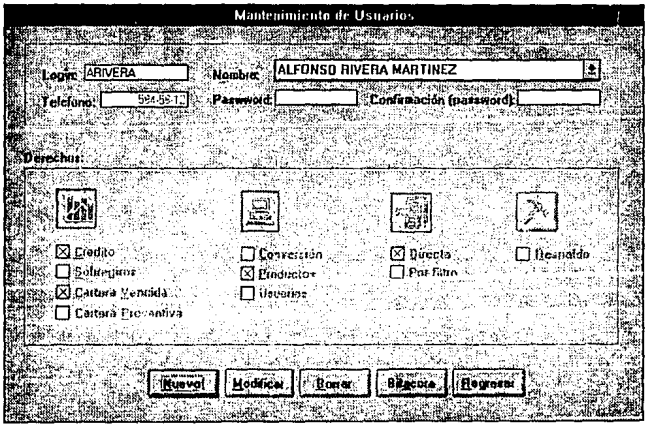


En esta pantalla se realizan altas, bajas, cambios y emisión de reportes de productos de tarjetas de crédito especificando las características de los mismos. La ejecución de las acciones se encuentran en la barra de botones ubicada en la parte inferior de la pantalla.

MANTENIMIENTO DE USUARIOS

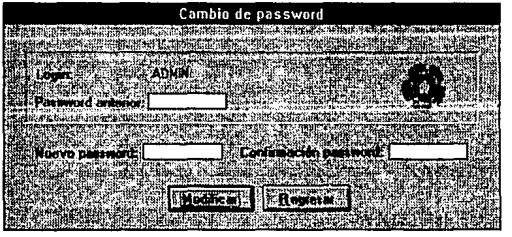
Otra opción del submenú de Administración es la correspondiente al mantenimiento de usuarios. En la pantalla siguiente usted podrá dar de alta, baja y modificar los datos y privilegios de cada usuario.

Los privilegios estan agrupados de tal manera que los cuatro botones con íconos que aparecen en la pantalla corresponden a los submenus del sistema donde usted prodrá especificar los derechos de cada opción del submenú seleccionando cada cuadro de verificación. En el botón perteneciente al submenú de análisis de cartera, también podrá restringir acciones de las pantallas como consultas y detalle de la información.



CAMBIO DE PASSWORD

Para la personalización de los accesos la opción **cambio de password** permite al usuario modificar el password en cualquier momento, ya sea el asignado por el administrador al darlo de alta o uno que él mismo estableció después de su alta.

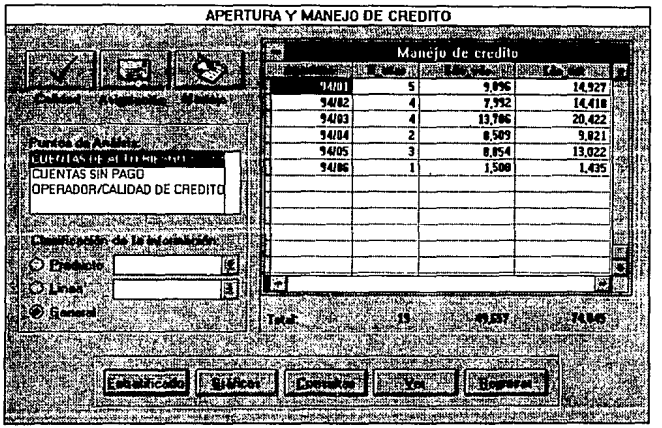


ANALISIS DE CARTERA

Dentro de este submenú se encuentran las opciones más representativas del sistema que a continuación se explican:

APERTURA Y MANEJO DE CREDITO

Para el análisis de la apertura y manejo de crédito se utiliza la siguiente pantalla.

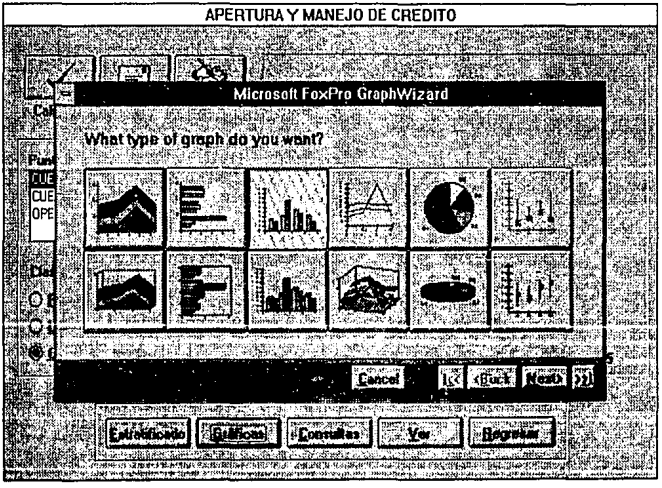


En ésta usted podrá elegir distintos puntos de análisis los cuales están agrupados por los tres botones con íconos. De acuerdo al botón seleccionado se elegirá de la lista inferior el punto de análisi deseado.

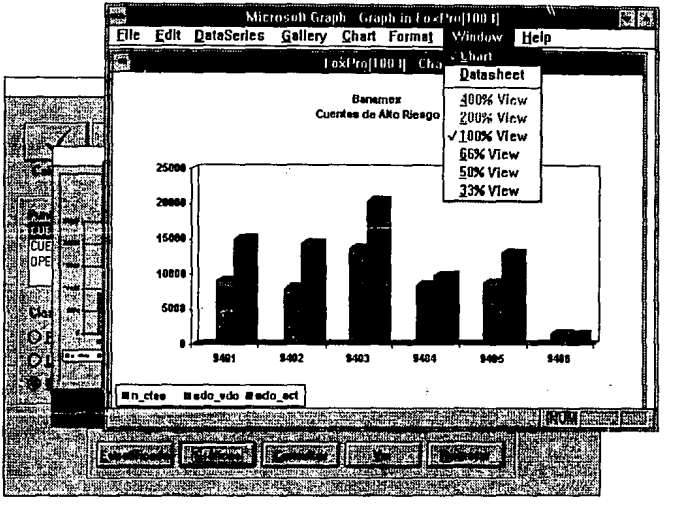
Posteriormente usted deberá de especificar a través de los botones de opción la manera en que será clasificada la información: por producto, por linea o general.

Una vez indicado lo anterior, en la barra de botones inferior se puede obtener el estratificado correspondiente de cada punto de análisis, realizar consultas, ver la información a detalle o relizar gráficas de la misma.

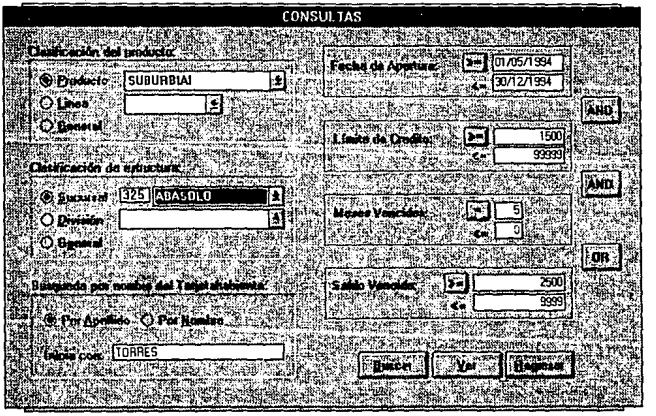
Cuando se elige la opción de gráficas aparece la siguiente pantalla en la que usted puede seleccionar el tipo de gráfica que requiera.



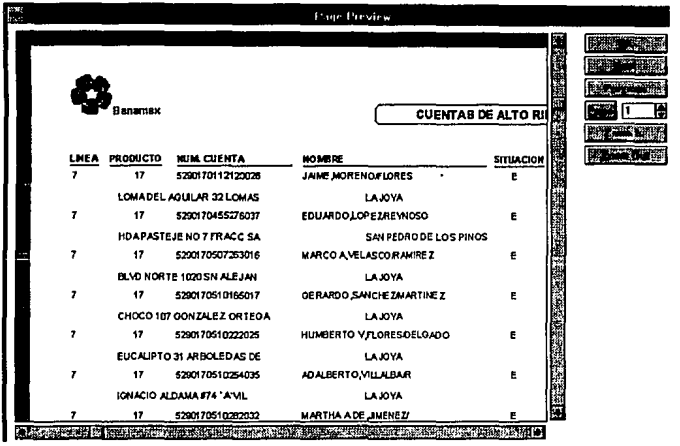
Al realizar una gráfica se cuentan con varias opciones para ajustar la presentación de su grafica como desee.



Si eligió en la barra de botones la opción de consultas aparecerá la siguiente pantalla. En la cual usted podrá realizar la serie de combinaciones necesarias para obetener información mas especifica.



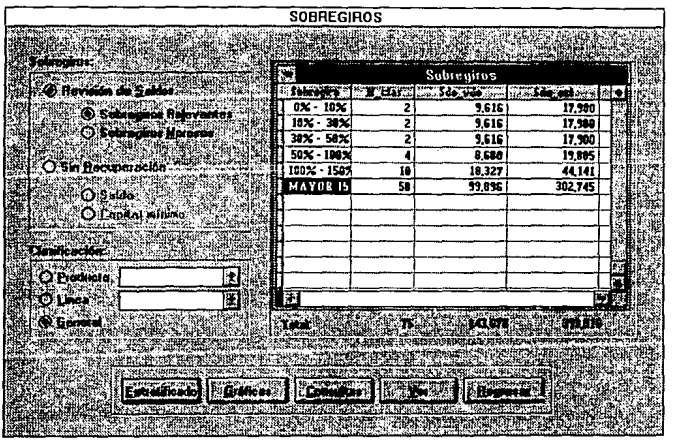
La pantalla de apertura y manejo de crédito también cuenta con la opcion para la emisión de reportes, una vez que se ha seleccionado el botón correspondiente usted podrá observar un vista prelimibar del reporte:



Despues de visualizar el reporte, cuando regresa de la vista preliminar el sistema pregunta por la impresión del reporte con la siguiente pantalla:

SOBREGIROS

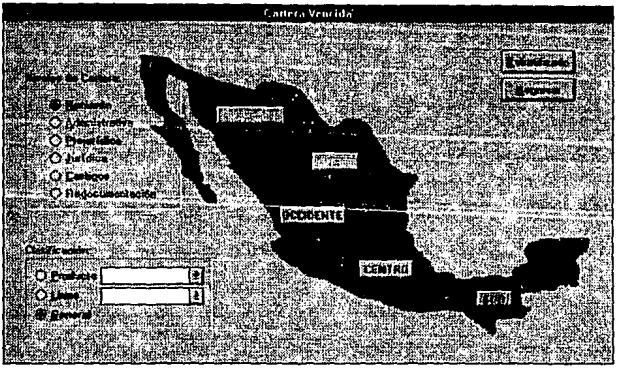
En la siguiente opción del submenú de **Análisis de Cartera** se tiene acceso a la siguiente pantalla:



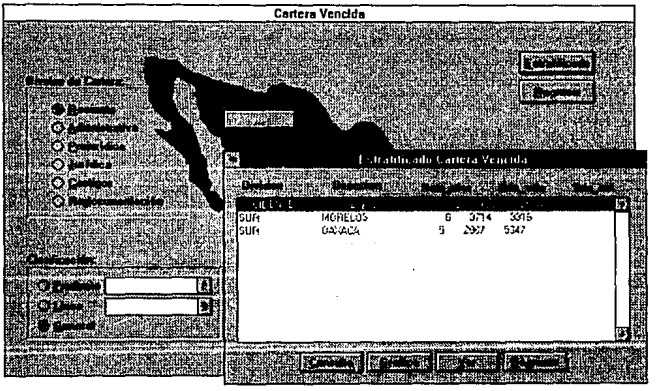
La operación de esta pantalla es similar a la de Apertura y manejo de crédito solo que la seleccion de los puntos de análisis se controla con los botones de opción que aparecen en la parte superior izquierda.

CARTERA VENCIDA

En la pantalla correspondiente a la **cartera vencida**, usted puede obtener información clasificándola por regiones a través de los botones una vez que se ha seleccionado el nivel de cartera.



Una vez que se presenta la información de **cartera vencida**, posteriormente se presenta un estratificado agrupado por las regiones especificadas, donde se tiene la posibilidad de obtener el desglose de sucursales de cada región, seleccionándola de la lista.

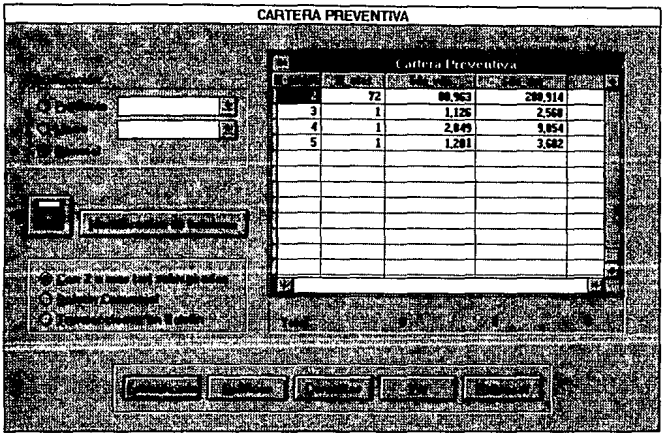


CARTERA PREVENTIVA

En este pantalla se lleva a cabo la identificación de nombres similares para detectar clientes con dos o más tarjetas de crédidito estratificando la información según los puntos de análisis que se presentan en ella.

Se cuenta con la opción de grabar en diskette dichos clientes en el botón representado con el ícono que aparece.

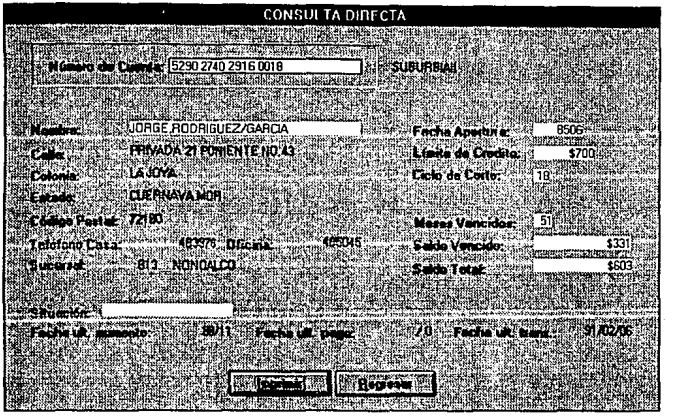
La operación de los botones que aparecen en la parte inferior



CONSULTAS

Existen dos tipos de consultas, la consulta directa y la consulta por filtro, la consulta directa consiste en la búsqueda de un número de cuenta de una tarjeta de crédito para desplegar la información correspondiente a la misma.

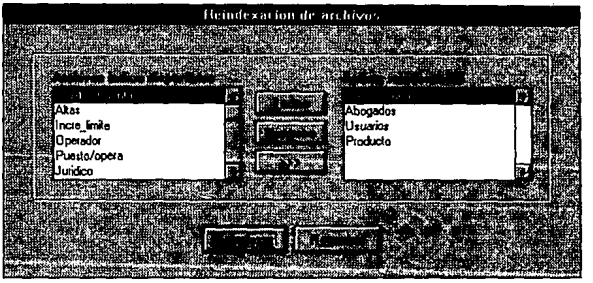
La pantalla por filtro es la misma que se explicó en la opción de apertura y manejo de crédito del submenú de análisis de cartera.



UTILERIAS

En este menú se pueden encontrar herramientas de apoyo para uso del sistema, tales como: reindexación de archivos, respaldo, configuración de impresora, calculadora y calendario

Para la reindexación de archivos, en la parte izquierda de la siguiente pantalla se encuentra una lista conteniendo todos los archivos que utiliza el sistema. Seleccione los que desee y páselos a la lista de archivos para reindexar con el botón (>>>) . Tambien puede seleccionar todos los archivos disponibles rápidamente con el Botón (Todos).



CALCULADORA Y CALENDARIO