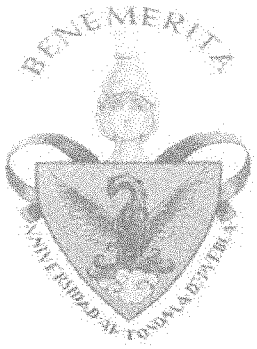


Benemérita Universidad Autónoma de Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Trabajo de tesis

Sistema de Control Administrativo Para la Bonetera DUTLA

Que para obtener el titulo de

Licenciado en Ciencias de la Computación

presenta

García Azcárate Leonardo

Asesora

M.C. María del Carmen Santiago Díaz

2007

Dedicatoria

A mi amado DIOS quien, a pezar de todos los obstáculos que he tenido que enfrentar, me ha permitido llegar hasta esta instancia.

A mi finado hermano Ricardo García Azcárate, mi compañero de alegrías y sin sabores de mi niñéz.

A mi finado padre, Don Leonardo García López, ejemplo de inquebrantable tenacidad y persistencia.

A mi amado retoño Jesús Leonardo García Arroyo y a mi amada madre Doña Emma Azcárate Reyes, quines son el principal motivo por el que he llegado hasta aquí.

A la M.C. Alma Delia Ambrosio Vázquez, a la M.E. Etelvina Archundia Sierra y a la M.C. María del Carmen Santiago Díaz, quienes tuvieron a bien brindarme el honor de ser mis sinodales.

A todas las profesoras y a todos los profesores que participaron en mi formación profesional.

A la Facultad de Ciencias de la Computación, de la Benemérita Universidad Autónoma de Puebla, que me albergó durante mi formación profesional.

Indice

Introducción

En el mercado de software existen varios programas de administración comercial, tales como creación de nóminas, programas de punto de venta, control de inventarios, control de la producción, contabilidad de costos, entre otros.

Aunque todos estos son muy buenos no suelen cubrir al 100% las necesidades de otras empresas, pues estos están hechos de acuerdo a los requerimientos de una empresa en particular, y han sido lanzados al mercado, suponiendo que todos los negocios son manejados y administrados de igual forma.

Pero esto es una suposición errónea, ya que la contabilidad de cada empresa depende de los criterios utilizados por los contadores que están al servicio de la misma.

Por otro lado no son muy amigables estos programas, y complican su uso, incluso algunos exigen que el usuario tenga conocimientos de computación para poder entender su aplicación, y a veces es necesario cerrar una ventana para poder imprimir algún informe.

En fin, el dueño de la empresa tiene que limitar su negocio a lo que el programa le proporcione, resultando un gasto infructuoso la adquisición del software, a grado tal que el propietario de la empresa termina por dejar de utilizarlo, y contrata a alguien que le desarrolle un programa ajustado a sus necesidades. Cabe mencionar que el desarrollo de un programa hecho a medida suele alargar su tiempo más allá de lo esperado, pues muchas veces surgen más necesidades que cubrir.

El presente trabajo es el resultado de la implementación de un programa que puede ser utilizado por empresas cuyo giro sea el ramo textil manufacturero, así como de la industria de la confección del vestido y del calzado.

Algunas de las actividades que efectúa dicho programa son: punto de venta (pedidos, remisiones y facturas), cuentas por pagar, cuentas por cobrar, comisiones e información estadìstica de ventas .

Este documento está estructurado de la siguiente forma:

En el capítulo I se contemplan algunos conceptos de administración, de las bases de datos y se incluyen instrucciones de los lenguajes SQL y Visual Basic 6.0, utilizados para el desarrollo del programa.

En el capítulo II se presenta el planteamiento del problema y la especificación de requerimientos, los modelos E-R y las tablas desarrolladas para el almacenamiento de la información, y la forma en que se relacionan entre sí.

En el capítulo III se efectúan la implementación y pruebas del programa a desarrollar, así como se explican algunos módulos que forman parte del programa .

Finalmente se exponen las conclusiones, y algunas perspectivas para hacer más completo el programa.

Capítulo 1
Marco teórico

1.1 Conceptos de Ingeniería de Software

Desde sus inicios (finales de los 60's) la ingeniería de software ha ido fortaleciéndose con el paso de los años, y desde entonces a la fecha, ha demostrado ser muy necesaria en la elaboración de proyectos de software. [Ref. 4]

Por lo que se puede decir que la ingeniería de software utiliza los principios de la ciencia de la computación, para lograr soluciones a los problemas de desarrollo de software, màs confiables y a bajo costo.

Para esto, es necesario llevar a cabo un proceso en el que las necesidades del solicitante son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado e4n código, el código es probado, documentado y certificado para su uso operativo.

Para realizar esta tarea se aplica un modelo, el cual consiste en una serie de procedimientos para la implementación de programas de software, y que se conoce como "Ciclo de vida del software".

1.1.1 Ciclo de vida del software

El modelo de ciclo de vida de software más utilizado es el llamado "Modelo clásico" o "Modelo de cascada" para la implementación de software, y el cual es un conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar un sistema de información.

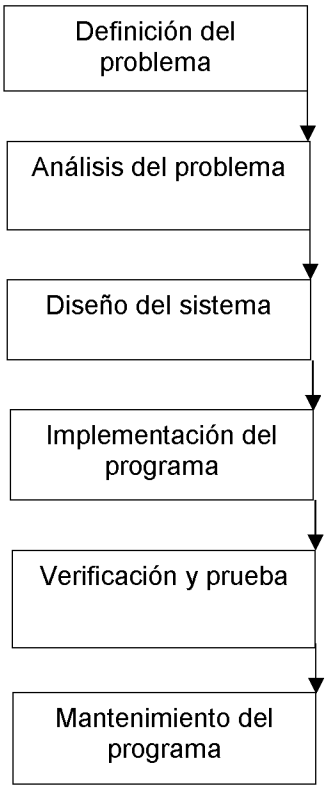


Figura 1.1
Modelo clásico del ciclo de vida del software

Las etapas de este modelo se muestran en la figura 1.1, y cada una de ellas se describe a continuación.

A) Definición del problema

En esta etapa se plantea el problema que se va a resolver.

- Se debe realizar una serie de entrevistas con quien haya solicitado la solución a su problema, y con cada uno de los usuarios que emplearán el programa que se va a desarrollar, en caso de ser necesario.
- En cada una de las entrevistas, es aconsejable llevar una serie de anotaciones, para considerar todos los requerimientos del cliente y de los usuarios.
- Esto se logra con una serie de preguntas que se debenm hacer a cada persona con la que se entreviste, teniendo en cuenta hasta los detalles que parezcan no tener importancia alguna.

B) Análisis del problema

De acuerdo a las entrevistas efectuadas con los futuros usuarios, y habiendo identificado sus necesidades, se deben analizar detenidamente los requerimientos recopilados, identificando:

- La información que se utiliza, y que debe ser guardada en la computadora.
- La información de entrada, la información de salida, los datos calculados, los datos de salida a pantalla y los datos de salida a

impresora.

- Los tipos de datos y las estructuras de datos que se deberán utilizar para las entradas de unos y salidas de otros.
- Las funciones y las operaciones que se deberán implementar con el fin de cumplir con los requerimientos del problema.

C) Diseño del sistema (diseño lógico)

En esta etapa:

- Se determinan las estructuras y los dispositivos que alojaran la información, que será capturada a través de la interfase de usuario..
- Se modelan las estructuras cuidando de establecer los tipos de datos que albergarán: si son de tipo texto proporcionar la longitud de estos, si son de tipo numérico indicar la cantidad de cifras decimales que manejará, etc.
- Se debe establecer el lenguaje de programación con el que se desarrollará el programa.

D) Implementación del programa (diseño físico)

A partir de lo desarrollado en la etapa de diseño:

- Se implementa el programa de computadora en el lenguaje de programación, establecido en la etapa de diseño.
- Crear las interfaces de usuario para:
  - Solicitud de clave de acceso al programa.
  - Menú de acceso a los módulos del programa.
  - La introducción y consulta de información.
  - Informes requeridos.

E) Verificación y prueba

Es la fase en la que se efectúan las pruebas de las funciones del programa, con el fin de comprobar que trabaje para el objetivo que fue creado.

Si el programa consta de varios módulos, se deben ir probando uno por uno, conforme se vayan desarrollando, esto es: introducir y modificar información, y efectuar consultas, con los datos que se realizarán las pruebas, hasta que el programa carezca de errores.

F) Mantenimiento del programa

Una vez concluías las pruebas y cubierto los requerimientos del cliente, se instala el programa en las computadoras indicadas por el usuario. Es cuando se puede considerar que el proyecto se ha concluido.

Es en esta fase cuando se efectúan las mejoras o cambios que no hayan sido considerados por el cliente, antes de entregar el programa.

1.2 Conceptos de administración

El control de las ventas se realiza por medio de documentos que se extiendan a los clientes. En estos documentos se especifica el destinatario, el detalle y el importe de la venta. Los documentos se conocen como Remisiones y Facturas. [Ref. 1]

La información de cada documento se guarda para posteriores consultas, tales como: Diario de ventas, Ventas por período, Facturas por período, etc.

También se registran las entradas y salidas de materia prima permitiendo, en un momento dado, saber la existencia de estas en el almacén, la cual se modifica conforme se facture o se realicen entradas.

Las deudas que quedan pendientes de pago o que de estas se efectúen en pagos parciales, se guardan como Cuentas por Pagar, y se registran los pagos que se van haciendo de cada una de ellas, hacia cada proveedor.

Las facturas que quedan pendientes de cobro o que de estas se efectúen en cobros parciales, se guardan como Cuentas por Cobrar, y se registran los cobros que se van haciendo de cada una de ellas, y de cada cliente.

A continuación se describen algunos de estos conceptos.

1.2.1 Pedidos, Remisiones y facturas

Para tener control de las ventas se utilizan documentos en los que se anota: la cantidad vendida, la descripción, el precio unitario de venta y el importe de cada artículo; también suelen incluirse: el porcentaje e importe de algún descuento o del impuesto, así como algunos datos del cliente.

Los documentos que se utilizan son: pedidos, remisiones y facturas, los cuales se describen a continuación.

❖ **Pedidos**

Se refiere a los pedidos que hacen los clientes y que quedan pendientes de entregar. En este documento se pueden considerar descuentos y no se incluye el cobro del impuesto. La información que se considera es:

- Del documento: número y fecha del pedido.
- Del cliente: nombre o razón social.
- Del detalle de venta de cada artículo: cantidad, clave y descripción, precio unitario e importe.
- Del total de la venta: suma de los importes, porcentaje e importe del descuento y total del pedido.

❖ **Remisiones**

Se refiere a las ventas al mayoreo que se realizan y que son pagadas de contado, o que quedan pendientes de pago. En este documento se pueden considerar descuentos y no se incluye el cobro del impuesto; la información que se considera es:

- Del documento: número y fecha de remisión.
- De la empresa que lo expide: nombre o razón social, y opcionalmente el número del vendedor.
- Del cliente: nombre o razón social, R.F.C., dirección, lugar de origen, número telefónico.
- Del detalle de venta de cada artículo: cantidad, clave y descripción, precio unitario e importe.
- Del total de la venta: suma de los importes, porcentaje e importe del descuento y total a pagar con número y letra.

❖ **Facturas**

- Se refiere a las ventas al mayoreo que se realizan y que son pagadas de contado, o que quedan pendientes de pago. En este documento se pueden considerar descuentos, se efectúa el cobro del impuesto, el cual debe ir desglosado. En este caso se registra la salida de mercancía al momento de facturar. La información que se considera es:

- Del documento: número y fecha de factura.

- De la empresa que lo expide: nombre o razón social, domicilio, código postal, número telefónico, lugar de procedencia, R.F.C., y opcionalmente el número del vendedor.
- Del cliente: nombre o razón social, R.F.C., dirección, lugar de origen, número telefónico.
- Del detalle de venta de cada artículo: cantidad, clave y descripción, precio unitario e importe.
- Del total de la venta: suma de los importes, porcentaje e importe del descuento, porcentaje e importe del impuesto, y total a pagar con número y letra.

1.2.2 Cuentas por pagar

Al efectuar compras en las que se solicita determinada cantidad de mercancía, se registra la entrada de la misma, actualizando su existencia.

Algunas de estas compras suelen quedar pendientes de pago y son archivadas. De esta forma se va creando lo que en materia de administración se conoce como "Cuentas por pagar".

Los pagos, ya sean totales o parciales, de los documentos correspondientes, se van anotando en tarjetas llamadas "kárdex", las cuales forman el "Estado de cuenta" para con los proveedores,

Dicho estado de cuenta se imprime en un concentrado con la siguiente información:

- Del informe: fecha de impresión.
- Del proveedor: nombre o razón social, domicilio, colonia, código postal, lugar de procedencia, R.F.C. y C.U.R.P.
- Del detalle: número del documento (remisión o factura), concepto (deuda/pago), fecha del concepto, fecha de vencimiento del documento, referencia del documento, debe (importe pagado al proveedor), haber (importe del documento), saldo (lo que resta por pagar del documento).
- Del total: Importe total de lo pagado al proveedor (debe), importe total de todos los documentos (haber), importe total de lo que la empresa adeuda de cada documento (saldo).

1.2.3 Cuentas por cobrar

Al efectuar ventas en las que el cliente solicita determinada cantidad de mercancía, se registra la salida de la misma, actualizando su existencia.

Algunas de estas ventas suelen quedar pendientes de pago y son archivadas. De esta forma se va creando lo que en materia de administración se conoce como "Cuentas por cobrar".

Los cobros, ya sean totales o parciales, de los documentos correspondientes, se van anotando en tarjetas llamadas "kárdex", las cuales forman el "Estado de cuenta" del cliente".

Dicho estado de cuenta se imprime en un concentrado con la siguiente información:

- Del informe: fecha de impresión.
- Del cliente: nombre o razón social, domicilio, colonia, código postal, lugar de procedencia, R.F.C. y C.U.R.P. y saldo a favor.
- Del detalle: número del documento, concepto (deuda/cobro), fecha del concepto, fecha de vencimiento del documento, referencia del documento, debe (importe del documento), haber (importe del cobro), saldo (lo que resta por cobrar del documento al cliente).
- Del total: Importe total de todos los documentos (debe), importe total de lo cobrado al cliente (haber), importe total de lo que el cliente adeuda de cada documento (saldo).

1.2.4 Comisiones

Si se cuenta con personal que se encargue de promover las mercancías que se ofrezcan, a este personal se les llama "vendedores", a los cuales se les asigna una determinada comisión por el monto de las ventas que efectúen.

La comisión es un porcentaje del importe de cada documento, y normalmente es entregada a los vendedores cada quincena del mes.

Para saber el monto de la comisión de cada vendedor, se imprime una lista con la siguiente información:

- Del informe: fecha de corte.
- Del vendedor: nombre del vendedor.
- Del detalle: fecha del documento, concepto, nombre del cliente, importe del documento, importe que se cobró del documento, importe de la comisión.
- De los totales: suma del importe de cada documento, suma del importe cobrado de cada documento y suma de la comisión que le corresponde de cada documento.

1.3 Conceptos de bases de datos

Una base de datos es un conjunto de datos estructurados, guardados en un dispositivo de almacenamiento y que se puede acceder a ella por medio de algún programa. Antes de diseñar una base de datos se debe establecer un proceso, a partir del mundo real, de tal forma que se pueda ver reflejado mediante un conjunto de datos. [Ref. 3]

A la imagen que se obtiene del mundo real se le conoce como "modelo conceptual", el cual consiste en una serie de elementos que definen lo que se quiere reflejar del mundo real en la base de datos.

El sistema gestor de la base de datos es un programa que proporciona las herramientas necesarias para trabajar con una base de datos, actuando como un intermediario entre los usuarios y la información almacenada en la base de datos.

Debe permitir definir los registros (tuplas), sus campos (atributos), relaciones entre tablas, agregar nuevos registros, borrar registros, modificar y consultar información, dando una interfaz interactiva y entendible para el usuario. La base de datos puede ser implementada con un programa diseñado especialmente para este fin.

1.3.1. Conceptos fundamentales

❖ **Entidad**

Es un objeto del que se obtiene información y que se representa en una base de datos. Ejemplos de entidad: clientes, artículos, vendedores, etc.

❖ **Campo**

Es la unidad básica de información de la entidad y sirve para identificar y describir a las entidades. Al definir los campos se debe tener cuidado con el significado de los datos que se almacenarán en cada campo. Ejemplos de campos: ClaveCliente, Domicilio, etc.

❖ **Dominio**

El dominio es el conjunto de valores que puede tomar cada uno de los campos. Esto es, no se debe confundir el tipo de dato del campo, con la información que almacenará el campo, por ejemplo: si se define un campo como "NombreCliente" y otro como "Sexo" y ambos son de tipo cadena, el primer campo únicamente contendrá "nombres" y el segundo solamente "Masculino" o "Femenino".

❖ **Tabla relacional**

Es la manera en que se organiza la información en forma de renglones (llamados tuplos o registros) y columnas (llamadas atributos o campos), con las siguientes características:

- Cada registro debe ser único.
- Cada campo debe ser único
- La información que almacenen los campos debe ser del dominio de cada campo
- El valor de cada campo para cada registro debe ser único

❖ **Relación**

Es la correspondencia entre entidades. Por ejemplo: a cada cliente le corresponde una o varias facturas.

1.3.2 Claves de relación

Una clave es un conjunto de campos que a un registro lo hace único a los demás, en una relación. Los campos que forman parte de la clave se llaman “campos primos”, y los campos que no forman parte de ella se llaman “campos no primos”.

Existen diferentes formas de definir una clave que cumple con la propiedad de unicidad.

❖ **Superclave**

Conjunto de campos que poseen la propiedad de unicidad y que hacen único a un registro; en sí, todos los campos de un registro forman una superclave, pues solamente un registro es único para toda la relación de campos; también es conocida como clave principal o clave primaria.

❖ **Clave de relación**

Conjunto de campos que tiene un valor único para cada registro en la relación, de tal manera que si se borra un campo de la clave, entonces el resto de los campos, que forman dicha clave, no tendrá la propiedad de unicidad, además de que ningún campo que forme parte de la clave debe ser nulo.

❖ **Clave extranjera**

Conjunto de campos que es clave de relación en otra tabla, a través del cual se relaciona con esta, una o más tablas.

❖ **Inconsistencia**

Se da cuando se obtienen diferentes resultados en diferentes consultas de una misma entidad

1.3.3 Relaciones entre entidades

Además de los campos de cada entidad, un modelo de datos debe especificar las relaciones existentes entre las entidades. Por ejemplo, la frase "los clientes compran artículos" indica que hay dos entidades, "Clientes" y "Artículos", y que están relacionadas a través de "compran".

La mayoría de las relaciones son binarias, como: "los clientes hacen pedidos", o bien "las empresas surten pedidos". Con las dos asociaciones binarias independientes es difícil saber a qué clientes se han surtido los pedidos que ha hecho la empresa; en este caso se necesita de una tercera tabla a través de la cual se relacionan las dos tablas anteriores

Las relaciones entre dos tablas pueden ser de tres tipos: uno a uno, uno a varios y varios a varios.

❖ **Relación uno a uno**

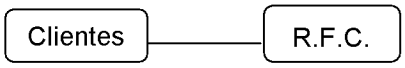


Figura 1.2: Relación uno a uno

Cada registro de una tabla X sólo puede coincidir con un registro de una tabla Y y viceversa. Este tipo de relación no es muy utilizada pues la mayoría de la información relacionada de esta forma está en una sola tabla. Se puede generar la relación uno a uno para dividir una tabla que tenga muchos campos, para aislar parte de una tabla por razones de seguridad o para almacenar información que sólo se aplica a un subconjunto de la tabla principal. Por ejemplo, se puede crear una tabla que contenga a los clientes y en otra tabla su Registro Federal de Contribuyente (R.F.C.), y relacionar ambas tablas por medio del campo "ClaveCliente"

Para representar este tipo de relación se traza una línea de una tabla a otra indicando la cardinalidad "uno a uno" entre estas, como se muestra en la Figura 1.2. En esta relación a cada cliente le corresponde un único R.F.C.

❖ **Relación uno a varios**

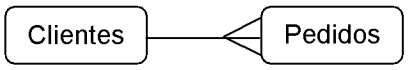


Figura 1.3: Relación uno a varios

Un registro de una tabla X puede coincidir con varios registros de una tabla Y, pero un registro de la tabla Y sólo puede coincidir con un registro de una tabla X. Por ejemplo, un cliente puede tener varios pedidos.

Para representar este tipo de relación se traza una línea de una tabla a otra indicando la cardinalidad "uno a varios" entre estas, como se muestra en la Figura 1.3. En esta relación cada cliente puede hacer varios pedidos.

❖ **Relación varios a varios**

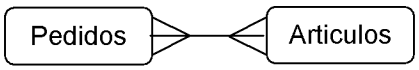


Figura 1.4: Relación varios a varios

Un registro de una tabla X puede coincidir con varios registros de una tabla Y, y viceversa. Por ejemplo, un pedido puede tener varios artículos, y un artículo puede estar en varios pedidos.

Para representar este tipo de relación se traza una línea de una tabla a otra indicando la cardinalidad "varios a varios" entre estas, como se muestra en la Figura 1.4

1.3.4 El modelo entidad-relación

El modelo entidad-relación es una herramienta de diseño de bases de datos, que incluye información relativa a los datos y la relación existente entre ellos, con el fin de representar una visión del mundo real sobre un soporte informático.

Sus características fundamentales son:

- ❖ Refleja la existencia de los datos sin expresar lo que se hace con ellos.
- ❖ Es independiente de las bases de datos y de los sistemas operativos
- ❖ Incluye los datos que se estudian sin considerar sus aplicaciones.

Los elementos utilizados en el modelo entidad-relación (E-R) se listan en la figura 1.5 (en el capitulo 2 se expone la aplicación del modelo entidad-relación).

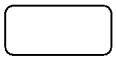
	Con los rectángulos redondeados se simboliza a las entidades y a las relaciones.
———	Con una línea continua se representa la "relación obligatoria". Con ella se representa la relación uno a uno
- - - - - - - -	Con una línea punteada se representa la "relación opcional". Con ella tambien se representa la relación uno a uno
	Con este símbolo representa la relación uno a varios
*	Con el asterisco antepuesto al nombre del atributo, se indica que su valor no debe ser nulo
o	Con el circulo antepuesto al nombre del atributo, se indica que su valor puede ser nulo o diferente de nulo
#	Con la almoadilla antepuesta al nombre del atributo, se indica que este es el identificador único, es decir, la clave principal (clave primaria)

Figura 1.5: Simbología utilizada en el modelo entidad-relación (E-R)

Dentro de los rectángulos redondeados se anota el nombre de la entidad (o de la relación), y debajo de este se anotan los nombres de los atributos que la forman, con sus características (contenido opcional o necesario, atributos que forman la clave primaria).

Y posteriormente se implementan las tablas correspondientes, a parir del modelo E-R obtenido.

1.3.5 Operaciones sobre tablas

Para poder tener acceso a las tablas se utiliza un lenguaje que facilita el trabajo, y que consta de una serie de operadores, los cuales pueden emplearse con una o más tablas, para obtener otra tabla.

Para explicar cada uno de los operadores considérese las tablas 1.3 y 1.4, en las que se muestra el contenido de cada una de ellas..

campo1	campo2	campo3
A	B	C
A	D	E
B	C	D

Tabla 1.3: Contenido de la tabla de ejemplo

campo1	campo2	campo3
A	B	C
B	D	E
A	C	D

Tabla 1.4: Contenido de la tabla de ejemplo

❖ **Selección**

Este operador se utiliza para efectuar consultas, y se obtiene un subconjunto de registros de una o más tablas que cumplen una condición determinada, extrayendo los campos especificados, y se ordena o se agrupan los registros resultantes en el orden deseado. Al hacer una selección en la tabla 1.3 para campo 1 = A, se obtienen la tabla 1.5.

campo1	campo 2	campo 3
A	B	C
A	D	E

Tabla I.5: Resultado cuando campo 1 = A

❖ **Inserción**

Con esta operación se logra agregar un nuevo registro a una tabla, cuidando que los campos del nuevo registro sean del mismo dominio al que pertenecen los campos de la tabla.

Al agregar un registro en la tabla 1.3, para: campo1 = B, campo2 = C y campo3 = E, se obtienen la tabla 1.6.

campo1	campo 2	campo 3
A	B	C
A	D	E
B	C	D
B	C	E

Tabla I.6: Resultado de agregar un registro

❖ **Modificación**

Con esta operación se logra modificar la información de uno o todos los campos de un registro de una tabla, cuidando que los campos del registro a modificar sean del mismo dominio al que pertenecen los campos de la tabla. Esto se realiza para un registro que cumpla con una determinada condición.

Si el contenido del segundo registro de la tabla 1.3, donde: campo1 = A, campo2 = D y campo3 = E, se modifica por: campo1 = D, campo2 = C y campo3 = B, se obtienen la tabla 1.7.

campo1	campo 2	campo 3
A	B	C
D	C	B
B	C	D

Tabla I.7: Resultado de modificar un registro

❖ **Eliminación**

Con esta operación se logra eliminar un registro de una tabla, Esto se realiza para un registro que cumpla con una determinada condición.

Si de la tabla 1.3 se elimina el segundo registro, se obtienen la tabla 1.8.

campo1	campo 2	campo 3
A	B	C
B	C	D

Tabla I.8: Resultado de eliminar un registro

1.4 Normalización

La normalización es un conjunto de reglas que se utilizan para implementar bases de datos libres de inconsistencias y de redundancias, de tal manera que los datos obtenidos tienen una estructura óptima para su gestión y explotación desde distintas aplicaciones. Para que se cumpla una regla debe cumplirse la anterior. [Ref. 3]

Pedidos

Pedido	Cliente		Articulo	
			Descrip	Cantidad
Pedido1	Nombre	Direc	Fibra	10
	Nombre1	Direc1	Algodón	20
			Descrip	Cantidad
Pedido2	Nombre	Direc	Fibra	15
	Nombre2	Direc2	Algodón	25

Tabla I.10: Tabla sin normalizar

El objetivo de la normalización es crear una base de datos funcional, y que pueda crecer y adaptarse fácilmente a los nuevos requisitos. Otra ventaja de la normalización de las bases es que hay menos repetición de información, lo cual reduce el consumo de espacio en disco.

Básicamente existen tres niveles de normalización: Primera forma normal (1FN), Segunda forma normal (2FN) y Tercera forma normal (3FN); cada una de las cuales tiene sus propias reglas.

Cuando una base de datos se normaliza a un nivel, se dice que está normalizada a esa forma de normalización. Por ejemplo, supóngase que una base de datos cumple con la regla del segundo nivel de normalización, entonces se dice que está en la segunda forma normal.

Para explicar cada una de las reglas, se considerará un ejemplo, partiendo de una tabla que no está normalizada, la cual se muestra en la tabla 1.10, en la que se almacenan los pedidos que hacen los clientes.

1.4.1 Primera forma normal (1FN)

La primera forma normal establece que todos los campos de una tabla deben ser simples, para que esté en la primera forma normal.

Como se puede observar en la tabla 1.10, el campo "Articulo" está formado por dos campos: "Descrip" y "Cantidad", es decir, no es un campo simple, por lo que no está en 1FN. Lo mismo sucede con el campo Cliente, al cual lo forman los campos Nombre y Direc.

Pedidos				
Pedido	**Nombre**	**Direc**	**Descrip**	**Cantidad**
Pedido1	Nombre1	Direc1	Fibra	10
Pedido1	Nombre1	Direc1	Algodón	20
Pedido2	Nombre2	Direc2	Fibra	15
Pedido2	Nombre2	Direc2	Algodón	25

Tabla I.11: Tabla normalizada a la 1FN

Para que la tabla 1.10 esté en 1FN, el campo "Artículos" debe dividirse en los campos que lo forman, al igual que el campo Cliente. Esto se puede observar en la tabla 1.11.

1.4.2 Segunda forma normal (2FN)

La tercera forma normal establece que una tabla está en 2FN, si los campos no primos dependen completamente de la clave principal.

En la tabla 1.11 la clave principal está formada por los campos: "Pedido", "Nombre y "Descrip". Como se puede observar, esta tabla no está en la segunda forma normal, pues el campo "Cantidad" no depende del campo "Nombre", sino únicamente de los campos "Pedido" y "Descrip".

Para que la tabla 1.11 esté en 2FN, debe dividirse en dos tablas y relacionarlas por medio de un campo común. Esto se muestra en las tablas Tabla 1.12a y Tabla 1.12b

Pedidos		
Pedido	**Nombre**	**Direc**
Pedido1	Nombre1	Direc1
Pedido2	Nombre2	Direc2

Tabla 1.12a Tabla de pedidos

Detalle		
Pedido	**Descrip**	**Cantidad**
Pedido1	Fibra	10
Pedido1	Algodón	20
Pedido2	Fibra	15
Pedido2	Algodón	25

Tabla 1.12b: Tabla de detalle de pedidos

En este caso ambas tablas se relacionan a través del campo "Pedido".

1.4.3 Tercera forma normal (3FN)

La tercera forma normal establece que una tabla está en 3FN, si los campos no primos son independientes mutuamente; es decir, si los campos no primos, no dependen de otros campos no primos.

Como se puede observar en la tabla 1.12a, la clave está formada por el campo "Pedido"; y además el campo "Direc" depende del campo "Nombre", y estos dos no forman parte de la clave.

Para que la tabla 1.12a esté en 3FN, el campo “Direc” debe separarse en otra tabla y agregar un campo en esta a través del cual se relacione con la tabla “Pedidos”.

Esto se muestra en las tablas 1.13a y 1.13b y 1.13c.

Pedidos

Pedido	Nombre
Pedido1	Nombre1
Pedido2	Nombre2

Tabla 1.13a
Tabla de pedidos

Clientes

Nombre	Direc
Nombre1	Direc1
Nombre2	Direc2

Tabla 1.13b
Tabla de clientes

Detalle

Pedido	Descrip	Cantidad
Pedido1	Fibra	10
Pedido1	Algodón	20
Pedido2	Fibra	15
Pedido2	Algodón	25

Tabla 1.13c
Tabla de detalle de pedidos

Pero según se puede observar en estas tablas, se repiten: el nombre del cliente y la descripción del artículo, lo cual produce redundancia en la información.

Para evitar esto, en la tabla “Clientes” y en la tabla “Pedidos” se agrega un nuevo campo, a través del cual se relacionarán. Y el detalle de pedidos se divide en dos tablas, las cuales se relacionan por medio de un campo.

Esto se muestra en las tablas 1.14a y 1.14b, 1.14c y 1.14d.

Clientes

ClvCliente	Nombre	Direc
Cliente1	Nombre1	Direc1
Cliente2	Nombre2	Direc2

Tabla 1.14a
Tabla de clientes

Articulos

ClvArticulo	Descrip
Articulo1	Fibra
Articulo2	Algodón

Tabla 1.14b
Tabla de artículos

Pedidos

Pedido	ClvCliente
Pedido1	Cliente1
Pedido2	Cliente2

Tabla 1.14c
Tabla de pedidos

DetalleDePedidos

Pedido	ClvArticulo	Cantidad
Pedido1	Articulo1	10
Pedido1	Articulo2	20
Pedido2	Articulo1	15
Pedido2	Articulo2	25

Tabla 1.14d: Tabla de detalle de pedidos

1.5 Lenguajes SQL y Visual Basic

Para poder tener acceso a la información que guarda la base de datos y de realizar las operaciones básicas en ella, tales como: agregar, modificar, consultar y borrar, es necesaria la implementación de un programa. [Ref. 2]

Para esta tarea es recomendable utilizar un lenguaje que acepte instrucciones del lenguaje SQL, pues éste permite realizar el procesamiento de la base de datos de una manera más sencilla y rápida, además de que reduce código de programación. O bien, utilizar el lenguaje que más se adapte a lo que tiene la empresa

Hoy en día la mayoría de los lenguajes, tales como Delphi, FoxPro, Java, Visual Basic, entre otros, permiten esta combinación.

A continuación se exponen algunas instrucciones de SQL y de Visual Basic,

1.5.1 SQL

❖ **Instrucción para agregar**

Se utiliza para agregar uno o varios registros al final de una tabla ya existente; su sintaxis es:

INSERT INTO *destino* [(*campo1*[, *campo2*[, ...]])] VALUES (*valor1*[, *valor2*[, ...])

En donde:

Parte de la instrucción	Explicación
Destino	Es el nombre de la tabla en donde se van a añadir los registros.
campo1, campo2, ...	Son los nombres de los campos donde se van a añadir los datos
Valor1, valor2, ...	Son los valores que se van a insertar en los campos especificados del nuevo registro. Cada valor se inserta en el campo que corresponde a la posición del valor en la lista: valor1 se inserta en campo1, valor2 dentro de campo2, y así sucesivamente. Se debe escribir los valores de tipo texto entre apóstrofos (' '), los valores de tipo fecha entre almohadillas (# #) y los valores numéricos no se encierran entre carácter alguno.

Se debe especificar cada uno de los campos del registro en los que se va a guardar un valor y dicho valor para ese campo. Cuando no se especifique un campo, automáticamente se inserta el valor predeterminado o bien el valor Null en el campo no especificado.

Si la tabla destino contiene una clave principal, se debe asegurar de añadir los valores en los campos que forman parte de la clave, para evita errores.

En el siguiente ejemplo se agrega el artículo cuya clave de control es NY001P y su descripción es HILO NYLON:

INSERT INTO Articulos (ClvArtículo, Artículo) VALUES ('NY001P', 'HILO NYLON')

Nota: Únicamente se puede agregar información en una tabla a la vez.

❖ **Instrucción para modificar**

Se utiliza para modificar valores de los campos en una tabla específica según un criterio determinado, su sintaxis es:

UPDATE tabla SET campo = nuevovalor WHERE condicion;

En donde

Parte de la instrucción	Explicación
Tabla	Es el nombre de la tabla cuyos datos se van a modificar.
campo	Es el campo cuyo contenido se va modificar
nuevovalor	Es el valor con el que se va a modificar el campo.
Criterio	Es una expresión que determina qué registros se modificarán. Sólo se modifican los registros que cumplen con la expresión.

En el siguiente ejemplo se selecciona el registro que cumple con la condición ClvArtículo = 'NY001P', se cambia NY001P por NYTEXT y también se cambia HILO NYLON por HILO TEXTURIZADO:

Update Articulos SET ClvArtículo = 'NYTEXT', Artículo = 'HILO TEXTURIZADO'
WHERE ClvArtículo = 'NY001P'

Nota: Únicamente se puede modificar información en una tabla a la vez.

❖ **Instrucción para consultar**

Se utiliza para efectuar consultas en la tabla o en las tablas especificadas, devuelve los campos elegidos, selecciona los registros que cumplen con las condiciones establecidas, y ordena o agrupa los registros resultantes en el orden especificado; su sintaxis es:
SELECT { *| tabla.* | [tabla.]campo1 [AS alias1] [,[tabla.]campo2 [AS alias2] [,...]]}
FROM listatablas [,...] [WHERE criterio] [ORDER BY listacampos]

En donde

Parte de la instrucción	Explicación
*	Se utiliza para seleccionar todos los campos de la tabla o tablas especificadas.
Tabla	Es el nombre de la tabla que contiene los campos de la cual se van a seleccionar los registros.
campo1, campo2	Son los nombres de los campos que se van a obtener.
Alias1, alias2	Son los nombres que se van a utilizar como encabezados de columnas en lugar de los nombres que tienen los campos en la tabla.
Listatablas	Es el nombre de las tablas.
Criterio	Es una expresión que determina qué registros se devolverán. Sólo se seleccionan los registros que cumplen con la expresión.
Listacampos	Lista de campos en base a la cual se ordenarán los registros devueltos en orden ascendente o descendente.

Ejemplos:

1) Selecciona todos los registros de la tabla “Articulos” y ordena la lista en base al campo “clvArtículo”: SELECT * FROM Articulos ORDER BY clvArtículo

2) Selecciona todos los registros de la tabla “Articulos”, devolviendo el campo “artículo”: SELECT artículo FROM Articulos

3) De la tabla “Articulos” selecciona todos los registros cuando el contenido del campo “clvArtículo” sea igual al valor AC001P:
SELECT artículo FROM Articulos WHERE clvArtículo = ‘AC001P’

4) Selecciona los artículos que tiene la factura 8052 de la empresa cuya clave de control es “DP”:

```
SELECT Facturas.EMPRESA, Facturas.FECHA, Facturas.FACTURA,
   Articulos_Factura.ARTICULO, Articulos.DESCRIPCION, Articulos_Factura.CANTIDAD,
   Articulos_Factura.PARTIDA, Facturas.CLIENTE, Clientes.NOMBRE
FROM Articulos_Factura, Facturas, Clientes, Articulos
WHERE Facturas.EMPRESA = 'DP' AND Facturas.FACTURA = 8052 AND
   Articulos_Factura.FACTURA = Facturas.FACTURA AND
   Articulos_Factura.EMPRESA = Facturas.EMPRESA AND
   Facturas.CLIENTE = Clientes.CLIENTE AND
   Facturas.EMPRESA = Clientes.EMPRESA AND
   Articulos_Factura.EMPRESA = Clientes.EMPRESA AND
   Articulos_Factura.ARTICULO = Articulos.ARTICULO AND
   Articulos_Factura.EMPRESA = Articulos.EMPRESA AND
   Facturas.EMPRESA = Articulos.EMPRESA AND
   Clientes.EMPRESA = Articulos.EMPRESA
```

ORDER BY Articulos_Factura.PARTIDA

En esta ejemplo:

1. En la cláusula SELECT se anotan los campos que se desean seleccionar de cada tabla.
2. En la cláusula FROM se anotan las tablas de las que se debe hacer la selección.
3. En la cláusula WHERE se anotan las condiciones que se deben cumplir, así como las relaciones que debe existir entre las tablas involucradas. En esta cláusula las constantes de tipo cadena se encierran entre apóstrofos (Facturas.EMPRESA = 'DP') , y las constantes de tipo numérico van sin apóstrofos (Facturas.FACTURA = 8052).
   En el caso de las constantes de tipo fecha, si estas se anotan entre apóstrofos el formato que se considerará es de la forma: mm/dd/yyyy. Pero para que sea considerado de la forma dd/mm/yyyy, la constante debe escribirse de la siguiente forma: Facturas.FECHA = DateValue('04/02/1984'). También se puede utilizar la almohadilla (#) en lugar del apostrofo: Facturas.FECHA = DateValue(#04/02/1984#)
4. En la cláusula WHERE también se puede hacer la selección de un conjunto de registros, cuyo contenido del campo FECHA esté ubicado en un rango de fechas, utilizando el operador Between...And, según se muestra en la siguiente línea: Facturas.FECHA BETWEEN DateValue('01/02/1984') And DateValue('29/04/1984').
5. En la cláusula ORDER BY se ordena la información en base a los campos deseados.
6. Debe observarse al seleccionar atributos de diferentes tablas, se escribe el nombre de la tabla seguida de un punto, y en seguida del mismo el nombre del atributo que se desea. El resultado será lo que se llama "vista" o "consulta".

❖ **Instrucción para borrar**

Se utiliza para borrar registros de una o más tablas listadas en la cláusula FROM y que cumplen con la condición de la cláusula WHERE.

DELETE [tabla.*] FROM tabla WHERE criterio

En donde

Parte de la instrucción	Explicación
*	Indica todos los campos de la tabla o tablas especificadas.
Tabla	El nombre de la tabla cuyos registros se van a borrar.
Criterio	Es una expresión que determina qué registros se van a borrar.

Ejemplo:

En el siguiente ejemplo se borra el registro que cumple con la condición ClvArtículo='NYTEXT':

DELETE * FROM Articulos WHERE ClvArtículo = 'NYTEXT'

Nota: Únicamente se puede borrar información en una tabla a la vez.

1.5.2 Visual Basic

A continuación se exponen algunas instrucciones más utilizadas en la programación con el lenguaje Visual Basic. [Ref. 2]

❖ Conexión a una base de datos

Cuando el programa y la base de datos residen en el mismo subdirectorio, y es para un ambiente monousuario, se puede utilizar el siguiente fragmento de código:

```
‘  Declaración de variables en un archivo tipo “.BAS”
‘  Para conexiones de tipo jerárquico, similar al llamado “maestro/detalle”:
    Global Const cadMSDataShape = "PROVIDER=MSDataShape;Data "

‘  Para conexiones de tipo simple:
    Global Const cadDBSimple_1 = “Provider=Microsoft.Jet.OLEDB.4.0;Data Source=”

‘  Complemento de la conexión:
    Global Const cadDBSimple_2 = "\Bdsad.mdb;Persist Security Info=False;
                                  Jet OLEDB: Database Password = ClaveDeAcceso;"

‘  Si la base de datos no tiene clave de acceso la cadena se reduce a la declaración
    Global Const cadDBSimple_3 = "\Bdsad.mdb;Persist Security Info=False;
'  Variable para establecer la conexión hacia la base de datos.
    Public conDB As New ADODB.Connection

‘  Variable que guardará la cadena de conexión hacia la base de datos
    Public cadconDB As String

‘ Subrutina para efectuar la conexión hacia la base de datos
‘ Normalmente se utiliza esta rutina para iniciar la ejecución de un programa, y debe
‘ estar en un archivo tipo “.BAS”
Sub Main()
  conDB.CursorLocation = adUseClient
  conDB.Open cadDSN_MSDASQL

‘ Se almacena en la variable cadconDB la cadena de conexión hacia la base de datos
‘ En esta línea la instrucción App.Path captura la ruta en la que reside el programa, y
‘ la unión de cadenas se puede efectuar con el carácter “&” como se muestra a continuación
    cadconDB = cadDBSimple_1 & App.Path & cadDBSimple_2

 ‘ Se establece la conexión
    conDB.CursorLocation = adUseClient
    conDB.Open cadconDB
End Sub
```

Es recomendable declarar y establecer la conexión hacia la base de datos al iniciar la ejecución del programa, pues con esto se evita el declarar una variable de conexión por cada formulario que se implemente, es decir, una vez que se ha efectuado la conexión con la base de datos es suficiente utilizar la variable conDB, tantas veces sea necesario para abrir una tabla de la base de datos.

La declaración:

```
Global Const cadDBSimple_1 = “Provider=Microsoft.Jet.OLEDB.4.0;Data Source=”
```

es preferible sustituirla por:

```
Global Const cadDBSimple_1 = "DRIVER=Microsoft Access Driver (*.mdb);DBQ="
```

pues ésta es más general y no tiene problemas de versión con el proveedor de datos Microsoft Jet de Access

.

Si la base de datos está instalada en ambiente multiusuario, es decir, trabajará como origen de datos ODBC, entonces la declaración de la variable cadDBSimple_1 se debe cambiar por la siguiente línea:

```
Global Const cadDBSimple_1 = _
    "PROVIDER=MSDASQL;dsn=NombreOrigenDeDatos;uid=;pwd=ClaveDeAcceso;
    database=NombreBaseDeDatos;"
```

y cambiar la expresión: conDB.Open cadconDB, por la expresión:

```
conDB.Open cadconDB cadDBSimple_1
```

❖ **Abrir una tabla de la base de datos**

Una vez que se ha efectuado la conexión con la base de datos ya se puede abrir alguna de sus tablas. Es recomendable utilizar las propiedades del control de acceso a datos ADO y no el control mismo, como se muestra en el siguiente fragmento de código:

```
' Declaración de la variable que contendrá el conjunto de registros
  Dim rsTabla As New ADODB.Recordset

' Declaración de la variable cadSQL de tipo cadena que contendrá la selección de registros
  Dim cadSQL As String

' Se asigna a cadSQL la selección de registros
  cadSQL = "SELECT * From Articulos ORDER BY ClvArtículo"

' Si ya se ha utilizado rsTabla, es conveniente cerrar la selección previa
  If rsTabla.State = adStateOpen Then rsTabla.Close
  rsTabla.Open cadSQL, conDB, adOpenDynamic, adLockOptimistic  ' Abre la selección
' Enlaza el control "datgrdTabla" a rsTabla para poder ver los registros seleccionados
  Set datgrdTabla.DataSource = rsTabla          ' Este control es un objeto DataGrid
```

La variable conDB ya tiene en dónde está ubicada la base de datos, y estará activa mientras esté funcionando el programa. Además la variable cadSQL tiene una instrucción del lenguaje SQL expresada como cadena.

❖ **Agregar información en una tabla de la base de datos**

Una vez que se ha establecido la conexión a la base de datos, es posible agregar información en alguna de sus tablas, como se muestra a continuación:

```
' Declaración de la variable que contendrá el conjunto de registros
    Dim rsAux As New ADODB.Recordset

' Declaración de la variable cadSQL de tipo cadena que contendrá la selección de registros
    Dim cadSQL As String
```

```
‘ Se asigna a cadSQL la instrucción SQL a efectuar
  cadSQL = "INSERT INTO Articulos (ClvArtículo, Artículo) " & _
       "VALUES ('" & Text1(0).Text & "', '" & Text1(1).Text & "')"
  rsAux.Open cadSQL, conDB, adOpenDynamic, adLockOptimistic ' Agrega el registro
  rsTabla.Requery    ‘ Actualiza los registros de rsTabla
```

La variable cadSQL tiene una instrucción del lenguaje SQL expresada como cadena, y en ella se intercalan el contenido de los controles Text1(0).Text y Text1(1).Text. Nótese la forma de concatenarlos para que sus contenidos estén dentro de la instrucción SQL.

El siguiente bloque de código también hace lo mismo:

```
‘ Declaración de la variable que contendrá el conjunto de registros
  Dim rsTabla As New ADODB.Recordset

‘ Declaración de la variable cadSQL de tipo cadena que contendrá la selección de registros
  Dim cadSQL As String

‘ Se asigna a cadSQL la instrucción SQL a efectuar
  cadSQL = "SELECT  ClvArtículo, Artículo FROM Artículos”
  If rsTabla.State = adStateOpen Then rsTabla.Close        ‘ Se cierra la selección previa

‘ Abre la nueva selección ordenada
  rsTabla.Open cadSQL, conDB, adOpenDynamic, adLockOptimistic
  rsTabla.AddNew   ‘ Agrega un nuevo registro en blanco

‘ El contenido del control Text1(0).Text se asigna al campo ClvArtículo
  rsTabla!ClvArtículo = Text1(0).Text
‘ El contenido del control Text1(1).Text se asigna al campo Artículo
  rsTabla!Artículo = Text1(1).Text
  rsTabla.Update          ‘ Guarda la información en la tabla Artículos
  rsTabla.Requery         ‘ Actualiza los registros de rsTabla
```

En este caso previamente debe estar abierta la tabla, además este método requiere de más líneas de instrucción.

❖ **Modificar información en una tabla de la base de datos**

Una vez que se ha abierto la conexión a la base de datos, también es posible modificar información en alguna de sus tablas, como se muestra a continuación:

```
‘ Declaración de la variable que contendrá el conjunto de registros
  Dim rsAux As New ADODB.Recordset

‘ Declaración de la variable cadSQL de tipo cadena que contendrá la selección de registros
  Dim cadSQL As String

‘ Se asigna a cadSQL la instrucción SQL a efectuar
  cadSQL = "UPDATE Articulos SET ClvArtículo = '" & Text1(0).Text & _
      "', Artículo = '" & Text1(1).Text & _
      "' WHERE ClvArtículo = '" & rsTabla!ClvArtículo & "'"
```

```
    rsAux.Open cadSQL, conDB, adOpenDynamic, adLockOptimistic       ' Modifica el registro
    rsTabla.Requery    ‘ Actualiza los registros de rsTabla
```

La variable cadSQL tiene una instrucción del lenguaje SQL expresada como cadena, y en ella se intercalan el contenido de los controles Text1(0).Text y Text1(1).Text, así como el contenido del campo rsTabla!ClvArtículo Nótese la forma de concatenarlos para que sus contenidos estén dentro de la instrucción SQL.

En este caso rsTabla es una tabla previamente abierta y cuyo campo ClvArtículo tiene la clave del artículo a modificar.

El siguiente bloque de código también hace lo mismo:

```
‘ Declaración de la variable que contendrá el conjunto de registros
    Dim rsAux As New ADODB.Recordset

‘ Declaración de la variable cadSQL de tipo cadena que contendrá la selección de registros
    Dim cadSQL As String

’ Si sucede algún error la secuencia de ejecución pasa a la siguiente línea que está
‘ después de la etiqueta Salir
    On Error GoTo Salir

‘ Se asigna a cadSQL el mensaje que aparecerá en la caja de diálogo
‘ Aquí vbCrLf efectúa un salto de línea
    cadSQL = "¿Se cambia el siguiente artículo?" & vbCrLf & _
        rsTabla!ClvArtículo & " - " & rsTabla!Artículo

‘ Manda un cuadro de mensaje solicitando confirmación
    If MsgBox(cadSQL, vbQuestion + vbYesNo, " C o n f i r m a r ") = vbNo Then
      Text1(0).SetFocus    ‘ Fija el enfoque en el control Text1(0)
      Exit Sub             ‘ Sale de la subrutina
    End If
‘ Se asigna a cadSQL la instrucción SQL a efectuar
    cadSQL = "SELECT  ClvArtículo, Artículo “ & _
        “FROM Artículos “ & _
        “WHERE  ClvArtículo = ‘” & rsTabla!ClvArtículo & “’”
    If rsAux.State = adStateOpen Then rsAux.Close   ‘ Se cierra la selección previa

‘ Abre la nueva selección ordenada
    rsAux.Open cadSQL, conDB, adOpenDynamic, adLockOptimistic

‘ El contenido del control Text1(0).Text se asigna al campo ClvArtículo
    rsAux!ClvArtículo = Text1(0).Text

‘ El contenido del control Text1(0).Text se asigna al campo Artículo
    rsAux!Artículo = Text1(1).Text
    rsAux.Update             ‘ Guarda la información en la tabla Artículos
    If rsAux.State = adStateOpen Then rsAux.Close   ‘ Se cierra la selección previa de rsAux
    rsTabla.Requery          ‘ Actualiza los registros de rsTabla
```

```
    Exit Sub            ‘ Sale de la subrutina
Salir:  ‘ Si ocurre algún error este es capturado y aparece el mensaje de error
    MsgBox Err.Number & " - " & Err.Description, vbCritical & vbOKOnly, "E R R O R ..."
```

En este caso rsTabla es una tabla previamente abierta y cuyo campo ClvArtículo tiene la clave del artículo a modificar, además este método requiere de más líneas de instrucción.

❖ **Borrar información de una tabla de la base de datos**

Una vez que se ha abierto la conexión a la base de datos, también es posible borrar información de alguna de sus tablas, como se muestra a continuación:

```
‘ Declaración de la variable que contendrá el conjunto de registros
    Dim rsAux As New ADODB.Recordset

‘ Declaración de la variable cadSQL de tipo cadena que contendrá la selección de registros
    Dim cadSQL As String

‘ Se asigna a cadSQL el mensaje que aparecerá en la caja de diálogo
‘ Aquí vbCrLf efectúa un salto de línea
    cadSQL = "¿Se borra el siguiente artículo?" & vbCrLf & _
        rsTabla!ClvArtículo & " - " & rsTabla!Artículo

‘ Manda un cuadro de mensaje solicitando confirmación de borrado
    If MsgBox(cadSQL, vbQuestion + vbYesNo, " C o n f i r m a r ") = vbNo Then
      Text1(0).SetFocus    ‘ Fija el enfoque en el control Text1(0)
      Exit Sub             ‘ Sale de la subrutina
    End If

‘ Se asigna a cadSQL la instrucción SQL a efectuar
cadSQL = "DELETE * FROM Articulos WHERE ClvArtículo = '" & rsTabla!ClvArtículo & "'"
rsAux.Open cadSQL, conDB, adOpenDynamic, adLockOptimistic          ' Borra el registro
rsTabla.Requery    ‘ Actualiza los datos de rsTabla
```

La variable cadSQL también tiene una instrucción del lenguaje SQL expresada como cadena, y en ella se intercala el contenido del campo rsTabla!ClvArtículo, el cual tiene la clave del artículo a borrar.

El siguiente bloque de código también hace lo mismo:

```
‘ Declaración de la variable que contendrá el conjunto de registros
    Dim rsAux As New ADODB.Recordset  ‘ Declaración de la variable

‘ Declaración de la variable cadSQL de tipo cadena que contendrá la selección de registros
    Dim cadSQL As String

‘ Se asigna a cadSQL el mensaje que aparecerá en la caja de diálogo
‘ Aquí vbCrLf efectúa un salto de línea
    cadClvAnterior = "¿Se borra el siguiente artículo?" & vbCrLf & _
        rsTabla!ClvArtículo & " - " & rsTabla!Artículo
```

```
‘ Manda un cuadro de mensaje solicitando confirmación de borrado
    If MsgBox(cadClvAnterior, vbQuestion + vbYesNo, " C o n f i r m a r ") = vbNo Then
      Text1(0).SetFocus    ‘ Fija el enfoque en el control Text1(0)
      Exit Sub             ‘ Sale de la subrutina
    End If

‘ Se asigna a cadSQL la instrucción SQL a efectuar
    cadSQL = "SELECT  ClvArtículo, Artículo “ & _
        “FROM Artículos “ & _
        “WHERE  ClvArtículo = ‘” & rsTabla!ClvArtículo & “’”
    If rsAux.State = adStateOpen Then rsAux.Close   ‘ Se cierra la selección previa

‘ Abre la nueva selección ordenada
    rsAux.Open cadSQL, conDB, adOpenDynamic, adLockOptimistic
    rsAux.Delete      ‘ Borra el registro
    If rsAux.State = adStateOpen Then rsAux.Close   ‘ Se cierra la selección previa de rsAux
    rsTabla.Requery         ‘ Actualiza los registros de rsTabla
    Exit Sub            ‘ Sale de la subrutina
Salir:  ‘ Si ocurre algún error este es capturado y aparece el mensaje de error
    MsgBox Err.Number & " - " & Err.Description, vbCritical & vbOKOnly, "E R R O R ..."
```

En este caso rsTabla es una tabla previamente abierta y cuyo campo ClvArtículo tiene la clave del artículo a borrar, además este método requiere de más líneas de instrucción.

En el anexo A se trata la implementación de una tabla virtual

Capítulo 2
Análisis y diseño

2.1 Planteamiento del problema

Aún hoy en día, muchas pequeñas empresas dedicadas a la industria del vestido y del calzado, llevan el control de sus inventarios y de sus ventas anotándolas en libretas, y en el mejor de los casos realizan esta tarea en Excel. De igual forma llevan sus: compras, cuentas por pagar, cuentas por cobrar; y hacen comparaciones estadísticas de una libreta a otra, con lo que obligan a sus contadores a trabajar largas horas en esta consulta.

Aunque en el mercado hay muchos programas que realizan estas tareas no todos se ajustan a las necesidades de las empresas, además de que son muy complicados de entender por parte de quienes no tiene conocimientos mínimos de computación, y algunos de estos programas exigen que el usuario sepa un poco de programación

Ante esta problemática es necesaria la implementación de un programa que sea más amigable y sin necesidad de saber algún lenguaje de programación.

El programa debe cumplir con los siguientes puntos:

- Conocer la existencia de las materias primas en cualquier momento.
- Establecer la cantidad de las materias primas con las que se fabrique cada artículo. Y poder cambiar el precio de venta de estos.
- Registrar las ordenes de compra dirigidas a los proveedores de materia prima, de las cuales, una vez recibidas, darles entrada a su existencia. Poder consultar las deudas que se tengan pendientes de pago hacia los proveedores, y realizar pagos parciales.
- Registrar los pedidos de artículos que soliciten los clientes.
- Distribuir por zonas a los vendedores, y registrar las ventas que cada uno de estos vaya realizando, con el fin de obtener la comisión que a cada uno de ellos corresponde.
- Registrar las ventas de artículos, que se hagan a través de remisiones y de facturas que soliciten los clientes. Al imprimir estos documentos se debe incluir los artículos de que consta y, el monto total con número y letra, además del nombre del cliente. En caso de quedar alguna venta pendiente de pago poder consultar las deudas que tengan los clientes hacia la empresa, y realizar cobros parciales. Que se puedan presentar en una misma lista los documentos que el cliente tiene pendientes con las empresas
- Dar acceso, a ciertas tareas del programa, a un determinado número de usuarios.
- Permitir modificar el porcentaje de los impuestos que determina la SHCP.
- Permitir modificar la numeración de pedidos remisiones y facturas.
- Que se pueda conocer qué artículos se han vendido a cada cliente en un determinado período, así como los artículos que se han vendido, en general, en un determinado período.

Estas actividades se deben considerar para más de una empresa.

❖ Objetivos

- **General**

Desarrollar un programa para controlar los elementos de más de una empresa dedicadas a la industria textil manufacturero, así como de la industria de la confección del vestido y del calzado.

Los elementos a controlar son: materias primas, artículos, ordenes de compra, proveedores, las deudas que tenga cada empresa, vendedores, clientes, pedidos, ventas

de artículos, las deudas que tengan los clientes hacia las empresas, usuarios, numeración de documentos.

- **Específicos**

  Se deben cubrir los siguientes puntos:

  - Materia prima: construir su catálogo con la información de cada materia prima, y poder conocer la existencia de cada materia prima
  - Artículos: construir su catálogo con la información de cada artículo, y establecer las materias primas con las que se fabrique cada artículo.
  - Cambio de precios: cambiar el precio de venta a un determinado conjunto de artículos, de una sola vez.
  - Entradas: registrar las entradas de materia prima que se compren, considerando al proveedor que la proporcione.
  - Salidas: registrar las salidas de materia prima que sean devueltas al proveedor, debido a algún defecto de estas.
  - Proveedores: construir su catálogo con la información de cada proveedor
  - Compras: registrar las ordenes de compra de materia prima, considerando el proveedor a quien va dirigida.
  - Clientes: construir su catálogo con la información de cada cliente.
  - Zonas: construir su catálogo en el que estarán distribuidos los vendedores.
  - Vendedores: construir su catálogo con la información de cada vendedor
  - Pedidos: registrar los pedidos de artículos, considerando el cliente que lo solicite.
  - Remisiones: registrar las remisiones de las ventas efectuadas, considerando los clientes que efectuaron la compra. En este caso la existencia de materia prima no se modifica. Además transferir cada remisión a Cuentas por cobrar.
  - Facturas: registrar las facturas de las ventas efectuadas, considerando los clientes que efectuaron la compra. En este caso la existencia de materia prima se modifica. Además transferir cada factura a Cuentas por cobrar.
  - Registrar las cuentas que queden pendientes de pagar, y efectuar pagos parciales de cada una de estas.
  - Registrar las cuentas que queden pendientes de pagar, y efectuar pagos parciales de cada una de estas, de cada empresa
  - Registrar las cuentas que queden pendientes de cobro, y efectuar pagos parciales de cada una de estas, de cada cliente.
  - Compras y ventas: reflejar el resumen de las compras y de las ventas que haya efectuado cada empresa, durante un determinado período.
  - Ventas por cliente: reflejar las ventas, de artículos, efectuadas a cada cliente durante un determinado período.
  - Ventas por artículo: reflejar las ventas efectuadas, de artículos, durante un determinado período.
  - Empresas: construir su catálogo con la información de cada empresa.
  - Usuarios: dar acceso, a ciertas tareas del programa, a un determinado número de usuarios.
  - Impuestos: poder modificar el porcentaje de los impuestos que determina la SHCP.
  - Numeración: poder modificar la numeración de pedidos remisiones y facturas.

2.2 Especificación de requerimientos

Para poder tener control de todo lo antes mencionado primero se desarrollará la base de datos, en la que se guardará toda la información que se procesará, para lo cual se utilizará Access; y para la implementación del programa que gestionará dicha base se empleará el lenguaje de programación Visual Basic versión 6.0, pues se cuenta con la licencia de este.

Además se requiere como mínimo de: una computadora con procesador Celeron de 400 Mhz., sistema operativo Windows 2000, el programa generador de reportes Crystal Report versión 4.6. Cabe mencionar que el programa puede funcionar muy bien en cualquier versión de Windows: Windows 98, Windows ME, Windows 2000, Windows 2003 y Windows XP.

Se debe contar con una computadora trabajando como servidor en la que se almacene la información, dicho servidor debe tener como mínimo: procesador Celeron de 1.8 Ghz., disco duro de 80 Gb., sistema operativo Windows 2000.

Cada estación de trabajo, que tendrá acceso al servidor, deberá tener como mínimo: procesador Celeron de 400 Mhz., sistema operativo Windows 2000, el programa para manejar la base de datos y el generador de reportes Cristal Report versión 4, para que se puedan ejecutar los reportes que se desarrollarán con este programa, pues se instala el runtime de este.

2.3 Análisis

Antes de implementar la base de datos se identifican las entidades y las relaciones que la formarán.

Las entidades identificadas son:

- Empresa
- Proveedores
- Materia prima
- Artículos
- Entradas
- Salidas
- Compras
- Clientes
- Vendedores
- Estados
- Pedidos
- Remisiones
- Facturas
- Cuentas por pagar.
- Cuentas por cobrar.
- Bancos
- Usuarios
- Impuestos
- Numeros

A continuación se exponen sus respectivos modelo E-R.

➢ Entidades

❖ Empresa

La información que se guardará de cada empresa son sus datos generales, y su diagrama de entidad se muestra en la figura 2.1

Figura 2.1 Diagrama de entidad de Empresas

Su clave primaria está formada por el campo “empresa” (campo primo).

La representación de la dependencia funcional entre sus campos es:

empresa → razonsocial, direccion, colonia, cp, rfc, email, telefono, fax, registrotext

❖ **Impuesto**

Únicamente hay un impuesto, el cual es el IVA, y el diagrama E-R de esta entidad se muestra en la figura 2.2

Figura 2.2 Diagrama de entidad Impuesto

Su clave primaria está formada por el campo clvimp (campo primo).
La representación de la dependencia funcional entre sus campos es:clave→nom, monto

❖ **Numeros**

La información que se guardará es la numeración de los documentos de cada empresa, y su diagrama de E-R se muestra en la figura 2.3

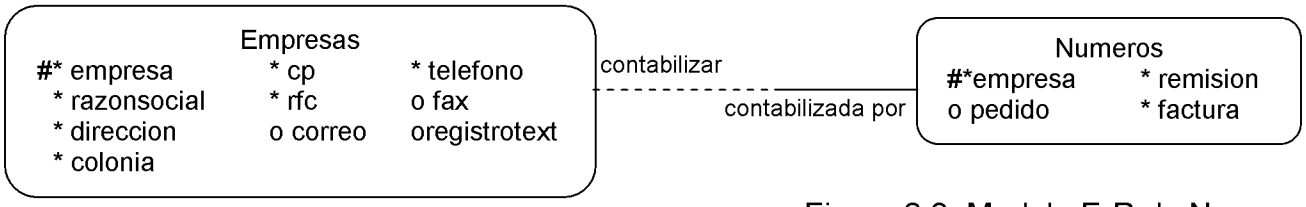


Figura 2.3: Modelo E-R de Numeros

Su clave primaria está formada por el campo: empresa (campo primo).
La representación de la dependencia funcional entre sus campos es:

empresa → pedido, remision, factura

❖ **Proveedores**

La información que se guardará de cada proveedor son sus datos generales, y su diagrama de E-R se muestra en la figura 2.4

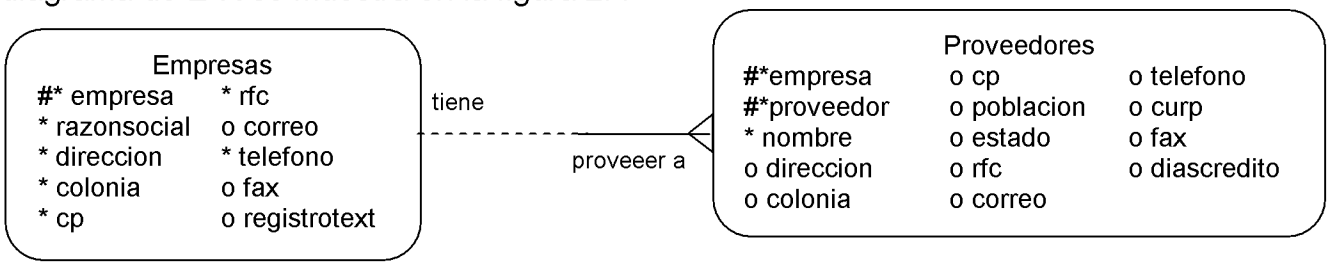


Figura 2.4: Modelo E-R de Proveedores

Su clave primaria está formada por los campos: empresa, proveedor (campos primos).
La representación de la dependencia funcional entre sus campos es:

empresa, proveedor → nombre, dirección, colonia, cp, rfc, correo, estado, población, telefono, fax, diascredito, curp

❖ **Materia prima**

La información que se guardará de cada materia prima se muestra en la figura 2.5, que es su modelo E-R de esta entidad.

Su clave primaria está formada por los campos: empresa, materiaprima (campos primos).

La representación de la dependencia funcional entre sus campos es:
empresa, materiaprima → descripción, unidad, existencia, stockmin, stockmax, coniva

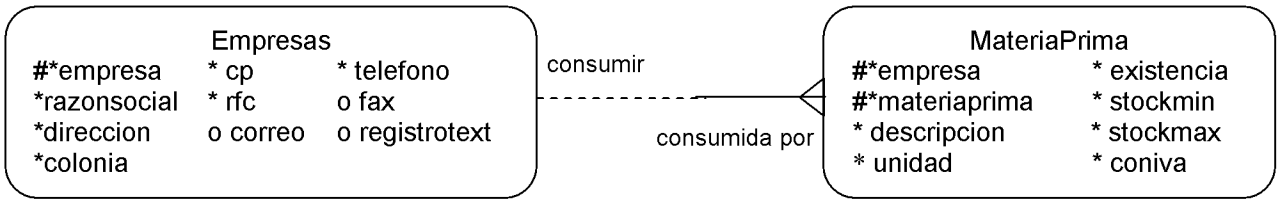


Figura 2.5: Modelo E-R de Materia prima

❖ **Articulos**

La información que se guardará de cada artìculo se muestra en la figura 2.6, que es el modelo E-R de esta entidad.

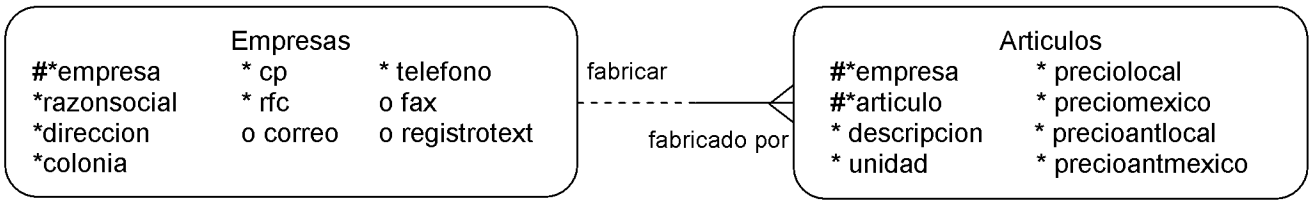


Figura 2.6: Modelo E-R de Artìculos

Su clave primaria está formada por los campos: empresa, articulo (campos primos).
La representación de la dependencia funcional entre sus campos es:

empresa, articulo → descripcion, unidad, preciolocal, preciomexico, precioantlocal, precioantméxico

❖ **Compras**

La información que se guardará de cada orden de compra se muestra en la figura 2.7, que es el modelo E-R de esta entidad.

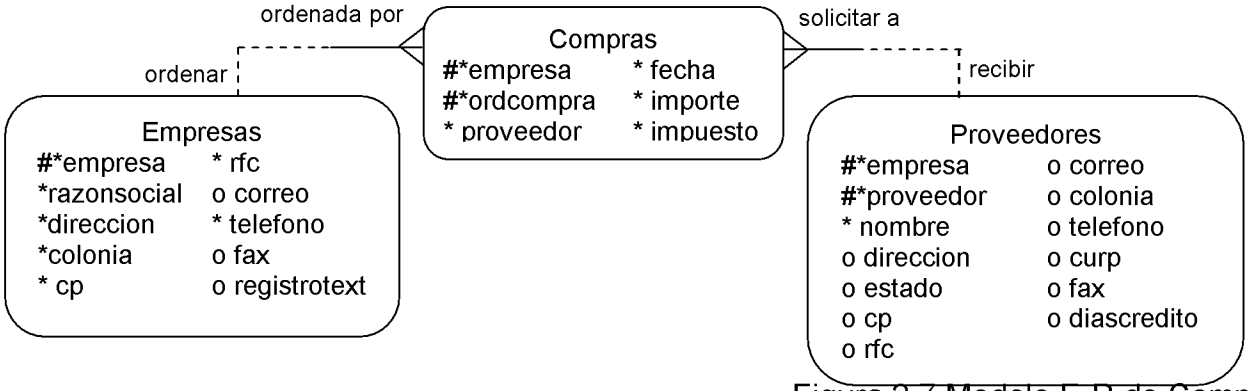


Figura 2.7 Modelo E-R de Compras

Su clave primaria está formada por los campos: empresa, ordcompra (campos primos).
La representación de la dependencia funcional entre sus campos es:

empresa, ordcompra → fecha, proveedor, importe

❖ Entradas

La información que se guardará de cada entrada se muestra en la figura 2.8, que es el modelo E-R de esta entidad.

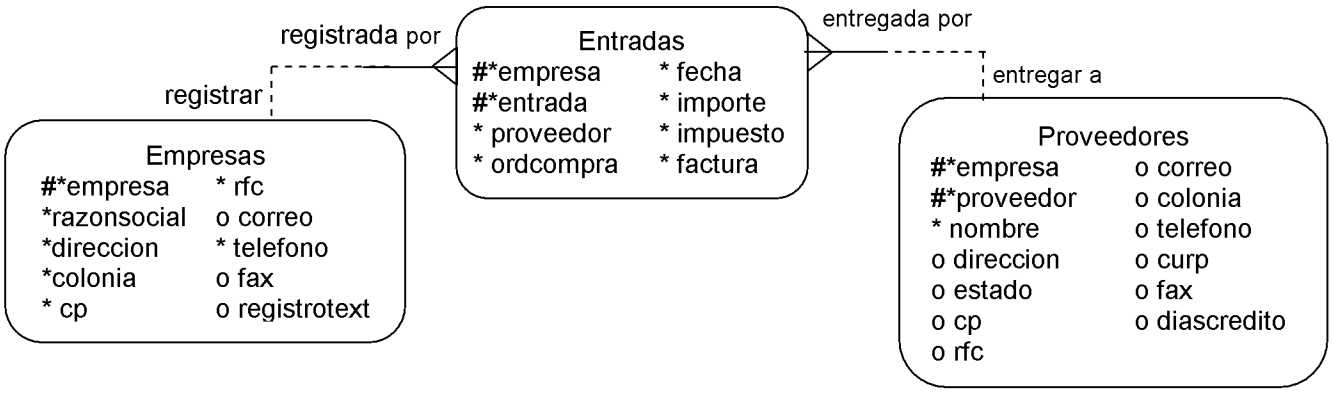


Figura 2.8: Modelo E-R de Entradas

Su clave primaria está formada por los campos: empresa, entrada (campos primos).
La representación de la dependencia funcional entre sus campos es:
empresa, entrada → fecha, proveedor, ordcompra, impuesto, importe, factura

❖ Salidas

La información que se guardará de cada salida se muestra en la figura 2.9, que es el modelo E-R de esta entidad. En este caso solo dará salida a la materia prima que sea devuelta a algún proveedor.

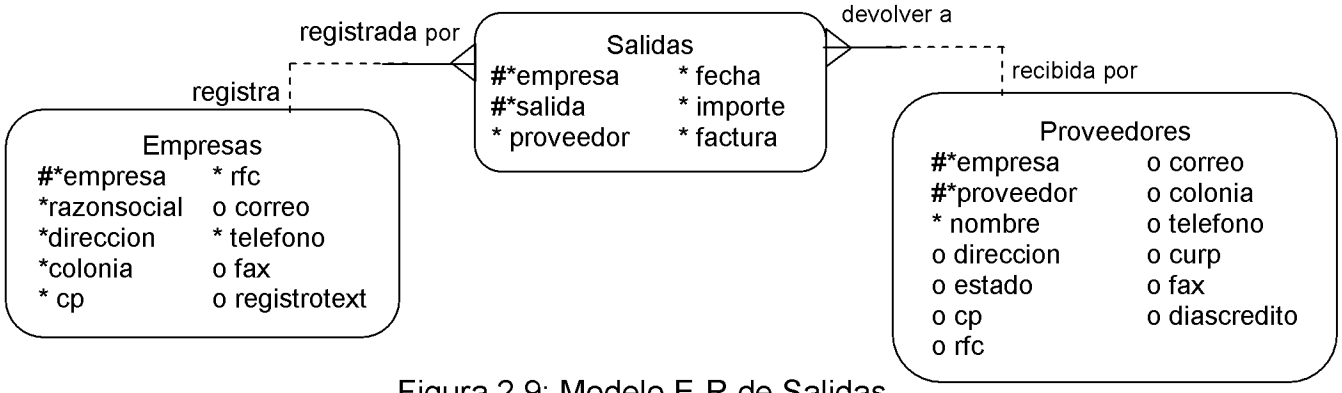


Figura 2.9: Modelo E-R de Salidas

Su clave primaria está formada por los campos: empresa, salida (campos primos).
La representación de la dependencia funcional entre sus campos es:
empresa, salida → fecha, proveedor, importe, factura

❖ TipoCxP

Las entidades TipoCxP son conceptos de cuentas por pagar, y su diagrama de E-R se muestra en la figura 2.10.

Su clave primaria está formada por el campo "clavecxp" (campo primo).
La representación de la dependencia funcional entre sus campos es: clvcxp→ descrip

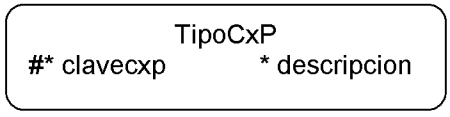


Figura 2.10 Diagrama de entidad de TipoCxP

❖ **Bancos**

Las entidades Bancos son las instituciones bancarias que se podrán utilizar, y su diagrama de E-R se muestra en la figura 2.11

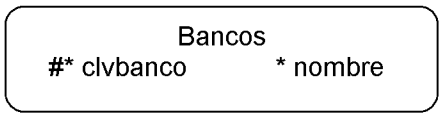


Figura 2.11 Diagrama de entidad de Bancos

Su clave primaria está formada por el campo "clvbanco" (campo primo).
La representación de la dependencia funcional entre sus campos es: clvbanco→nombre

❖ **Cuentas por pagar (CxP)**

La información que se guardará de cada cuenta por pagar se muestra en la figura 2.12, que es el modelo E-R de esta entidad.

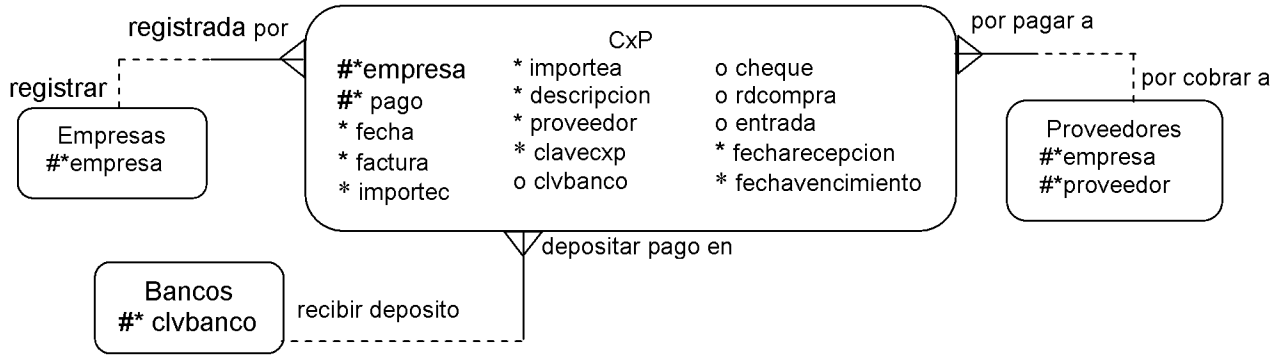


Figura 2.12 Diagrama de entidad de CxP

Su clave primaria está formada por los campos empresa, pago (campos primos).
La representación de la dependencia funcional entre sus campos es:

empresa, pago → fecha, factura, fechavencimiento, fecharecepcion, descripción, importec, importea, proveedor, clavecxp, clvbanco, ordcompra, entrada

❖ **Estados**

Las entidades Estados son los estados de la República Mexicana, y en la figura 2.13 se muestra su diagrama de E-R.

Su clave primaria está formada por el campo clvedo (campo primo).
La representación de la dependencia funcional entre sus campos es: clvedo → estado

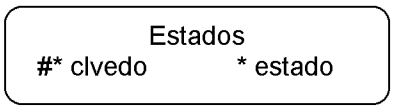


Figura 2.13 Diagrama de entidad de Estados

❖ **Municipios**

Las entidades Municipios son los municipios de cada Estado de la República Mexicana, y su diagrama de E-R se muestra en la figura 2.14.

Su clave primaria está formada por los campos: clvedo, clvmun (campos primos).
La representación de la dependencia funcional entre sus campos es:

clvedo, clvmun → Municipio

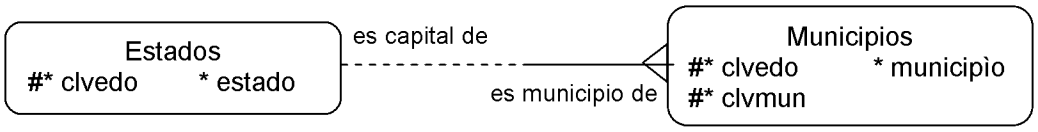


Figura 2.14: Modelo E-R de Municipios

Su clave primaria está formada por los campos: clvedo, clvmun (campos primos).
La representación de la dependencia funcional entre sus campos es:

clvedo, clvmun → Municipio

❖ **Zonas**

Las zonas es la forma en que se distribuirán los vendedores, y en las cuales se localizan los clientes, y su diagrama de E-R se muestra en la figura 2.15

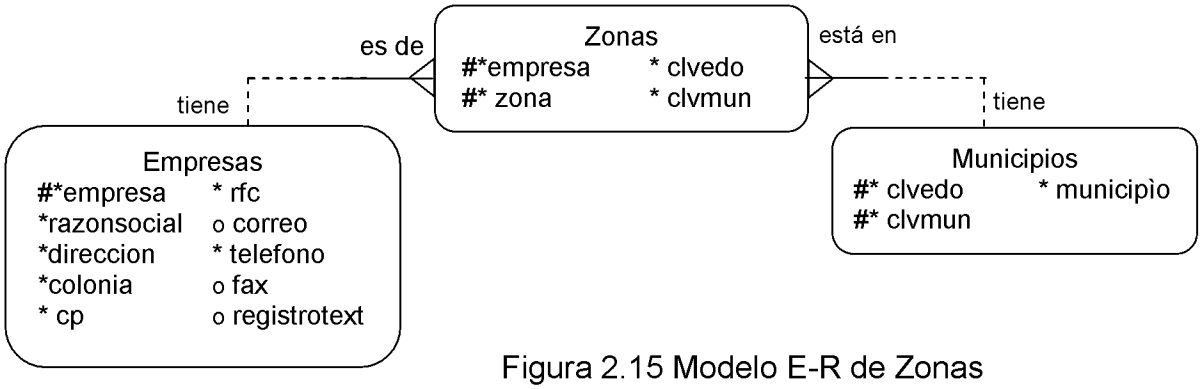


Figura 2.15 Modelo E-R de Zonas

Su clave primaria está formada por los campos: empresa, clvestado, clvmunicipio (campos primos).
La representación de la dependencia funcional entre sus campos es:

empresa, zona → clvestado, clvmunicipio

❖ **Vendedor**

La información que se guardará de cada vendedor son sus datos generales, y su diagrama de E-R se muestra en la figura 2.16.

Su clave primaria está formada por los campos: empresa, vendedor (campos primos).
La representación de la dependencia func7ional entre sus campos es:

empresa, vendedor → nombre,dirección,colonia,cp,rfc, telefono, comision

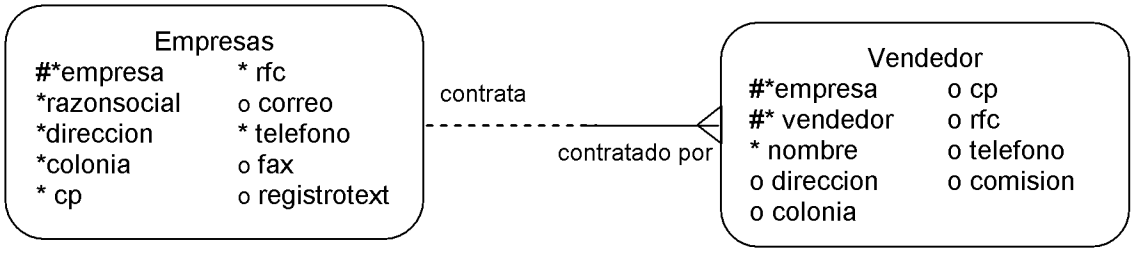


Figura 2.16 Modelo E-R de Vendedores

Su clave primaria está formada por los campos: empresa, vendedor (campos primos).
La representación de la dependencia funcional entre sus campos es:

empresa, vendedor → nombre,dirección,colonia,cp,rfc, telefono, comision

❖ **Clientes**

La información que se guardará de cada cliente son sus datos generales, y su diagrama de E-R se muestra en la figura 2.17

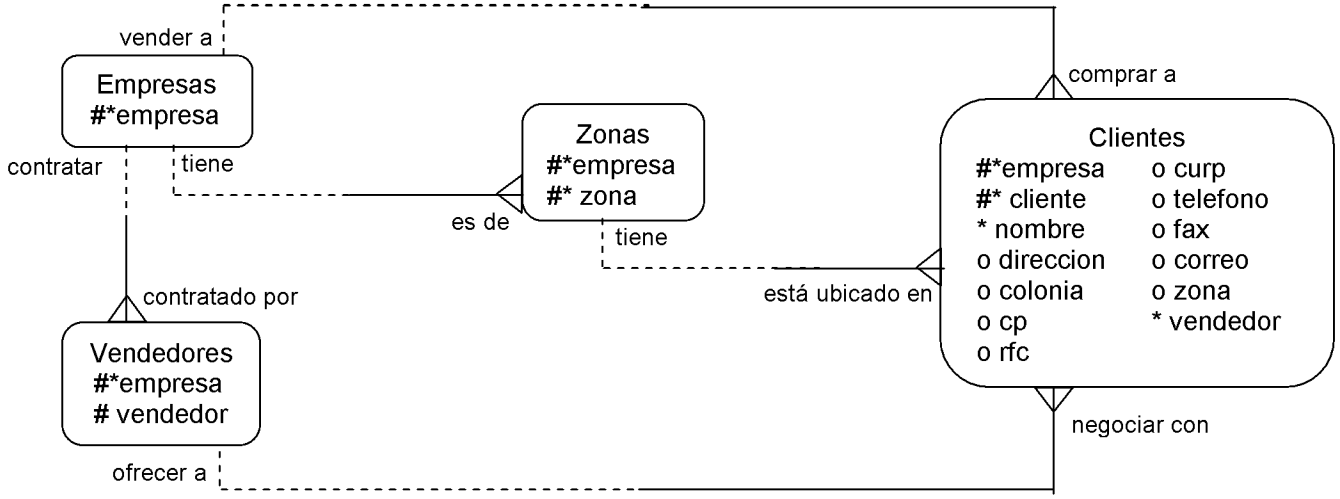


Figura 2.17 Modelo E-R de Clientes

Su clave primaria está formada por los campos: empresa, cliente (campos primos).
La representación de la dependencia funcional entre sus campos es:

empresa, cliente → nombre, dirección, colonia, cp, rfc, correo, estado, población, telefono, fax, diascredito, curp, zona, vendedor

❖ **Pedidos**

La información que se guardará de cada pedido se muestra en la figura 2.18, que es el modelo E-R de esta entidad.

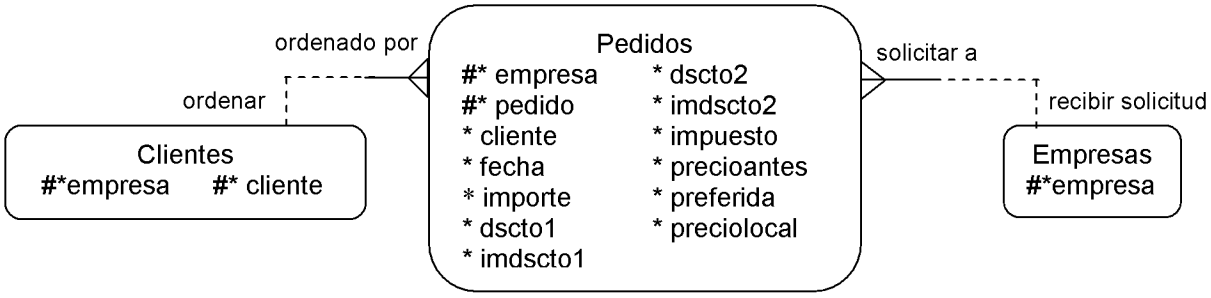


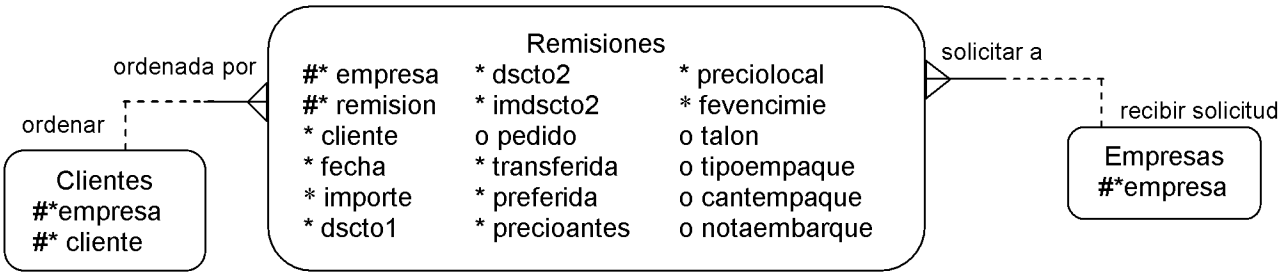
Figura 2.18 Modelo E-R de Pedidos

Su clave primaria está formada por los campos: empresa, pedido (campos primos).
La representación de la dependencia funcional entre sus campos es:

empresa, pedido → cliente, fecha, importe, impuesto, dscto1, imdscto1, dscto2, imdscto2, precioantes, preferida, preciolocal

❖ **Remisiones**

La información que se guardará de cada remisión se muestra en la figura 2.19, que es el modelo E-R de esta entidad.



Fgura 2.19 Modelo E-R de Remisiones

Su clave primaria está formada por los campos: empresa, remision (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, remision → cliente, fecha, fevencimie, importe, dscto1, imdscto1, dscto2, imdscto2, transferida, pedido, precioantes, preciolocal, preferida, talon, tipoempaque, cantempaque, notaembarque

❖ **Facturas**

La información que se guardará de cada factura se muestra en la figura 2.20, que es el modelo E-R de esta entidad.

Su clave primaria está formada por los campos: empresa, factura (campos primos).

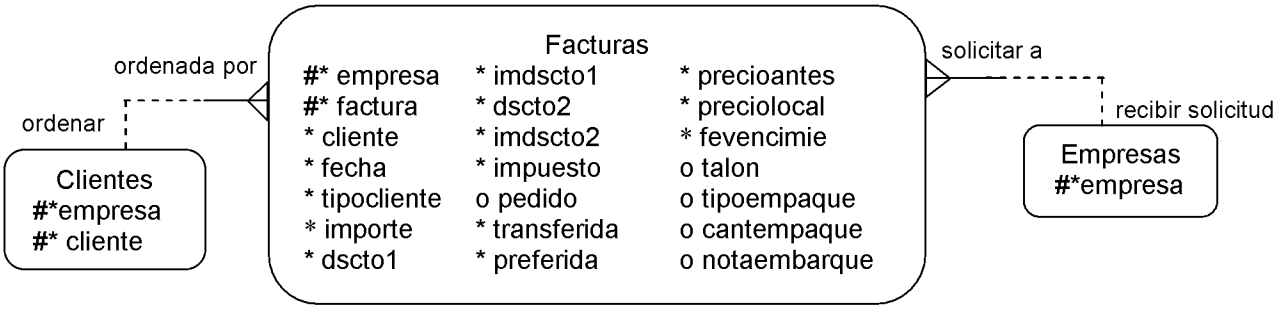


Figura 2.20 Modelo E-R de Facturas

La representación de la dependencia funcional entre sus campos es:

empresa factura → cliente, fecha, fevencimie, importe, clvimp, impimpuesto, dscto1, imdscto1, dscto2, imdscto2, transferida, precioantes, pedido, preferida, preciolocal, talon, tipoempaque, cantempaque, notaembarque, tipocliente

❖ **TipoCXC**

Las entidades TipoCxC son conceptos de cuentas por cobrar, y su diagrama de E-R se muestra en la figura 2.21

Su clave primaria está formada por el campo “clavecxc” (campo primo).

La representación de la dependencia funcional entre sus campos es: clvcxc→ descrip

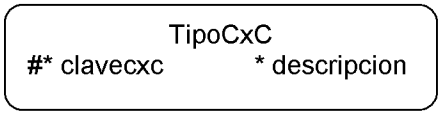


Figura 2.21 Diagrama de entidad de TipoCxC

❖ **CXC**

La información que se guardará de cada cuenta por cobrar se muestra en la figura 2.22, que es el modelo E-R de esta entidad.

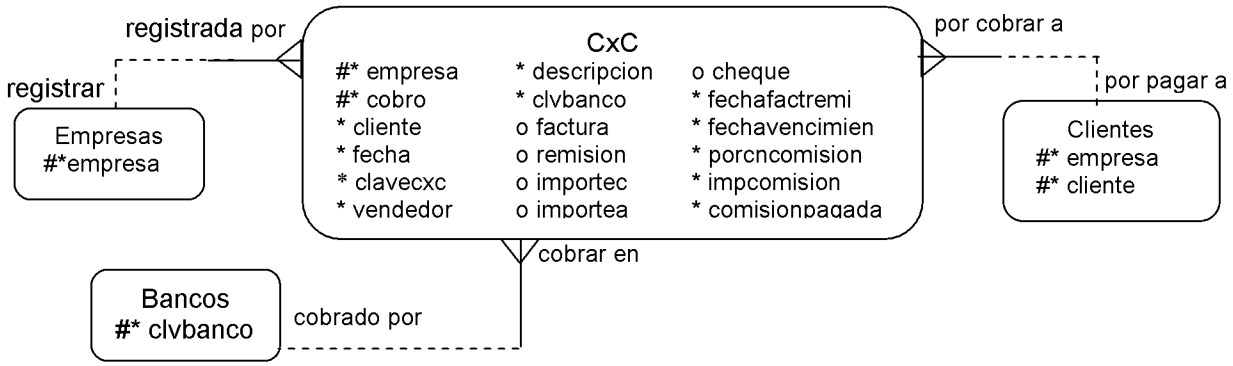


Figura 2.22 Diagrama de entidad de CxC

Su clave primaria está formada por los campos empresa, cobro (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, cobro → fecha, factura, remision, fechafactremi fechavencimien, fecharecepcion, descripción, importec, importea, clvbanco, cheque, impcomision, porcncomision, comisionpagada, clavecxc, cliente, vendedor

❖ **Usuarios**

Las entidades Usuarios son quienes tendrán acceso al uso del programa y a cada uno de sus módulos; su diagrama de E-R se muestra en la figura 2.23

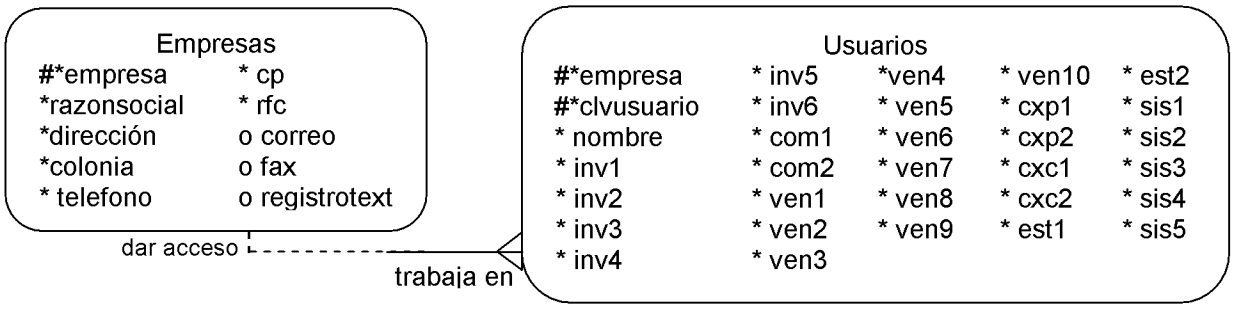


Figura 2.23 Modelo E-R de Usuarios

Su clave primaria está formada por los campos: empresa, clvusuario (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, clvusuario → nombre

➢ **Relaciones**

El modelo E-R de las relaciones, así como el modelo E-R general, se muestran en el apéndice B

2.4 Diseño

Posteriormente se construyen las tablas en las que será almacenada la información que se va a procesar de cada empresa. A continuación se exponen las tablas.

En cada tabla se muestra el tipo de dato y la longitud de cada uno de sus atributos.

➢ Entidades

❖ Empresa

Su representación en la base de datos se muestra en la tabla 2.1, y en ella se almacenarán los datos de las empresas cuya información se va procesar.

Su clave primaria está formada por el campo: empresa (obsérvese la figura 2.1).

Empresas

Campo	Tipo dato	Longitud	Campo	Tipo dato	Longitud
empresa	Texto	2	rfc	Texto	15
razonsocial	Texto	50	correo	Texto	40
direccion	Texto	35	telefono	Texto	18
colonia	Texto	35	fax	Texto	18
cp	Texto	5	registrotext	Texto	10

Tabla 2.1: Tabla en la que se almacenan los datos de cada empresa

❖ Impuesto

Su representación en la base de datos se muestra en la tabla 2.2, y en ella se almacenará el porcentaje del IVA que las empresas, y los clientes, deben cubrir en cada compra que efectúen.

Su clave primaria está formada por el campo: clave (obsérvese la figura 2.2).

Impuesto

Campo	Tipo dato	Longitud	Campo	Tipo dato	Longitud
clvimp	Texto	2	monto	Moneda	
nom	Texto	50			

Tabla 2.2: Tabla en la que se almacenará el porcentaje del IVA

❖ Numeracion

Su representación en la base de datos se muestra en la tabla 2.3, y en ella se almacenará la numeración de los documentos de cada empresa.

Su clave primaria está formada por el campo: empresa (obsérvese la figura 2.3). Además este mismo atributo es clave extranjera, pues forma parte de la tabla “Empresa”, con la cual se relaciona a través de este campo.

Numeros

Campo	Tipo dato	Longitud	Campo	Tipo dato	Longitud
empresa	Número	2	remision	Número	Entero largo
pedido	Número	Entero largo	factura	Número	Entero largo

Tabla 2.3: Tabla en la que se almacenan la numeración de los documentos

❖ **Proveedores**

Su representación en la base de datos se muestra en la tabla 2.4, y en ella se almacenarán los datos de los proveedores a quienes cada empresa compra las materias primas necesarias.

Proveedores

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	estado	Texto	3
proveedor	Texto	4	rfc	Texto	15
nombre	Texto	50	correo	Texto	40
direccion	Texto	35	telefono	Texto	18
colonia	Texto	35	fax	Texto	18
cp	Texto	5	curp	Texto	20
poblacion	Numérico	Entero	diascredito	Numérico	Entero

Tabla 2.4: Tabla en la que se almacenan los proveedores

Su clave primaria está formada por los campos: empresa, proveedor (obsérvese la figura 2.4). Además el atributo Empresa es clave extranjera, pues forma parte de la tabla "Empresa", con la cual se relaciona a través de este campo.

❖ **MateriaPrima**

Su representación en la base de datos se muestra en la tabla 2.5, y en ella se almacenarán los datos de las materias primas que cada empresa compra para fabricar los productos que posteriormente serán vendidos.

Su clave primaria está formada por los campos: empresa, materiaprima (obsérvese la figura 2.5). Además el atributo Empresa es clave extranjera, pues forma parte de la tabla "Empresa", con la cual se relaciona a través de este campo.

MateriaPrima

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	existencia	Texto	3
materiaprima	Texto	6	stockmin	Moneda	3
descripcion	Texto	25	stockmax	Moneda	3
unidad	Texto	3	coniva	Sí/No	

Tabla 2.5: Tabla en la que se almacenan las materias primas

❖ **Articulos**

Articulos

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	preciolocal	Moneda	2
articulo	Texto	7	preciomexico	Moneda	2
descripcion	Texto	50	precioantlocal	Moneda	2
unidad	Texto	3	precioantmexico	Moneda	2

Tabla 2.6: Tabla en la que se almacenan los artículos fabricados

Su representación en la base de datos se muestra en la tabla 2.6, y en ella se almacenarán los datos de los artículos fabricados.

Su clave primaria está formada por los campos: empresa, articulo (obsérvese la figura 2.6). Además el atributo Empresa es clave extranjera, pues forma parte de la tabla "Empresa", con la cual se relaciona a través de este campo.

❖ **Compras**

Su representación en la base de datos se muestra en la tabla 2.7, y en ella se almacenarán la información de cada orden de compra.

Compras

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	fecha	Fecha	
ordcompra	Numérico	Entero largo	importe	Moneda	2
proveedor	Texto	6	impuesto	Moneda	2

Tabla 2.7: Tabla en la que se almacenan las ordenes de compra

Su clave primaria está formada por los campos: empresa, ordcompra (obsérvese la figura 2.7). Además los atributos Empresa y Proveedor son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "proveedor" con la tabla Proveedores

❖ **Entradas**

Su representación en la base de datos se muestra en la tabla 2.8, y en ella se almacenarán la información de cada numero de entrada de materia prima de las empresas.

Entradas

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	fecha	Fecha	
entrada	Numérico	Entero largo	importe	Moneda	2
proveedor	Texto	6	impuesto	Moneda	2
ordcompra	Numérico	Entero largo	factura	Numérico	Entero largo

Tabla 2.8: Tabla en la que se almacenan los números de entrada de materia prima

Su clave primaria está formada por los campos: empresa, entrada (obsérvese la figura 2.8). Además los atributos Empresa y Proveedor son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "proveedor" con la tabla Proveedores

❖ **Salidas**

Su representación en la base de datos se muestra en la tabla 2.9, y en ella se almacenarán la información de cada numero de salida de materia prima.

Salidas

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	fecha	Fecha	
salida	Numérico	Entero largo	importe	Moneda	2
proveedor	Texto	6	factura	Numérico	Entero largo

Tabla 2.9: Tabla en la que se almacenan las salidas de materia prima

Su clave primaria está formada por los campos: empresa, salida (obsérvese la figura 2.9). Además los atributos Empresa y Proveedor son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "proveedor" con la tabla Proveedores

❖ **TipoCXP**

Su representación en la base de datos se muestra en la tabla 2.10, y en ella se almacenarán los conceptos de cuentas por pagar. Su clave primaria está formada por el campo clavecxp (obsérvese la figura 2.10).

TipoCXP

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
clavecxp	Texto	2	descripcion	Texto	25

Tabla 2.10: Tabla en la que se almacenan los conceptos de cuentas por pagar

❖ **Bancos**

Su representación en la base de datos se muestra en la tabla 2.11, y en ella se almacenarán las instituciones bancarias. Su clave primaria está formada por el campo clvbanco (obsérvese la figura 2.11).

Bancos

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
clvbanco	Numérico	Entero	nombre	Texto	25

Tabla 2.11: Tabla en la que se almacenan las instituciones bancarias

❖ **CxP**

Su representación en la base de datos se muestra en la tabla 2.12, y en ella se almacenarán la información de cada cuenta por pagar.

Su clave primaria está formada por los campos: empresa, pago (obsérvese la figura 2.12). Además las claves extranjeras son los atributos: empresa, proveedor, clavecxp, clvbanco, ordcompra, entrada; y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "proveedor" con la tabla Proveedores
- El campo "ordcompra" funciona como referencia
- A través del campo "banco" con la tabla Bancos
- A través del campo "clavecxp" con la tabla TipoCxP
- El campo "entrada" funciona como referencia

CXP

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	clavecxp	Texto	2
pago	Numérico	Entero largo	clvbanco	Texto	2
fecha	Fecha		cheque	Texto	12
factura	Numérico	Entero largo	ordcompra	Numérico	Entero largo
importec	Moneda	2	entrada	Numérico	Entero largo
importea	Moneda	2	fechavencimiento	Fecha	
descripcion	Texto	60	fecharecepcion	Fecha	
proveedor	Texto	6			

Tabla 2.12 Tabla en la que se almacenan los datos de cada cuenta por pagar

❖ **Estados**

Su representación en la base de datos se muestra en la tabla 2.13, y en ella se almacenarán los estados de la República Mexicana. Su clave primaria está formada por el campo clvedo (obsérvese la figura 2.13).

Estados

Campo	Tipo dato	Longitud	Campo	Tipo dato	Longitud
clvedo	Texto	3	estado	Texto	22

Tabla 2.13: Tabla en la que se almacenan estados de la República Mexicana

❖ **Municipios**

Su representación en la base de datos se muestra en la tabla 2.14, y en ella se almacenarán los municipios de cada Estado de la República Mexicana.

Municipios

Campo	Tipo dato	Longitud	Campo	Tipo dato	Longitud
clvedo	Texto	3	Municipio	Texto	40
clvmun	Numérico	Entero			

Tabla 2.14: Tabla en la que se almacenan los Municipios de la República Mexicana

Su clave primaria está formada por los campos: clvedo, clvmun (obsérvese la figura 2.14). Además el atributo Clvedo es clave extranjera, pues forma parte de la tabla "Estados", con la cual se relaciona a través de este campo.

❖ **Zonas**

Zonas

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
zona	Texto	6	clvedo	Texto	3
empresa	Texto	2	clvmun	Numérico	Entero

Tabla 2.15: Tabla en la que se almacenan las zonas

Su representación en la base de datos se muestra en la tabla 2.15, y en ella se almacenarán las zonas en que se distribuirán los vendedores, y en las cuales se localizan los clientes.

Su clave primaria está formada por los campos: empresa, zona (obsérvese la figura 2.15). Además las claves extranjeras son los atributos: empresa, clvedo, clvmun; y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla empresas.
- A través del campo "clvedo" con la tabla Estados
- A través del campo "clvmun" con la tabla Municipios

❖ **Vendedor**

Su representación en la base de datos se muestra en la tabla 2.16, y en ella se almacenarán los datos de los vendedores

Vendedor

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	cp	Texto	5
vendedor	Numérico	Entero largo	rfc	Texto	15
nombre	Texto	50	telefono	Texto	18
direccion	Texto	35	comision	Moneda	2
colonia	Texto	35			

Tabla 2.16: Tabla en la que se almacenan los datos de los vendedores

Su clave primaria está formada por los campos: empresa, vendedor (obsérvese la figura 2.16). Además el atributo Empresa es clave extranjera, pues forma parte de la tabla "Empresa", con la cual se relaciona a través de este campo.

❖ **Clientes**

Clientes

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	curp	Texto	20
cliente	Numérico	Entero largo	telefono	Texto	18
nombre	Texto	50	fax	Texto	18
direccion	Texto	35	correo	Texto	40
colonia	Texto	35	zona	Texto	6
cp	Texto	5	vendedor	Texto	4
rfc	Texto	15			

Tabla 2.17: Tabla en la que se almacenan los datos de los clientes

Su representación en la base de datos se muestra en la tabla 2.17, y en ella se almacenarán los datos de los clientes.

Su clave primaria está formada por los campos: empresa, cliente (obsérvese la figura 2.17).

Además las claves extranjeras son los atributos: empresa, vendedor, zona; y se relaciona de la siguiente forma:

- A través del campo “empresa” con la tabla Empresas.
- A través del campo “zona” con la tabla Zonas
- A través del campo “vendedor” con la tabla Vendedores

❖ **Pedidos**

Su representación en la base de datos se muestra en la tabla 2.18, y en ella se almacenarán la información de cada pedido.

Pedidos

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	dscto2	Moneda	2
pedido	Numérico	Entero largo	imdscto2	Moneda	2
cliente	Numérico	Entero largo	impuesto	Moneda	2
fecha	Fecha		precioantes	Sí/No	
importe	Moneda	2	preferida	Sí/No	
dscto1	Moneda	2	preciolocal	Sí/No	
imdscto1	Moneda	2			

Tabla 2.18: Tabla en la que se almacenan la información de los pedidos

Su clave primaria está formada por los campos: empresa, pedido (obsérvese la figura 2.18). Además los atributos Empresa y Cliente son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo “empresa” con la tabla Empresas
- A través del campo “cliente” con la tabla Clientes

❖ **Remisiones**

Remisiones

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	transferida	Sí/No	
remision	Numérico	Entero largo	preferida	Sí/No	
cliente	Numérico	Entero largo	precioantes	Sí/No	
fecha	Fecha		preciolocal	Sí/No	
importe	Moneda	2	fevencimie	Fecha	
dscto1	Moneda	2	talon	Numérico	Entero largo
imdscto1	Moneda	2	tipoempaque	Texto	7
dscto2	Moneda	2	cantempaqe	Numérico	Entero
imdscto2	Moneda	2	notembarqe	Memo	
pedido	Numérico	Entero largo			

Tabla 2.19: Tabla en la que se almacenan la información de las remisiones

Su representación en la base de datos se muestra en la tabla 2.19, y en ella se almacenarán la información de cada remisión.

Su clave primaria está formada por los campos: empresa, remisión (obsérvese la figura 2.19). Además los atributos Empresa, Cliente y Pedido son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo “empresa” con la tabla Empresas.
- A través del campo “cliente” con la tabla Clientes
- A través del campo “pedido” con la tabla Pedidos

❖ **Facturas**

Su representación en la base de datos se muestra en la tabla 2.20, y en ella se almacenarán la información de cada factura.

Facturas

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	pedido	Numérico	Entero largo
factura	Numérico	Entero largo	transferida	Sí/No	
cliente	Numérico	Entero largo	preferida	Sí/No	
fecha	Fecha		precioantes	Sí/No	
tipocliente	Texto	1	preciolocal	Sí/No	
importe	Moneda	2	fevencimie	Fecha	
dscto1	Moneda	2	talon	Numérico	Entero largo
imdscto1	Moneda	2	tipoempaque	Texto	7
dscto2	Moneda	2	cantempaque	Numérico	Entero
imdscto2	Moneda	2	notaembarque	Memo	
impuesto	Moneda	2			

Tabla 2.20: Tabla en la que se almacenan la información de las facturas

Su clave primaria está formada por los campos: empresa, factura (obsérvese la figura 2.20).

Además los atributos Empresa, Cliente y Pedido son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo “empresa” con la tabla Empresas.
- A través del campo “cliente” con la tabla Clientes
- A través del campo “pedido” con la tabla “Pedidos”

❖ **TipoCXC**

Su representación en la base de datos se muestra en la tabla 2.21, y en ella se almacenarán los conceptos de cuentas por cobrar. Su clave primaria está formada por el campo clavecxc (obsérvese la figura 2.21).

TipoCXC

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
clavecxc	Texto	2	descripción	Texto	25

Tabla 2.21: Tabla en la que se almacenan los conceptos de cuentas por cobrar

❖ **CXC**

Su representación en la base de datos se muestra en la tabla 2.22, y en ella se almacenarán la información de cada cuenta por cobrar.

Su clave primaria está formada por los campos: empresa, cobro (obsérvese la figura 2.22).

CXC

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	remision	Numérico	Entero largo
cobro	Numérico	Entero largo	importeC	Moneda	2
cliente	Numérico	Entero largo	importeA	Moneda	2
fecha	Fecha		cheque	Texto	12
clavecxc	Texto	2	fechafactremi	Fecha	
vendedor	Texto	4	fechavencimien	Fecha	
descripcion	Texto	60	porcncomision	Moneda	2
clvbanco	Texto	2	impcomision	Moneda	2
factura	Numérico	Entero largo	comisionpagada	Sí/No	

Tabla 2.22: Tabla en la que se almacenan las cuentas pendientes de cobrar

Además las claves extranjeras son los atributos: empresa, clvbanco, clavecxc, cliente, vendedor; y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "banco" con la tabla Bancos
- A través del campo "clavecxc" con la tabla Tipocxc
- A través del campo "vendedor" con la tabla Vendedores
- A través del campo "cliente" con la tabla Clientes
- El campo "factura" funciona como referencia
- El campo "remision" funciona como referencia

❖ **Usuarios**

Su representación en la base de datos se muestra en la tabla 2.23, y en ella se almacenarán los usuarios que tendrán acceso al uso del programa.

Usuarios

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	nombre	Texto	40
clvusuario	Texto	4			

Usuarios (continuación)

Campo	Tipo dato	Campo	Tipo dato	Campo	Tipo dato
inv1	Sí/No	ven3	Sí/No	cxc1	Sí/No
inv2	Sí/No	ven4	Sí/No	cxc2	Sí/No
inv3	Sí/No	ven5	Sí/No	cxc3	Sí/No
inv4	Sí/No	ven6	Sí/No	est1	Sí/No
inv5	Sí/No	ven7	Sí/No	est2	Sí/No
inv6	Sí/No	ven8	Sí/No	sis1	Sí/No
com1	Sí/No	ven9	Sí/No	sis2	Sí/No
com2	Sí/No	ven10	Sí/No	sis3	Sí/No
ven1	Sí/No	cxp1	Sí/No	sis4	Sí/No
ven2	Sí/No	cxp2	Sí/No	sis5	Sí/No

Tabla 2.23: Tabla en la que se almacenan los usuarios del programa

Su clave primaria está formada por los campos: empresa, clvusuario (obsérvese la figura 2.23). Además el atributo empresa es clave extranjera, pues forma parte de la tabla Empresas, con la cual se relaciona a través de este campo.

- **Relaciones**

  Las tablas de las relaciones se muestran en el anexo B

Capítulo 3
Implementación y pruebas

Después de haber efectuado el análisis y el diseño, se utilizó Access de Microsoft para el desarrollo de la base de datos, en la cual se almacenará la información.

Posteriormente se desarrollo el programa con el que se gestionará la base de datos, y con el que se cubrirán las necesidades del cliente.

Para diferenciar una empresa de las demás, a cada una de ellas se le asignará una clave única, la cual se utilizará para .identificar las actividades que cada empresa efectuará, tales como: ventas, compras, pedidos, etc. De las actividades efectuadas se obtendrán informes de cada empresa, según lo desee el usuario.

El programa se dividió en módulos, y se diseñaron las interfases de usuario que dará acceso a cada módulo del programa, los cuales se listan a continuación:

- **Inventario:** Materia prima, Artículos, Cambio de precios, Entradas de materia prima, Salidas de materia prima.
- **Compras:** Proveedores, Compras.
- **Ventas:** Clientes, Zonas, Vendedores, Pedidos, Remisiones y Facturas.
- **Cuentas por pagar:** Abonos (pagos efectuados) Cargos (pagos pendientes).
- **Cuentas por cobrar:** Abonos (cobros efectuados) Cargos (cobros pendientes).
- **Estadísticas:** Importe de compras y de ventas que haya efectuado cada empresa, así como las ventas de artículos, durante un determinado período.
- **Sistema:** Empresas, Usuarios, Impuesto, Numeración.

Para cada módulo también se diseñó la interfase para los procesos de: agregar, modificar, consultar y borrar registro.

Para el desarrollo del programa se utilizó el lenguaje Visual Basic Ver. 6.0, pues se cuenta con la licencia de ese lenguaje de programación.

Conforme se terminó cada una de las tareas en cada módulo se realizaron pruebas de funcionamiento, con datos reales, hasta que el estuviera libre de errores. En las pruebas se incluyó desde la captura de información hasta la salida de informes en pantalla y en impresora.

3.1 Conexión a la base de datos.-

Para efectuar la conexión a la base de datos a través del programa, se llevaron a cabo los siguientes pasos:

❖ Se declararon constantes y variables globales en el archivo “modUtilVar.BAS”, las cuales se muestran a continuación:

- Esta constante se utiliza como apoyo para la conexión a tablas de tipo jerárquico (maestro/detalle):
  Global Const cadDSN_MSDataShape = "PROVIDER=MSDataShape;Data"
- Esta constante se utiliza para la conexión a tablas simples:
  Global Const cadDSN_MSDASQL = "PROVIDER=MSDASQL;dsn=Bdsad;” & _
  “uid=;pwd=das;database=bdsad;"
- La siguiente variable se utiliza para establecer la conexión a la base de datos: Public conDataBase As New ADODB.Connection

❖ Una vez declarada la variable se establece la conexión a la base de datos con las siguientes instrucciones:

```
conDataBase.CursorLocation = adUseClient
conDataBase.Open cadDSN_MSDASQL
```

Notas:

- El nombre de cada variable puede ser a gusto del programador, es decir, no necesariamente se deben nombrar como se muestran en el ejemplo, pero se debe procurar proporcionarles un nombre que identifique con qué fin se van a utilizar.
- Se debe tener en cuenta que estas variables se deben colocar en el código de cada formulario en donde se requiera abrir alguna tabla de la base de datos.
- Si el programa funcionará en ambiente monousuario no se debe utilizar la cadena de conexión:

  Provider=Microsoft.Jet.OLEDB.4.0;Data Source=...

  sino que es preferible sustituirla por: la cadena:

  DRIVER=Microsoft Access Driver (*.mdb);DBQ=...

  pues ésta es más general y no tiene problemas de la versión de Access que se tenga disponible, es decir, este formato funciona perfectamente desde Access de Office 97, hasta Access de Office XP.

3.2 Acceso a las tablas de una base de datos.-

Después de haber establecido la comunicación con la base de datos es posible tener acceso a las tablas que la forman, como se explica a continuación.

❖ Para tener acceso a una tabla de la base de datos primero se declara la variable que la contendrá, de la siguiente forma: Dim rsTabla As New ADODB.Recordset

❖ Después se abre la tabla con la siguiente instrucción:

rsTabla.Open “Empresas”, conDataBase, OpenDynamic, adLockOptimistic, adCmdTable

En donde:

- **Empresas:** Es el nombre de la tabla a la cual se tendrá acceso.
- **conDataBase:** es la variable que indica la conexión a la base de datos.
- **OpenDynamic:** indica que **Empresas** se debe abrir con propiedad dinámica, esto es, que cualquier modificación que otro usuario haga en otro registro, podemos verlo.
- **AdLockOptimistic:** Se utiliza para bloquear un registro en el momento de guardarlo, esto se da al utilizar la instrucción **Empresas.Update**

Esta forma es recomendable para abrir únicamente una tabla, tal como: Clientes, MateriaPrima, Vendedores, Facturas, etc.

Pero si únicamente se desea consultar o hacer modificaciones a un solo registro, o bien, si nada más se quiere consultar, por ejemplo, la clave y la RazonSocial de la tabla de Empresas, es inconveniente utilizar esta instrucción pues se incluyen todos los campos que forman la tabla, así como los registros que esta contiene,

Si se reflexiona sobre esto se puede observar que se utiliza innecesariamente una cantidad de memoria, y la carga de la tabla es mas lenta.

Para seleccionar un solo registro se podría pensar en utilizar la instrucción:

```
rsTabla,Filter = "clvEmpresa = ‘DP’"
```

pero es doble trabajo, pues después de abrir la tabla se le ordena que solo presente el registro deseado, y el resto de los registros queden ocultos, es decir, aun están activos, por así decirlo, en espera de seleccionar alguno de ellos.

Para esto basta emplear la instrucción rsTabla,Filter = "", con el fin de quitar el filtrado y seleccionar otro registro.

O bien, si se desea ver los campos clvEmpresa y RazonSocial de la tabla Empresas e incluir todos los registros de la misma, se puede emplear el objeto DataGrid, en el que se indica los campos a presentar, pero esto también únicamente oculta el resto de los campos.

Lo más indicado es utilizar la instrucción "Select ... From ... " de SQL, y en la que es posible seleccionar los campos deseados, esto se logra de la siguiente forma:

- Primero se declara la variable que contendrá el conjunto de campos de la tabla que se desea abrir, esto es: Dim cadSQL As String
- Posteriormente se inicializa la variable cadSQL de la siguiente forma:
  - Para seleccionar campos:
    cadSQL = "SELECT Empresa, RazonSocial From Empresas Order By Empresa"
  - Para seleccionar un registro:
    cadSQL = "SELECT Empresa, RazonSocial From Empresas Where Empresa = 'DP'"
- Después se abre la tabla con la siguiente instrucción:
  rsTabla.Open cadSQL, conDataBase, adOpenDynamic, adLockOptimistic
- Y por último se utiliza un objeto para observar la información contenida en la tabla, esto se logra con la siguiente instrucción:
  Set DataGrid1.DataSource = rsTabla

A continuación se expone cada uno de los módulos de que consta el programa.

3.3 Inicio del programa

Cuando se inicia la ejecución del programa el usuario debe pulsar la tecla F12, para que se presente el catálogo de empresas, mover la barra de la tabla hasta iluminar la empresa deseada y accionar la tecla S, con el fin de seleccionar la empresa con la que va a trabajar.

Al hacer esto la tabla desaparecerá y se activará el menú de opciones del programa (véase figura 3.1), cada una da acceso a un módulo diferente, según lo elija el usuario

Inventarios	Compras	Ventas	CxP	CxC	Estadísticas	Sistema

Figura 3.1 Menú de opciones del Sistema de Control Administrativo

Nota:

La opción Empresas también activa la tabla que se presenta al pulsar la tecla F12.

Despúes de estalecer la conexión a la base de datos y de abrir la tabla, como se explicó en el apartado 3.2, se puede emplear un control DataGrid para lograr la presentación de la información seleccionada, con la instrucción: Set DataGrid1.DataSource = rsTabla

En este caso se utilizó la cadena:

```
cadSQL = "Select Empresa, RazonSocial From Empresas " & _
         "Where Empresa <> 'AL'  Order By Empresa"
```

Aquí 'AL' se utiliza como bandera para presentar todas las empresas, en algunas partes del programa.

Posteriormente se enmascaran, por así decirlo, el nombre de los campos con otro nombre deseado, lo cual se hace con el siguiente código:

```
DataGrid1.Caption = "Catálogo de empresas"      ' Titulo de la tabla
DataGrid1.Columns(0).Caption = "Clave"          ' Titulo de la columna Empresa
DataGrid1.Columns(0).Width = 1050               ' Ancho de la columna Empresa
```

O bien, se puede utilizar el siguiente bloque de instrucciones:

```
With DataGrid1
   .Caption = "Catálogo de empresas"
   .Columns(0).Caption = "Clave"
   .Columns(0).Width = 1050
End With
```

Notas:

- Para la conexión a la base de datos y a las tablas de la misma se utilizó el objeto ADODB.
- También se puede emplear el control ADODC para el mismo fin, esto es:

```
AdodcCatalogo.ConnectionString = conDataBase
AdodcCatalogo.RecordSource = cadSQL
```

Al seleccionar la opción Inventarios se abre el módulo que tiene las opciones:

Materia prima	Entradas de materia prima
Artículos	Salidas de materia prima
Cambio de precios	

Y en las cuales se pueden efectuar los movimientos necesarios.

Nota:

En el código de un formulario (llamado Padre) se puede abrir otro formulario (llamado Hijo), de la siguiente forma:

- En el formulario Padre utilizar la instrucción frmNombre.Show vbModal, la cual presenta al formulario Hijo.
- Para cerrar el formulario Hijo utilizar la instrucción Unload Me, en su código.
- En el formulario padre utilizar la instrucción Set frmNombre = Nothing, debajo de la instrucción que invoca al formulario Hijo.
- La secuencia queda de la siguiente forma:

```
        Formulario Padre            Formulario Hijo
               :                          :
frmNombre.Show vbModal                    :
Set frmNombre = Nothing           Unload Me
               :                          :
                                          :
```

3.4 Descripción de un módulo del programa

Los menús de cada módulo de que consta el programa tiene el conjunto de opciones que se muestra en la figura 3.2

Registro	Ordenar	Buscar	Varios

Figura 3.2 Opciones del menú del módulo inventarios

En donde:

- Registro: En esta opción se pueden efectuar las tareas de : agregar, modificar y borrar
- Ordenar: Se puede ordenar por clave o por descripción.
- Buscar: Se puede buscar alguna clave o descripción.
- Varios: Se puede imprimir la tabla que esté activa, o bien cerrar esta ventana.

En cada una de estas opciones se presenta un menú descolgante para poder efectuar las tareas descritas.

También se utiliza una barra de herramientas para tales fines (véase figura 3.3). Por ejemplo, tanto el menú como la barra de herramientas, se utilizan para efectuar movimientos en el catálogo de inventario, cuyos títulos se muestran en la figura 3.4.

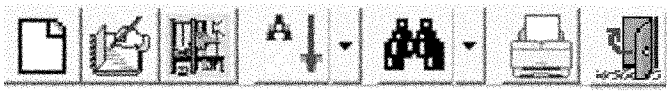


Figura 3.3 Barra de herramientas

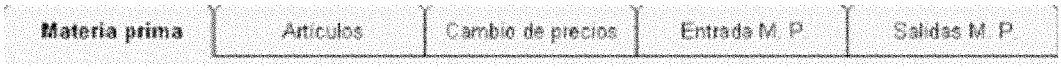


Figura 3.4 Pestañas que dan acceso al catálogo deseado del módulo Inventarios

3.4.1 Descripción de la barra de herramientas

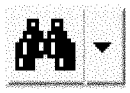
A continuación se describe la función de cada icono de la barra de herramientas

Agregar: Con este icono se abre otra ventana en la que se captura la información necesaria para agregar un nuevo registro. Los campos de captura que dicha ventana presenta, están en blanco. El mismo efecto se logra al pulsar la tecla de la letra N.

Modificar: Con este icono se abre la misma ventana en la que se capturó la información para agregar un nuevo registro. Los campos de captura que dicha ventana presenta, tienen la información antes capturada, para efectuar las modificaciones necesarias. El mismo efecto se logra al pulsar la tecla de la letra M.

Borrar: Con este icono se abre otra ventana en la que se pide la confirmación o cancelación, del borrado del registro seleccionado. El mismo efecto se logra al pulsar la tecla con la palabra **Supr**. En el caso de remisiones y facturas estas son canceladas, en vez de borradas.

Ordenar: Con este icono se abre un menú descolgante para seleccionar en base a qué columna se va a ordenar la información.



Buscar: Con este icono se abre un menú descolgante para seleccionar en base a qué columna se va a efectuar la búsqueda de algún registro.

Imprimir: Con este icono se abre otra ventana en la que se presenta la información que se enviará a la impresora. Dicha ventana se conoce como vista

previa.

Transferir: Este icono solo se presenta en la barra de herramientas de remisiones y de facturas, y con él se abre otra ventana en la que se solicita el rango de remisiones o de facturas que se van a transferir a cuentas por cobrar.

Salir: Con este icono se cierra la ventana en la que se está trabajando.

El menú de opciones y la barra de herramientas implementadas en cada módulo del programa es igual al utilizado en este módulo, con las mismas funciones.

Nota:

- Al colocar el apuntador del mouse en alguno de estos iconos, aparecerá un pequeño mensaje que indica su función.

3.4.2 Descripción del control SSTab

El control que proporciona esta forma de ver un menú de opciones, es el control SSTab1, al cual se le pueden agregar tantas opciones sean necesarias y cada una de estas pueden albergar otros controles. Para tener acceso a cada una de estas basta dar un clic en la opción deseada.

A cada una de estas opciones comúnmente se les conoce como Pestañas o Fichas. Esta forma de presentar un menú de opciones es recomendable cuando se efectuarán los mismos movimientos (agregar, modificar, consultar y borrar) a un conjunto de catálogos.

Como se puede observar, las pestañas que se presentan son:

- Materia prima: Da acceso al catálogo de la materia prima que adquiere cada empresa.
- Artículos: Da acceso al catálogo de artículos que cada empresa fabrica.
- Cambio de precios: Da acceso a efectuar cambio de precio a determinados artículos.
- Entradas: Da acceso al catálogo de entradas de materia prima de cada empresa.
- Salidas: Da acceso al catálogo de salidas materia prima de cada empresa

Este formato de presentación también se utilizó en los módulos de: Compras, Ventas y Cuentas por cobrar, como se muestra a continuación.

❖ Para el módulo Compras: (figura 3.5)

- Proveedores: Da acceso al catálogo de proveedores.
- Compras: Da acceso al catálogo ordenes de compra que cada empresa efectúa

Figura 3.5 Pestañas que dan acceso al catálogo deseado del módulo Compras

❖ Para el módulo Ventas: (figura 3.6)

- Clientes: Da acceso al catálogo de clientes..
- Zonas: Da acceso al catálogo de zonas.
- Vendedores: Da acceso al catálogo de vendedores.
- Pedidos: Da acceso catálogo de pedidos.
- Remisiones: Da acceso catálogo de remisiones

- Facturas: Da acceso catálogo de facturas

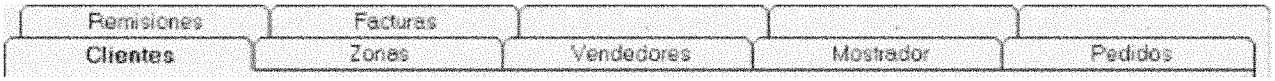


Figura 3.6 Pestañas que dan acceso al catálogo deseado del módulo Ventas

❖ Pare el módulo Cuentas por cobrar: (figura 3.7)
- Abono: Da acceso a los abonos de los clientes.
- Cargo: Da acceso a los cargos hacia los clientes (en caso de devolución de cheques).

Figura 3.7 Pestañas que dan acceso al catálogo deseado del módulo Cuentas por cobrar

Al seleccionar la pestaña deseada el catálogo se presenta en forma de tabla, por ejemplo, si al abrir el programa se ha seleccionado el módulo Ventas, y después se selecciona la pestaña Facturas se observa una parte de la tabla de las facturas

El nombre del control en el que se muestran las facturas y la forma de utilizarlo ya se ha mencionado en esta misma sección, y se emplea uno para cada catálogo, en cada pestaña.

El programa se ha implementado de tal manera que en cada modulo se utiliza una sola variable declarada de tipo ADODB.Recordset para abrir el catálogo deseado y en el que se realizarán los movimientos: agregar, modificar, borrar y buscar.

Para esto se empleó la variable numTabla de tipo entero, para determinar el catálogo deseado. A dicha variable se le asigna el valor cuando se selecciona una pestaña del control SSTab1, esto es, supóngase que se está utilizando el módulo Ventas:
- Al seleccionar la pestaña Clientes se le asigna numTabla = 0.
- Al seleccionar la pestaña Zonas se le asigna numTabla = 1.
- Al seleccionar la pestaña Vendedores se le asigna numTabla = 2.
- Al seleccionar la pestaña Pedidos se le asigna numTabla = 3.
- Al seleccionar la pestaña Remisiones se le asigna numTabla = 4.
- Al seleccionar la pestaña Facturas se le asigna numTabla = 5.

Posteriormente al control SSTab1 se le ordena presentar el contenido de la pestaña elegida, con la instrucción: SSTab1.Tab = numTabla.

Ahora bien, se utilizó una variable exclusiva para abrir el catálogo que se seleccione, y a la que se le proporcionó el nombre de cada módulo, es decir:
- rsInventario para el módulo Inventarios.
- rsCompras para el módulo Compras
- rsVentas para el módulo Ventas
- rsCUxCO para el módulo Cuentas por cobrar
- rsCxP para el módulo Cuentas por pagar

3.5 Descripción de facturas del módulo ventas

A continuación se explica la forma de cómo se combinaron estas variables, para abrir un catálogo, para lo cual continuará tomándose como ejemplo el módulo de Ventas, y cuyo nombre de formulario es frmVentas

Al inicio del código del formulario debe estar la instrucción que hace que todas las variables utilizadas en el módulo sean declaradas, la cual es:

```
Option Explicit
```

Se declaran las variables necesarias, que son las variables del módulo, esto es:

```
Public rsVentas As New ADODB.Recordset     ‘ Variable que contendrá la tabla que se abrirá
Dim cadSQL As String      ‘ Variable de tipo cadena, cuya función se explica más adelante
Dim objColumn As Column      ‘ Variable de tipo Column para utilizar con el control DataGrid
Dim i As Integer                ‘ Variable de tipo entero utilizada como contador
```

La variable rsVentas es declarada de esta forma para que pueda ser reconocida por otros formularios en los que se requiera (formulario Hijo). Como únicamente se desea presentar unas cuantas columnas de la tabla de facturas se declara la variable cadSQL que contendrá los campos deseados y que cumplan con cierta condición. Esta variable debe contener cláusulas del lenguaje SQL, como se muestra a continuación:

```
cadSQL = "SELECT Facturas.FACTURA, Facturas.FECHA, Facturas.CLIENTE, " & _
        "Clientes.NOMBRE, Facturas.Total, Facturas.CONDESCTO, " & _
        "iif(Facturas.STATUS = 'C', Facturas.STATUS,'') As S, " & _
        "Facturas.Transferida, Facturas.EMPRESA " & _
        "FROM Facturas, Clientes " & _
        "WHERE Facturas.CLIENTE = Clientes.CLIENTE AND " & _
        "Clientes.EMPRESA = '" & cadClvEmpresa & _
        "' AND Clientes.EMPRESA = Facturas.EMPRESA Order By FACTURA Desc"
```

Se puede observar el par de caracteres “& _” al final de cada línea, esto indica la continuación de la cadena.

Posteriormente se debe preguntar si la variable rsVentas tiene alguna tabla en uso, de ser así esta se debe cerrar; esto se logra con la siguiente línea de instrucción:

```
If rsVentas.State = adStateOpen Then rsVentas.Close
```

Se abre la tabla definida en la variable cadSQL, de la siguiente manera:

```
rsVentas.Open cadSQL, conDataBase, adOpenDynamic, adLockOptimistic
```

Notas:

- Se podría pensar cerrar la tabla en uso antes de abrir otra tabla, lo cual es correcto, pero es mejor esta forma, en caso de que hayamos olvidado que la variable rsVentas previamente tiene una tabla en uso.
- Los apóstrofos que inician cada línea indican un comentario, dentro de Visual Basic.
- El mismo procedimiento se siguió para todos los módulos de que consta el programa, incluso en las ventanas de captura de información.

Se conecta el control datgrdFactura a la nueva tabla abierta, para ver la información contenida en esta, con la siguiente línea de instrucción:

```
Set datgrdFactura.DataSource = rsVentas
```

Por último se enmascara el título de cada columna del control datgrdFactura, y se da formato de visualización a la información

```
With datgrdFactura
    For Each objColumn In .Columns     ‘ Para cada columna de datgrdFactura hacer ...
```

```
        objColumn.Visible = False          ' Se ocultan las columnas
    Next

    .Columns(0).Visible = True             ' Se hace visible la columna
     .Columns(0).Alignment = dbgCenter     ' Se alinea al centro la información
     .Columns(0).Caption = "Factura" '     ' Titulo de la columna
     .Columns(0).Width = 1000              ' Ancho de la columna

     .Columns(1).Visible = True
     .Columns(1).Alignment = dbgCenter
     .Columns(1).Caption = "Fecha"
     .Columns(1).NumberFormat = "dd/mm/yyyy"   ' Presentación para tipo fecha
     .Columns(1).Width = 1200

     .Columns(3).Visible = True
     .Columns(3).Caption = "Cliente"
     .Columns(3).Width = 6800

     .Columns(4).Visible = True
     .Columns(4).Alignment = dbgRight      ' Se alinea a la derecha la información
     .Columns(4).Caption = "Importe"
     .Columns(4).NumberFormat = "$ ###,##0.00" ' Presentación para tipo moneda
     .Columns(4).Width = 1500

     .Columns(5).Visible = True
     .Columns(5).Alignment = dbgCenter
     .Columns(5).Caption = " "                    ' Columna sin titulo
     .Columns(5).Width = 200
     .Columns(6).Visible = True
     .Columns(6).Alignment = dbgCenter
     .Columns(6).Caption = "S"
     .Columns(6).Width = 300
```

Se establece el estilo de la barra de iluminación del apuntador en el control datgrdFactura

```
        .MarqueeStyle = 3
End With
```

O bien se puede utilizar el siguiente ciclo para ocultar columnas

```
With datgrdPedidos
  For i = 0 To .Columns.Count – 1        '.Count – 1  es el número de columnas del control
    .Columns(i).Visible = False
  Next I
End With
```

Con el anterior bloque de código es suficiente para ver un conjunto de datos en un control DataGrid.

Ahora bien, como se utilizó el mismo objeto rsVentas para: Clientes, Zonas, Vendedores, Pedidos, Remisiones y Facturas, del módulo ventas, es necesario utilizar el bloque de código anteriormente descrito para abrir el catálogo deseado, de acuerdo a la pestaña seleccionada en el control SSTab1, esto es, antes de abrir un catálogo se deben emplear las siguientes líneas de instrucción, con el fin de no causar error al pasar de una pestaña a otra (supóngase que se cambia a la pestaña Remisiones).

```
Set datgrdFactura.DataSource = Nothing
datgrdFactura.Clear
cadSQL = "SELECT Remisiones.REMISION, Remisiones.FECHA, " & _
  "Remisiones.CLIENTE, Clientes.NOMBRE, Remisiones.Total, " & _
  "Remisiones.CONDESCTO, " & _
  "iif(Remisiones.STATUS = 'C', Remisiones.STATUS,") As S, " & _
  " Remisiones.Transferida, Remisiones.EMPRESA " & _
  "FROM Remisiones, Clientes " & _
  "WHERE Remisiones.EMPRESA = '" & cadClvEmpresa & "' AND " & _
  "Remisiones.CLIENTE = Clientes.CLIENTE AND " & _
  "Clientes.EMPRESA = Remisiones.EMPRESA Order By Remisiones Desc"
If rsVentas.State = adStateOpen Then rsVentas.Close
rsVentas.Open cadSQL, conDataBase, adOpenDynamic, adLockOptimistic
Set datgrdRemisiones.DataSource = rsVentas
```

La primera línea de este bloque de código es necesaria para no saturar el uso de la memoria.

Ahora bien, analizando la cadena cadSQL se puede observar que:

- El operador ternario de Visual Basic: iif(condicion, valor1, valor2), está dentro de una cláusula de SQL, esto demuestra la forma de cómo combinar instrucciones de ambos lenguajes entre si, y de cómo hacer referencia a un atributo de una tabla dentro de este operador y representarlo con otro nombre de columna. Esto es, dicho operador devuelve el contenido del atributo Remisiones.STATUS si este es igual al carácter 'C', de lo contrario devuelve el carácter de espacio; al resultado lo representa con el nombre de columna 'S'.

  No todas las instrucciones de Visual Basic se pueden insertar en una cláusula de SQL, sino únicamente las que ocupan una sola línea de instrucción, algunas de las cuales se muestran en la tabla 3.1

Instrucción	Descripción
iif(condicion, valor1, valor2)	Si se cumple la condicion, devuelve valor1, de no ser así, devuelve valor2
Round(num, n)	Devuelve **num** redondeado en **n** lugares decimales
Mid(cad, m, n),	Devuelve una subcadena de cad, ubicada del carácter **m**, hasta el carácter **n**
Len(cad)	Devuelve la longitud de caracteres de la cadena cad
Trim(cad)	Devuelve la cadena cad sin espacios en blanco a la izquierda y a la derecha de cad

Tabla 3.1: Instrucciones de Visual Basic que se pueden combinar con SQL

Instrucción	Descripción (continuaciòn)
InStr(1, cad1, cad2)	Devuelve la posición en la que se encuentra la cad2, dentro de cad1
Int(num)	Devuelve la parte entera de num
Left(cad,n)	Devuelve los primeros **n** caracteres que están dentro de **cad**
Right(cad,n)	Devuelve los últimos **n** caracteres que están dentro de **cad**

Tabla 3.1: Instrucciones de Visual Basic que se pueden combinar con SQL

Nota:

- Para más información referirse a la ayuda en línea de Visual Basic

❖ Continuando con el análisis de la cadena cadSQL, también se puede observar que en la cláusula WHERE se incluye una cadena declarada como pública (cadClvEmpresa), y que se utiliza como parte de la condición para la selección de registro, como se muestra en el siguiente fragmento de código:

```
"WHERE Remisiones.EMPRESA = '" & cadClvEmpresa & "' AND " & _
 "Remisiones.CLIENTE = Clientes.CLIENTE AND " & _
 "Clientes.EMPRESA = Remisiones.EMPRESA ..."
```

3.5.1 Otras funciones que se realizan en facturas

Además de que se puede agregar, modificar y consultar información (véase sección 3.4.1), también es posible realizar otras funciones, las cuales se listan a continuación.

❖ **Buscar:** Para efectuar la búsqueda de algún registro en particular primero se debe ordenar la información en base a la columna que se va a buscar, ya sea ordenar por número de facturas, o bien por nombre de cliente (véase figura 3.4 y la sección 3.4.1).

Posteriormente se debe dar un clic en el icono que representa unos binoculares (véase figura 3.4 y la sección 3.4.1), o bien en la barra de menú seleccionar la opción Buscar, y después la opción deseada.

Después aparecerá un cuadro de diálogo que solicita la información a buscar, como se observa en la figura 3.8.

Figura 3.8: Cuadro de diálogo para capturar la información a buscar

En el cuadro de diálogo se observa un campo de captura, en la cual se debe introducir la información que se va a buscar.

Después de la captura, es suficiente con pulsar la tecla Enter (Intro) para que el cuadro de diálogo se cierre y en el catálogo se haga la búsqueda, o también dando un clic en al botón Aceptar..

Si se pulsa la tecla Esc o si se da un clic en el botón Cancelar, también se cierra el cuadro de diálogo sin efectuar la búsqueda.

Para que aparezca este cuadro de diálogo se utiliza la instrucción:

InputBox(cad1, cad2)

en donde cad1 es el mensaje y cad2 es el titulo, y se puede utilizar de la siguiente manera:

```
cadBuscar = vbCrLf & vbCrLf & vbCrLf & vbCrLf & vbCrLf & "Buscar lo siguiente"
cadResp = InputBox(cadBuscar, 'Búsqueda')
```

La información a buscar se almacena en la variable cadResp. Pero si el campo de captura del cuadro de diálogo está en blanco, o bien si se pulsó la tecla Esc, o si se dio un clic en el botón Cancelar, en la variable cadResp se almacena una cadena de longitud cero, por tanto no se efectúa la búsqueda.

Nota:

- La instrucción vbCrLf causa un salto de renglón.

Para buscar la información requerida se utilizó el siguiente bloque de código:

```
Dim mvBookMark As Variant        ‘ Se declara una variable

On Error GoTo Salir ‘ Si hay algún error el flujo del programa salta a la etiqueta Salir

‘ Se asigna el lugar en el que está el apuntador de registro de la tabla
mvBookMark = rsVentas.Bookmark
rsVentas.MoveFirst           ‘ Se mueve el apuntador de registro hacia el inicio de la tabla
rsVentas.Find "FACTURA = '" & cadResp & "'"
If rsVentas.EOF Then         ‘ Se mueve el apuntador de registro hacia el lugar en el que
        rsVentas.Bookmark = mvBookMark        ‘estaba antes de efectuar la búsqueda

‘ Aparece un cuadro de diálogo con un mensaje que indica que no está la información
        MsgBox "No está: " & vbCrLf & cadResp, vbCritical & vbOKOnly, " A v i s o "
End If
Salir:
  MsgBox Err.Number & " - " & Err.Description, vbInformation & vbOKOnly, "A V I S O"
```

Si la información está, el apuntador de registro se coloca en el lugar en donde se encuentra. En caso de que resulte algún error inesperado, el flujo del programa salta hasta la instrucción que está después de la etiqueta Salir, y se muestra una caja de diálogo indicando el error que ha ocurrido.

Notas:

- Para más información acerca del uso de los cuadros de diálogo InputBox(cad1, cad2) y MsgBox(Mensaje**,** Estilo**,** Título**,** Ayuda**,** Ctxt) referirse a la ayuda en línea del lenguaje Visual Basic.
- Para conocer el aspecto de este cuadro véa la figura 3.8

❖ **Cancelar:** Para cancelar algún registro, primero se debe colocar el apuntador de registro (barra de la tabla, véase figura 3.9) en la factura que se va a cancelar.

Posteriormente se debe dar un clic en el icono que representa un verdugo (véase figura 3.4 y la sección 3.4.1 y la sección 3.4.1), o bien en la barra de menú seleccionar la opción Registro, y después la opción Borrar, o también se puede pulsar la tecla Supr.

Después aparecerá un cuadro de diálogo que solicita la confirmación de la acción solicitada, como se observa en la figura 3.9.

Para que aparezca este cuadro de diálogo se utiliza la instrucción MsgBox de la siguiente forma:

```
cadMensaje = "¿Desea cancelar: " & vbCrLf & Factura: " & rsVentas!FACTURA & _
    vbCrLf & "Con fecha: " & rsVentas!FECHA & vbCrLf & "Cliente: " & _
    rsVentas!Cliente & " - " & rsVentas!NOMBRE & " ?"

' Se activa el cuadro de diálogo
If MsgBox(cadMensaje, vbQuestion + vbYesNo, " C o n f i r m a r ") = vbNo Then Exit Sub
```

Como se puede observa el cuadro de diálogo MsgBox también se puede utilizar como función, que devuelve como respuesta el valor del botón que en èl que se haya dado un clic.

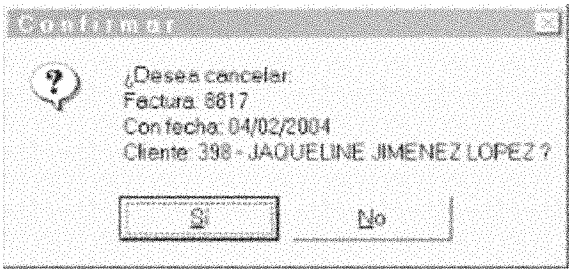


Figura 3.9: Cuadro de diálogo que solicita la confirmación de la acción

Si la respuesta es negativa (vbNo) el control, del programa sale de la subrutina en la que se encuentra. Pero si la respuesta es afirmativa (vbYes) se realiza la cancelación de la factura.

Pero antes de cancelar la factura primero se comprueba si ésta no ha sido transferida a cuentas por cobrar, lo cual se logra con la siguiente instrucción:

```
If rsVentas!Transferida="T" Then
    :
End If
```

Si se cumple la condición, se activa un cuadro de diálogo con en el que se avisa que la factura ya fue transferida, por lo cual no será cancelada (ver figura 3.10).

Nota:

- Para hacer referencia a un campo de un objeto Recordset se utiliza el nombre del objeto, seguido del símbolo "!" y después el nombre del campo. Por ejemplo, para hacer referencia al campo Factura del objeto rsVentas se escribe de la siguiente forma rsVentas!Factura.

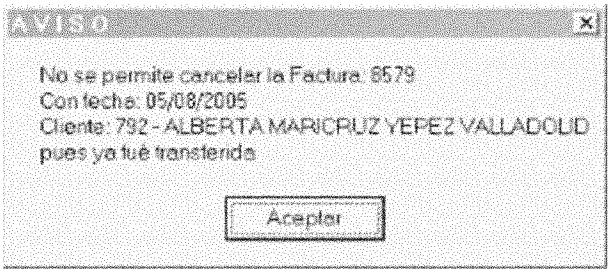


Figura 3.10: Cuadro de diálogo que envía un mensaje

Ahora bien, para cancelar una factura se debe guardar el caracter "C" en el atributo de STATUS de la tabla Facturas, pero esto no se puede realizar directamente en rsVentas debido a que los atributos que lo forman provienen de diferentes tablas, además de que al intentarlo se produce un error. Para esto es necesario utilizar un objeto Recorset auxiliar y después actualizar el objeto rsVentas, como se muestra a continuación.

```
Dim rsBorra As New ADODB.Recordset, cadSQL As String ' Declaración de variables
cadSQL = "UPDATE Facturas SET STATUS = 'C' WHERE FACTURA = " & numFactura & _
         " AND EMPRESA = '" & cadClvEmpresa & "'"
rsBorrar.Open cadSQL, conDataBase, adOpenDynamic, adLockOptimistic
rsVentas.Requery   ' Se actualiza el objeto. Después de esto se deben volver
                   ' a enmascarar los títulos del DataGrid
```

O bien, si se desea borrar un registro, la cadena de la variable cadSQL se sustituye con la siguiente:

```
cadSQL = "DELETE * From Facturas WHERE FACTURA = " & numFactura & _
         " And EMPRESA = '" & cadClvEmpresa & "'"
```

Nota:

- No olvidar que para efectuar alguna modificación en la información de algún objeto Recorset en uso, primero se debe declarar y utilizar otro objeto Recorset auxiliar, y después actualizar el objeto Recorset que está en uso.

❖ **Imprimir:** Para imprimir alguna factura, primero se debe colocar el apuntador de registro (barra de la tabla) en la factura que se va a imprimir.

Posteriormente se debe dar un clic en el icono que representa una impresora (véase figura 3.4 y la sección 3.4.1), o bien en la barra de menú seleccionar la opción Varios y después la opción Imprimir.

Con esto se abre otra ventana en la que se presenta la información que se enviará a la impresora. El generador de reportes que se utilizó en este caso es el Data Report, de Visual Basic (referirse a la ayuda en línea de este lenguaje para más información acerca del uso de este generador de reportes).

Para que esto sea posible es necesario crear el formato del reporte que se desea, después, por medio de código, enlazarlo a la fuente de datos que contiene la factura que se desea imprimir, y por último hacer el llamado del reporte, como se muestra en el siguiente bloque de código.

```
Dim conDB As New ADODB.Connection ' Variable para conexión con la base de datos

' Variable que contendrá el encabezado de la factura
Dim rsFactura As New ADODB.Recordset
Dim rsDetalle As New ADODB.Recordset  ' Variable que contendrá el detalle de la factura
Dim cadSQL As String        ' Variable que contendrá la cadena de la cláusula SQL
conDB.CursorLocation = adUseClient  ' Establece conexión a la base de datos

' Conexión tipo jerárquico o maestro/detalle (véase sección 3.1)
conDB.Open cadDSN_MSDataShape & cadDSN_MSDASQL

' Se inicializa la variable.
' Esto se puede crear con el Diseñador de entorno de datos de Visual Basic
cadSQL = "SHAPE {SELECT Facturas.*, Clientes.NOMBRE, " & _
  "Clientes.DIRECCION, Clientes.COLONIA, Clientes.CP, " & _
  "Clientes.POBLACION, Clientes.ESTADO, Clientes.RFC, " & _
  "Clientes.TELEFONO1, TipoPago.Descripcion " & _
  "FROM Facturas, Clientes, TipoPago " & _
  "WHERE Facturas.Empresa = '" & datgrdFactura.Columns(8).Text & "' And " & _
  "Facturas.FACTURA = " & Val(datgrdFactura.Columns(0).Text) & " And " & _
  "Facturas.CLIENTE = Clientes.CLIENTE AND " & _
  "Facturas.Empresa  = Clientes.Empresa AND " & _
  "Facturas.TIPOPAGO = TipoPago.Clave} AS cmdFactura " & _
  "APPEND ({SELECT Articulos_Factura.FACTURA, Articulos_Factura.CANTIDAD, " & _
```

```
  "Articulos_Factura.ARTICULO, Articulos.DESCRIPCION, " & _
  "Articulos_Factura.PRECIO AS Precio, " & _
  "Articulos_Factura.CANTIDAD * Precio AS Importe " & _
  "FROM Articulos, Articulos_Factura " & _
  "WHERE Articulos_Factura.Empresa = '" & datgrdFactura.Columns(8).Text & _
  "' And Articulos_Factura.FACTURA = " & Val(datgrdFactura.Columns(0).Text) & _
  " And Articulos.ARTICULO = Articulos_Factura.ARTICULO " & _
  " ORDER BY Articulos_Factura.PARTIDA} AS cmdDetalle " & _
  "RELATE FACTURA TO FACTURA) AS cmdDetalle"

rsFactura.Open cadSQL, conDB, adOpenDynamic, adLockOptimistic     ' Se abre la tabla
' Si está vacía la tabla se sale de la subrutina
If rsFactura.BOF And rsFactura.EOF Then Exit Sub

' Se establece la fuente del detalle de la factura
Set rsDetalle = rsFactura ("cmdDetalle").UnderlyingValue
DoEvents     ' Se pasa el control  al sistema operativo para evitar que se sature la memoria
rptFactura.TituloInf "Factura"     ' Titulo que aparecerá en la barra de título de la ventana

Set rptFactura.DataSource = rsFactura    ' Se establece la fuente de datos para el reporte
rptFactura.DetailMember = "cmdDetalle" ' Se establece el detalle de la factura para el reporte
rptFactura.Show     ' Se presenta una vista previa de la factura
Set rptFactura.DataSource = Nothing
```

Nota:

- Si el reporte que se imprimirá es simple, es decir, no es de tipo jerárquico, no es necesario declarar otra variable para la conexión a la base de datos, ni otra variable cadSQL, ni tampoco abrir otra tabla, sino que basta conectar al reporte la tabla previamente abierta, es decir rsVentas, como se muestra a continuación.

  ```
  Set rptCatalogo.DataSource = rsVentas ' Establece la fuente de datos para el reporte.
  rptCatalogo.Show     ' Se presenta una vista previa del catálogo
  ```

❖ Uso de Crystal Report

Como es sabido, las remisiones y las facturas van a nombre del cliente y estos documentos se imprimen en hojas preimpresas, es decir, la hoja en la que se imprimen estos documentos ya tienen el nombre de la empresa que la emite, por lo que es suficiente con un mismo formato rptFactura para emitir las remisiones y las facturas de cualquier empresa.

Pero no sucede lo mismo si se desea saber, por separado, lo que cada empresa vendió en un período determinado. En este caso se debe imprimir un informe al que .en materia de administración se le llama Diario de ventas detallado, en el que se debe incluir la siguiente información:

- Nombre de la empresa que emitió la factura
- Período de consulta
- Nombre del cliente de cada factura,
- Detalle de cada factura
- Importe de cada factura
- Total del día de todas las facturas emitidas
- Total final del período consultado

Para este tipo de informes el reporteador de Visual Basic está muy limitado, pues se tendría que hacer un reporte por empresa, además no tiene forma de colocar el período de consulta por medio de código, por lo que se utilizó Crystal Report, el cual es más práctico, y fácil de operar por código.

Para utilizar un reporte hecho con Cristal Report, se puede realizar como se muestra en el siguiente bloque de código.

```
‘ Declaración de variables
‘ Variable que contendrá la condición que debe cumplir la información a consultar
Dim cadSQL As String
Dim Dia1 As Variant, Mes1 As Variant, Agno1 As Variant
Dim Dia2 As Variant, Mes2 As Variant, Agno2 As Variant

‘Las siguientes asignaciones son necesarias para comparar fechas dentro del Cristal Rpt

‘Se asigna el número del día del control DTPicker1(0) a la variable Dia1
Dia1 = Day(DTPicker1(0).Value)

‘ Se asigna el número del mes del control DTPicker1(0) a la variable Mes1
Mes1 = Month(DTPicker1(0).Value)

‘ Se asigna el número del año del control DTPicker1(0) a la variable Agno1
Agno1 = Year(DTPicker1(0).Value)

‘ Se asigna el número del día del control DTPicker1(1) a la variable Dia2
Dia2 = Day(DTPicker1(1).Value)

‘ Se asigna el número del mes del control DTPicker1(1) a la variable Mes2
Mes2 = Month(DTPicker1(1).Value)

‘ Se asigna el número del año del control DTPicker1(1) a la variable Agno2
Agno2 = Year(DTPicker1(1).Value)

DoEvents     ‘ Se pasa el control  al sistema operativo para evitar que se sature la memoria
With frmInicio.CrystalReport1      ‘ Invoca al control CrystalReport1 para su uso
    .DiscardSavedData = 1          ‘ Se ignora la información guardada al crear el informe
    .Connect = cadDSN_MSDASQL      ‘ Establece la conexión a la base de datos

 ‘ Se fija el titulo que aparecerá en la barra de título de la ventana del reporte
    .WindowTitle = "Diario de ventas detallado"
    .ReportFileName = App.Path & "\rdvdetal.rpt"  ‘ Especifica el informe que se va a imprimir

' Escribe el período que comprende el informe en la fórmula de nombre "Periodo"
 .Formulas(0) = " Periodo = "' & "De " & DTPicker1(0).Value & " A " & DTPicker1(1).Value & "'"

' Establece condiciones
    cadSQL = "{Facturas.Fecha} in Date(" & Agno1 & "," & Mes1 & "," & Dia1 & ") to " & _
        Date(" & Agno2 & "," & Mes2 & "," & Dia2 & ") AND " & _
```

```
        "{Facturas.Empresa} = '" & cadClvEmpresa & "'"
    .ReplaceSelectionFormula (cadSQL)    ' Establece la selección para el informe
    .WindowState = crptMaximized         ' Indica que se presente la ventana maximizada
    .Action = 1          ' Se imprime en una vista previa el informe
    .DiscardSavedData = 1          ' Se descarta guardar la información presentada
    .Connect = ""     ' Se borra la cadena de conexión a la base de datos
    .LogOffServer 0, True  ' Se cierra la conexión a la base de datos
  End With
```

Notas:

- Crystal Report utiliza el formato de fecha aaaa/mm/dd
- Formulas(0) es un campo que se debe crear dentro de Cristal Report y en el que se puede insertar texto por medio de código. Si se utiliza otro campo de este tipo se hace referencia a él como Formulas(1), etc.
- La forma de hacer referencia a un campo de una tabla que está dentro del reporte hecho en Cristal Report es, por ejemplo: {Facturas.Fecha}
- ReplaceSelectionFormula contiene la cadena de condición, es decir, lo que en SQL es la cláusula WHERE, y por programa no se puede alterar la clausula SELECT
- DTPicker1(0). Y DTPicker1(1). son controles fechadores de Visual Basic
- Dentro de la cláusula WHERE, Crystal Report suele no relacionar correctamente más de tres tablas de la base de datos, pero esto se compensa al establecer las condiciones en ReplaceSelectionFormula a través de la variable cadSQL
- Para más información referirse a la ayuda en línea de Crystal Report

3.5.2 Captura de información

La captura de la información se realiza en otra ventana en donde el usuario debe introducir los datos solicitados por el programa. Esta ventana se activa de cualquiera de las siguientes formas (véase figura 3.4 y la sección 3.4.1):

- Para agregar una nueva factura:
  - Pulsar la tecla de la letra N
  - En la barra de herramientas dar un clic en el icono que representa una hoja en blanco
  - En la barra de menú dar un clic en la opción Registro, y después seleccionar la opción Nuevo
- Para modificar una factura:
  - Pulsar la tecla de la letra M
  - En la barra de herramientas dar un clic en el icono que representa una mano con una libreta
  - En la barra de menú dar un clic en la opción Registro, y después seleccionar la opción Modificar

Cuando se elige agregar factura los campos de captura aparecen en blanco, y cuando se elige modificar estos aparecen con la información capturada anteriormente.

Ahora bien, los campos en los que se efectúa la captura son controles de distintos tipos, los cuales se listan a continuación:

:

TextBox	Se puede editar datos de tipo numérico y de tipo carácter
MaskEdBox	Se puede editar datos de tipo numérico y de tipo carácter, con la diferencia de que se puede restringir la captura únicamente a datos numéricos

DTPicker	Se captura la fecha. Este control presenta un calendario del mes actual
ComboBox	Contiene una lista de elementos de la cual se puede seleccionar uno
CheckBox	Control en el que se capturo información de tipo bolean
DataGrid	Control en el que se captura el detalle de la factura
Label	Control en el que se puede colocar algún mensaje

A continuación se muestran algunos ejemplos de cómo cargar la información ya existente para una determinada factura.

```
 ‘ Declaración de la variable que contendrá el encabezado de la factura
Dim rsTabla As New ADODB.Recordset

‘ Declaración de la variable que contendrá el detalle de la factura
Dim rsDetalle As New ADODB.Recordset

‘ Declaración de la variable para obtener información adicional
Dim rsTabla As New ADODB.Recordset

‘ Variable que contendrá los campos deseados y que cumplan con cierta condición
Dim cadSQL As String

 ‘ Se inicializa la variable cadSQL
cadSQL = "Select * From Facturas Where FACTURA = " & numFactura & _
     " And EMPRESA = '" & cadClvEmpresa & "'"

‘ Si el objeto rsFactura está en uso se cierra
If rsFactura.State = adStateOpen Then rsFactura.Close

‘ Se abre la tabla
rsFactura.Open cadSQL, conDataBase, adOpenDynamic, adLockOptimistic

‘ Se asigna el contenido de cada campo de la tabla rsFactura, al campo de edición que
‘ le corresponde
Text1(0).Text = rsFactura!Factura
DTPicker1(0).Value = IIf(VarType(rsFactura!FECHA) = vbNull, Date, rsFactura!FECHA)
DTPicker1(1).Value = IIf(VarType(rsFactura!FECHA) = vbNull, Date, rsFactura!FEVENCIMIE)
Text1(2).Text = rsFactura!Cliente
:
‘ Se asigna el resto de la información del encabezado de la factura
:
‘ Asignación a la variable para obtener información adicional
cadSQL = "Select CLIENTE, NOMBRE, DIRECCION, COLONIA, RFC, TELEFONO1, " & _
  "POBLACION, ESTADO, DIASCREDI, CREDITO, SALDO, TIPOCLIENTE, " & _
  “VIAEMBARQUE " & _
  " From Clientes Where CLIENTE = " & Val(Trim(Text1(2).Text)) & _
  " And EMPRESA = '" & cadClvEmpresa & "'"
If rsTabla.State = adStateOpen Then rsTabla.Close
rsTabla.Open cadSQL, conDataBase, adOpenDynamic, adLockOptimistic
```

```
Label2(0).Caption = rsTabla!NOMBRE
:
‘ Se asigna el resto de la información a las etiquetas Label
‘ Se crea la tabla virtual que contendrá el detalle de la factura
With rsDetalle
   If .State = adStateOpen Then .Close
   .Fields.Append "Clave", adChar, 7, adFldUpdatable ‘ Se crea la estructura de la tabla
   .Fields.Append "Cantidad", adCurrency, , adFldUpdatable
   .Fields.Append "Descripción", adChar, 50, adFldUpdatable
   .Fields.Append "Precio", adCurrency, , adFldUpdatable
   .Fields.Append "Importe", adCurrency, , adFldUpdatable
   .Fields.Append "ImpDSCTO", adCurrency, , adFldUpdatable
‘ Se utiliza para efectos de modificación en la cantidad de existencia de materia prima
   .Fields.Append "CantidadAnt", adCurrency, , adFldUpdatable´

 ‘ Se utilizan por si la captura del detalle se efectúa a través de un pedido
   .Fields.Append "CANTXSUR", adCurrency, , adFldUpdatable
   .Fields.Append "CantPedido", adCurrency, , adFldUpdatable

 ‘ Se ordena de acuerdo a como se vayan capturando el detalle
   .Fields.Append "Partida", adInteger, , adFldUpdatable

’ Para efectos de borrado de algún artículo: A = Activo, B = Borrado
   .Fields.Append "cadEstado", adChar, 1, adFldUpdatable
   .CursorType = adOpenKeyset
   .LockType = adLockOptimistic
   .Open        ‘ Se pone en uso la tabla
   .Filter = "cadEstado = 'A'"          ‘ se presenta únicamente lo que está activo
End With

‘ Se carga el detalle de la factura
cadSQL = "SELECT * From Articulos_Factura Where FACTURA = " & numFactura & _
     " And EMPRESA = '" & cadClvEmpresa & "' Order By PARTIDA"
If rsTabla.State = adStateOpen Then rsTabla.Close
rsTabla.Open cadSQL, conDataBase, adOpenDynamic, adLockOptimistic
rsTabla.MoveFirst
Do     ‘ Se copia el detalle de la factura a la tabla virtual
   rsArticulos.MoveFirst
   rsArticulos.Find "ARTICULO = '" & rsTabla!ARTICULO & "'"
   rsDetalle.AddNew
   rsDetalle!Clave = rsTabla!ARTICULO
   rsDetalle!CANTIDAD = rsTabla!CANTIDAD
   rsDetalle!Descripción = rsArticulos!DESCRIPCION
   rsDetalle!PRECIO = rsTabla!PRECIO
   rsDetalle!Importe = rsTabla!CANTIDAD * rsTabla!PRECIO
   rsDetalle!ImpDSCTO = rsTabla!ImpDSCTO
   rsDetalle!CantidadAnt = rsTabla!CANTIDAD
   rsDetalle!Partida = rsTabla!Partida
```

```
   rsDetallePF!cadEstado = "A"
   rsDetalle.Update
   rsDetalle.MoveFirst
   rsTabla.MoveNext
Loop Until rsTabla.EOF

‘ Se establece la fuente de datos para el DataGrid que contiene el detalle de la factura
Set datgrdDetalle.DataSource = rsDetalle
datgrdDetalle.setFocus      ‘ Se establece el enfoque en el detalle de la factura
SendKeys "{DOWN}+{UP}", True ‘ Se envía una secuencia de teclas par asegurar el enfoque
```

Para la captura del detalle es más recomendable utilizar una tabla virtual con el fin de prevenir escribir directamente en la base de datos, por si el usuario desea no efectuar la factura, y se tenga que borrar físicamente en la base de datos.

En el anexo C se expone la forma en como utilizar por código cada uno de estos controles.

Anexo A
Implementación de una tabla virtual

Una tabla virtual es una tabla en la que se pueden hacer operaciones de: agregar, modificar y borrar, sin cambiar la información contenida en la base de datos, a menos que el programador lo decida al efectuar operaciones en esta tabla, o bien al cerrarla (siendo esto último lo más recomendable).

Una tabla virtual se puede declarar de la siguiente forma:

```
Dim rsVirtual As New ADODB.Recordset
```

Posteriormente se declara cada uno de los campos que contendrá, incluyendo el tipo y longitud de cada uno de estos, según se muestra en la siguiente línea de instrucción:

```
rsVirtual.Fields.Append "Empresa", adChar, 2, adFldUpdatable
```

Aquí:

rsVirtual	es la variable que contendrá los registros
Fields.Append	se utiliza para declarar y agregar un campo
“Empresa”	es nombre de uno de los campos que se agregará a la tabla
adChar	es el tipo de dato (cadena) que guardará el campo “Empresa”
2	es la longitud del campo “Empresa”
adFldUpdatable	Indica que se puede escribir en el campo

Después se abre la tabla como se muestra a continuación:

```
‘ Se establece el tipo de cursor que se utilizará
        rsVirtual.CursorType = adOpenDynamic

‘ Se establece el tipo de bloqueo que se pondrá durante la edición de cada uno de los
‘ registros
        rsVirtual.LockType = adLockOptimistic
        rsVirtual.Open          ‘ Se activa la tabla para hacer operaciones en ella
```

Nota:

- Para mayor información de adOpenDynamic y de adLockOptimistic referirse a la ayuda en línea de visual basic

Una vez activada la tabla es posible agregar, modificar o borrar información en ella, como se muestra a continuación:

```
        rsVirtual.AddNew            ‘ Agrega registro
        rsVirtual! Empresa = “DP”   ‘ Asigna la cadena DP al campo Empresa
        rsVirtual.Update            ‘ Actualiza la información del registro
        rsVirtual.MovePrevious      ‘ Mueve el apuntador al anterior registro
        rsVirtual.MoveNext          ‘ Mueve el apuntador al siguiente registro
```

Nota:

- Las dos últimas líneas son necesarias para asegurar que se guarde la información

A continuación se muestra un ejemplo de cómo emplear una tabla virtual, si se desea copiar información de una tabla rsVentas previamente abierta

```
    Dim rsVirtual As New ADODB.Recordset ‘ Se declara la variable
‘ Se declara cada uno de los campos que contendrá, incluyendo el tipo y longitud de cada
‘ uno de estos
  rsVirtual.Fields.Append "Empresa", adChar, 2, adFldUpdatable
  rsVirtual.Fields.Append "Cliente", adInteger, , adFldUpdatable
  rsVirtual.Fields.Append "Nombre", adChar, 50, adFldUpdatable
  rsVirtual.Fields.Append "Fecha", adDate, , adFldUpdatable
  rsVirtual.Fields.Append "Clave", adChar, 7, adFldUpdatable
  rsVirtual.Fields.Append "Descrip", adChar, 60, adFldUpdatable
  rsVirtual.Fields.Append "Cantidad", adInteger, , adFldUpdatable
  rsVirtual.Fields.Append "Precio", adCurrency, , adFldUpdatable
  rsVirtual.Fields.Append "Importe", adCurrency, , adFldUpdatable
  rsVirtual.CursorType = adOpenDynamic      ‘ Se establece el tipo de cursor que se utilizar

‘ Se establece el tipo de bloqueo que se pondrá durante la edición de cada uno de los
‘ registros
  rsVirtual.LockType = adLockOptimistic
  rsVirtual.Open     ‘ Abre la tabla
```

Aquí:

rsVirtual	es la variable que contendrá los registros
Fields.Append	se utiliza para declarar y agregar un campo
adChar	es el tipo de dato (cadena) que guardará el campo
adInteger	es el tipo de dato (entero) que guardará el campo
adDate	es el tipo de dato (fecha) que guardará el campo
adCurrency	es el tipo de dato (moneda) que guardará el campo
adFldUpdatable	indica que se puede escribir en el campo

Nota:

- En los campos de tipo fecha y de tipo numérico no se especifica longitud, aunque en estos últimos se utilicen cifras decimales

```
rsVentas.MoveFirst              ‘ Mueve el apuntador al primer registro de la tabla rsVentas

‘ Se inicia la copia del contenido de la tabla rsVentas a la tabla virtual rsVirtual
  Do  ‘ Abre el ciclo
    rsVirtual.AddNew            ‘ Agregar registro
    rsVirtual!Fecha = rsVentas!FECHA          ‘ Asigna rsVentas!FECHA a  rsVirtual!Fecha
    rsVirtual!Clave = rsVentas!ARTICULO       ‘ Asigna rsVentas!ARTICULO a rsVirtual!Clave
    rsVirtual!Descrip = rsVentas! Descrip     ‘ Asigna rsVentas!Descrip a rsVirtual!Descrip
    rsVirtual!Cantidad = rsVentas! Cantidad   ‘ Asigna rsVentas!Cantidad a rsVirtual!Cantidad
    rsVirtual!Precio = rsVentas! Precio       ‘ Asigna rsVentas!Precio a rsVirtual!Precio
    rsVirtual!Importe = rsVentas!Importe      ‘ Asigna rsVentas!Importe a rsVirtual!Importe
    rsVirtual.Update        ‘ Actualiza registro
    rsVirtual.MoveFirst     ‘ Mueve el apuntador al primer registro de la tabla rsVirtual
    rsVentas.MoveNext       ‘ Mueve el apuntador al siguiente registro de la tabla rsVentas
  Loop until rsVentas.EOF‘ Cierra el ciclo
```

```
‘ Se enlaza el control “datgrdTabla” a rsVirtual para poder ver los registros copiados
  Set datgrdTabla.DataSource = rsVirtual        ‘ Este control es un objeto DataGrid
```

Ahora bien, si se desea copiar el contenido de una tabla fuente hacia una tabla virtual destino, y no se conocen los campos de la primera tabla, se puede hacer de la siguiente forma (en este caso rsFuente es una tabla previamente abierta, de donde se efectuara la copia):

```
  Dim rsDest As New ADODB.Recordset ‘ Se declara la variable en la que se hará  la copia
  Dim Campo As Field      ‘ Se declara la variable Campo del tipo Field
  Dim i As Integer        ‘ Se declara una varia ble de tipo entero

  If rsDest.State = adStateOpen Then rsDest.Close   ‘ Se cierra la selección previa de rsDest

‘ Se declara cada uno de los campos que contendrá rsDest, incluyendo el tipo y longitud de
‘ cada uno de estos
  For Each Campo In rsFuente.Fields    ‘ Inicia el ciclo For para crear la tabla virtual
    rsDest.Fields.Append Campo.Name, Campo.Type, Campo.ActualSize, adFldUpdatable
  Next          ‘ Termina el ciclo For, se concluye la creación de la tabla virtual
```

Aquí:

For Each Campo In rsFuente.Fields	Se indica que para cada Campo en rsFuente se efectúe el siguiente bloque de instrucciones
Fields Append	se utiliza para declarar y agregar un campo
Campo.Name	es el nombre del campo
Campo.Type	es el tipo de dato que guardará el campo
Campo.ActualSize	es la longitud del campo
adFldUpdatable	indica que se puede escribir en el campo

Nota:

- Esta información se toma de la estructura que tiene rsFuente, y se hace una copia de esta hacia la tabla rsDest para crear esta última

```
  rsDest.CursorType = adOpenDynamic         ‘ Se establece el tipo de cursor que se utilizará

‘ Se establece el tipo de bloqueo que se pondrá durante la edición de cada uno de los
‘ registros
  rsDest.LockType = adLockOptimistic
  rsDest.Open       ‘ Abre la tabla
  rsFuente.MoveFirst        ‘ Mueve el apuntador al primer registro de la tabla rsFuente
‘ Se inicia la copia del contenido de la tabla rsFuente a la tabla virtual rsDest
  Do          ‘ Abre el ciclo
    rsDest.AddNew         ‘ Agregar registro

‘ Inicia el ciclo For para copiar el contenido de cada campo de la tabla rsFuente, a cada
‘ campo de la tabla rsDest
    For i = 0 To rsFuente.Fields.Count – 1
‘ Al campo “i” de la tabla rsDest, le asigna el contenido del campo “i” de la tabla rsFuente
      rsDest.Fields(i).Value = rsFuente.Fields(i).Value
    Next              ‘ Termina el ciclo For, se concluye la copia
```

```
rsDest.Update              ‘ Actualiza registro
    rsDest.MoveFirst       ‘ Mueve el apuntador al primer registro de la tabla rsDest
    rsFuente.MoveNext      ‘ Mueve el apuntador al siguiente registro de la tabla rsFuente
  Loop until rsTblFuente.EOF     ‘ Cierra el ciclo

rsFuente.MoveFirst ‘ Mueve el apuntador al primer registro de la tabla rsFuente

‘ Se enlaza el control “datgrdTabla” a rsDest para poder ver los registros copiados
  Set datgrdTabla.DataSource = rsDest
```

En esta tabla rsDest se pueden hacer operaciones de agregar, modificar y borrar, sin alterar la información contenida en la tabla rsFuente, incluso su contenido se puede enviar a la impresora, si se proporciona esta opción al usuario.

Aunque se puede realizar la operación de borrado de registros, esta no es muy recomendable aplicarla para borrar el contenido de toda la tabla dentro de un ciclo, tal como:

```
rsDest.MoveFirst          ‘ Mueve el apuntador al primer registro
Do                        ‘ Inicia ciclo
  rsDest.Delete           ‘ Borra registro actual
  rsDest.MoveNext         ‘ Mueve el apuntador al siguiente registro
Loop Until rsDest.Eof     ‘ Cierra el ciclo hasta que sea fin de la tabla
```

Este método suele dejar el último registro, lo cual causará problemas si se desea guardar nueva información en esta tabla. Lo más recomendable es cerrar y volver a crear la tabla; para esto se debe crear un subrutina en la que se haga tal tarea.

Además, como se puede observar, una tabla virtual también se puede utilizar para guardar ciertos datos, pues en realidad es un arreglo de MxN (Renglones x Columnas)

Anexo B
Relaciones

B.1 Análisis

Las relaciones identificadas son:

- Componentes de artículos
- Detalle de entrada de materia prima
- Detalle de salida de materia prima
- Detalle de compra de materia prima
- Detalle de pedido de artículos
- Detalle de remisión de artículos
- Detalle de factura de artículos

A continuación se exponen sus respectivos modelo E-R.

❖ Componentes de artículos (MpArticulos)

La información de los componentes de cada artículo se guardará en MpArticulos, y se muestra en la figura B1, que es el modelo E-R de esta relación.

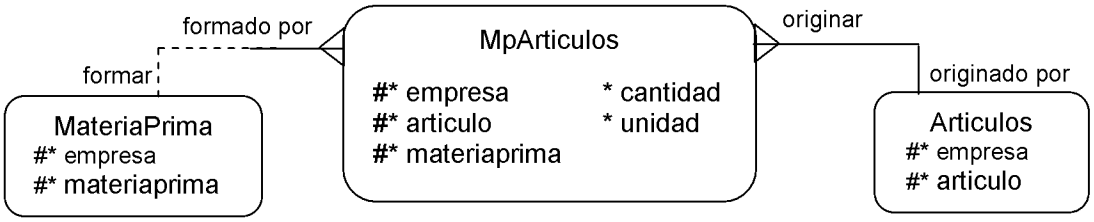


Figura B1 Modelo E-R de MpArticulos

Su clave primaria está formada por los campos: empresa, articulo, materiaprima (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, articulo, materiaprima → cantidad, unidad

❖ Detalle de entrada de materia prima (EntradasMpdetalle)

La información que se guardará de las entradas de materias primas se muestra en la figura B2, que es el modelo E-R de esta relación.

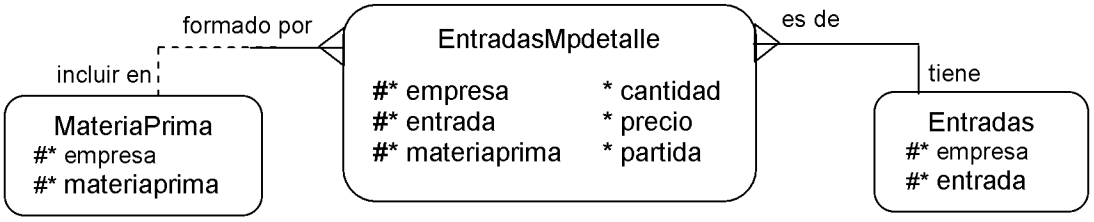


Figura B2 Modelo E-R de EntradasMpdetalle

Su clave primaria está formada por los campos: empresa, entrada, materiaprima (campos primos).

La representación de la dependencia funcional entre sus campos es:

Empresa, entrada, materiaprima → cantidad, precio, partida

❖ **Detalle de salida de materia prima (SalidasMpdetalle)**

La información que se guardará de las salidas de materias primas se muestra en la figura B3, que es el modelo E-R de esta relación.

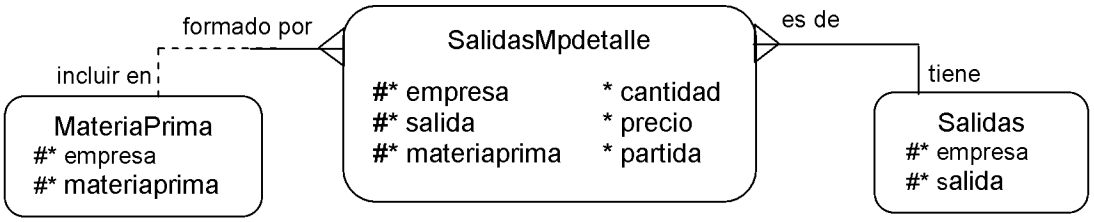


Figura B3 Modelo E-R de SalidasMpdetalle

Su clave primaria está formada por los campos: empresa, salida, materiaprima (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, salida, materiaprima → cantidad, precio, partida

❖ **Detalle de compra de materia prima (ComprasMpdetalle)**

La información que se guardará de las compras de materias primas se muestra en la figura B4, que es el modelo E-R de esta relación.

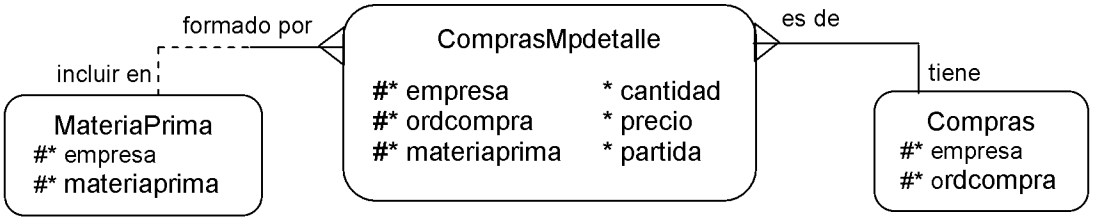


Figura B4 Modelo E-R de ComprasMpdetalle

Su clave primaria está formada por los campos: empresa, ordcompra, materiaprima (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, ordcompra, materiaprima → cantidad, precio, partida

❖ **Detalle de pedido de artículos (ArticulosPedido)**

La información que se guardará de los pedidos de artículos que efectúan los clientes se muestra en la figura B5, que es el modelo E-R de esta relación.

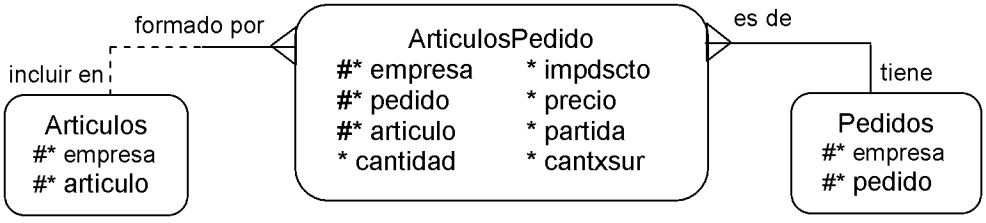


Figura B5 Modelo E-R de ArticulosPedido

Su clave primaria está formada por los campos: empresa, pedido, articulo (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, pedido, articulo → cantidad, precio, cantxsur, impdscto, partida

❖ **Detalle de remisión de artículos (ArticulosRemision)**

La información que se guardará de los artículos que compran los clientes, a través de las remisiones, se muestra en la figura B6, que es el modelo E-R de esta relación.

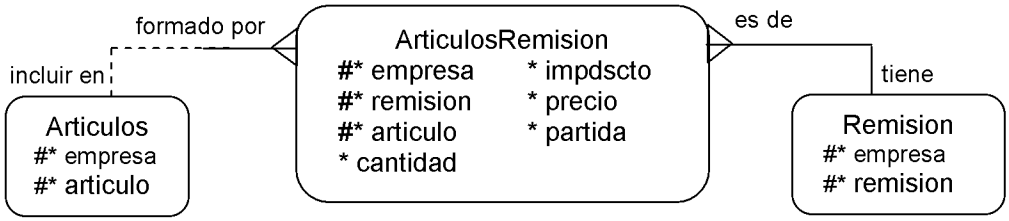


Figura B6 Modelo E-R de ArticulosRemision

Su clave primaria está formada por los campos: empresa, remision, articulo (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, remision, articulo → cantidad, precio, impdscto, partida

❖ **Detalle de factura de artículos (ArticulosFactura)**

La información que se guardará de los artículos que compran los clientes, a través de las facturas, se muestra en la figura B7, que es el modelo E-R de esta relación.

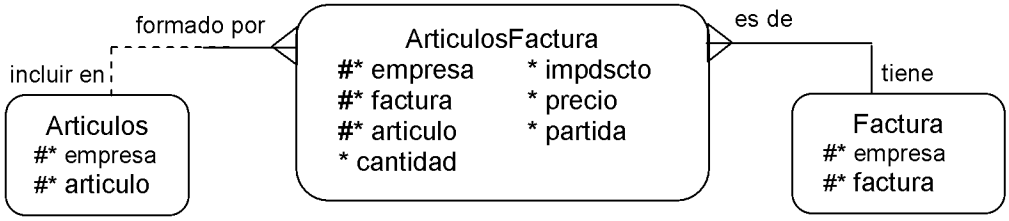


Figura B7 Modelo E-R de ArticulosFactura

Su clave primaria está formada por los campos: empresa, factura, articulo (campos primos).

La representación de la dependencia funcional entre sus campos es:

empresa, factura, articulo → cantidad, precio, impdscto, partida

B.2 Diseño

Posteriormente se construyen las tablas en las que será almacenada la información de cada relación. A continuación se exponen las tablas.

En cada tabla se muestra el tipo de dato y la longitud de cada uno de sus atributos.

❖ **MpArticulos**

Su representación en la base de datos se muestra en la tabla B1, y en ella se almacenarán las materias primas que forman cada artículo.

MpArticulos

Campo	Tipo dato	Long/Decim
empresa	Texto	2
articulo	Numérico	Entero largo
materiaprima	Texto	6
cantidad	Moneda	2
unidad	Texto	3

Tabla B1: Tabla en la que se almacenan los componentes de cada artículo

Su clave primaria está formada por los campos: empresa, articulo, materiaprima (obsérvese la figura B1). Además los atributos empresa, articulo y materiaprima son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "articulo" con la tabla Articulos
- A través del campo "materiaprima" con la tabla Materiaprima

❖ **EntradasMpdetalle**

Su representación en la base de datos se muestra en la tabla B2, y en ella se almacenará el detalle de las entradas de materia prima.

EntradasMpdetalle

Campo	Tipo dato	Long/Decim
empresa	Texto	2
entrada	Numérico	Entero largo
materiaprima	Texto	6
cantidad	Moneda	2
precio	Moneda	2
partida	Autonumérico	

Tabla B2: Tabla en la que se almacenan el detalle de las entradas de materia prima

Su clave primaria está formada por los campos: empresa, entrada y materiaprima (obsérvese la figura B2). Además los atributos empresa, entrada y materiaprima son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "entrada" con la tabla EntradasMp
- A través del campo "materiaprima" con la tabla MateriaPrima

❖ **SalidasMpdetalle**

Su representación en la base de datos se muestra en la tabla B3, y en ella se almacenará el detalle de las salidas de materia prima.

Su clave primaria está formada por los campos: empresa, salida y materiaprima (obsérvese la figura B3).

SalidasMpdetalle

Campo	Tipo dato	Long/Decim
empresa	Texto	2
salida	Numérico	Entero largo
materiaprima	Texto	6
cantidad	Moneda	2
precio	Moneda	2
partida	Autonumérico	Entero

Tabla B3: Tabla en la que se almacena el detalle de las salidas de materia prima

Además los atributos empresa, entrada y materiaprima son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "salida" con la tabla SalidaMp
- A través del campo "materiaprima" con la tabla MateriaPrima

❖ **ComprasMpdetalle**

Su representación en la base de datos se muestra en la tabla B4, y en ella se almacenará el detalle de las ordenes de compra de materia prima.

ComprasMpdetalle

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	cantidad	Moneda	2
ordcompra	Numérico	Entero largo	precio	Moneda	2
materiaprima	Texto	6	partida	Numérico	Entero

Tabla B4: Tabla en la que se almacena el detalle de las ordenes de compra

Su clave primaria está formada por los campos: empresa, ordcompra y materiaprima (obsérvese la figura B4). Además los atributos empresa, ordcompra y materiaprima son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "ordcompra" con la tabla Compras
- A través del campo "materiaprima" con la tabla MateriaPrima

❖ **ArticulosPedido**

Su representación en la base de datos se muestra en la tabla B5 y en ella se almacenará el detalle de los pedidos de artículos que efectúan los clientes.

ArticulosPedido

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	impdscto	Moneda	2
pedido	Numérico	Entero largo	precio	Moneda	2
articulo	Texto	7	partida	Numérico	Entero
cantidad	Moneda	2	cantxsur	Numérico	Entero

Tabla B5: Tabla en la que se almacena el detalle de cada pedido

Su clave primaria está formada por los campos: empresa, pedido y articulo (obsérvese la figura B5). Además los atributos empresa, pedido y articulo son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "pedido" con la tabla Pedidos
- A través del campo "articulo" con la tabla Artículos

❖ **ArticulosRemision**

Su representación en la base de datos se muestra en la tabla B6 y en ella se almacenará el detalle de los artículos que compran los clientes, a través de las remisiones.

ArticulosRemision

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	impdscto	Moneda	2
remision	Numérico	Entero largo	precio	Moneda	2
articulo	Texto	7	partida	Numérico	Entero
cantidad	Moneda	2			

Tabla B6: Tabla en la que se almacena el detalle de cada remisión

Su clave primaria está formada por los campos: empresa, remision y articulo (obsérvese la figura B6). Además los atributos empresa, remision y articulo son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "remision" con la tabla Remision
- A través del campo "articulo" con la tabla Articulos

❖ **ArticulosFactura**

Su representación en la base de datos se muestra en la tabla B7 y en ella se almacenará el detalle de los artículos que compran los clientes, a través de las facturas.

ArticulosFactura

Campo	Tipo dato	Long/Decim	Campo	Tipo dato	Long/Decim
empresa	Texto	2	impdscto	Moneda	2
factura	Numérico	Entero largo	precio	Moneda	2
articulo	Texto	7	partida	Numérico	Entero
cantidad	Moneda	2			

Tabla B7: Tabla en la que se almacena el detalle de cada factura

Su clave primaria está formada por los campos: empresa, factura y articulo (obsérvese la figura B7). Además los atributos empresa, factura y articulo son claves extranjeras, y se relaciona de la siguiente forma:

- A través del campo "empresa" con la tabla Empresas
- A través del campo "factura" con la tabla Facturas
- A través del campo "articulo" con la tabla Articulos

Anexo C
Utilizar por código algunos de los controles

A continuación se expone cómo emplear por programa cada uno de los controles empleados en la captura de una factura y de su detalle de artículos.

Una vez que se ha establecido la conexión a la base de datos, y después de abrir la tabla de facturas, se puede agregar o modificar alguna de estas. En ambos casos el código que se utiliza es el mismo, como se muestra a continuación.

❖ Eventos de controles de tipo TextBox

Este control se puede utilizar para la captura de datos de tipo texto o de tipo numérico, sin formato, pudiéndose tener un arreglo se estos. Contiene algunos procedimientos (eventos) a través de los cuales puede ser manejado.

❖ Control Text1

Se utilizaron otros procedimientos para la captura del detalle de la factura, y estos se listan a continuación.

- **Evento Text1_KeyPress**

Con este evento se interceptan las pulsaciones de teclas, para establecer los caracteres que únicamente pueden ser introducidos, como se muestra en el siguiente procedimiento:

```
Private Sub Text1_KeyPress(Index As Integer, KeyAscii As Integer)
  Dim cadRango As String

  KeyAscii = Asc(Chr(KeyAscii))
  Select Case Index
    Case 0, 1, 2, 4 : cadRango = "0123456789"    ‘ Factura, Pedido, Cliente, Cantidad
    Case 3 : cadRango = "0123456789"     ‘ Vendedor
    Case 5
      cadRango="0123456789” & “abcdefghijklmnopqrstuvwxyzáéíóúüñ” &
          “ABCDEFGHIJKLMNOPQRSTUVWXYZ” & “-.,() "
  End Select

  If KeyAscii > 26 Then   ' Si no es un código de control.
    If InStr(cadRango, Chr(KeyAscii)) = 0 Then KeyAscii = 0   ‘
  End If
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1
KeyAscii	Es el código ascii de la tecla pulsada
cadRango	Contiene la cadena de caracteres permitidos en la captura

- **Evento Text1_KeyUp**

Este evento se activa cuando se deja de pulsar alguna tecla, e intercepta su código, como se muestra en el siguiente procedimiento:

```
Private Sub Text1_KeyUp(Index As Integer, KeyCode As Integer, Shift As Integer)
```

```
Select Case KeyCode
  Case vbKeyReturn          ' Si se pulsó la tecla Enter (Intro) se toma esta opción
    Select Case Index
      Case 0 : DTPicker1(0).SetFocus     ' Establece el enfoque en DTPicker1(0)
      Case 1 : Text1(2).SetFocus         ' Establece el enfoque en Text1(2)
      Case 2 : Text1(3).SetFocus         ' Establece el enfoque en Text1(3)
      Case 3 : MaskEdBox1(0).SetFocus    ' Establece el enfoque en MaskEdBox1(0)
      Case 4 : Text1(5).SetFocus         ' Establece el enfoque en Text1(5)
    End Select
  Case vbKeyDown            ' Si se pulsó la tecla avance de linea se toma esta opción
    Select Case Index
      Case 0 : DTPicker1(0).SetFocus     ' Establece el enfoque en DTPicker1(0)
      Case 1 : Text1(2).SetFocus         ' Establece el enfoque en Text1(2)
      Case 2 : Text1(3).SetFocus         ' Establece el enfoque en Text1(3)
      Case 3 : MaskEdBox1(0).SetFocus    ' Establece el enfoque en MaskEdBox1(0)
      Case 4 : Text1(5).SetFocus         ' Establece el enfoque en Text1(5)
    End Select
  Case vbKeyUp              ' Si se pulsó la tecla retroceso de linea se toma esta opción
    Select Case Index
      Case 1 : DTPicker1(1).SetFocus     ' Establece el enfoque en DTPicker1(1)
      Case 2 : Text1(1).SetFocus         ' Establece el enfoque en Text1(1)
      Case 3 : Text1(2).SetFocus         ' Establece el enfoque en Text1(2)
      Case 4 : Combo1(0).SetFocus        ' Establece el enfoque en Combo1(0)
    End Select
  Case vbKeyTab             ' Si se pulsó la tecla Tab se toma esta opción
    If Index = 5 Then       ' Si el control Text1(5) tiene el enfoque ...
      DatgrdDetalle.SetFocus      ' Establece el enfoque en DatgrdDetalle
      DatgrdDetalle.Col = 0 ' El cursor se coloca en la columna 0 del control
DatgrdDetalle
    End If
  End Select
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1
KeyCode	Es el código de la tecla que se deja de pulsar
Shift	Es el código de la tecla que se está pulsando: MAYÚS, CTRL, ALT

❖ **Control txtMatPrim**

Este control se utilizó para capturar el detalle de la factura, y algunos de sus eventos empleados se listan a continuación

- **Evento txtMatPrim_LostFocus**

Este evento se activa cuando txtMatPrim pierde el enfoque, y se oculta como se muestra en el siguiente procedimiento:

```
Private Sub txtMatPrim_LostFocus()
  txtMatPrim.Visible = False   'Se hace invisible
End Sub
```

- **Evento txtMatPrim_KeyUp**

Este evento se activa cuando se deja de pulsar alguna tecla, e intercepta su código, como se muestra en el siguiente procedimiento:

```
Private Sub txtMatPrim_KeyUp(KeyCode As Integer, Shift As Integer)
  Dim objTxt As TextBox        ' Se declara una variable de tipo TextBox
  Dim cadSQL As String

  Select Case KeyCode
    Case vbKeyReturn           ' Si se pulsó la tecla Enter (Intro) se toma esta opción
      Select Case numColumna
        Case 0   ' Clave del artículo
          txtMatPrim.Text = UCase(txtMatPrim.Text)
          rsArticulos.MoveFirst        ' La tabla de artículos debe estar previamente abierta
          rsArticulos.Find "ARTICULO like '" & txtMatPrim.Text & "%'"
          If rsArticulos.EOF Then      ' Si no se encontró el artículo
Tabla:
' En esta parte se presenta la tabla de artículos, para seleccionar el deseado
                :
          Else    ' Si se encontró el artículo
            Asigna_MP          ' Se agrega al detalle de la factura
          End If
          DatgrdDetalle.Col = 1        ' Se coloca el enfoque en la columna 1 de DatgrdDetalle
        Case 1   ' Cantidad
          FijaCantidad  ' Se guarda la cantidad solicitada del artículo
          DatgrdDetalle.SetFocus
          DatgrdDetalle.Col = 3        ' Se coloca el enfoque en la columna 3 de DatgrdDetalle
        Case 3   ' Cambio de precio del artículo
          rsDetalleVista!Precio = Val(txtMatPrim.Text)     ' Se guarda el precio del artículo
          txtMatPrim.Visible = False ' Se oculta txtMatPrim
' En esta parte se llama a un procedimiento para actualizar los totales de la factura
          :
          :
          DatgrdDetalle.SetFocus
          DatgrdDetalle.Col = 0        ' Se coloca el enfoque en la columna 0 de DatgrdDetalle
      End Select
    Case vbKeyEscape           ' Si se pulsó la tecla Escape (Esc) se toma esta opción
      txtMatPrim.Visible = False       ' Se oculta txtMatPrim
      DatgrdDetalle.SetFocus' Se coloca el enfoque en DatgrdDetalle
  End Select
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1
KeyCode	Es el código de la tecla que se deja de pulsar
Shift	Es el código de la tecla que se está pulsando: MAYÚS, CTRL, ALT

- **Evento txtMatPrim_KeyPress**

Con este evento se interceptan las pulsaciones de teclas, para establecer los caracteres que únicamente pueden ser introducidos, como se muestra en el siguiente procedimiento:

```
Private Sub txtMatPrim_KeyPress(KeyAscii As Integer)
  Dim cadRango As String

  KeyAscii = Asc(Chr(KeyAscii))
  Select Case numColumna
    Case 0 : cadRango = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"  ' Clave
    Case 1, 3 : cadRango = "0123456789."          ' Cantidad, Precio
  End Select

  If KeyAscii > 26 Then   ' Si no es un código de control.
    If InStr(cadRango, Chr(KeyAscii)) = 0 Then KeyAscii = 0
  End If
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1
KeyAscii	Es el código ascii de la tecla pulsada
cadRango	Contiene la cadena de caracteres permitidos en la captura

❖ **Eventos del control de tipo DTPicker**

Este control se utilizó para capturar los datos de tipo fecha de la factura, pudiéndose tener un arreglo se estos. Contiene algunos procedimientos (eventos) a través de los cuales puede ser manejado.

- **Evento DTPicker1_KeyUp**

Este evento se activa cuando se deja de pulsar alguna tecla, e intercepta su código, como se muestra en el siguiente procedimiento:

```
Private Sub DTPicker1_KeyUp(Index As Integer, KeyCode As Integer, Shift As Integer)
  Select Case KeyCode
    Case vbKeyReturn          ' Si se pulsó la tecla Enter (Intro) se toma esta opción
      Select Case Index
        Case 0 : DTPicker1(1).SetFocus     ' Establece el enfoque en DTPicker1(1)
        Case 1 : Text1(1).SetFocus         ' Establece el enfoque en Text1(1)
      End Select
  End Select
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1
KeyCode	Es el código de la tecla que se deja de pulsar
Shift	Es el código de la tecla que se está pulsando: MAYÚS, CTRL, ALT

❖ Eventos del control de tipo ComboBox

Este control también se utilizó para capturar una parte de la información de la factura, pudiéndose tener un arreglo se estos. Contiene algunos procedimientos (eventos) a través de los cuales puede ser manejado.

- **Evento Combo1_KeyUp**

Este evento se activa cuando se deja de pulsar alguna tecla, e intercepta su código, como se muestra en el siguiente procedimiento:

```
Private Sub Combo1_KeyUp(Index As Integer, KeyCode As Integer, Shift As Integer)
  Select Case KeyCode
    Case vbKeyReturn          ' Si se pulsó la tecla Enter (Intro) se toma esta opción
      Select Case Index
        Case 0 : Combo1(1).SetFocus       ' Establece el enfoque en Combo1(1)
        Case 1 : Text1(4).SetFocus        ' Establece el enfoque en Text1(4)
      End Select
  End Select
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1
KeyCode	Es el código de la tecla que se deja de pulsar
Shift	Es el código de la tecla que se está pulsando: MAYÚS, CTRL, ALT

❖ Eventos del control de tipo MaskEdBox

Este control se utilizó para capturar una parte de la información de la factura, se puede emplear para introducir datos de tipo texto o de tipo numérico, con formato, pudiéndose tener un arreglo se estos. Contiene algunos procedimientos (eventos) a través de los cuales puede ser manejado.

- **Evento MaskEdBox1_KeyPress**

Con este evento se interceptan las pulsaciones de teclas, para establecer los caracteres que únicamente pueden ser introducidos, como se muestra en el siguiente procedimiento:

```
Private Sub MaskEdBox1_KeyPress(Index As Integer, KeyAscii As Integer)
  Dim cadRango As String

  KeyAscii = Asc(Chr(KeyAscii))
  Select Case Index
    Case 0, 1, 2 : cadRango = "0123456789."
  End Select

  If KeyAscii > 26 Then          ' Si no es un código de control.
    If InStr(cadRango, Chr(KeyAscii)) = 0 Then KeyAscii = 0
  End If
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1

KeyAscii	Es el código ascii de la tecla pulsada
cadRango	Contiene la cadena de caracteres permitidos en la captura

- **Evento MaskEdBox1_KeyUp**

Este evento se activa cuando se deja de pulsar alguna tecla, e intercepta su código, como se muestra en el siguiente procedimiento:

```
Private Sub MaskEdBox1_KeyUp(Index As Integer, KeyCode As Integer, Shift As Integer)
  Select Case KeyCode
    Case vbKeyReturn          ' Si se pulsó la tecla Enter (Intro) se toma esta opción
      Select Case Index
        Case 0 : MaskEdBox1(1).SetFocus ' Establece el enfoque en MaskEdBox1 (1)
        Case 1 : Combo1(0).SetFocus     ' Establece el enfoque en Combo1(0)
      End Select
    Case vbKeyDown            ' Si se pulsó la tecla avance de linea se toma esta opción
      Select Case Index
        Case 0 : MaskEdBox1(1).SetFocus ' Establece el enfoque en MaskEdBox1 (1)
        Case 1 : Combo1(0).SetFocus     ' Establece el enfoque en Combo1(0)
      End Select
    Case vbKeyUp              ' Si se pulsó la tecla retroceso de linea se toma esta opción
      Select Case Index
        Case 0 : Text1(3).SetFocus      ' Establece el enfoque en Text1(3)
        Case 1 : MaskEdBox1(1).SetFocus ' Establece el enfoque en MaskEdBox1(1)
      End Select
  End Select
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1
KeyCode	Es el código de la tecla que se deja de pulsar
Shift	Es el código de la tecla que se está pulsando: MAYÚS, CTRL, ALT

❖ **Eventos del control de tipo DataGrid**

Este control se puede utilizar para consultar el contenido de una tabla, incluso para la captura de información; también se puede tener un arreglo se estos. Contiene algunos procedimientos (eventos) a través de los cuales puede ser manejado. Se utilizó para mostrar el detalle de la factura

- **Evento DatgrdDetalle_KeyUp**

Este evento se activa cuando se deja de pulsar alguna tecla, e intercepta su código, como se muestra en el siguiente procedimiento:

```
Private Sub DatgrdDetalle_KeyUp(KeyCode As Integer, Shift As Integer)
  Dim varBookmark As Variant, cadMensaje As String

  Select Case KeyCode
    Case vbKeyDelete          ' Si se pulsó la tecla Supr (Delete) se toma esta opción
      If rsDetalleVista.BOF And rsDetalleVista.EOF Then Exit Sub
      cadMensaje="¿Desea borrar: " & vbCrLf & DatgrdDetalle.Columns(0).Text & " - " & _
        DatgrdDetalle.Columns(2).Text & " ?"
```

```
        If MsgBox(cadMensaje, vbQuestion + vbYesNo, "Confirmar") = vbNo Then Exit Sub
        rsDetalleVista!cadEstado = "C"
        rsDetalleVista.Update
        rsDetalleVista.MoveFirst
        Set DatgrdDetalle.DataSource = Nothing
        Set DatgrdDetalle.DataSource = rsDetalleVista
        MascaraDatGrd
        rsDetalleVista.Bookmark = varBookmark
    End Select
End Sub
```

En donde

Index	Es el índice en el arreglo de controles Text1
KeyCode	Es el código de la tecla que se deja de pulsar
Shift	Es el código de la tecla que se está pulsando: MAYÚS, CTRL, ALT

- **Evento DatgrdDetalle_BeforeColEdit**

Este evento se activa cuando se deja de pulsar alguna tecla, e intercepta su código, como se muestra en el siguiente procedimiento:

```
Private Sub DatgrdDetalle_BeforeColEdit(ByVal ColIndex As Integer, _
  ByVal KeyAscii As Integer, Cancel As Integer)
  Dim cadRango As String
  Cancel = True       ' Cancela modificación (edición)
  If numPedido > 0 Then
    If rsDetalleVista.RecordCount - 1 = numRegsPedido Then Exit Sub
  End If

  If rsDetalleVista.RecordCount >= 25 Then Exit Sub
  If ColIndex Like "[25]" Then Exit Sub ' Columna 2 – Descripción, Columna 5 - Importe
  If ColIndex Like "[13]" Then   ' Columna 1 - Cantidad, Columna 3 - Precio
    If Len(DatgrdDetalle.Columns(0).Text) = 0 Then
      MsgBox "Falta clave del artículo", vbInformation & vbOKOnly, " A v i s o"
      DatgrdDetalle.Col = 0
    End If
  End If

  If numPedido > 0 And ColIndex = 0 Then ' Si existe pedido y la columna de captura es la
    If rsDetalleVista.EOF Then                ‘ clave del artículo
      If rsDetalleVista.RecordCount - 1 = numRegsPedido Then Exit Sub
    Else
      Exit Sub
    End If
  End If

  KeyAscii = Asc(Chr(KeyAscii))
  Select Case ColIndex
    Case 0 : cadRango = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"     ' Clave
    Case 1, 3 : cadRango = "0123456789."          ' Cantidad
```

```
    End Select

    If KeyAscii > 26 Then   ' Si no es un código de control.
      If InStr(cadRango, Chr(KeyAscii)) = 0 Then
        KeyAscii = 0
        Exit Sub
      End If
    End If
    CantidadMP ColIndex, KeyAscii
  End Sub
```

En donde

Colindex	Es el índice de la columna donde se ubica el cursor, en el DatgrdDetalle
KeyAscii	Es el código ascii de la tecla pulsada
Cancel	Entero que se puede establecer a True para evitar modificar la celda

❖ **Otros procedimientos**

Se utilizaron otros procedimientos para la captura del detalle de la factura, y estos se listan a continuación.

- **Evento Asigna_MP**

Este se utilizó para guardar la clave del artículo seleccionado, en el detalle de la factura, como se muestra en el siguiente procedimiento:

```
Private Sub Asigna_MP()
  Dim cadBuscar As String, numPartida As Integer

  If rsDetalleVista.RecordCount = 0 Then ' Si la tabla está vacía
    rsDetalleVista.AddNew          ' Agrega nuevo registro
    rsDetalleVista!cadEstado = "A"
    rsDetalleVista!Partida = rsDetalleVista.RecordCount
  Else                ' La tabla no está vacía
' Si la tabla no está vacía se busca la clave del artículo seleccionado,
' para evitar duplicidad en el detalle de la factura
    rsDetalleVista.MoveFirst       ' Se coloca el apuntador en el primer registro
    rsDetalleVista.Find "Articulo = '" & txtMatPrim.Text & "'"
    If rsDetalleVista.EOF Then     ' El apuntador está en el final de la tabla (no encontró)
      rsDetalleVista.AddNew        ' Agreega nuevo registro
      rsDetalleVista!cadEstado = "A"
      rsDetalleVista!Partida = rsDetalleVista.RecordCount
    Else  ' Encontró la clave
      DatgrdDetalle.SetFocus       ' Establece el enfoque en DatgrdDetalle
      Exit Sub   ' Sale del procedimiento sin guardar, para evitar duplicidad
    End If
  End If

  rsDetalleVista!ARTICULO = txtMatPrim.Text
  rsDetalleVista!DESCRIPCION = rsArticulos!DESCRIPCION
```

```
  rsDetalleVista!Cantidad = 0
  rsDetalleVista!CantidadAnt = 0
  rsDetalleVista!Precio = Round(rsArticulos!PRECIO * (1 - MaskEdBox1(0).Text / 100), 2)
  rsDetalleVista!ImpDSCTO = Round(rsArticulos!PRECIO * MaskEdBox1(0).Text / 100, 2)
  rsDetalleVista.Update
  rsDetalleVista.MoveFirst
  rsDetalleVista.MoveLast
  txtMatPrim.Visible = False   ' Se oculta txtMatPrim
  MascaraDatGrd      ' Se da formato a la presentación de la información en
DatgrdDetalle
  DatgrdDetalle.SetFocus          ' Establece el enfoque en DatgrdDetalle
  DatgrdDetalle.Col = 1           ' Establece el enfoque en la columna 1 de
DatgrdDetalle
End Sub
```

- **Evento CantidadMP**

Este se utilizó para situar txtMatPrim en la celda del DatgrdDetalle, en donde se va a hacer la captura, como se muestra en el siguiente procedimiento:

```
Private Sub CantidadMP(ColIndex As Integer, KeyAscii As Integer)
  Select Case ColIndex
    Case 0        ' Para la clave del artículo
      txtMatPrim.MaxLength = 7 ' Se establece la longitud máxima que tendrá txtMatPrim
    Case 1, 3     ' Para la cantidad, y el precio
      txtMatPrim.MaxLength = 10 ' Se establece la longitud máxima que tendrá txtMatPrim
  End Select

  With DatgrdDetalle  ' Se realiza la ubicación de txtMatPrim en la celda del
DatgrdDetalle
    txtMatPrim.Height = .RowHeight ' Se fija la altura de txtMatPrim
    txtMatPrim.Width = .Columns(ColIndex).Width   ' Se fija el ancho de txtMatPrim
    txtMatPrim.Left = .Left + .Columns(ColIndex).Left   ' Se fija la abscisa de txtMatPrim
    txtMatPrim.Top = .Top + .RowTop(.Row)          ' Se fija la ordenada de txtMatPrim
  End With

  txtMatPrim.Visible = True    ' Se hace visible

' Se convierte el valor Ascii de la tecla pulsada y se asigna a txtMatPrim.Text
  txtMatPrim.Text = Chr(KeyAscii)
  txtMatPrim.ZOrder 0 ' Se coloca en  primer plano txtMatPrim
  txtMatPrim.SetFocus          ' Establece el enfoque en txtMatPrim
  txtMatPrim.SelStart = 1 ' La captura del resto de los carateres inicia en la posición 1
End Sub
```

En donde

Colindex	Es el índice de la columna donde se ubica el cursor, en el DatgrdDetalle
KeyAscii	Es el código ascii de la tecla pulsada

- **Evento FijaCantidad**

Este se utilizó para guardar la cantidad del artículo seleccionado, en el detalle de la factura, como se muestra en el siguiente procedimiento:

```
Private Sub FijaCantidad()
  Dim numCantidad As Currency

  numCantidad = Val(txtMatPrim.Text)
  If Val(Text1(1).Text) > 0 Then        ' Si existe pedido
 ' Si la cantidad es mayor a la del pedido
    If rsDetalleVista!CantPedido < numCantidad Then
      MsgBox "La cantidad en factura es mayor a la cantidad del pedido", _
        vbInformation & vbOKOnly, " A v i s o"
      DatgrdDetalle.Col = 1    ' Establece el enfoque en la columna 1 de DatgrdDetalle
      Exit Sub    ' Sale del procedimiento cancelándose la captura de la cantidad
    End If
 ' Se guarda la cantidad faltante del artículo por entregar al cliente
    rsDetalleVista!CANTXSUR = rsDetalleVista!CantPedido - numCantidad
  End If

  rsDetalleVista!Cantidad = numCantidad
  rsDetalleVista!Importe = numCantidad * rsDetalleVista!Precio
  rsDetalleVista.Update
  txtMatPrim.Visible = False   ' Se oculta txtMatPrim
' En esta parte se llama a un procedimiento para actualiza los totales de la factura
    :
    :
  DatgrdDetalle.Col = 3         ' Establece el enfoque en la columna 3 de DatgrdDetalle
End Sub
```

- **Evento MascaraDatGrd**

Este se utilizó para guardar la cantidad del artículo seleccionado, en el detalle de la factura, como se muestra en el siguiente procedimiento:

```
Private Sub MascaraDatGrd()
  Dim objColumn As Column, i As Integer

  With DatgrdDetalle
    For Each objColumn In .Columns     ' Para cada columna de DatgrdDetalle hacer ...
      objColumn.Visible = False          ' Se ocultan las columnas
    Next

    For i = 0 To 4     ' Para las columnas 0 a 4 de DatgrdDetalle hacer ...
      Columns(i).Visible = True              ' Se hace visible la columna i
    Next

    .Columns(0).Width = 1000                 ' Ancho de la columna

    .Columns(1).Alignment = dbgRight       ' Se alinea a la derecha la información
    .Columns(1).NumberFormat = "###,##0.00"  ' Presentación para cantidad
```

```
        .Columns(1).Width = 1000                ‘ Ancho de la columna

        .Columns(2).Width = 6600                ‘ Ancho de la columna

        .Columns(3).Alignment = dbgRight        ‘ Se alinea a la derecha la información
        .Columns(3).NumberFormat = "$ ###,##0.00" ‘ Presentación para tipo moneda
        .Columns(3).Width = 1100                ‘ Ancho de la columna

        .Columns(4).Alignment = dbgRight        ‘ Se alinea a la derecha la información
        .Columns(4).NumberFormat = "$ ###,##0.00" ‘ Presentación para tipo moneda
        .Columns(4).Width = 1200                ‘ Ancho de la columna

        .MarqueeStyle = 2  ‘ Se establece el estilo de la barra del apuntador de renglón
        .Col = 0           ‘ Establece el enfoque en la columna 0 de DatgrdDetalle
    End With
End Sub
```

Como se podrá observar en el módulo de la captura de la factura se utilizó rsDetalleVista, que es la tabla virtual en donde se guarda temporalmente el detalle de la factura, dicha tabla virtual debe ser declarada y creada previamente; se recomienda declararla como variable global del formulario, y crearla en el procedimiento From_Load del formulario. Posteriormente establecer rsDetalleVista como origen de datos para DatgrdDetalle.

Para mayor información de los controles, y de sus procedimientos, aquí expuestos, referirse a la ayuda en línea del lenguaje Visual Basic.

Conclusión

Se cumplieron los objetivos planteados para el desarrollo de un programa administrativo, que puede ser utilizado por empresas cuyo giro sea el ramo textil manufacturero, así como de la industria de la confección del vestido y del calzado.

Aunque en el mercado de software existe una gran cantidad de programas de administración comercial, estos no siempre suelen cubrir al 100% las necesidades de una empresa en particular, pues están muy generalizados.

Esto hace que los dueños de otras empresas acudan a personas especializadas en el desarrollo de software, para que les implementen un programa que cubran los requerimientos para poder administrar sus negocios.

El desarrollar un programa de computadora no solo se centra en una serie de entrevistas con los solicitantes, y después sentarse frente a una computadora y desarrollar un programa, sino que esto va más allá.

Antes de todo esto, se deben tener conocimientos de lo que son las bases de datos, y de cómo se deben implementar para evitar redundancias en la información que se almacenará en ellas.

También se debe tener conocimientos de ingeniería de software, para tomar la decisión correcta acerca del método a utilizar, por muy tedioso que parezca el desarrollar diagramas, pues esto ahorra muchos dolores de cabeza al realizar posteriores modificaciones al programa.

Perspectivas

A este programa se le puede integrar:

- Código de barras
- Contabilidad de costos
  - Nómina
  - Gasolina
  - Luz
  - Transporte de la materia prima
  - Etc.

- Declaración de I.V.A. de proveedores terceros
- Censores de consumo de materia prima
- Algunas opciones se ejecuten al tacto
- Anualizar el programa

Además no todo está escrito en materia de computación, como muchos piensan.

Bibliografía

1) Del Río González
Introducción a la Contabilidad de Costos.
Editorial: Limusa.

2) Curtis Smith y Michael Amundsen
Aprendiendo Programación de Bases de datos con Visual Basic.
Editorial: Prentice Hall.

3) I. T. Hawryszkiewycz
Análisis y Diseño de Bases de Datos.
Editorial: Noriega Editores.

M.C. Alma Delia Ambrosio Vázquez
Diseño de Bases de Datos.Relacionales (curso taller)
F.C.C., B.U.A.P.

4) Ian Sommerville
Ingeniería de Software.
Editorial: Addison – Wesley

5) Microsoft
Ayuda en línea del lenguaje Visual Basic 6.0 y del programa Access