21

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERÍA

"SISTEMA DE CONTROL DE ALMACEN PARA UNA DEPENDENCIA DE LA U.N.A.M."

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

JORGE GALVEZ CASTILLO

VICTOR MANUEL REYES SAENZ

ASESOR: M.I. ADOLFO MILLAN NAJERA

MEXICO, D.F. 2001

Agradecimientos

A mis padres.

Gracias papá y mamá por haberme dado el regalo de mi educación, pero sobre todo por el cariño, amor, comprensión y paciencia que siempre me han tenido. Los quiero mucho.

A mis hermanos.

Gracias Norma y Ramón por el gran cariño que siempre nos hemos tenido y por el gran apoyo que he recibido de ustedes, tanto en las buenas como en las malas.

A mis tíos y primos.

Gracias por estar siempre al pendiente de mi y motivarme a seguir adelante.

A mis entrañables amigos de la Facultad de Ingeniería.

A ustedes que los considero como mis hermanos, ya que durante la carrera vivimos grandes momentos que nunca olvidaré, en especial a Francisco C. y a Susana.

A mis amigos del trabajo.

Que siempre me están motivando a seguir adelante en todas mis metas, especialmente a Germán, Adriana, Julio, Francisco O., Rocío, Alejandra y Nelly.

A la familia Reyes Sáenz.

Que durante todo este tiempo de la realización de la tesis me adoptaron como un miembro mas de su familia, así como por la paciencia que nos tuvieron, gracias.

A Víctor.

Por enseñarme el gran valor de una amistad y en particular por el apoyo que siempre he recibido de ti, gracias "Pollo".

A la Universidad.

Por haberme formado profesionalmente y darme los conocimientos que me ayudarán en mi vida. No te defraudaré.

Jorge Gálvez Castillo

AGRADECIMIENTO ESPECIAL A:

Ing. Adolfo Millán Najera

Gracias por su confianza, paciencia y tiempo que nos brindo durante todo el desarrollo de esta tesis. Su presencia fue de gran valor para el termino de este sueño que hoy ya es una realidad. Esperamos contar con su experiencia en la facultad durante muchos años más.

Atentamente sus alumnos:

Jorge Gálvez Castillo
Víctor Manuel Reyes Sáenz

Agradecimientos

Dedico esta tesis a todas las personas que me apoyaron durante el desarrollo de mi carrera profesional, en especial:

A ti dios

Por estar a mi lado cuando te necesito y poder ayudarme a salir adelante. Gracias.

A ti Mamá

Por darme la vida, enseñarme a valorar las cosas y el amor que me has dado durante todos estos años. Te quiero. Gracias. Por fin la termine Mamá!!!!!

A ti Papá

Por todo los sacrificios que has hecho por darnos una carrera a mi y a mis hermanos y por todo el apoyo que he tenido en ti. Te quiero
Gracias por esta educación que me has dado.

A mis hermanos

Daniel "danielo", Quique "flaco" y Hugo "hugose", que sin su compañía nunca hubiera logrado llegar hasta aquí, gracias por todo lo que hemos compartido juntos. Gracias "los hermanos".

A mis familiares

Abuelitos, Tíos, primos, a todos muchas gracias. Gracias por mostrame el valor de la unidad familiar que tenemos.

A mis amigos

Freddy, Patolin, Ana, Manolo, Fidencio, Toñote, Miriam, Verenice, Vero, Yuriria y a todos los que en este momento no recuerdo, gracias por brindarme su amistad, por su apoyo en las buenas y en las malas, gracias.

A todos los que yo no estan con nosotros

Nunca los olvidare

A ti Maribel

Por todo este tiempo que hemos estado juntos, este es un gran logro "chip".

A ti Jorge

Gracias "chavo", por esta amistad que ahora tenemos pudimos sacar adelante este proyecto de muchos que esperamos realizar. Gracias por tu paciencia.

A ti Universidad

Gracias por tu sabiduría los cuales me brindaste para poder llegar a ser ingenierio.

A la Fam. Gálvez Castillo

Gracias a por su apoyo y paciencia para la realización de esta tesis.

Victor Manuel Reyes Sáenz

Prólogo

En la actualidad, los avances tecnológicos han ido superando las expectativas de muchos de nosotros, debido a la gran velocidad con que éstos han surgido, beneficiando a múltiples áreas de interés para el hombre, tales como la medicina, las telecomunicaciones y en particular la informática, que es el área que principalmente nos interesa mencionar para el desarrollo de nuestro tema. Este crecimiento tan acelerado en la tecnología, ha provocado que ciertas herramientas y procesos hayan quedado limitados por no ser los más óptimos, y en el caso de la computación podemos mencionar los sistemas informáticos que utilizan múltiples organizaciones del sector público y privado, ya que los lenguajes de programación con los que fueron desarrollados son obsoletos a comparación con los de hoy en día.

Por otra parte, es preciso mencionar que en la actualidad, para muchas organizaciones, los sistemas de información basados en las computadoras son el corazón de las actividades cotidianas y objeto de gran consideración en la toma de decisiones. Sin ayuda automatizada, las dependencias gubernamentales tendrían que hacer un alto ante el volumen de trabajo que abrumaría a sus administradores y empleados. Y esto, en la U.N.A.M., no es la excepción, ya que los diferentes institutos y dependencias, cuentan con sus propios sistemas que les permiten desarrollar múltiples tareas en lo que se refiere al procesamiento de la información que ellos requieren.

Durante la realización de nuestro servicio social, en la Dirección General de Normatividad y Sistemas Administrativos, hemos tenido la oportunidad de visitar diferentes dependencias de la U.N.A.M., y se observó la complejidad de las situaciones implicadas al considerar una posible actualización al sistema. Tales dificultades pueden ser abordadas de manera efectiva con una herramienta actual, que proporcione un ambiente gráfico, eficiente y completo, y con la cual se pueda construir soluciones potentes con una inversión razonable de dinero, tiempo y esfuerzo, atendiendo a la posibilidad de aplicar lo aprendido en las labores de servicio social de forma que redunden en el beneficio de la Universidad.

Por todo lo anterior, el objetivo principal del presente trabajo de tesis es dar una exposición detallada de cada una de las etapas de desarrollo, utilizando un lenguaje de programación orientado a objetos, así como una base de datos relacional, los cuales permitirán un manejo de la información de una manera más ágil, eficiente y segura. Dichas etapas son: el análisis, eldiseño, la codificación y la implantación, las cuales serán explicadas y desarrolladas en nuestro tema de tesis de la siguiente manera.

En el primer capítulo, se mencionarán los conceptos generales relacionados con el desarrollo de sistemas, desde el ciclo vital de desarrollo hasta el manejo de librerías exclusivas del lenguaje de programación Power Builder. En el capítulo dos se hablará sobre el análisis y diseño en que se desarrolló el sistema, aplicando la teoría que se vio con anterioridad. En el tercer capítulo se verá la codificación del sistema en sus diferentes módulos, aplicados en el lenguaje de programación Power Builder, así como el lenguaje nativo de la base de datos que se utilizó (SQL Anywhere Sybase). En el capítulo cuatro se mencionarán los por menores de las pruebas, así como también la implantación y puesta en marcha del sistema de control de almacén.

El resultado esperado del presente trabajo es dar a los usuarios finales el beneficio completo que se obtenga al implantar, en las varias dependencias de la U.N.A.M., el sistema de control de almacén del cual se detalla aquí el desarrollo. Y que, de aprobarse, abrirá una amplia posibilidad de mantener actualizados los sistemas, alejando la amenaza de la obsolescencia que ya empieza a pasar sobre los programas que están actualmente en uso.

Introducción

Conforme la tecnología avanza la necesidad de manejar la información adecuadamente encuentra cada vez mejores soluciones, es por eso que el estudio del óptimo manejo de la información debe mantenerse en constante revisión, a la par de los avances de la tecnología. La importancia del uso de sistemas que ofrecen la posibilidad de adaptarlos a las particularidades especificas de cada usuario se hace patente en las empresas, negocios, escuelas, etc., con lo que resulta necesario, con cada nuevo sistema implantado, contar con más gente capacitada para el análisis, diseño y desarrollo de tales sistemas.

Dentro de la universidad mas grande de América Latina, la Universidad Nacional Autónoma de México campus C.U., a través de los años que llevade operar se ha llegado a un consenso entre los usuarios de esos sistemas, de que no satisfacen sus necesidades de manejo de la información como se podria esperar, pues no se cuenta con los medios para que los usuarios encuentren atendidas sus necesidades de la actualización o adaptación de sus sistemas, mismos que no se rescatan de la obsolescencia que ya es visible en ellos con problemas como los que siguen:

- El lento procesamiento de la información.
- Un alto consumo en cuanto a recursos de la computadora.
- La captura de información, por parte de los usuarios, es repetitiva, tediosa y en gran volumen.
- Los sistemas no reconocen impresoras de otro tipo diferente a las de matriz de puntos para la emisión de reportes.
- Solo reconoce la impresora conectada al puerto local, sin que pueda reconocer a las impresoras remotas que estén en la red.
- La interfaz con el usuario no esta actualizada para usarse en los ambientes gráficos más populares, por lo que el usuario debe reiniciar la computadora cada vez que ha de usar uno u otro, de tipo carácter o gráfico.

Cabe mencionar que las herramientas que sirvieron para programar los sistemas que actualmente se encuentran en funcionamiento no están al nivel de los adelantos tecnológicos más recientes en el ámbito del desarrollo de sistemas. Aunque sigue siendo una herramienta efectiva para el desarrollo de sistemas en ambiente de caracteres, el lenguaje de programación Clipper (ver. 5.2) carece de un ambiente de desarrollo de sistemas suficientemente amigable para realizar la programación de los sistemas de forma que no resulte una tarea tediosa y poco gráfica. Esto ocasiona que se haga cada vez más escasa la posibilidad de contar con personal calificado o por lo menos interesado, en el uso cabal de las características propias de Clipper como herramienta de desarrollo, por lo cual, el mantenimiento y actualización de los sistemas no se realiza con la rapidez que se necesita.

Del servicio social, por parte de los que escriben, surge el deseo de apoyar a las dependencias de la U.N.A.M., con el desarrollo de una versión actualizada, gráfica, cómoda, eficiente y mantenible del sistema de almacén.

Este sistema llevará el control de la siguiente manera:

Dentro de la U.N.A.M., existe un departamento de distribución y almacenaje de artículos, el cual debe llevar su control cuando éstos son ingresados, almacenados y distribuidos a las diferentes áreas asignadas. La distribución se realiza por medio de solicitudes de material que se hacen llegar al almacén por el personal que cada departamento, dentro de las dependencias, tiene asignado para tal efecto. Mientras que el almacenaje se realiza haciendo llegar la solicitud de material a diferentes distribuidores en función de los volúmenes de demanda de cada artículo. El proceso descrito requiere un manejo confiable de las partidas, artículos, empleados y departamentos, así como el control preciso de las existencias de los artículos en el almacén, con transacciones que reflejen los movimientos de inventario inicial, entradas y salidas y cierre de mes.

Departamento de almacén

Uno de los departamentos más importantes, dentro de la universidad, es el almacén, considerando esta importancia en la justa dimensión del valor de los grandes volúmenes de artículos que se manejan en las diferentesinstancias de este departamento. Las escalas de valores de los artículos que se manejan van desde unos cuantos centavos, en el caso de los clips y grapas, hasta varios miles de pesos, en el caso de sofisticados equipos electrónicos de los

laboratorios, en cuanto a valor monetario, y desde algunos gramos, en el caso de reactivos químicos, hasta miles de unidades como toneladas de arena para construcciones, en cuanto a las cantidades de artículos que es preciso controlar, sobre todo por el valor monetario que representan. Es por esto que, para lograr un control efectivo de los movimientos de entrada y salida del material, las herramientas, y demás artículos, todas las adquisiciones deben canalizarse por este departamento. Por otra parte, cabe mencionar que el control de facturas y de resguardos (inventario o activo fijo) para cada empleado se manejan en otros departamentos.

Los gastos dentro de una facultad tienen un flujo especial, el cual varía de acuerdo al semestre o año académico, mientras que en cualquier otra dependencia el gasto sólo se efectúa una vez al año (por lo regular en el mes de noviembre).

Al observar el ambiente que regularmente se presenta en un almacén, resulta evidente que, en el caso ideal, los movimientos del almacén se sintetizan en un par de eventos discretos, que son entradas (compras o devoluciones al almacén) y salidas (consumos o devoluciones al proveedor) de artículos. Sin embargo, al extender el alcance de la observación, el sistema de control de almacén debe incluir las funciones administrativas de registro y control de un presupuesto y las partidas que lo conforman, lo cual cambia la perspectiva de un enfoque meramente ideal de un control de movimientos de artículos, para incluir funciones de análisis financiero, por ejemplo, los proveedores con los que se adquieran los materiales que cada dependencia requiere, periodicidad con que se hacen llegar los requerimientos, el catálogo del inventario de acuerdo al método de conteo utilizado, etc.

La descripción del sistema que se hace corresponde al funcionamiento del almacén en la facultad de Ciencias Políticas y Sociales, dado que es esa instancia del Departamento de almacén con la que se tuvo mayor contacto y la cual servirá para poder iniciar la sistematización de esta área en toda la universidad.

Adquisición de mercancía (Entradas)

La adquisición de materiales, herramientas, equipos, etc., se efectúa por dos vías principales, la compra directa y la proveeduría, mismas que a continuación se describen.

Compra Directa

La compra directa se refiere, dentro de la U.N.A.M., al hecho de que se paga con cheque a un distribuidor, siempre que la compra no exceda los 100,001 pesos porque, de lo contrario, esta compra estará condicionada a hacerse por medio de proveeduría, la cual puede o no surtir este producto. Una de las ventajas que puede representar una compra directa, es el tiempo de entrega, la cual puede llegar a ser desde dos horas hasta cinco días, algo que pone en demasiada desventaja a proveeduría. Además, la calidad del producto, al ser menor la cantidad, se verifica y se tienen productos sumamente valiosos en cuanto a la calidad, pero por supuesto, esto tiene un costo. Para este tipo de compra podríamos citar dos inconvenientes los cuales son: que el precio se eleva a comparación de proveeduría y la cantidad de distribuidores también, ya que se tendría, por citar algunos ejemplos, un distribuidor para la papelería, otro para material de limpieza, otro para refacciones automotrices, otro para muebles, etc., es decir, este catálogo crecería de acuerdo a los productos que se pidan.

Proveeduría (U.N.A.M.)

Proveeduría, no es otra cosa más que otro distribuidor, la diferencia sería que además de vender mucho más barato a ésta no se le paga con cheque, si no que se va descontando del presupuesto de la dependencia y asignándoselo a proveeduría de la universidad. Para que esta instancia pueda vender mucho más barato y pueda distribuir mercancía a todas las dependencias de la U.N.A.M., ésta compra a distribuidores en enormes cantidades (mayoreo) por lo que el precio se disminuye notablemente (no es lo mismo comprar 5,000 millares de hojas, aunque es una cantidad considerable, que comprar 700,000 millares, es por eso la diferencia en cuanto al precio). Aunque por otro lado, se tiene el inconveniente que esta instancia no verifica la calidad de los productos y muchas veces por el hecho de comprar en

grandes cantidades, por un mal cálculo en cuanto a cantidad, puede dañarse la mercancía (un ejemplo muy claro de ésto, son los plumones o marcadores; que muchas veces se llegan a secar por estar almacenados por mucho tiempo o de ser de pésima calidad).

Cuando el valor de un producto, que alguna dependencia desea adquirir, excede los 100,001 pesos, se hace una solicitud de compra a proveeduría, buscando la forma de que ésta consiga, junto con pedidos similares de otras dependencias, un precio más económico. Proveeduría por su parte, si no consigue un precio menor, se lo comunica a la dependencia y autoriza a ésta para que la misma realice la compra. Además, el tiempo de entrega, del equipo o material solicitado, aumenta de acuerdo a la diversidad de materiales, por lo regular se distribuye una o dos veces por mes y muy rara es la ocasión en que esto se realice tres o más veces.

Inventario inicial

Al inicio de cada año se tiene que hacer un conteo de la mercancía que se encuentra en existencia, a esta cantidad o informe se le conoce como inventario inicial. De igual manera, mes a mes se efectúa un conteo de mercancía donde se considera que las compras o adquisiciones efectuadas por mes (entradas) menos los consumos (salidas), darán como resultado un inventario inicial mensual. Cabe aclarar que es muy diferente el inventario inicial que se efectúa a principios de año. Y por otra parte, es preciso mencionar que al sumar el precio de todos los artículos nos dará una cantidad que representa de igual forma el presupuesto inicial, y conforme avance el mes éste crecerá y/o disminuirá paulatinamente.

Distribución del material

La forma en que la mercancía es distribuida dentro de la facultad es más simple, y se realiza de la siguiente forma: una persona lleva al almacén una lista con el material que ocupará durante el mes, la cual elaboró el jefe del departamento del que proviene, y que, según un estudio previo de éste, es el material a ocupar durante un mes. En el transcurso de este mes la lista puede ser ampliada por artículos que no se contemplaron en un inicio y pueden ser agregados a la lista para ser solicitados.

Al finalizar el periodo o mes, se hace un balance de la cantidad de mercancía que fue solicitada por departamento, por empleado y por partida, esto se realiza en toda la dependencia, con el fin de calcular el presupuesto ocupado por cada departamento y partida.

1

Conceptos Generales.

Sobre el entorno que hay en la construcción de un sistema, que va desde el análisis, diseño y desarrollo, existen conceptos que se aplican directa e indirectamente, los cuales nos permitirán crear un sistema sólido y robusto basados en la documentación tecnológica mas reciente, por lo cual el siguiente capítulo muestra, en una forma breve, los conceptos con los cuales se involucró el desarrollo del sistema de almacén.

1.1 Introducción al ciclo vital del desarrollo de un sistema

El ciclo de vida del desarrollo de un sistema es un conjunto de pasos que un programador recorre para su creación y que, tradicionalmente, incluye las siguientes etapas:

1. **Análisis de los requerimientos.** En la fase de análisis se trabaja con los usuarios para conocer y especificar los requerimientos de su sistema. Durante esta etapa, se pueden desarrollar prototipos de la interfaz del usuario, así como completar modelos lógicos de datos.

2. **Arquitectura y diseño.** En la fase de arquitectura y diseño se desarrolla un proyecto técnicamente válido para aplicar los requerimientos del sistema.

3. **Desarrollo.** En la fase de desarrollo es donde se codifica realmente la aplicación utilizando el diseño y los requerimientos ya definidos previamente.

4. **Garantía de calidad.** En la fase de garantía de calidad se asegura que la aplicación cumple con todos los requerimientos y no contiene errores.

5. **Instalación.** En esta fase se instala la aplicación en los puestos de trabajo del usuario.

6. **Seguimiento.** Durante esta fase de seguimiento se verifica como funciona el sistema con los operadores. Es una fase importante de cara a definir posibles nuevos requerimientos para ediciones posteriores de la aplicación.

1.2 Conocimiento de los requerimientos del Usuario

El análisis de los requerimientos es una fase muy importante en el ciclo de desarrollo de un sistema. Antes de empezar el diseño hay que comprender bien los requerimientos, porque aunque parece una cosa intuitiva, los programadores a veces empiezan con la codificación antes de que estén definidos.

En los requerimientos deben estar identificadas todas las reglas comerciales importantes, entradas y salidas (suministros del sistema) y frecuentemente incluir interfaces del usuario o prototipos de desplazamiento interno. Los suministros del sistema pueden consistir en instrucciones sobre lo que un sistema debe hacer, por ejemplo: "Quiero que el sistema sea capaz de seguir un componente desde el pedido hasta la entrega", así como en ejemplos de documentos de entrada o salida, como facturas o propuestas de pedido, por mencionar algunos ejemplos.

Los documentos deben expresar lo que el sistema ha de hacer, no cómo se consigue. Durante la fase del análisis hay que completar tanto el modelo de datos conceptual como el lógico. Antes de empezar el diseño hay que documentar un protocolo con la comprensión de los requerimientos del usuario.

En la fase de diseño se define cómo se consiguen esos requerimientos, se selecciona la tecnología y se revisan los modelos de datos lógicos bajo consideraciones de rendimiento para desarrollar un modelo de datos físico.

Antes de empezar cualquier desarrollo, con algún lenguaje de programación, ha de haberse completado el diseño y se ha de disponer de una base de datos física.

1.3 Introducción al análisis

El análisis se encarga principalmente de conocer y entender las necesidades comerciales del sistema. En la fase de análisis del proyecto es prematuro pensar en el entorno de software y hardware que se ha de usar para la solución (Lenguajes de programación, sistemas operativos y tipos de computadoras personales), sobre estos elementos se decidirá en la fase de diseño.

1.4 Programación Orientada a Objetos

Actualmente una de las áreas más candentes en la industria y en el ámbito académico es la orientación a objetos. La orientación a objetos promete mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software: la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas.

Un lenguaje orientado a objetos ataca estos problemas. Tiene tres características básicas: debe estar basado en objetos, basado en clases y capaz de tener herencia de clases. Muchos lenguajes cumplen uno o dos de estos puntos; muchos menos cumplen los tres. La barrera más difícil de sortear es usualmente la herencia.

El concepto de programación orientada a objetos (POO) no es nuevo, lenguajes clásicos como SmallTalk se basan en ella. Dado que la POO. Se basa en la idea natural de la existencia de un mundo lleno de objetos y que la resolución del problema se realiza en términos de objetos, un lenguaje se dice que está basado en objetos si soporta objetos como una característica fundamental del mismo.

El elemento fundamental de la POO es, como su nombre lo indica, el objeto. Podemos definir un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización.

1.4.1 Conceptos Generales de POO

1.4.1.1 Objeto

Las personas tienen una idea clara de lo que es un objeto: conceptos adquiridos que nos permiten sentir y razonar acerca de las cosas del mundo. Un objeto podría ser real o abstracto, por ejemplo una organización, una factura, una figura en un dibujo, una pantalla de usuario, un avión, un vuelo de avión, etc.

En el análisis y diseño orientados a objetos (OO), interesa el comportamiento del objeto. Si se construye software, los módulos de software OO se basan en los tipos de objetos. El software que implanta el objeto contiene estructuras de datos y operaciones que expresan dicho comportamiento. Las operaciones se codifican como métodos. La representación en software OO del objeto es entonces una colección de tipos de datos y objetos.

Entonces, dentro del software orientado a objeto, un objeto es cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos.

Un objeto puede estar compuesto por otros objetos. Estos últimos a su vez también pueden estar compuestos por otros objetos. Esta intrincada estructura es la que permite construir objetos muy complejos.

1.4.1.2 Tipo de Objeto

Los conceptos que poseemos se aplican a tipos determinados de objetos. Por ejemplo, empleado se aplica a los objetos que son personas empleadas por alguna organización. Algunas instancias de empleado podrían ser Juan Pérez, José Martínez, etc. En el análisis orientado a objetos, estos conceptos se llaman tipos de objetos; las instancias se llaman objetos.

Así, un tipo de objeto es una categoría de objeto, mientras que un objeto es una instancia de un tipo de objeto.

En el mundo de las bases de datos existen los tipos de entidad, como cliente o empleado. Existen muchas instancias de cada tipo de entidad (como Juan Pérez o José Martínez para el tipo de entidad empleado). Del mismo modo, en OO se define tipos de objetos e instancias de tipo de objeto.

Sin embargo, el término objeto tiene diferencias fundamentales con el término entidad, ya que la entidad sólo se refiere a los datos, mientras que objeto se refiere a los datos y a los métodos mediante los cuales se controlan a los propios datos. En OO, la estructura de datos y los métodos de cada tipo de objeto se manejan juntos. No se puede tener acceso o control de la estructura de datos excepto mediante los métodos que forman parte del tipo de objeto.

1.4.1.3 Métodos

Los métodos especifican la forma en que se controlan los datos de un objeto. Los métodos en un tipo de objeto sólo hacen referencia a la estructura de datos de ese tipo de objeto. No deben tener acceso directo a las estructuras de datos de otros objetos. Para utilizar la estructura de datos de otro objeto, deben enviar un mensaje a éste. El tipo de objeto empaca juntos los tipos de datos y su comportamiento.

Un objeto entonces es una cosa cuyas propiedades están representadas por tipos de datos y su comportamiento por métodos.

Un método asociado con el tipo de objeto factura podría ser aquel que calcule el total de la factura. Otro podría transmitir la factura a un cliente. Otro podría verificar de manera periódica si la factura ha sido pagada y, en caso contrario, añadir cierta tasa de interés.

1.4.1.4 Encapsulado

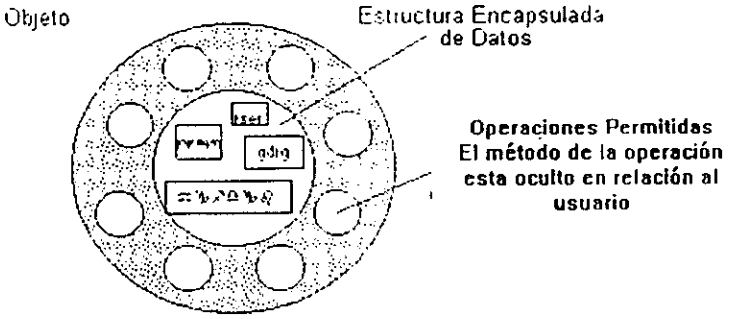
El empaque conjunto de datos y métodos se llama encapsulado. El objeto esconde sus datos de los demás objetos y permite el acceso a los datos mediante sus propios métodos. Esto recibe el nombre de ocultación de información. El encapsulamiento evita la corrupción de los datos de un objeto. Si todos los programas pudieran tener acceso a los datos de cualquier forma

que quisieran los usuarios, los datos se podrían corromper o utilizar de mala manera. El encapsulado protege los datos del uso arbitrario y no pretendido.

El encapsulado oculta los detalles de su implantación interna a los usuarios de un objeto. Los usuarios se dan cuenta de las operaciones que puede solicitar del objeto, pero desconocen los detalles de cómo se lleva a cabo la operación. Todos los detalles específicos de los datos del objeto y la codificación de sus operaciones están fuera del alcance del usuario.

Así, encapsulado es el resultado (o acto) de ocultar los detalles de implantación de un objeto respecto de su usuario.

El encapsulado, al separar el comportamiento del objeto de su implantación, permite la modificación de ésta sin que se tengan que modificar las aplicaciones que lo utilizan.



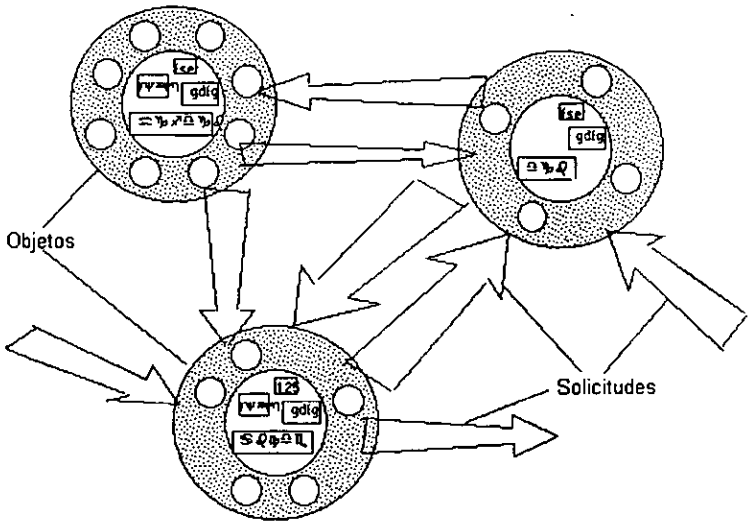
1.4.1.5 Mensajes

Para que un objeto haga algo, le enviamos una solicitud. Ésta hace que se produzca una operación. La operación ejecuta el método apropiado y, de manera opcional, produce una respuesta. El mensaje que constituye la solicitud contiene el nombre del objeto, el nombre de una operación y, a veces, un grupo de parámetros.

La programación orientada a objetos es una forma de diseño modular en la que con frecuencia el mundo se piensa en términos de objetos, operaciones, métodos y mensajes que se transfieren entre tales objetos. Un mensaje es una

solicitud para que se lleve a cabo la operación indicada y se produzca el resultado.

Los objetos pueden ser muy complejos, puesto que pueden contener muchos subobjetos, éstos a su vez pueden contener otros, etc. La personaque utilice el objeto no tiene que conocer su complejidad interna, sino la forma de comunicarse con él y la forma en que responde.



1.4.1.6 Clase

El término clase se refiere a la implantación en software de un tipo de objeto.

El tipo de objeto es una noción de concepto. Especifica una familia de objetos sin estipular la forma en que se implanten. Los tipos de objetos se especifican durante el análisis OO.

Así, una clase es una implantación de un tipo de objeto. Especifica una estructura de datos y los métodos operativos permisibles que se aplican a cada uno de sus objetos.

1.4.1.7 Herencia

Un tipo de objeto de alto nivel puede especializarse en tipos de objeto de bajo nivel. Un tipo de objeto puede tener subtipos. Por ejemplo, el tipo de objeto persona puede tener subtipos estudiante y empleado. A su vez, el tipo de objeto estudiante puede tener como subtipo estudiante de pregrado y estudiante de postgrado, mientras que empleado puede tener como subtipo a académico y administrativo. Existe, de este modo, una jerarquía de tipos, subtipos, subsubtipos, etc.

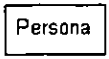
Una clase implanta el tipo de objeto. Una subclase hereda propiedades de su clase padre; una sub-subclase hereda propiedades de las subclases; etc. Una subclase puede heredar la estructura de datos y los métodos, o algunos de los métodos, de su superclase. También tiene sus métodos e incluso tipos de datos propios.

1.4.2 Análisis de la Estructura de Objetos

El análisis de la estructura de objetos (AEO) define las categorías delos objetos que percibimos y las formas en que los asociamos.

1.4.2.1 Objetos y Tipos de Objetos.

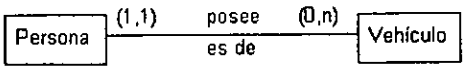
En el análisis se trata de identificar los tipos de objeto más que los objetos individuales en un sistema. Los tipos de objetos se definen sobre la base de la comprensión del analista de nuestro mundo. Un objeto puede categorizarse de variadas formas.



Representación para Tipo de Objeto (Persona).

1.4.2.2 Asociaciones de Objetos.

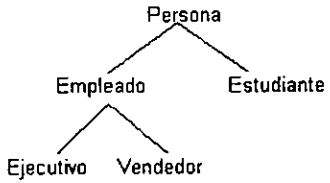
Es importante modelar la forma como los objetos se asocian entre sí. Además es necesario identificar el significado de la asociación y la cantidad de objetos con los que un objeto dado puede y debe asociarse (cardinalidad).



Representación para la Asociación entre dos Tipos de Objetos. Un objeto del tipo persona posee cero o muchos objetos del tipo vehículo. Un objeto del tipo vehículo es de un y sólo un objeto del tipo persona.

1.4.2.3 Jerarquías de Generalización.

Una de las vías de sentido común por las que el hombre organiza su volumen de conocimiento es el de las jerarquías, dé lo más general a lo más específico.

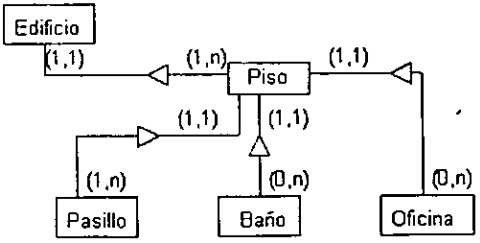


Representación de una Jerarquía de generalización, para el tipo de objeto Persona.

En las jerarquías se habla de subtipo o especialización de un supertipo o generalización. En el caso anterior, persona es el supertipo para Empleado y Estudiante, que son sus subtipos. Por otra parte, Empleado es el supertipo para los subtipos Ejecutivo y Vendedor. Los subtipos (niveles inferiores de la jerarquía) heredan las características de sus supertipos, además, cada instancia de un tipo de objeto lo es también de sus supertipos.

1.4.2.4 Jerarquías Compuestas.

Un objeto se denomina complejo si está formado por otros. Las jerarquías compuestas permiten realizar agregaciones de objetos.



Representación de una Jerarquía Compuesta.

Un objeto del tipo edificio se compone de a lo menos un objeto del tipo piso. A su vez un objeto del tipo piso se compone de a lo menos un objeto del tipo pasillo, podría tener varios (o ninguno) objetos del tipo baño y oficina.

1.4.2.5 Diagramas de relación entre los objetos.

Los tipos de objetos están relacionados con otros tipos de objeto. Por ejemplo, un empleado trabaja en una sucursal, o un cliente realiza un pedido de varios productos.

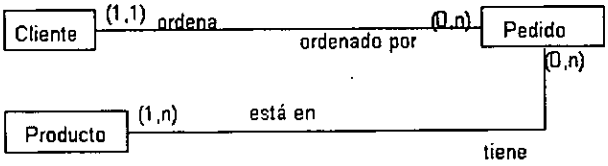


Diagrama de Relación entre objetos.

Un objeto del tipo cliente puede ordenar muchos objetos del tipo pedido, y un objeto del tipo pedido es ordenado por un y sólo un objeto del tipo cliente. Un objeto del tipo producto está en muchos o ningún objeto del tipo pedido, mientras que un objeto del tipo pedido tiene al menos un objeto del tipo producto.

1.4.2.6 Esquemas de Objetos.

La comprensión de un modelo suele ser más sencilla si los tipos de objetos y relaciones se presentan mediante un diagrama de relación entre objetos; los supertipos y subtipos se presentan en un diagrama de jerarquías de generalización y las estructuras compuestas en un diagrama compuesto. Sin embargo, para los usuarios más sofisticados puede ser útil presentarlo todo en un mismo diagrama, el que se denomina esquema de objetos.

1.4.3 Análisis del Comportamiento de Objetos.

En el análisis del comportamiento de objetos (ACO) realizamos esquemas de eventos que muestran eventos, la secuencia en que ocurren y cómo los eventos cambian el estado de los objetos.

1.4.3.1 Estados de un Objeto.

Un objeto puede existir en varios estados. Por ejemplo, un objeto reservación aérea puede ser una instancia de alguno de los siguientes tipos de objeto:

- Reservación solicitada,
- Reservación en lista de espera,
- Reservación confirmada,
- Reservación cancelada,
- Reservación satisfecha,
- Reservación archivada.

Tales tipos de objetos suelen percibirse como estados posibles del ciclo vital de un objeto. Sin embargo, un objeto puede tener una gran variedad de perspectivas de ciclos vitales. Por ejemplo, el mismo objeto reservación aérea también puede tener los siguientes estados relacionados con el pago:

- Reservación no liquidada,
- Reservación con un pago de depósito,
- Reservación totalmente pagada,
- Reservación reembolsada.

Así, el estado de un objeto es la colección de asociaciones que tiene un objeto.

1.4.3.2 Eventos.

El mundo está lleno de eventos: una coneja tiene conejitos, llega el pesado del vecino en forma inesperada, un cliente solicita un préstamo, el servidor se cae, se termina la tarea, etc.

En el análisis orientado a objetos el mundo se describe en términos de los objetos y sus estados, así como los eventos que modifican esos estados.

Un evento produce un cambio en el estado de un objeto. Los eventos sirven como indicadores de los instantes en que ocurren los cambios de estado.

Para saber de los cambios y reaccionar adecuadamente ante ellos, debemos entender y modelar los eventos.

1.4.3.3 Tipos de Eventos

El analista no necesita conocer cada evento que ocurra en una organización, sólo los tipos de eventos.

Por ejemplo, el tipo de evento reservación en lista de espera confirmada es la colección de eventos donde un objeto cambia de una reservación en lista de espera a una reservación confirmada.

Los tipos de eventos indican los cambios sencillos en el estado de un objeto; por ejemplo, cuando se deposita dinero en una cuenta bancaria o se actualiza el sueldo de un trabajador. Básicamente, los tipos de eventos describen las siguientes formas de cambios de estado:

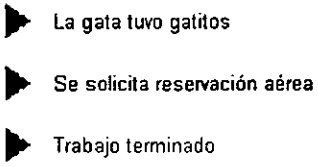
- Un objeto se crea.
- Un objeto se termina.
- Un objeto se clasifica como una instancia de un tipo de objeto.
- Un objeto se desclasifica como una instancia de un tipo de objeto.
- Un objeto cambia de clasificación.
- Un atributo de un objeto se cambia.

Los objetos pueden asociar un objeto con otro. Por ejemplo, en la mayoría de las organizaciones, cuando un objeto se clasifica como empleado, debe estar asociado con un departamento. Un evento clasificará al objeto como empleado. Otro evento creará una asociación entre el objeto empleado y un objeto departamento (las asociaciones son objetos como los demás).

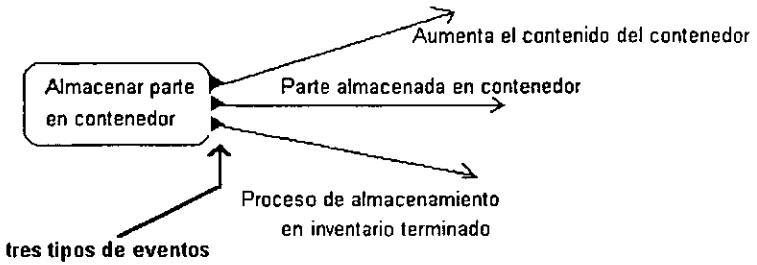
Algunos eventos requieren que antes ocurran otros. Por ejemplo, antes de cerrar un departamento, todos los empleados deben ser asignados a otra parte, las oficinas que ocupaban deben tener otro uso, etc.

Algunas veces, un evento puede provocar la reacción en cadena de otros eventos. Por ejemplo, el cambio de circuito a las conexiones de un avión, puede exigir cambios a otros varios objetos.

Una operación hace que los eventos ocurran. Dibujamos la operación como un cuadro con esquinas redondeadas, puesto que los eventos indican los puntos en el tiempo en que se da el cambio de estado de un objeto. Los tipos de eventos se representan como triángulos negros llenos, generalmente unidos a la caja de operación.



Según el área que se modele, puede ocurrir más de un evento al terminar una operación, y cada uno de éstos puede activar operaciones independientes.



1.4.3.4 El Ciclo Vital de un Objeto

La mayoría de los objetos tienen un ciclo vital en el que una sucesión de eventos puede ocurrirles y cada uno de éstos modifica su estado.

En este análisis, se dibuja un diagrama que muestre el ciclo vital de un objeto, incluyendo los estados posibles de los objetos, además de los cambios de estado permisibles. Éste se denomina diagrama de reja.

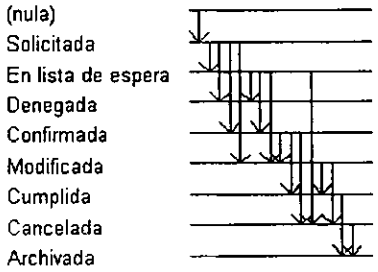
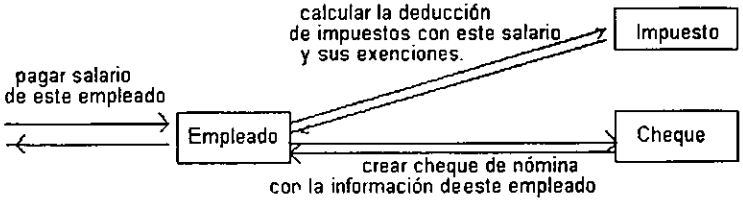


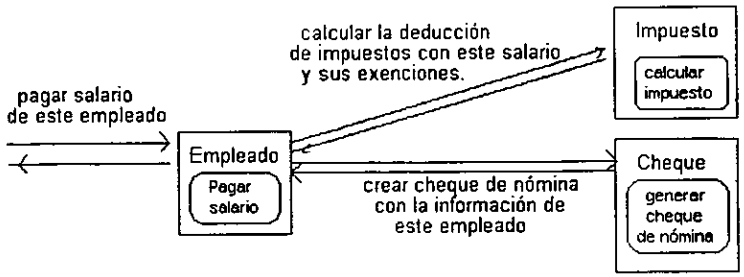
Diagrama de reja que muestra los estados posibles de un objeto reservación aérea. Las líneas horizontales representan estados y las verticales muestran las transiciones entre estados.

1.4.3.5 Interacciones entre tipos de objetos

La mayoría de los procesos requieren la interacción de varios objetos.



En esta otra figura, se desarrolla el diagrama anterior para mostrar las operaciones necesarias.

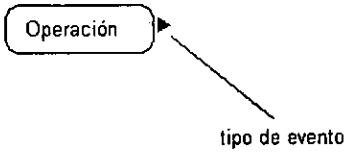


1.4.3.6 Operaciones.

En el análisis OO, una operación se refiere a una unidad de procesamiento que puede ser solicitada. El procedimiento se implanta mediante un método. El método es la especificación de cómo llevar a cabo la operación.

A nivel de programa, el método es el código que implanta la operación.

Las operaciones se invocan. Una operación invocada es una instancia de una operación. Una operación puede o no cambiar el estado de un objeto, si lo cambiara ocurriría un evento.



1.4.3.7 Fuentes externas de eventos

Los eventos son cambios de estado que un sistema debe conocer y reaccionar ante ellos de algún modo. Muchas de las operaciones que producen estos eventos son externas al sistema.

1.4.3.8 Reglas de activación

Cuando ocurre un evento, lo normal es que el cambio de estado active el llamado a una o más operaciones. Por ejemplo, si se retiran bienes de un

almacén y la cantidad baja de cierto nivel, ello puede activar una operación para volver a realizar un pedido.

Las reglas de activación definen la relación entre la causa y el efecto. Siempre que ocurra un evento de cierto tipo, la regla de activación invoca a una operación ya definida.

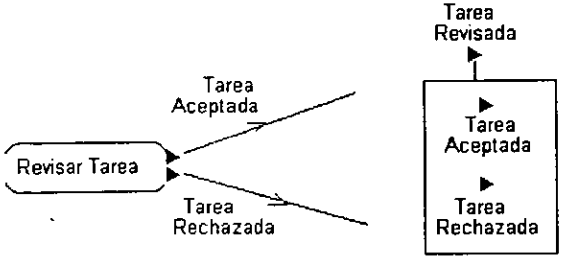
Un tipo de evento puede tener varias reglas de activación, cada una de las cuales invoca a su operación en paralelo. Las operaciones paralelas pueden producir diferentes cambios de estado en forma simultánea.

1.4.3.9 Condiciones de Control

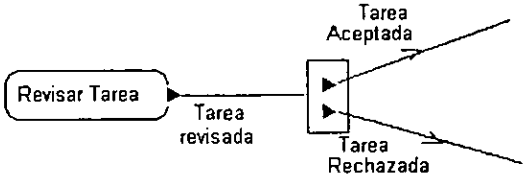
Una operación puede ser invocada por una o varias reglas de activación. Sin embargo, antes de invocar de hecho a la operación se puede verificar una condición de control. Si el resultado de evaluación de la condición es verdadera se invoca su operación, en otro caso no.

Las condiciones de control también pueden actuar como puntos de sincronización para el procesamiento en paralelo, pues garantizan que un conjunto de eventos esté completo antes de proceder con una operación.

1.5.3.10 Subtipos y Supertipos de Eventos.



Los eventos pueden dividirse en subtipos mediante diagramas independientes; o bien, es posible expresar la misma información en un diagrama ampliado.



La operación revisar tarea produce dos eventos: tarea aceptada o tarea rechazada. Sólo se puede dar uno de estos tipos de evento al revisar una tarea.

Aquí tarea revisada es un supertipo de tarea aceptada y tarea rechazada, que son los subtipos. Siempre se entiende que existe una relación de exclusividad entre los subtipos.

Las particiones de eventos no son operaciones independientes que coordinan las condiciones de bifurcación para las formas de activación ajenas, sino que indican los objetivos y distintos subobjetivos de las operaciones a las que están asociadas.

1.4.4 Diseño de la estructura y comportamiento de un objeto.

En el diseño de la estructura y comportamiento de objetos se identifican los componentes siguientes:

- Clases que se implantarán. Los tipos de objetos en el AEO serán la guía en esta decisión.
- Estructuras de Datos que utilizará cada clase. Se puede hacer un diagrama para representar la estructura de datos.
- Operaciones que ofrecerá cada clase y cuáles serán sus métodos. Se enumeran las operaciones y se especifican los métodos.
- Forma de Implantación de la herencia de clases y efecto sobre las especificaciones de los datos y operaciones.
- Identificación de variantes de clases ("igual que, excepto...").

1.4.4.1 Clase.

En el análisis de estructura de objetos, se identificaron tipos de objetos; en el diseño de estructura de objetos nos centramos en la implantación de esos tipos de objetos.

Clase es la implantación de un tipo de objeto. Especifica la estructura de datos y los métodos operativos permitidos que se aplican a cada uno de sus objetos.

La clase especifica la estructura de datos de cada uno de sus objetos y las operaciones que se utilizan para tener acceso a los objetos. La especificación de cómo se llevan a cabo las funciones de una clase se llama método. Los objetos se pueden utilizar exclusivamente con métodos específicos.

Una instancia de una clase, almacena sus datos dentro de él. Se tiene acceso a los datos y se les modifica sólo mediante operaciones permisibles. Esta restricción al acceso se debe al encapsulado. El encapsulado protege los datos del uso arbitrario o no permitido. El acceso o la actualización directa de los datos de un objeto por parte del usuario violaría el encapsulado.

Los usuarios observan el "comportamiento" del objeto en términos de las operaciones que se pueden aplicar a los objetos, así como los resultados de tales operaciones. Estas operaciones forman la interfaz del objeto con sus usuarios.

1.4.4.2 Diferencia entre operación y método.

Las operaciones son procesos que se pueden solicitar como unidades. Los métodos son especificaciones del procedimiento de una operación dentro de una clase. Es decir, la operación es el tipo de servicio solicitado y el método es su código de programación.

Por ejemplo una operación asociada con la clase pedido podría ser aquella que calcule el total del pedido. El método especificaría la forma de calcular el total. Para esto, el método podría obtener el precio de cada artículo del pedido al enviar una solicitud a los objetos artículo asociados. A su vez,

cada objeto artículo regresaría su precio al método pedido mediante un método de la clase artículo.

Los métodos de una clase controlan solamente a los objetos de esa clase. No pueden tener acceso directo a las estructuras de datos de un objeto en una clase distinta. Para utilizar las estructuras de datos en una clase diferente, deben enviar una solicitud a ese objeto.

1.4.4.3 Herencia de Clase.

La generalización es una noción conceptual. La herencia de clase (que sólo se conoce como herencia) es una implantación de la generalización. La generalización establece que las propiedades de un tipo se aplican a sus subtipos.

La herencia de clase hace que la estructura de datos y operaciones sean disponibles para su reutilización por parte de sus subclases. La herencia de las operaciones de una superclase permite que las clases compartan código. La herencia de la estructura de datos permite la reutilización de la estructura.

1.4.4.3.1 Herencia Múltiple.

En la herencia múltiple, una clase puede heredar estructuras de datos y operaciones de más de una superclase.

Por ejemplo supóngase que existe un tipo de objeto cuenta, que tiene como subtipos a los tipos de objetos cuenta de cliente y cuenta vencida. A su vez, cuenta de cliente tiene como a subtipo a cuenta de cliente vencida y cuenta vencida también tiene como subtipo a cuenta de cliente vencida.

1.4.4.4 Selección del Método.

Cuando se envía una solicitud a un objeto, el software selecciona los métodos por utilizar. El método no se almacena en el objeto, pues esto causaría una réplica múltiple y pérdida de espacio. En vez de esto, el método se asocia con la clase. El método puede no estar en la clase de la que el objeto es una instancia, sino en una superclase.

En ese caso, el mecanismo de selección buscará la operación en su superclase y en todas las superclases de la jerarquía hasta que lo encuentre, nivel por nivel. Si la encuentra, selecciona la operación. Si la operación no se encuentra en ningún nivel de la superclase, se considera inválida la fuente de la solicitud.

De esta forma, los usuarios sólo deben especificar lo que se debe hacer, dejando que sea el mecanismo de selección el que determine la forma de localizar la operación y la ejecución. El mecanismo de selección deja en manos de la aplicación OO el problema de localizar la operación y la ejecución.

1.4.4.5 Polimorfismo.

Uno de los objetivos principales de las técnicas OO es utilizar otra vez el código. Sin embargo, algunas de las operaciones requieren adaptación para resolver necesidades particulares.

Esta necesidad, se da generalmente entre superclases y subclases, donde una subclase es una especialización de su superclase, y puede requerir alcanzar los mismos objetivos, pero con distintos mecanismos. Por ejemplo, una superclase rectángulo podría tener una operación área cuyo objetivo es calcular el área del rectángulo, definida como la multiplicación de los largos de dos lados contiguos. A su vez, la clase cuadrado es una subclase de rectángulo que también tiene una operación área cuyo objetivo es calcular el área del cuadrado, pero que está definida especialmente para los objetos del tipo cuadrado como la multiplicación del largo de uno de sus lados por sí mismo.

El fenómeno recién descrito se conoce como polimorfismo, y se aplica a una operación que adopta varias formas de implantación según el tipo de objeto, pero cumple siempre el mismo objetivo.

Una de las ventajas del polimorfismo es que se puede hacer una solicitud de una operación sin conocer el método que debe ser llamado. Estos detalles de la implantación quedan ocultos para el usuario; la responsabilidad descansa en el mecanismo de selección de la implantación OO.

1.4.5 Beneficios que se obtienen del desarrollo con OO

Día a día los costos del Hardware decrecen. Así surgen nuevas áreas de aplicación cotidianamente: procesamiento de imágenes y sonido, bases de datos relacionales, automatización de oficinas, ambientes de ingeniería de software, etc. Aún en las aplicaciones tradicionales encontramos que definir interfaces hombre-máquina suele ser bastante conveniente.

Sin embargo, los costos de producción de software siguen aumentando; el mantenimiento y la modificación de sistemas complejos suelen ser una tarea trabajosa; cada aplicación, (aunque tenga aspectos similares a otra) suele encararse como un proyecto nuevo, etc.

Todos estos problemas aún no han sido solucionados en forma completa. Pero como los objetos son portables (teóricamente) mientras que la herencia permite la reusabilidad del código orientado a objetos, es más sencillo modificar código existente porque los objetos no interaccionan excepto a través de mensajes; en consecuencia un cambio en la codificación de un objeto no afectará la operación con otro objeto siempre que los métodos respectivos permanezcan intactos. La introducción de tecnología de objetos como una herramienta conceptual para analizar, diseñar e implementar aplicaciones permite obtener aplicaciones más modificables, fácilmente extendibles y a partir de componentes reusables. Esta reusabilidad del código disminuye el tiempo que se utiliza en el desarrollo y hace que el desarrollo del software sea más intuitivo porque la gente piensa naturalmente en términos de objetos más que en términos de algoritmos de software.

1.4.6 Problemas derivados de la utilización de OO en la actualidad

Un sistema orientado a objetos, por lo visto, puede parecer un paraíso virtual. El problema sin embargo surge en la implementación de tal sistema. Muchas compañías escuchan acerca de los beneficios de un sistema orientado a objetos e invierten gran cantidad de recursos luego comienzan a darse cuenta que han impuesto una nueva cultura que es ajena a los programadores actuales. Específicamente los siguientes temas suelen aparecer repetidamente:

Curvas de aprendizaje largas. Un sistema orientado a objetos ve al mundo en una forma única. Involucra la conceptualización de todos los elementos de un programa, desde subsistemas a los datos, en la forma de objetos. Toda la comunicación entre los objetos debe realizarse en la forma de mensajes. Esta no es la forma en que están escritos los programas orientados a objetos actualmente; al hacer la transición a un sistema orientado a objetos la mayoría de los programadores deben capacitarse nuevamente antes de poder usarlo.

Dependencia del lenguaje. A pesar de la portabilidad conceptual de los objetos en un sistema orientado a objetos, en la práctica existen muchas dependencias. Muchos lenguajes orientados a objetos están compitiendo actualmente para dominar el mercado. Cambiar el lenguaje de implementación de un sistema orientado a objetos no es una tarea sencilla; Por ejemplo C++ soporta el concepto de herencia múltiple mientras que SmallTalk no lo soporta; en consecuencia la elección de un lenguaje tiene ramificaciones de diseño muy importantes.

Determinación de las clases. Una clase es un molde que se utiliza para crear nuevos objetos. En consecuencia es importante crear el conjunto de clases adecuado para un proyecto. Desdichadamente la definición de las clases es más un arte que una ciencia. Si bien hay muchas jerarquías de clase predefinidas usualmente se deben crear clases específicas para la aplicación que se este desarrollando. Luego, en 6 meses ó 1 año se da cuenta que las clases que se establecieron no son posibles; En ese caso será necesario reestructurar la jerarquía de clases devastando totalmente la planificación original.

Performance. En un sistema donde todo es un objeto y toda interacción es a través de mensajes, el tráfico de mensajes afecta el performance. A medida que la tecnología avanza y la velocidad de microprocesamiento, potencia y tamaño de la memoria aumentan, la situación mejorará; pero en la situación actual, un diseño de una aplicación orientada a objetos que no tiene en cuenta la performance no será viable comercialmente.

Idealmente, habría una forma de atacar estos problemas eficientemente al mismo tiempo que se obtienen los beneficios del desarrollo de una estrategia orientada a objetos. Debería existir una metodología fácil de

aprender e independiente del lenguaje, y fácil de reestructurar que no drene el performance del sistema.

1.5 Modelo de datos conceptual

Un modelo de datos conceptual es una descripción, un diagrama, un dibujo o algo más sofisticado, que explique, en el ámbito del problema, cuáles son los conceptos (datos) del negocio a resolver. No es una representación técnica y sólo muestra los elementos de la trama y su relación entre ellos.

Estos elementos pueden ser personas, lugares, cosas, sucesos o puramente conceptos, sobre los que el negocio quiere y puede manejar datos. Por ejemplo, clientes y facturas serán los elementos del sistema seguimiento de Inventarios.

Una relación es una conexión lógica entre dos elementos del negocio que comunican alguna información sobre el entramado interno. Por ejemplo. Un pedido es para uno y sólo uno de los clientes y una factura es para uno y sólo uno de los pedidos.

El modelo de datos conceptual que se desarrollará, será un diagrama Entidad/Relación (E/R) establecido durante la fase de análisis. Los diagramas E/R son representaciones gráficas de elementos de negocio y sus relaciones. El objetivo de estos diagramas es representar con exactitud, en un nivel conceptual los datos que un negocio necesita conocer y recordar.

Un diagrama E/R se efectúa en cuatro fases:

1. Identificar, cualificar y relacionar todos los elementos posibles.
2. Esquematizar los elementos y sus interrelaciones.
3. Documentar las reglas del negocio.
4. Revisar el diagrama E/R y actualizarlo, si procede.

1.5.1 Bases de Datos

Una base de datos se podría definir como un conjunto de información organizada, donde un archivo lógico es igual a "n" cantidad de archivos

físicos. Los archivos podrán ser almacenados en forma secuencial, mediante direcciones de memoria de disco o mediante la forma secuencial indexada.

El almacenamiento de datos es la parte medular de los sistemas de información, los objetivos para el diseño de base de datos son: integridad de los datos, disponibilidad, actualización y recuperación eficiente, almacenamiento eficiente y recuperación de información para un propósito.

La información obtenida debe estar en un formato útil para la administración, planeación, control o toma de decisiones.

Hay dos enfoques para el almacenamiento de datos:

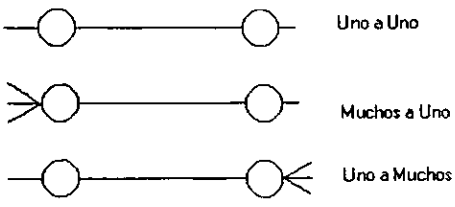
1) Guardar los datos en archivos individuales, un archivo para cada aplicación, lo que implica que están guardados en más de un lugar.
2) Desarrollar una base de datos que pueda ser compartida por muchos usuarios para varias aplicaciones, de esta manera los datos redundantes se guardan una sola vez, lo que implica una desventaja, ya que es más vulnerable a catástrofes y hay riesgos de que el administrador de la base sea el único privilegiado para acceder a los datos.

El enfoque del archivo convencional puede ser más eficiente, debido a que el archivo puede ser específico de la aplicación, el enfoque de base de datos puede ser más adecuado, ya que los mismos datos necesitan ser capturados, almacenados y actualizados una sola vez.
Para comprender el almacenamiento de datos es necesario tener conocimiento de tres conceptos, la realidad, los datos y los metadatos.

1.5.1.1 Concepto de Datos

a) Entidades: cualquier objeto o evento acerca del cual alguien escoge o recolecta datos, puede ser una persona, cosa, etc.

b) Relaciones: son asociaciones entre entidades, existen distintos tipos, uno a uno (un paquete de producto para cada producto), uno a muchos (un médico para todos) y muchos a muchos (un estudiante puede tener muchos cursos y muchos cursos son tomados por varios estudiantes)



c) Atributos: Características de una entidad (apellido, nombre, calle, etc.)

d) Registros: Conjunto conceptos que tienen algo en común con la entidad descrita.

e) Llaves: se usan para identificar a un registro. Cuando la llave candidato es elegida para identificar una entidad como única se le denomina como llave primaria. Y es llave foránea cuando una llave primaria nos sirve para relacionar a otra entidad.

f) Metadatos: Los metadatos describen a los datos, el nombre, la longitud y composición de cada registro y pueden contener restricciones acerca del valor de un concepto de datos.

Los conceptos de datos pueden tener valores y pueden ser organizados en registros y acceder mediante una llave o clave.
La efectividad de la base de datos incluye, asegurarse que la base pueda ser compartida entre los diversos usuarios, mantener datos precisos y consistentes, fácilmente disponibles, permitir que la base evolucione y que las necesidades crezcan, permitir que los usuarios construyan su vista personal sin preocuparse de cómo estén guardados.

1.5.1.2 Organización de archivos

Un archivo contiene grupos de registros que proporcionan información para una operación, planeación, administración o toma de decisiones.

- Tipos de archivos:

a) Archivos maestros: contienen registros de un grupo de entidades, son relativamente permanentes.

b) Archivos de tablas: contienen datos usados para calcular datos o medidas de desempeño.

c) Archivos de transacción: Se usa para capturar cambios y actualizar el archivo maestro y para producir reportes, son transitorios.

d) Archivos de trabajo: Aquel que ha sido reordenado para que los registros puedan ser accedidos más rápidamente.

e) Archivos de reporte: cuando se envía la salida a un archivo en lugar de mandarla a una impresora.

- Organización secuencial: se dice que es un archivo secuencial cuando esta ordenado físicamente, para actualizar es necesario recorrerlo todo, (una cinta magnética es un dispositivo secuencial) la organización secuencial es usada normalmente para todos los tipos de archivos a excepción de los maestros por su longitud.

- Listas Encadenadas: cuando son ordenados en forma lógica y no física.

- Organización de archivos de dispersión: dispositivos que permiten el acceso directo a un registro.

- Organización Indexada: un índice es diferente de un apuntador que es guardado en un archivo aparte.

- Organización secuencial indexada: archivos que permiten que los programas lean registros directamente sin pasar por otros, usando un método secuencial indexado.

1.5.2 Organización de Bases de Datos

Visión Lógica y Física.- Cada usuario ve los datos de forma diferente, el modelo lógico debe ser transformado en físico, involucrado con la manera en que son accedidos, guardados y relacionados.

Hay 3 tipos de bases de datos estructurados lógicamente:

a) Estructuras de datos jerárquicos: implican que una entidad no puede tener más de una entidad que la posea.

b) Estructuras de datos en red: permite a cualquier entidad tener cualquier cantidad de subordinados o superiores, conectados con enlaces de red, alivian los problemas de las estructuras jerárquicas.

c) Estructura de datos relacional: Consiste en una o más tablas de dos dimensiones a las que se les llama relaciones, los renglones contienen registros y las columnas atributos. Es bastante simple mantener estas tablas y además para que estas estructuras sean eficientes deben ser normalizadas.

1.5.3 Normalización

Es el proceso que transforma las vistas de los usuarios en estructuras menos complejas. Además de ser más simples y estables, las estructuras de datos normalizadas son más fáciles de mantener.

Los 3 Pasos para la normalización:

1) Primero son eliminados todos los grupos repetidos y se identifica la llave primaria.

2) Segundo, se asegura que todos los atributos que no son llave, sean dependientes de la llave primaria, las dependencias parciales son eliminadas y puestas en otra relación.

3) El tercer paso elimina cualquier dependencia transitiva, donde atributos que no son llave, son dependientes de otros que tampoco son llaves.

El objetivo principal de la normalización es simplificar todos los conceptos de datos complejos que se encuentran en las vistas de los usuarios. Después de estos tres pasos el resultado es la creación de muchas relaciones que están en la tercera forma normal.

Los lineamientos para el diseño de relaciones de bases de datos:

1) Cada entidad de datos separada debe crear un archivo maestro. No combine

dos entidades distintas en un solo archivo. (vendedores, artículos en forma separada)

2) Un campo de datos específico debe existir solamente en un archivo maestro. (el nombre del cliente debe estar solamente en el archivo maestro clientes)

3) Cada archivo maestro o relación de base de datos debe tener programas para crear, leer, actualizar y borrar registros, lo ideal es que sólo un programa añada registros y otro borre. (actualización del campo saldo actual de clientes)

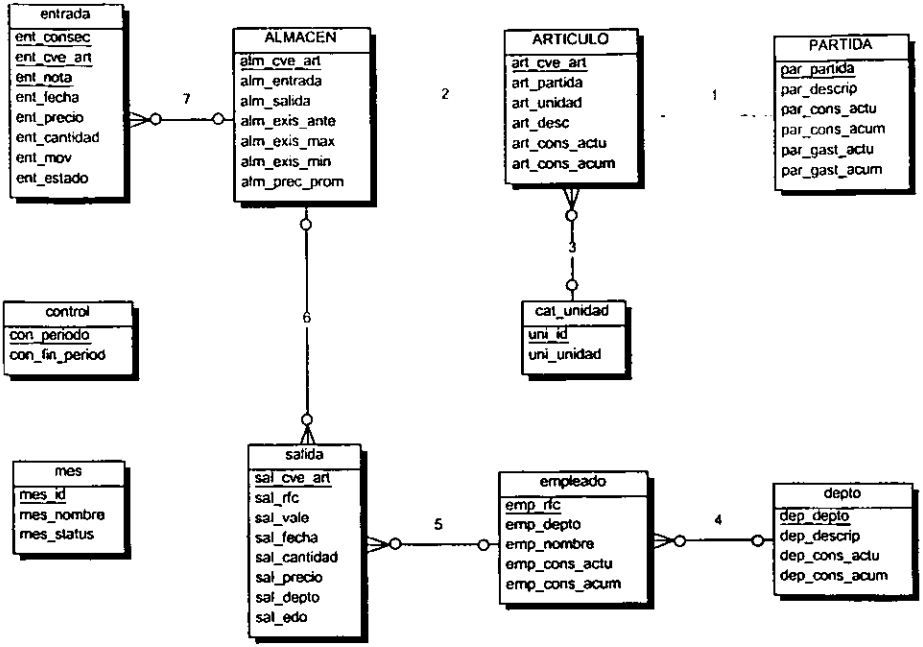
1.5.4 Pasos para la recuperación y presentación de datos

El proceso de recuperación puede involucrar hasta 8 pasos:

1) Escoger una o más relaciones de la base de datos.
2) Unir dos relaciones: Deben tener un atributo común.
3) Proyectar columnas para la relación: Es la extracción de determinadas columnas de una tabla relacional.
4) Seleccionar renglones de la relación: Se extraen renglones de la tabla relacional.
5) Derivar nuevos atributos: Implica la manipulación de datos existentes más algunos parámetros adicionales para derivar nuevos datos.
6) Indexar u ordenar renglones: ordenamiento lógico de renglones de acuerdo a una llave (alfabético)
7) Calcular totales y medidas de desempeño.
8) Presentar datos: Puede tomar muchas formas, en forma tabular, gráfica o como respuesta.

1.5.5 Diagrama Entidad/Relación

El diagrama Entidad/Relación utilizado para el diseño físico de la base de datos utilizado por el sistema de control del almacén es el siguiente:



La sintaxis de la estructura de relaciones del diagrama anterior es la siguiente:

1) Cada artículo debe estar asignado a una y sólo una partida.
   Cada partida puede ser asignada a uno o más artículos.

2) Cada almacén puede contener uno o más artículos.
   Cada artículo debe estar almacenado en un y sólo un almacén.

3) Cada artículo tiene asignada un cat_unidad.
   Cada cat_unidad puede estar asignado en muchos artículos.

4) Cada empleado debe estar asignado a uno y sólo un depto.
   Cada depto puede ser responsable de uno o más empleados.

5) Cada empleado puede solicitar una o más salidas.
Cada salida debe ser asignada a uno y sólo un empleado

6) Cada almacén puede tener una o más salidas.
Cada salida debe de ser generada por un y sólo un almacén

7) Cada almacén puede tener una o más entradas.
Cada entrada debe ser generada por un y sólo un almacén.

1.6 Interfaz del usuario

Los objetivos de diseñar interfaces para ayudar a los usuarios a proporcionar información que necesitan son: la efectividad para acceder al sistema de la forma que necesitan, el aumento de la velocidad en la captura de datos y la reducción de errores, el logro de retroalimentación del sistema a los usuarios y la productividad.

Tipos de Interfaz: Tiene dos componentes principales, el lenguaje de presentación (de la computadora al usuario) y el lenguaje de acción (la parte del usuario a la computadora)

- Interfaces de lenguaje natural: permite que los usuarios interactúen en su lenguaje con la computadora. No se requieren habilidades especiales del usuario. Los problemas de implementación son mínimos.
- Interfaces de pregunta y respuesta: la computadora muestra una pregunta, el usuario teclea una respuesta y la computadora actúa sobre esa información en forma preprogramada, moviéndose a la siguiente pregunta.
- Interfaces de llenado de forma (Formas de entrada/salida): consisten en formas en pantalla que despliegan campos que contienen conceptos comunicados al usuario. La ventaja principal es que la versión impresa proporciona excelente documentación, la desventaja es que los usuarios pueden impacientarse con las formas y querer formas para capturar datos más eficientes.
- Interfaces de lenguaje comando: permite al usuario controlar la aplicación con una serie de tecleos, comandos, frases o secuencia de ellos. No tiene significado inherente al usuario, requieren memorización de las reglas y puede ser un obstáculo, lo prefieren los usuarios experimentados por la rapidez.

- Interfaces gráficas de usuario: permiten el manejo de la representación gráfica en pantalla, requiere más sofisticación del sistema, hace muchas más fácil el trabajo del analista para la creación de diagramas, algunas interfaces gráficas utilizan iconos, como Windows.
- Otras interfaces de usuario: incluyen dispositivos apuntadores como plumas ópticas, pantallas sensibles al tacto, reconocimiento de voz. La ventaja es la aceleración en la captura de datos.

1.6.1 Retroalimentación para usuarios

Es necesaria la retroalimentación a los usuarios por parte del sistema, para que sepan si su entrada está siendo aceptada, con datos correctos, si el procesamiento está avanzando, si las peticiones pueden ser o no procesadas y si se encuentra disponible información más detallada y cómo obtenerla.

Diseño de Consultas:
Las consultas están diseñadas para permitir a los usuarios extraer datos significativos de la base de datos, hay 6 tipos básicos de consultas y pueden ser combinadas usando lógica para formar consultas más complejas. Cada consulta involucra 3 conceptos, una entidad, un atributo y un valor. En cada caso se dan dos de ellos y el objetivo es encontrar el restante.

Tipos de Consultas:

1) Se dan la entidad y un atributo, se busca el valor, ejemplo: Cuánto ganó el empleado número 7 (entidad) en el año 95 (atributo).

2) Se dan el atributo y el valor y se busca la entidad, ejemplo: Cuáles empleados ganaron más de $100 en el año.

3) Cuáles atributos concuerdan con la entidad y el valor, ejemplo: en qué años el empleado z ganó más de x.

4) Similar a la consulta 1, la diferencia es que se desean los valores de todos los atributos, ejemplo: Consulta sobre el empleado número 9, la respuesta es el nombre.

5) Similar a la 2 pero global, liste todas las entidades que tienen un valor

específico para todos los atributos, ejemplo. Todos los empleados con ingresos superiores a x en los años disponibles.

6) Similar a la 3 listas, todos los atributos de todas las entidades

1.6.2 Diseño de procesamientos para la captura de datos precisa

La calidad de los datos de entrada al sistema de información es crítico para asegurar la calidad de la salida.

Objetivos de la captura de datos para mejorar la calidad de los datos:

1) Codificación efectiva: una de las mejores formas de agilizar la captura es con la codificación que pone datos en secuencias cortas de dígitos o letras, estos datos requieren menos tiempo y espacio en memoria para su captura y facilita el ordenamiento. Los objetivos de la codificación son:
   a) Hacer el seguimiento de algo (usando códigos de secuencia simple o alfabética).
   b) Clasificar la información (clases mutuamente excluyentes, con códigos de clasificación y códigos de secuencia en bloque agrupados por características comunes).
   c) Ocultación de información restringida (son útiles los códigos de cifrado sustituyendo una letra por otra o por un número).
   d) Revelar información utilizando códigos de subconjuntos de dígitos significativos con números o códigos neumónicos alfanuméricos para recordarlos.
   e) Solicitar una acción adecuada utilizando códigos de función. Los puntos a tener en cuenta para la codificación son, ser conciso (implica menor probabilidad de error), mantener los códigos estables y uniformes, hacer códigos que sean únicos, que sean ordenables, evitar códigos confusos, tratar que los códigos sean modificables y adaptables y que sean significativos.

Captura de datos efectiva y eficiente: se debe seleccionar un dispositivo de entrada teniendo en cuenta la velocidad y confiabilidad, entre éstos están:

- Teclado a almacenamiento que requiere un microcomputador y software.
- Reconocimiento óptico de caracteres (mediante lectores, se agiliza la

entrada y se evitan errores).
- Reconocimiento de caracteres por tinta magnética (es confiable y veloz, se utilizan en los cheques y pagarés).
- Formas de marcas sensibles (se utilizan digitalizadores para sentir marcas con lápiz o formas especiales).
- Formas perforadas (rápida y conveniente para capturar datos limitados).
- Código de barras (metacódigos que codifican códigos, bandas que aparecen en una etiqueta que codifican números y letras leído por un rayo de luz o pluma óptica, alto grado de precisión, ahorra costos).
- Terminales inteligentes (dispositivos de entradas que aunque usan un microprocesador liberan a la CPU).

3) Asegurar la calidad por medio de la validación: Pueden suceder tres problemas con las transacciones de entradas, envío de datos erróneos, envío de datos por una persona no autorizada y pedir que el sistema realice una función inaceptable. La validación de los datos de entrada se incorpora al sistema de diferentes formas, prueba de datos faltantes, de longitud de campo correcta, de clase o composición (números o letras), prueba de rango o razonabilidad, de valores inválidos, de comparación con datos almacenados o poner códigos autovalidantes usando un dígito verificador en el código mismo.

1.7 Aseguramiento de la calidad por medio de la ingeniería del software

Se supone que cuando se está implementando un sistema no se deben de cometer errores, es por eso que hay que hacer varias pruebas para que en cada caso se hagan las correcciones pertinentes de los errores que van surgiendo. La ingeniería de software se apoya en los resultados obtenidos en el diseño lógico, hoy en día hay aplicación de normas para el desarrollo de un sistema.

Existen tres enfoques para el aseguramiento de la calidad mediante la ingeniería de software:

1) Asegurar la calidad por medio del enfoque modular de arriba hacia abajo en el diseño de sistemas de software.

El Concepto de calidad, hoy, es muy importante ya que gran responsabilidad la tienen los usuarios y el administrador del sistema de información.

Diseño y desarrollo de sistemas: la calidad total puede ser implementada tomando un enfoque de arriba hacia abajo para el diseño, es decir, primero los objetivos organizacionales generales y luego se divide el sistema en subsistemas y sus requerimientos, teniendo en cuenta las interrelaciones e interdependencias de los subsistemas.

Desarrollo modular: es la división de la programación en partes o módulos lógicos y manejables, cada uno debe ser funcionalmente coherente, está encargado del logro de una sola función, el desarrollo modular hace que la programación, depuración y mantenimiento sea más fácil de lograr. Hay sistemas que enlazan programas en el ambiente windows (intercambio dinámico de datos) conservando todas las propiedades. Una herramienta recomendada para un sistema modular de arriba a abajo es la gráfica de estructura, el cual es un diagrama compuesto por cuadros rectangulares que representan módulos conectados por flechas. Estas flechas pueden ser de dos tipos, acoples de datos y bandera de control, que indican que algo pasa de un módulo inferior a otro superior y viceversa.

Para los tipos de Módulos existen tres categorías, las cuales son:

- Control, están cerca de la parte superior de la gráfica de estructura y contienen la lógica para ejecutar los módulos inferiores no deben ser de tamaño muy grande.
- Transformacionales o trabajadores, creados a partir de un diagrama de flujo de datos, generalmente ejecutan una sola tarea (aunque una secundaria puede asociarse a una tarea primaria), tienen un lugar inferior en los módulos de la gráfica de estructura.
- Funcionales o Especializados, son los más bajos de la estructura, teniendo rara vez módulos subordinados, ejecutan solamente una tarea.

La cantidad de módulos se puede explicar de la siguiente manera, cada módulo tiene una función, cada función tiene una operación específica y esto es una rutina, por lo cual no es conveniente que existan más de 4 ó 5 módulos. Cada módulo tiene instrucciones que generalmente son alrededor de cincuenta. Cuando se realiza el mantenimiento se va directamente al módulo

que tiene el problema y sólo se revisan cincuenta instrucciones, por dar un ejemplo.

2) Documentar el software con las herramientas adecuadas

Parte de la administración de calidad total es ver que los programas estén diseñados, documentados y mantenidos adecuadamente, la documentación permite que los usuarios, programadores y analistas, vean el sistema, su software y procedimientos sin tener que interactuar con él, existen varias técnicas de diseño y documentación: Diagramas de Flujo, Seudo código, Manuales de procedimiento, etc. Los analistas deben escoger una técnica que sea compatible con la documentación existente, comprendida por todos, adecuada por el tamaño del sistema, que permita un enfoque de diseño estructurado, y que sea de fácil modificación. Para la generación de código y el proceso del uso del software, con el que se creará todo o parte de un programa de computadora, existen muchos generadores de código, los cuales están disponibles comercialmente. La reingeniería y la ingeniería inversa se refieren al uso del software para analizar código de programas existentes y crear elementos de diseño "case" a partir del código. El diseño "case" puede ser luego modificado y usado para generar nuevo código de programa de computadora.

3) Mantener y auditar el software

Cuanto mejor sea el diseño del sistema, más fácil será mantenerlo y menos costoso. El mantenimiento se realiza para mejorar el software existente, en vez de responder a una crisis o falla del sistema. Conforme cambian los requerimientos de los usuarios, debe cambiar el software y la documentación como parte de trabajo de mantenimiento, también se realiza para actualizar el software en respuesta a una organización cambiante. El mantenimiento antes llevaba mucho tiempo, y costo, las etapas iniciales del ciclo de vida no eran muy consideradas, en el nuevo enfoque se pretende que se use más tiempo en las primeras etapas que implica un menor tiempo en las etapas finales y un menor costo de mantenimiento, aplicando controles de calidad en cada etapa.

La Auditoría es otra forma de asegurar la calidad de la información contenida en el sistema, se refiere a tener un experto que no esté involucrado en el ajuste o uso de un sistema para que examine la información para

asegurar su confiabilidad. Hay dos tipos de auditores, los internos que trabajan para la misma organización dueña del sistema, y externos que son contratados del exterior de la organización, que auditan el sistema para asegurar la legalidad de los estados financieros. También en los casos en que ocurren cosas fuera de lo normal y que involucra a los empleados de la compañía.

1.8 Lenguaje de programación Power Builder

Power Builder es un entorno de programación que está compuesto por diferentes herramientas, para el desarrollo rápido de una aplicación en el ambiente cliente-servidor.

Permite usar ventanas, botones y todas las herramientas que presenta windows, facilitando su manejo.

Se maneja básicamente con el ratón, y el lenguaje de programación llamado PowerScript.

Esta herramienta de desarrollo está completamente orientada a objetos, lo que permite a equipos de programadores crear aplicaciones gráficas sofisticadas con acceso a información de base de datos locales o en servidores de red.

1.8.1 Características Básicas

Power Builder es un desarrollador de aplicaciones, cuyas características básicas son:

- Soporta una gran variedad de sistemas de gestión de base de datos, tales como: Sybase, Informix, Oracle, Watcom, entre otras.
- Tiene capacidad de acceder a información de múltiples bases de datos y mostrar esa información en una única ventana.
- Se trabaja en ambiente cliente - servidor.
- Posee un objeto inteligente llamado Datawindow que realiza directamente la interfase con la base de datos, sin requerir que el programador conozca SQL.
- Capacidad de utilizar sentencias SQL combinadas en el código.

- Se puede trabajar en múltiples plataformas, ya que soporta diferentes sistemas operativos y posee drivers nativos para las bases de datos más comerciales.
- La creación de aplicaciones es sencilla.
- Se realiza poca programación.
- Permite usar ventanas, botones y todas las herramientas que presenta el windows facilitando su manejo.
- Para construir la aplicación se utilizan painters, allí se definen las propiedades de los objetos y se agregan los controles.
- Se puede trabajar con múltiples ventanas.

Power Builder, ofrece a los programadores:

- Un entorno de desarrollo profesional.
- La Orientación a Objetos.
- Rapidez de aprendizaje y de desarrollo.
- Generación automática de código SQL.
- Acceso a las aplicaciones de Windows.
- Una herramienta de desarrollo asequible y rentable en explotación.
- Un entorno abierto de desarrollo que dispone de interfaces inteligentes a otras tecnologías Cliente/Servidor.
- Abierto a la mayoría de Base de Datos del mercado.
- Abierto a diferentes herramientas CASE.
- Abierto a Librerías de objetos y control de versiones.
- Abierto a aplicaciones Windows: DLL, DDE, OLE.

1.8.2 Eventos y Scripts

Un evento se ejecuta, por ejemplo, cuando se hace Click en un botón o al salir de cualquier control.

En el Script se escriben los comandos y funciones que se realizan cuando se ejecuta algún evento.

1.8.3 Funciones

Permite al programador hacer más fácil la programación. Power Builder posee una gran cantidad de funciones que se pueden usar en los objetos, por ejemplo se pueden usar para abrir o cerrar una ventana o para activar algún botón.

1.8.4 Librerías

Las librerías o PBL es el contenedor básico de objetos Power Builder (ventanas, menús, datawindows). Cualquier cosa que se cree deberá estar contenida en una PBL o librería. Una aplicación puede tener una o más librerías.

De una PBL se puede generar una PBD, que es lo mismo que una PBL pero sin el código fuente, o una DLL típica de windows.

2

Análisis y Diseño.

En la construcción de todo sistema es imprescindible hacer un análisis y diseño previos al desarrollo, los cuales permitirán tener un mayor orden y control, así como una mejor comprensión de lo que se tendrá que realizar, todo esto utilizando los conceptos anteriormente estudiados y que se aplicarán a la practica, para que de esta forma se logre llegar a la meta final que es la puesta en marcha del sistema.

2.1 Metodología de diseño

Antes de la llegada de las bases de datos, en la mayoría de los sistemas de información, resultaba muy difícil hacer modificaciones, así como el modo de utilizar los datos se complicaba. Los programadores ven los datos, cada uno, a su modo y quieren siempre modificarlos a medida que varían sus necesidades. Pero cualquier modificación es capaz de desatar una verdadera reacción en cadena de cambios en los programas existentes y resulta así inadmisible y oneroso. Dos aspectos del diseño de la base de datos son importantes con miras a lograr la flexibilidad de uso, que es esencial en la mayoría de las aplicaciones comerciales. Estos aspectos son los siguientes:

- Los datos deben ser independientes de los programas que los utilizan, de modo que se les pueda enriquecer y reestructurar sin que resulten necesarios los programas existentes.

- Debe ser posible interrogar y explorar la base de datos sin necesidad de recurrir a la tediosa operación de escribir programas utilizando los lenguajes convencionales de programación. En lugar de éstos se utilizarán lenguajes especiales para averiguación.

La tarea de diseñar una base de datos es cada vez más difícil, en particular cuando se requiere alcanzar soluciones óptimas. El software es cada vez más

elaborado y a menudo se comprenden mal sus aptitudes, se le emplea erróneamente o no se le aprovecha a fondo. En el caso particular de esta tesis se ocupará el lenguaje Power Builder debido a su gran facilidad de programación, interfaces con el usuario y por ser uno de los lenguajes más completos en la actualidad. De igual forma hay muchas maneras de estructurar los datos y cada uno de ellos ofrecen ventajas y desventajas.

2.1.1 Clasificación de las metodologías

El diseño efectivo del software se logra mejor utilizando una metodología consistente de diseño. Hay una gran cantidad de metodologías de diseño desarrolladas y que se utilizan en diferentes aplicaciones. Estas metodologías se pueden clasificar en:

- Diseño funcional descendente.

  El sistema se diseña desde un punto de vista funcional, empezando con una visión de alto nivel y redefiniéndola de forma progresiva hasta llegar a un diseño más detallado. Dicha metodología esta ejemplificada por el diseño estructurado (Constantine y Yourdon 1979), y el refinamiento por pasos (Wirth 1971;1976)

- Diseño orientado a objetos.

  El sistema se ve más como una colección de objetos que como funciones que pasan mensajes de un objeto a otro. Cada objeto tiene su propio conjunto de funciones de operaciones asociadas. El diseño orientado a objetos se basa con la idea de la ocultación de información propuesta por primera vez por Parnas (1972), y que ha sido descrita en fecha reciente por Robson y Booch (1983)

- Diseño controlado por los datos.

  Esta metodología, propuesta por Jackson (1975) y Warnir (1977), plantea que la estructura de un sistema de software debe reflejar la estructura de datos. Por lo tanto, el diseño de software se obtiene de un análisis de los datos del sistema de entrada y salida.

2.1.2 Etapas del diseño

Durante la etapa del diseño, se analiza la definición de requisitos y se diseñan los componentes del software para proporcionar los servicios al usuario. Este diseño se expresa de manera que se puedan realizar estos componentes en un lenguaje de programación. La especificación del software es parte de este proceso, en el cual el diseño se presenta de forma abstracta de alto nivel. En sistemas pequeños, es posible pasar directamente de la definición de requisitos a un diseño detallado de componentes, pero en cuanto a grandes sistemas de software, la actividad de diseño puede dividirse en tres etapas:

1. Asociar componentes abstractos de software con los servicios establecidos en la definición de requisitos y construir especificaciones precisas para esos componentes.

2. Construir un diseño de alto nivel que muestre la relación recíproca de los componentes abstractos del software.

3. Formular un diseño detallado para cada componente abstracto. Este diseño se expresa en términos de abstracciones más simples fácilmente traducibles a código ejecutable.

El enfoque de diseño del software fue la causa de muchos fallos, por lo regular graves y caros en los proyectos. Se han desarrollado varias notaciones de diseño como diagramas de flujo, diagramas de datos, diagramas de estructura y lenguajes para descripción del diseño, que son más apropiados que los organigramas que se usaban anteriormente para expresar los diseños del software. Este diseño que satisfaga los requisitos para el sistema de Control de Inventario se realizará mediante las siguientes etapas:

1. Se establecerán los módulos que componen el sistema de programación.
2. Cada módulo se dividirá en componentes individuales y se establecerá la especificación de los módulos definiendo la operación de sus componentes.
3. Posteriormente, cada módulo se diseñará de forma que sus componentes interactúen recíprocamente.

4. Se refinará cada componente (reingeniería) Esto podría implicar la especificación de cada componente como una jerarquía de módulos, mejor conocida como herencia.
5. En algún momento de este proceso de reingeniería hay que especificar con detalle los algoritmos utilizados en cada componente y los que son comunes en una familia de componentes.

2.1.3 Notaciones de diseño

Las notaciones para el diseño del sistema de control de inventario son muy parecidas a las de creación de objetos abstractos de software como los sistemas de software. Dichas notaciones que se ocuparán en el diseño de este sistema son:

a. Diagrama de flujo de datos

Son diagramas que se utilizan para describir un diseño de sistemas de alto nivel; muestran como se transforman los datos al pasar de un componente del sistema a otro.

Estos diagramas de flujo de datos se dividen a su vez en dos partes:
- Diagrama de flujo de datos lógico
- Diagrama de flujo de datos físico

Estos dos unidos son conocidos también como:
- Diagrama E - R

El diagrama de datos lógico se enfoca en el negocio y a la manera en que opera el mismo, describe los eventos que suceden y los datos requeridos y producidos por cada evento. En forma inversa, un diagrama físico, muestra como esta implementado el sistema, incluyendo el hardware, software, archivos y personas involucradas en el sistema. El modelo lógico refleja al negocio, ilustra el proceso involucrado sin detallar la implementación física de las actividades, el físico representa el sistema, por ejemplo, muestra qué y cómo se usa, un código de barras, etc.

Los diagramas de Flujo de datos (DFD's) son generalmente familiares y usan una extensa notación para las funciones especificas de un sistema de información. Éstos describen a los sistemas como colección de datos que son manipulados por funciones. Los datos son organizados en varias direcciones: éstos pueden ser almacenados en lugares específicos y pueden transferirse desde un ambiente externo.

Una de las razones por las que han tenido éxito los DFD's es porque éstos pueden ser expresados por medio de una atractiva notación gráfica que hace más fácil su uso.

Los elementos básicos de un DFD's son:

Burbujas: Se usan para representar funciones.

Flechas: Se usan para representar flujo de datos. Las flechas que entran a las burbujas representan la entrada de flujo de valores que pertenecen al dominio de la función representada por la burbuja. Las flechas que salen representan los resultados de la función.

Cajas abiertas: Se usan para almacenamiento de datos. Las flechas de entrada a las cajas abiertas representan inserción de los datos a éstos y las flechas que salen representan extracción de datos de dichos almacenamientos.

Cajas de I/O Se usan para representar adquisición de datos y el proceso de la interacción entre el humano y la máquina.

b. Diagramas de estructura.
Son gráficas de jerarquía que muestran la relación estructural de los componentes de un sistema de software. Nuevamente esta notación es más útil para describir el diseño de sistemas de alto nivel.

c. Un lenguaje para la descripción del diseño.
Es una notación con algunos atributos de los lenguajes de programación, adecuada para describir operaciones de control y de diseño detallado. Se puede usar cualquier nivel de diseño de sistemas.

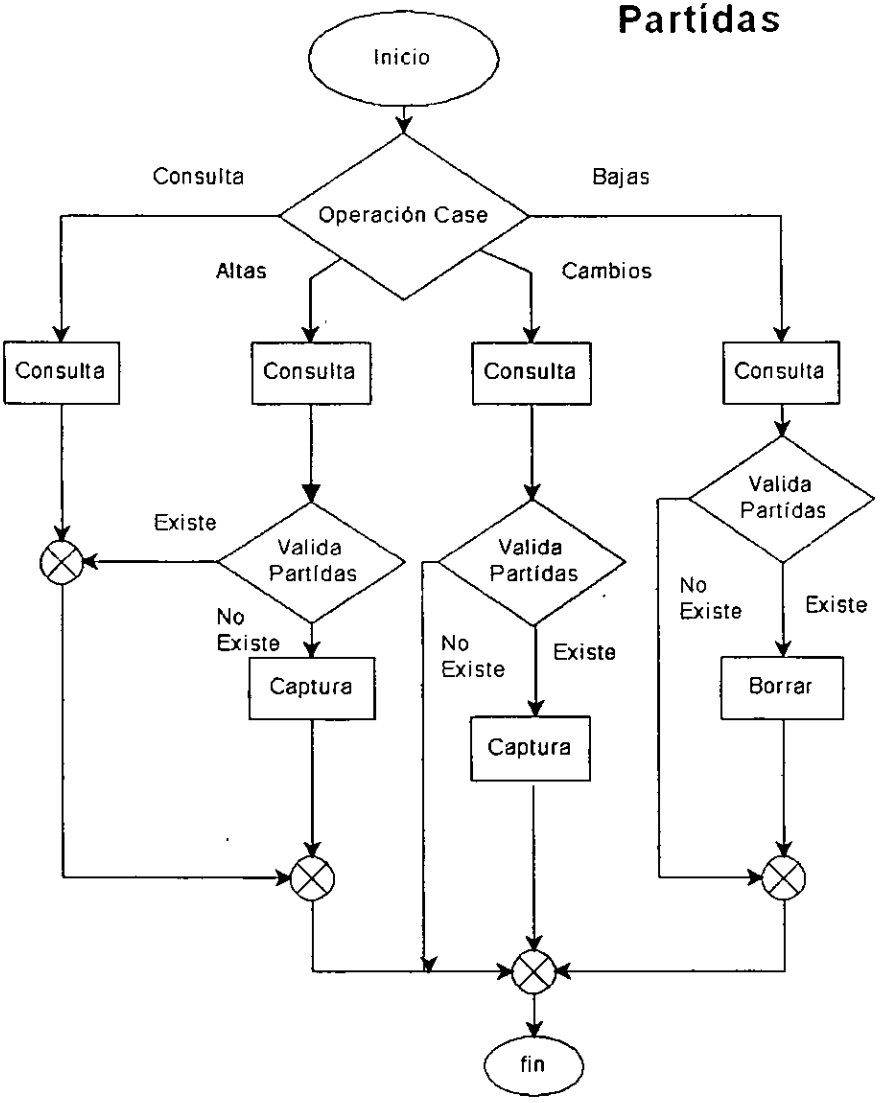
2.2 Módulos del sistema

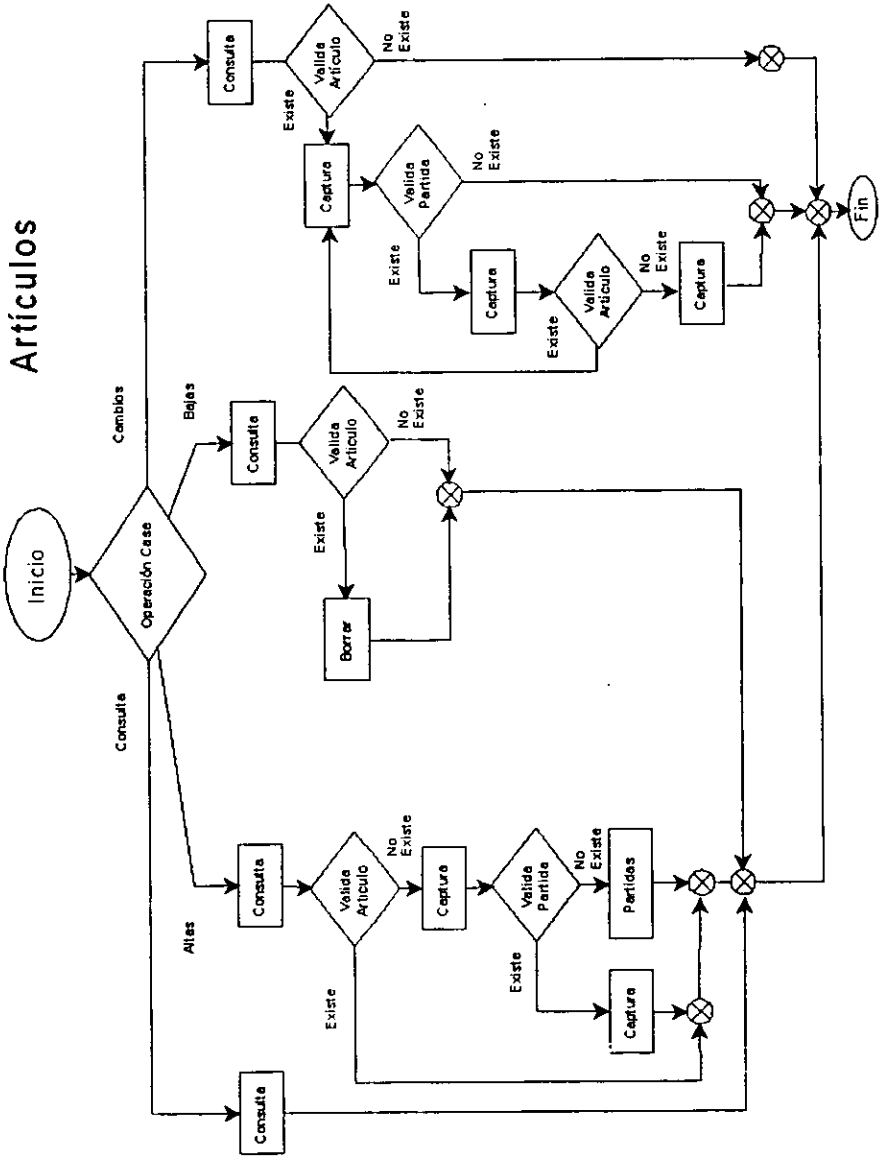
Todo sistema debe estar estructurado de forma que cuando se necesite hacer una modificación, no haya ninguna dificultad para encontrar la forma de hacerlo, en este caso el Sistema de Control de inventarios esta dividido principalmente en cinco módulos:

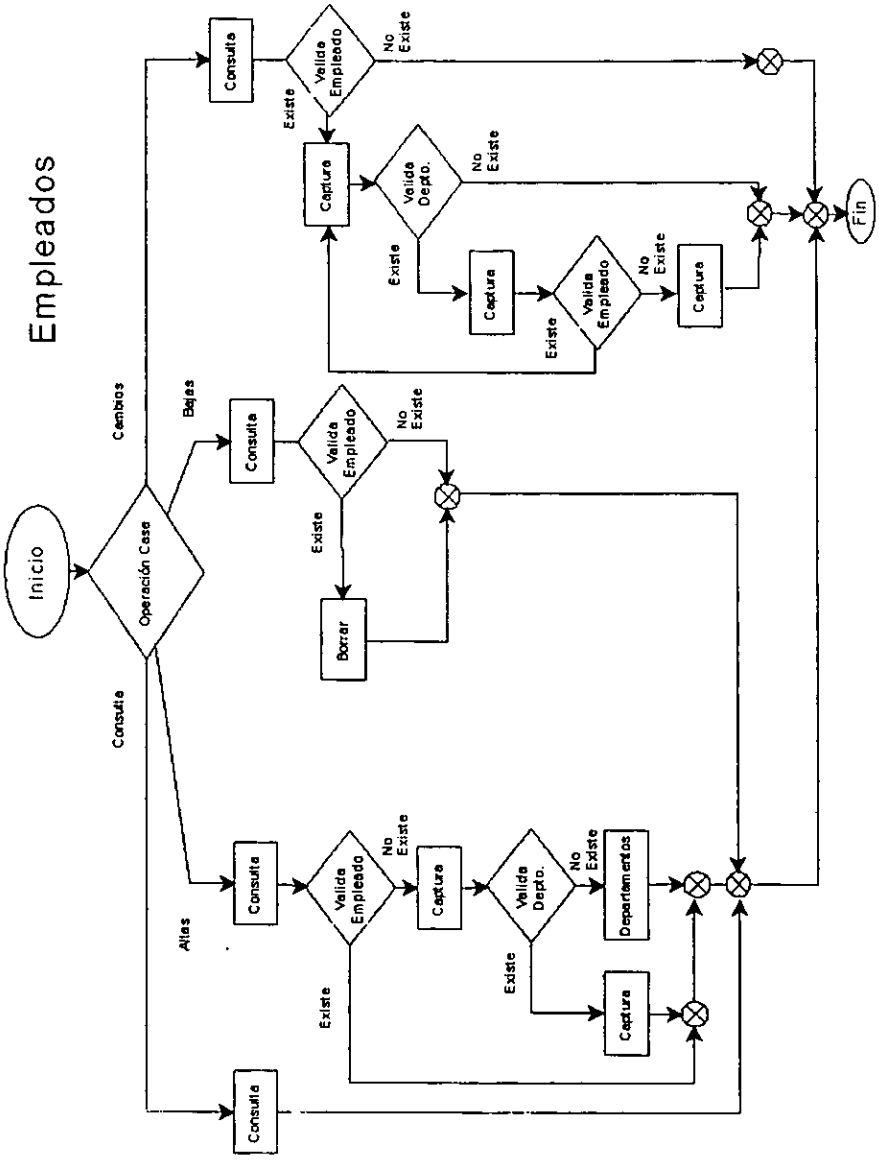
- Catálogos
- Movimientos
- Consultas
- Mantenimiento

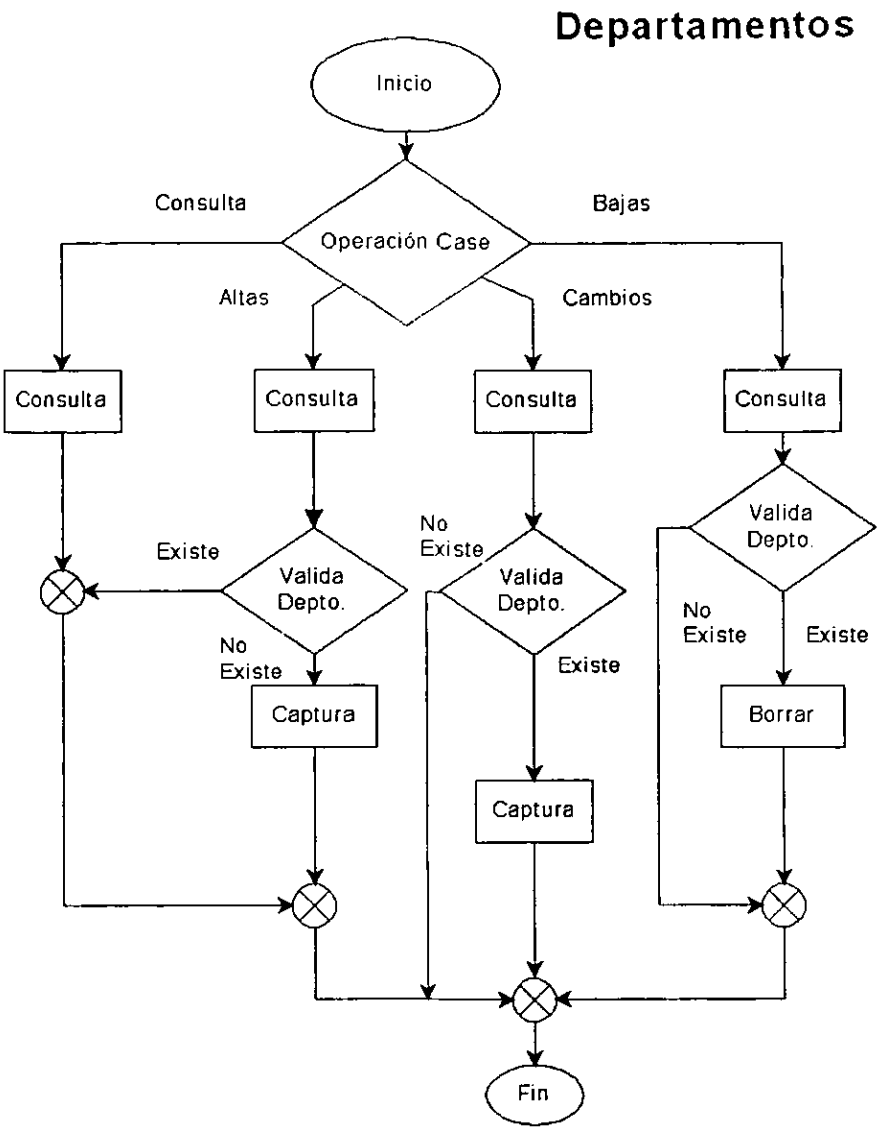
Ésta es la estructura general de división de un sistema, es decir son reglas establecidas por Ingeniería de Software. Al programar con el lenguaje Power Builder nos permite saber de una manera más clara (es decir en forma gráfica) los módulos del sistema.

2.3 Diagrama de flujo de datos.

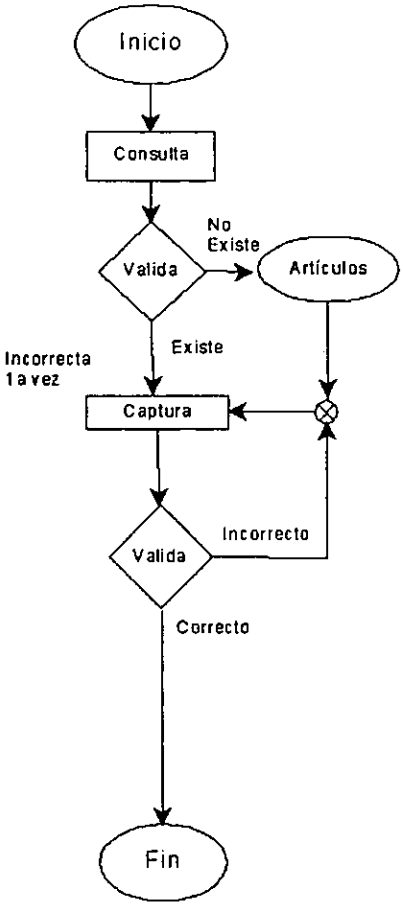



Artículos
Inicio
Operación Case
Cambios
Bajas
Consulta
Altas
Consulta
Valida Artículo
No Existe
Existe
Captura
Valida Partida
Captura
Valida Artículo
Captura
Consulta
Valida Artículo
Existe
Borrar
Consulta
Valida Artículo
Captura
Valida Partida
Partidas
Captura
Consulta
Fin


Empleados
Inicio
Operación Case
Cambios
Bajas
Consulta
Altas
Consulta
Valida Empleado
No Existe
Existe
Captura
Valida Depto.
No Existe
Existe
Captura
Valida Empleado
Existe
No Existe
Captura
Fin
Consulta
Valida Empleado
No Existe
Existe
Borrar
Consulta
Valida Empleado
No Existe
Existe
Captura
Valida Depto.
No Existe
Existe
Departamentos
Captura
Consulta


Departamentos
Inicio
Operación Case
Consulta
Altas
Cambios
Bajas
Consulta
Consulta
Consulta
Consulta
Existe
Valida Depto.
No Existe
Captura
No Existe
Valida Depto.
Existe
Captura
Valida Depto.
No Existe
Existe
Borrar
Fin

Inventario Inicial

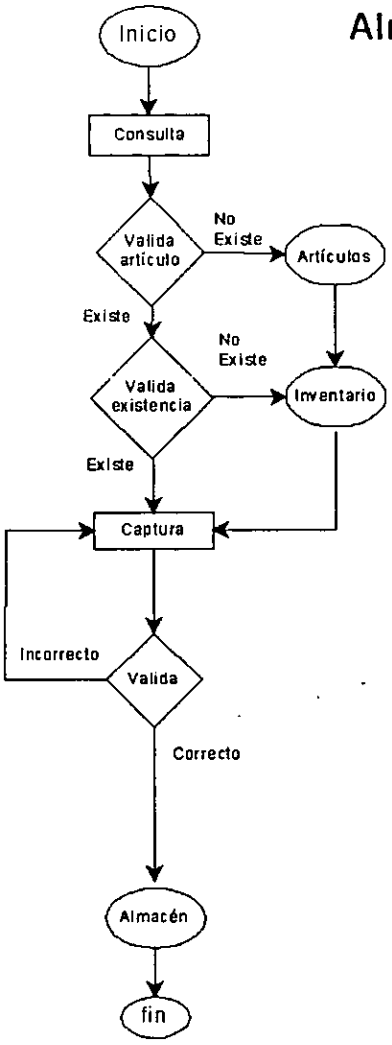


Se consultan los artículos que no tienen inventario inicial

Personal del departamento verifica si existe el artículo, si no existe capturarlo

Se ingresan los datos del artículo (precio y cantidad).

Personal del departamento de almacén verifica la informacion. En caso de estar mal capturados corregir (1a o 2a vez de acuerdo a esta toma una u otra ruta)



Almacén Entradas

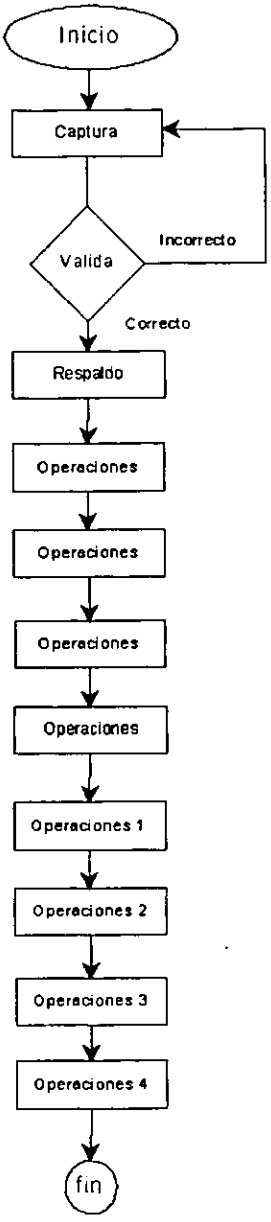
Se consulta el catálogo de artículos para ingresar los datos requeridos de los artículos que ya tienen inventario inicial

Personal del departamento valida si existe el artículo, además de que tenga inventario inicial, si no realizar las operaciones necesarias

Captura las cantidades que ingresaron por artículo.

Personal del departamento de almacén, valida la información capturada, si no realizar las operaciones necesarias

Se pueden realizar salidas desde este momento



Almacén Cierre de Mes

Se captura la fecha de cierre de mes.

Valida que sea el último día del mes.

Respalda la información del sistema.

Se suman
Entradas + Invéntario inicial = entradas.

Inventario inicial del mes siguiente =
Entradas - Salidas

Se hacen cálculos
por partida de acuerdo al artículo

Se hacen cálculos de empleados y
departamentos de acuerdo al departamento

Se realizan las siguientes
operaciones en todas las tablas

XX_acum = XX_acum + XX_actu

XX_actu = 0

Emisión de reportes

2.4 Diccionario de datos.

En esta parte del capítulo se hará una descripción de las tablas que componen a nuestra base de datos, es decir, campos que los componen, significado de éstos mismos así como llaves primarias y llaves foráneas.

Tabla PARTIDA

Columna	par_partida	par_descrip	par_cons_actu	par_cons_acum	par_gast_actu	parl_gast_acum
Tipo de Llave	PK					
Nulos/Unicos	NN,U	NN				
Ejemplo	100	PAPELERIA	0	1183.30	0	2925.00

Tabla ARTÍCULO

Columna	art_cve_art	art_partida	art_unidad	art_desc	art_cons_actu	art_cons_acum
Tipo de Llave	PK	FK1	FK2			
Nulos/Unicos	NN,U	NN	NN	NN		
Ejemplo	3000006	300	1	BOLSA DE JABON	0	0

Tabla CAT_UNIDAD

Columna	uni_id	uni_unidad
Tipo de Llave	PK	
Nulos/Unicos	NN,U	NN
Ejemplo	1	PIEZA

Tabla ALMACEN

Columna	alm_cve_art	alm_entrada	alm_salida	alm_exis_ante	alm_exis_max	alm_exis_min	alm_exis_prom
Tipo de Llave	PK, FK						
Nulos/Unicos	NN,U	NN	NN				
Ejemplo	1000001	15.00	0	25.00	80.00	5.00	118.33

Tabla DEPTO

Columna	dep_depto	dep_descrip	dep_cons_actu	dep_cons_acum
Tipo de Llave	PK			
Nulos/Unicos	NN,U			
Ejemplo	001	SISTEMAS	0	1183.30

Tabla EMPLEADO

Columna	emp_rfc	emp_depto	emp_nombre	emp_cons_actu	emp_cons_acum
Tipo de Llave	PK	FK			
Nulos/Unicos	NN,U				
Ejemplo	GACJ740613	001	JORGE GALVEZ CASTILLO	0	1183.30

Tabla ENTRADA

Columna	ent_consec	ent_cve_art	ent_nota	ent_fecha	ent_precio	ent_cantidad	ent_mov	ent_estado
Tipo de Llave	PK	PK, FK	PK					
Nulos/Unicos	NN,U	NN,U	NN,U	NN		NN	NN	
Ejemplo	1	1000001	0001	15/03/99	100	10	1	"

Tabla SALIDA

Columna	sal_cve_art	sal_rfc	sal_vale	sal_fecha	sal_cantidad	sal_precio	sal_depto	sal_edo
Tipo de Llave	PK, FK1	PK, FK2						
Nulos/Unicos	NN,U	NN,U	NN	NN	NN		NN	
Ejemplo	1000001	GACJ740613	0001	16/04/00	10.00	118.33	001	

Tabla MES

Columna	mes_id	mes_nombre	mes_status
Tipo de Llave	PK		
Nulos/Unicos	NN,U	NN	
Ejemplo	1	ENERO	1

Tabla CONTROL

Columna	con_periodo	con_fin_periodo
Tipo de Llave	PK	
Nulos/Unicos	NN,U	
Ejemplo	1	31/01/00

Tabla: PARTIDA.

Es un catálogo presupuestal en el cual se almacenarán las partidas contables, con su descripción, así como su consumo y gasto que le corresponde a cada una de éstas.

par_partida: Es un elemento presupuestario en el cual se dividen los subgrupos de gastos y que clasifican las distribuciones de acuerdo con el objeto específico del gasto.

par_descrip: Es la descripción de la partida.

par_cons_actu: Es el consumo actual de la partida.

par_cons_acum: Es el consumo acumulado de la partida hasta la fecha de corte.

par_gast_actu: Es el gasto actual de la partida.

par_gast_acum: Es el gasto acumulado de la partida hasta la fecha de corte.

Tabla: ARTÍCULO.

Es un catálogo que contiene la clave del artículo, su descripción, así como su consumo que tienen dentro del almacén.

art_cve_art: Es el elemento primario al cual le será asociado un número de artículo, que debe de cumplir con la regla de pertenecer a una partida (La cual debe existir en la tabla partida) y un número consecutivo, de acuerdo a las necesidades de la dependencia, donde este número deberá constar de 5 dígitos.

art_partida: Es un elemento presupuestario en el cual se dividen los subgrupos de gastos y que clasifican las distribuciones de acuerdo con el objeto específico del gasto, sirve para identificar la partida del artículo.

art_unidad: Es la medida, volumen o peso del artículo.

art_desc: Es la descripción del artículo.

art_cons_actu: Es el consumo actual del artículo.

art_cons_acum: Es el consumo acumulado del artículo hasta la fecha de corte.

Tabla: DEPTO.

Es un catálogo que contiene los departamentos de la dependencia, su descripción, así como su consumo.

dep_depto: Es una clave para identificar los departamentos.

dep_descrp: Es la descripción del departamento.

dep_cons_actu: Es el consumo actual del departamento.

dep_cons_acum: Es el consumo acumulado del departamento hasta la fecha de corte.

Tabla: EMPLEADO.

Es el catálogo que contiene a los empleados que pertenecen a los departamentos (que ya deben estar registrados en la tabla depto) y que serán los que realizarán los movimientos que se registren en el almacén.

emp_rfc: Es la clave del empleado.

emp_depto: Es la clave del departamento al cual pertenece, solo podrá ser si ya esta registrado en la tabla DEPTO.

emp_nombre: Es el nombre del empleado.

emp_cons_actu: Es el consumo actual del empleado.

emp_cons_acum: Es el consumo acumulado del empleado hasta la fecha de corte.

Tabla: ENTRADA.

Contiene la información de los artículos que entran en el almacén que van desde el precio, cantidad, fecha y referencia de la nota donde compró.

ent_cve_art: Es la clave del artículo que ya esta registrado en la tabla de ARTÍCULO.

ent_nota: Es el número de nota que se generó con la entrada del artículo al almacén, puede ser consecutiva.

ent_fecha : Es la fecha del registro de la nota de la entrada del artículo.

ent_precio: Es el precio del artículo (este precio debe ser de acuerdo a la unidad de medida con que se registró en la tabla de ARTÍCULO).

ent_cantidad: Es la cantidad de artículos que entraron al almacén.

ent_mov: Es el identificador del tipo de movimiento que se realizó, el cual puede ser Inventario Inicial (I) o entrada (E).

Tabla: SALIDA.

Contiene la información de los artículos que salen del almacén que van desde quién lo solicitó, departamento al que pertenece, el precio, cantidad, fecha y referencia del vale de salida.

sal_cve_art: Es la clave del artículo que ya esta registrado en la tabla de ARTÍCULO.

sal_rfc: Es la clave del empleado que solicito el artículo.

sal_vale: Es el número del vale que se generó con la salida del artículo del almacén, puede ser consecutiva.

sal_fecha : Es la fecha del registro del vale a la salida del artículo.

sal_cantidad: Es la cantidad de artículos que salieron del almacén.

sal_precio: Es el precio promedio del artículo que salió del almacén.

sal_depto: Es el departamento del empleado que solicito el artículo, debe estar registrado en la tabla DEPTO.

Tabla: ALMACÉN

Contiene la información referente a los movimientos que han sufrido los artículos, ya sea entradas o salidas, es donde se consulta la existencia de cada uno de éstos.

alm_cve_art: Es la clave del artículo.

alm_entrada: Es una cantidad acumulable de artículos que han entrado al almacén.

alm_salida: Es una cantidad acumulable de artículos que han salido del almacén.

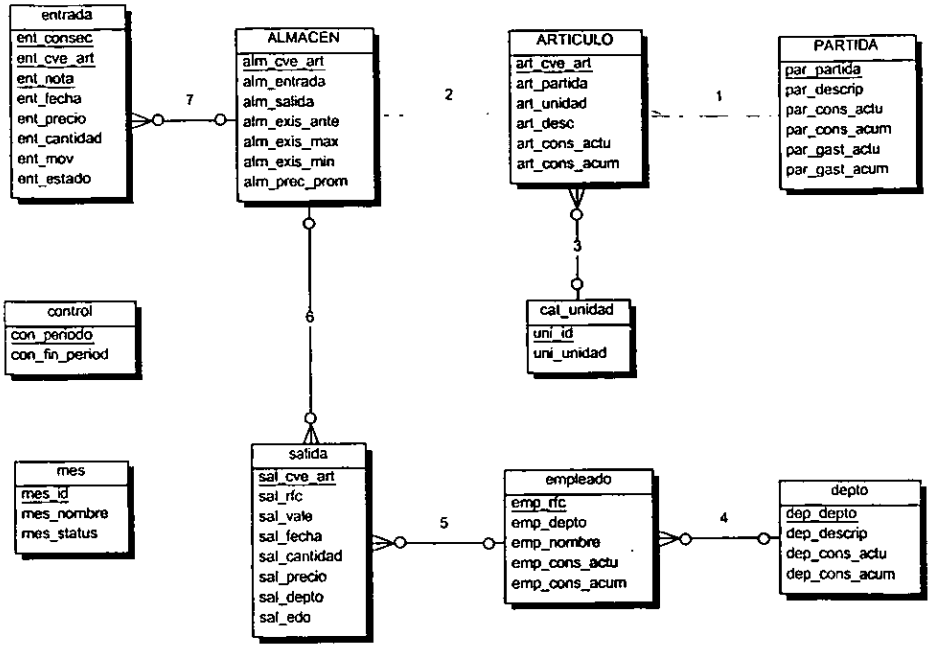
alm_ exis_ante: Es la existencia final de un periodo anterior (inventario inicial).

alm_exis_max: Es la existencia máxima que se puede almacenar de un artículo.

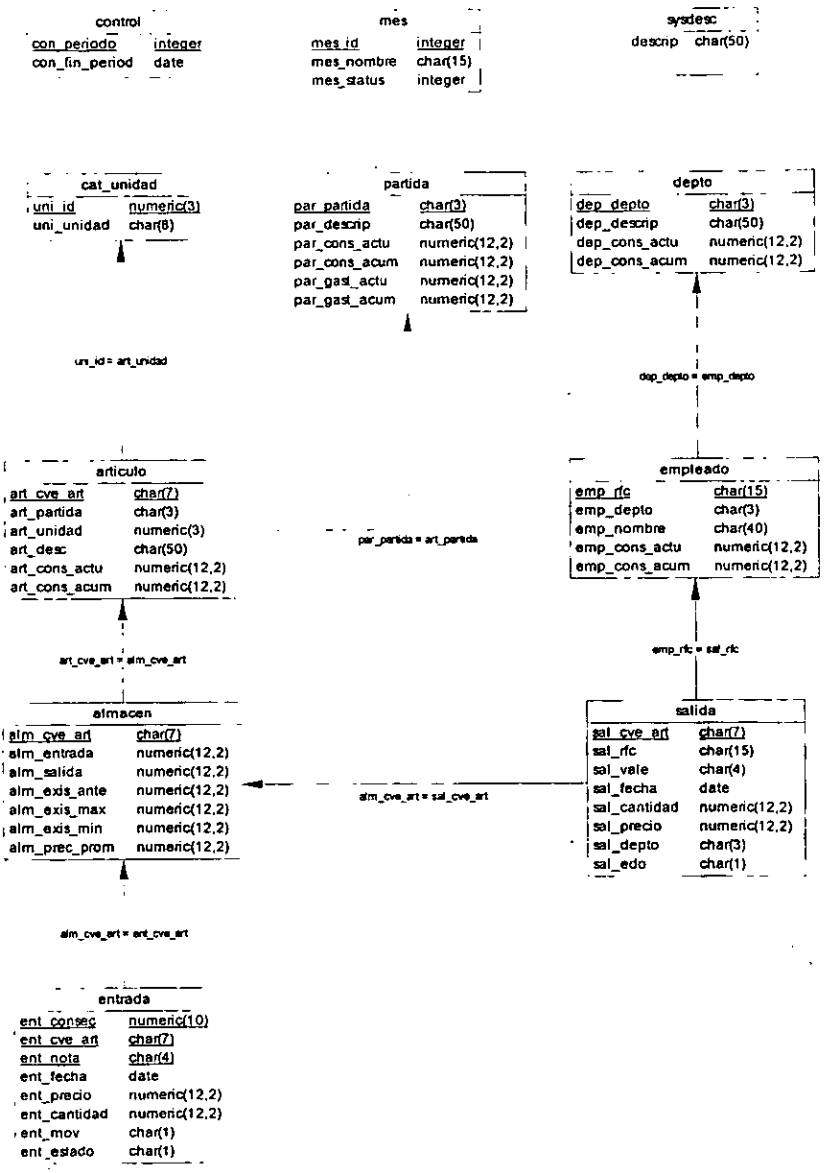
alm_exis_min: Es la existencia mínima que se puede almacenar de un artículo.

alm_prec_prom: Es el precio promedio del artículo.

2.5 Diagrama entidad relación



El diagrama entidad/relación anterior muestra las tablas que se utilizan en la base de datos del sistema de almacén, con sus respectivas relaciones entre si.



El diagrama entidad/relación anterior es otra variación del primero, con la diferencia que aquí se muestran los campos con los cuales existe la relación entre las tablas (llaves primarias y llaves foráneas).

2.6 Cohesión

El concepto de cohesión se puede entender como la medición de la fuerza de asociación de los elementos internos de los módulos, esto se toma como una extensión del ocultamiento de información. Idealmente un módulo coherente ejecuta una sola tarea y la realiza por medio de un procedimiento de programación, el cual necesita poca interacción con otros procedimientos que se ejecuten en otras partes del programa. En otras palabras, en situaciones ideales el módulo coherente sólo debe realizar una única tarea

La cohesión es una propiedad que se utiliza para medir la fuerza funcional de un módulo individual. Esto es, medir en que grado, el módulo realiza una sola función dentro del sistema y al terminar de ejecutarla, pasa el resultado a otro módulo. La cohesión interna de un módulo ocurre de una escala de la más débil (la menos deseada) a la más fuerte (la más deseada) en el siguiente orden:

Cohesión coincidental
Cohesión lógica
Cohesión temporal
Cohesión en la comunicación
Cohesión secuencial
Cohesión funcional
Cohesión informacional

2.6.1 Cohesión coincidental

Se presenta cuando los elementos internos de un módulo no tienen relación aparente entre ellos; por ejemplo, cuando un programa monolítico de gran tamaño es modularizado, esto es, separando en varias partes de forma arbitraria y considerarlas como módulos de menor tamaño. También es cuando se crea un módulo con un conjunto de instrucciones que no tengan una relación aparente. Es por esto que, un módulo que ejecute un conjunto de tareas que estén relacionadas con otras débilmente, tiene una cohesión coincidental.

2.6.2 Cohesión lógica

Este tipo de cohesión implica que la existencia de alguna actividad similar entre los elementos del módulo, por ejemplo, un módulo que desempeñe todas las funciones de entrada / salida y otro que se dedique a la edición general de los datos. Un módulo con cohesión lógica comúnmente combina varias funciones interrelacionadas en otros módulos.

2.6.3 Cohesión temporal

En este tipo de Cohesión todos los elementos son ejecutados en un momento dado, sin requerir de ningún parámetro o lógica alguna, para determinar que elementos deben ejecutarse.

2.6.4 Cohesión en la comunicación

Esta cohesión se presenta cuando algunos elementos de un módulo desarrollan diferentes funciones, pero cada función es alimentada por la misma entrada de datos. En otras palabras, esto es cuando varios elementos de un módulo se concentran sobre un área de una estructura de datos, en estos casos se presenta una cohesión de comunicación.

2.6.5 Cohesión secuencial

Esta cohesión ocurre cuando la salida de un elemento es la entrada para el siguiente. Cuando los elementos de procesamiento de un módulo están relacionados y deben ejecutarse en un orden especifico, existe cohesión secuencial.

2.6.6 Cohesión funcional

Esta cohesión representa un tipo fuerte, por lo tanto es deseable que exista, aquí todos los elementos de un módulo deben de estar relacionados al desempeño de una sola función. Si él módulo realiza una sola función se dice que tiene una cohesión funcional.

2.6.7 Cohesión informacional

Esta cohesión ocurre cuando el módulo contiene una estructura de datos compleja. Esta cohesión es la realización de la abstracción total de los datos.

2.7 Acoplamiento

El acoplamiento es una medida de la interconexión entre módulos en una estructura de programa. El acoplamiento depende de la complejidad de las interfaces entre los módulos, además del punto donde se hace la referencia a una entrada o algún módulo. También se considera a los datos que pasan a través de las interfaces. Al desarrollar un producto de software se busca que cumpla con el acoplamiento más bajo posible. La conectividad sencilla entre módulos da como resultado un software que es más fácil de comprender y menos propenso a efecto onda, causado cuando los errores ocurren en una parte del programa y se propagan a lo largo del sistema.

La fuerza del acoplamiento entre dos módulos esta influida por la complejidad de la interfaz y el tipo de comunicación existente entre ellos, a medida que es menor la complejidad de los elementos sé obtienen relaciones más sencillas de comprender a diferencia de los elementos más grandes y complejos que se hacen más difíciles es su comprensión. Por ejemplo, la modificación de un bloque común de datos o de control puede requerir de modificaciones en todas las rutinas que se encuentran acopladas a ese bloque; por otro lado, si los módulos se comunican, solamente por los parámetros y si las interfaces entre módulos permanecen constantes, los detalles internos de los módulos pueden ser modificados sin tener que modificar las rutinas que usan los módulos modificados.

El acoplamiento entre módulos puede ser considerado en una escala del más fuerte (el menos deseable) al más débil (él más deseable) de la siguiente forma:

Acoplamiento por contenido
Acoplamiento por zonas compartidas
Acoplamiento por control
Acoplamiento por zonas de datos
Acoplamiento por datos

2.7.1 Acoplamiento por contenido

Éste se presenta cuando un módulo modifica los valores locales o las instrucciones de otro módulo. Este acoplamiento se puede presentar en programas en lenguaje ensamblador.

2.7.2 Acoplamiento por zonas compartidas

Los módulos son amarrados en forma conjunta por medio de zonas globales para las estructuras de datos, en este caso la misma zona está disponible para ser utilizada por otros módulos.

2.7.3 Acoplamiento por control

Este acoplamiento incluye el paso de banderas de control y de parámetros en forma global entre los módulos, de tal forma que un módulo controla la secuencia de un proceso a otro.

2.7.4 Acoplamiento por zonas de datos

Es similar al de zonas compartidas, excepto que los elementos globales son compartidos en forma selectiva entre las diversas rutinas que requieren los datos

2.7.5 Acoplamiento por datos

Incluye el uso de listas de parámetros para pasar a los elementos entre rutinas. La forma más deseada de acoplamiento es ciertamente una combinación de zonas de datos y de acoplamiento de datos.

3

Codificación.

En el siguiente capítulo se hará una breve descripción técnica referente a los módulos que componen al sistema de almacén, es decir, se verá la codificación que es esencial para el buen funcionamiento del sistema. Para dicha descripción se tomarán las ventanas más representativas de cada módulo del sistema, sin embargo, primeramente se explicarán conceptos referentes al lenguaje de programación que se utilizó para el desarrollo de este sistema (Power Builder) a fin de facilitar un poco mejor la comprensión del presente capítulo.

3.1 Conceptos generales del entorno de programación

Debido a que en la codificación del sistema de almacenes se utilizan diferentes tecnicismos propios del lenguaje de programación (Power Builder), es necesario dar al lector una pequeña introducción acerca de éstos mismos, a fin de facilitar su comprensión a lo que se refiere, ya propiamente, al tema de este capítulo, que es la codificación.

3.1.1 Introducción a los DataWindows

El datawindow es una de las características más fuertes de Power Builder. Los Datawindows, que son como controles limitados a los datos, incorporan interfaces de usuario tales como editores de una línea y campos de texto estático, así como muchos de los accesos y actualizaciones que las aplicaciones hacen a la base de datos.

Una diferencia importante entre los DataWindows y las ventanas tradicionales que contienen controles como los mencionados, es que un datawindow se considera como un control único. Aunque pueda parecer una colección de controles para editores de una línea y campos de texto estático, realmente sólo muestra una representación de mapa de bits del diseño del datawindow. A pesar de lo complejo que parezca, es un control único con una

serie de sucesos asociados con él. Para un datawindow actualizable, se utiliza un segundo control de edición para tabular las columnas con posibilidad de actualización, de manera que se permita la introducción de datos. Por el uso del datawindow se puede reducir considerablemente el número de controles contenidos en una ventana y, por lo tanto, mejorar sensiblemente su rendimiento.

Los DataWindows realizan las recuperaciones y actualizaciones desde una base de datos en una forma más sencilla de aplicar que codificando querys (SQL) complejos.

3.1.2 Introducción a los sucesos (eventos)

La programación para eventos (sucesos) es un aspecto importante del desarrollo de aplicaciones cliente / servidor con interfaces gráficas de usuario. La programación tradicional de marco principal es procedimental, en la que el flujo de la lógica está en gran parte bajo el control del programa.

Las aplicaciones gobernadas por eventos responden a éstos, los cuales pueden ser acciones que el usuario emprende con el ratón o con el teclado (eventos de interfaz de usuario) y que pueden no seguir una secuencia lógica y predeterminada, o eventos iniciados dentro de la propia aplicación. Por lo anterior podemos definir a un evento como un suceso que responde a la acción de un usuario.

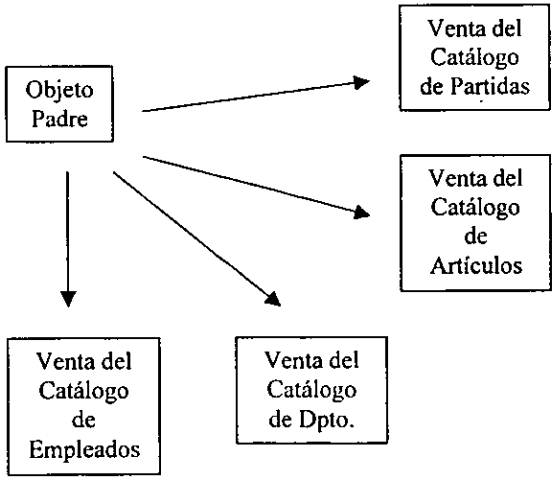
La codificación en Power Builder implica scripts (textos) escritos para los sucesos. Por ejemplo, si se pulsa en un botón de la barra de herramientas, se ejecutan ciertos códigos de PowerScript (lenguaje de Power Builder) en otro evento. La programación gobernada por eventos concede al usuario más control sobre el flujo de la aplicación, resultando así una de las muchas fortalezas de Power Builder.

Por otra parte, la definición de un objeto (clase) en Power Builder se realiza en un pintor de texto. Ventanas, menús, DataWindows y aplicaciones, son ejemplos de clases o definiciones de objetos. Los controles de ventanas son objetos contenidos dentro de otros objetos, normalmente ventanas. Los botones de mandato, los textos estáticos y los DataWindows son ejemplos de controles de ventana.

Cada control de ventana, así como las definiciones de objeto para ventanas, los menús, los objetos de usuario y el objeto de aplicación, pueden contener códigos escritos en PowerScript, llamados también scripts o textos.

3.2 Catálogos

El módulo de catálogos, del sistema de almacén, se compone por los siguientes elementos: partidas, artículos, departamentos y empleados. Los cuales corresponden, cada uno, a una ventana del sistema, que por la similitud del manejo de la información que tienen entre sí, se construyeron usando una metodología en común, utilizando los conceptos de herencia que nos permite la programación orientada a objetos y que se explicó en un capítulo anterior, es decir, a partir de un objeto padre, se fueron construyendo los objetos hijos, que en este caso serían las ventanas correspondientes al módulo de catálogos.



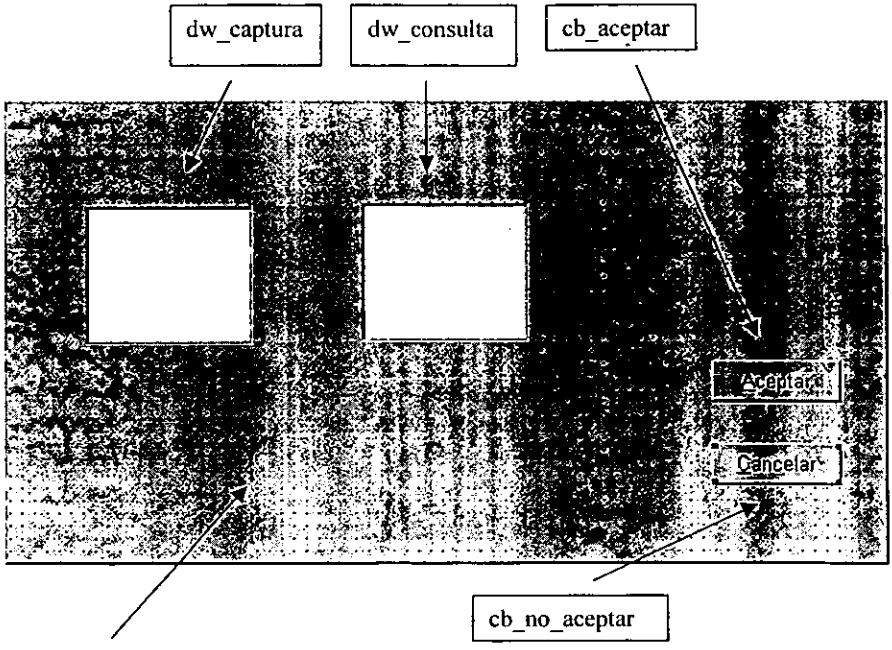
El objeto padre, tiene a su vez, dentro de él, a otros objetos, así como eventos y código que heredarán los objetos hijos, en este caso las ventanas del módulo, y cada uno de los hijos tendrán su propio código de acuerdo a la información que tengan que manejar cada uno de éstos.

Ya entrando más a detalle, el objeto padre que construimos para él módulo de catálogos, ya en el entorno del lenguaje de programación Power Builder, consta de cuatro objetos y seis eventos que podríamos denominar de

usuario, es decir, que fueron creados por los programadores para fines específicos y no son propios del objeto padre. Dichos objetos y eventos son los siguientes.

Objetos:

- dw_captura
- dw_consulta
- cb_aceptar
- cb_no_aceptar



Ventana Padre

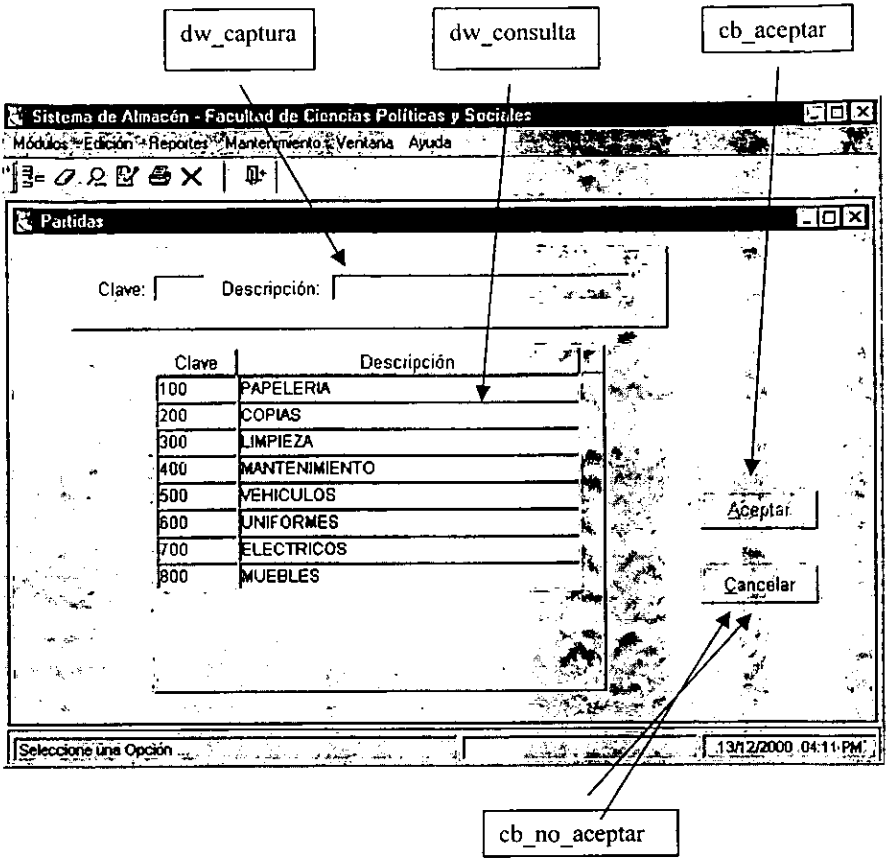
Eventos:

- aceptar
- borrar
- consultar
- insertar
- modificar
- imprimir

Retomando otra vez el tema referente al objeto padre, diremos que los objetos "dw_captura" y "dw_consulta" son objetos datawindow, los cuales son controles limitados para los datos, que incorporan interfaces de usuario tales como editores de línea y campos de texto estáticos, así como muchos de los accesos y actualizaciones que la aplicación hace a la base de datos. Cabe mencionar que todos los objetos que están dentro del ambiente de power builder son portadores de sus eventos nativos, por llamarles de alguna manera, y el programador codifica los scripts para dichos eventos, y en los datawindows no es la excepción.

Para complementar todo lo explicado anteriormente veremos un ejemplo de una ventana ya terminada, que en este caso será la correspondiente al catálogo de partidas, en donde veremos más a detalle el código asociado a los eventos de los objetos que hemos hecho mención.

La ventana u objeto denominado w_partidas tiene todas las características que heredó de la ventana u objeto padre, nada más que con su propio concepto, en este caso orientado al manejo de la información de la tabla "partidas" de la Base de Datos.



Si se observa con detalle, en la ilustración anterior se muestra la ventana de partidas ya construida, en donde los datawindows ya muestran la interfaz gráfica con las tablas de las bases de datos, correspondientes a los datos de las partidas (tabla partidas). Al abrirse la ventana por primera vez, los datawindows ejecutan el suceso contenido en el evento "constructor" propio de los mismos datawindow, los cuales contienen el siguiente código:

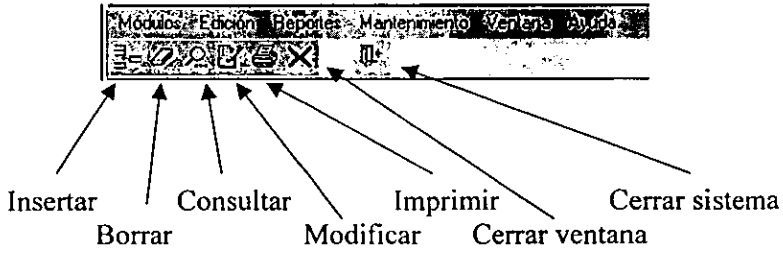
```
this.SetTransObject(SQLCA)
```

Éste código tiene la función de establecer la conexión de la base de datos con las interfaces gráficas.

El datawindow "dw_captura", como su nombre lo indica, nos servirá para que el usuario introduzca los datos en las opciones de inserción y

modificación. A su vez el datawindow "dw_consulta", nos servirá para hacer las consultas de los registros y/o información contenida en la tabla de partidas.

Por otra parte, en la siguiente ilustración se podrá observar la barra del menú que aparece en la tabla de partidas.



Como se observa, en la barra de herramientas del menú podemos acceder a los distintos eventos contenidos en nuestra ventana "w_partidas" las cuales tienen distintos scripts (código), dependiendo de la opción que se elija, así por ejemplo, si el usuario presiona el botón insertar se ejecutará el evento insertar, con el siguiente código:

```
is_opcion = 'A'

wf_asigna_tab(is_opcion)
wf_asigna_menu(is_opcion)
cb_aceptar.enabled = true
cb_no_aceptar.enabled = true
w_mdi_comp2.uo_sle_gral.sle_gral1.text = " Opción de Inserción de Partidas ..."

dw_captura.SetColumn ("par_partida")
dw_captura.setfocus()
```

La primera línea es una asignación de un valor tipo carácter 'A' a una variable de instancia. Se denomina variable de instancia a aquellas que conservan su valor mientras la ventana permanezca abierta y podrá ser utilizada por otros objetos y eventos que pertenezcan a dicha ventana. Entonces, la variable is_opcion será reconocida en cualquier script de cualquier objeto de la ventana, y en este caso se utiliza como si fuera una bandera que nos indica el estado en el que se encuentra trabajando la ventana, en este caso el estado de inserción es asociado con la letra 'A'. Posteriormente, encontramos la siguiente línea:

wf_asigna_tab(is_opcion)

la cual es una función implementada por los desarrolladores del sistema, la cual tiene la tarea de habilitar o deshabilitar los ítems o recuadros en los cuales el usuario tiene acceso a escribir y/o a consultar información. A dicha función hay que pasarle un parámetro de entrada, el cual, en este caso fue la variable de instancia que anteriormente mencionamos, "is_opcion", que en este caso tiene asignado el valor del carácter 'A' (opción de alta u/o inserción de datos), con lo cual se entiende que se habilitarán los recuadros para que el usuario pueda escribir en ellos. Esta misma función se usará también para los casos de modificación o borrado de la información, para los cuales la variable "is_opcion" tomará los valores de 'M' y/o 'B', respectivamente. Por lo tanto, cuando desde el menú se seleccione, por ejemplo, la opción de modificar, únicamente se habilitarán los recuadros en los cuales es permitido realizar las modificaciones, y lo mismo es para el caso de las consultas y borrado de registros.

La función wf_asigna_tab() ejecuta el siguiente código o script:

```
string ls_oscuro, ls_claro      //Declaración de variables
long ll_blanco

ls_oscuro = String(this.backcolor)    //Asignación del color para el fondo
ll_blanco = RGB(255,255,255)
ls_claro = String(ll_blanco)   //Asignación del color blanco


CHOOSE CASE operacion  //opción case para seleccionar múltiples opciones

CASE "A", "M"


    dw_captura.SetTabOrder("par_partida", 1)  //habilita campo del datawindow
    dw_captura.SetTabOrder("par_descrip", 2)

    dw_captura.Modify("par_partida.Background.Color=" + ls_claro +"")
    dw_captura.Modify("par_descrip.Background.Color=" + ls_claro +"")

    // Asignan colores a los campos del datawindow

        CASE "N"

                dw_captura.SetTabOrder("par_partida", 0)
```

```
            dw_captura.SetTabOrder("par_descrip", 0)

            dw_captura.Modify("par_partida.Background.Color='" + ls_oscuro +"'")
            dw_captura.Modify("par_descrip.Background.Color='" + ls_oscuro +"'")

    CASE "C"

            dw_captura.SetTabOrder("par_partida", 1)
            dw_captura.SetTabOrder("par_descrip", 0)

            dw_captura.Modify("par_partida.Background.Color='" + ls_claro +"'")
            dw_captura.Modify("par_descrip.Background.Color='" + ls_oscuro +"'")

END CHOOSE
```

La función wf_asigna_menu(), que es la que sigue en el código o script que estamos revisando, tiene la función de habilitar y deshabilitar las opciones del menú principal, es decir, se habilitarán las opciones del menú que requiera la ventana y obviamente las que no sean requeridas quedarán inhabilitadas. El código de dicha función es el siguiente:

```
choose case operacion

    case 'N'
            m_almacen.m_modulos.m_cerrar.enabled    = true
            m_almacen.m_modulos.m_catalogos.m_partidas.enabled  = false
            m_almacen.m_edicion.m_insertar.enabled  = true
            m_almacen.m_edicion.m_modificar.enabled = true
            m_almacen.m_edicion.m_consultar.enabled = true
            m_almacen.m_edicion.m_borrar.enabled = true
            m_almacen.m_modulos.m_imprimir.enabled    = true

    case 'A'
            m_almacen.m_modulos.m_cerrar.enabled    = true
            m_almacen.m_edicion.m_insertar.enabled  = false
            m_almacen.m_edicion.m_modificar.enabled = false
            m_almacen.m_edicion.m_consultar.enabled = false
            m_almacen.m_edicion.m_borrar.enabled = false
            m_almacen.m_modulos.m_imprimir.enabled = false

end chose
```

en el código anterior se observa, en cada línea, la manera en como se habilita y deshabilita un objeto, que en este caso fue el menú. La sintaxis para cualquier tipo de objeto es la siguiente:

"nombre del objeto".enabled = "true / false"

en donde

"true" habilita y "false" deshabilita
Enabled es la propiedad del objeto para habilitar y deshabilitar.

Para concluir, mencionaremos la funcionalidad de las últimas líneas que aparecen en el script que estamos analizando, las cuales son las siguientes:

```
w_mdi_comp2.uo_sle_gral.sle_gral1.text = " Opción de Inserción de Partidas ..."
```

En la anterior línea se esta cambiando la propiedad "text", es decir, se le esta cambiando el texto al objeto y al igual que la propiedad "enabled", que se vio anteriormente, la sintaxis consiste en poner el nombre, la propiedad y en este caso el nuevo texto que se le asignará al objeto.

```
dw_captura.SetColumn ("par_partida")
dw_captura.setfocus()
```

Las dos líneas anteriores tienen la función, en conjunto, de posicionar el cursor en el item o recuadro que esta asociado a una columna de la tabla asociada al datawindow.

Después de dar una explicación general del script del evento "insertar" solo nos resta decir que el manejo para los eventos "modificar", "borrar" y "consultar" es muy similar, ya que lo que se pretende hacer con este código es solamente dar un ambiente a la ventana de acuerdo a la opción que el usuario ha seleccionado.

Cabe mencionar que una vez que el usuario a ejecutado alguna de las opciones de la ventana (insertar, modificar, consultar, borrar) éste tiene que empezar a manipular la información dependiendo de la opción que haya escogido y posteriormente debe presionar el botón "aceptar", el cual disparará el evento "aceptar" y se ejecutará la parte correspondiente del script que corresponda a la opción seleccionada. Dicho script es el siguiente:

```
long ll_row_cap                    // Declaración de las variables del script
int li_resul_upd1
String ls_descrip, ls_partida
integer li_response

ll_row_cap = dw_captura.GetRow()          // obtiene renglón activo del datawindow

CHOOSE CASE is_opcion              // Opción case para seleccionar opción

CASE 'A'    // Altas

        IF dw_captura.AcceptText() = 1 THEN
                is_partida = dw_captura.GetItemString(ll_row_cap,"par_partida")
                ls_descrip = dw_captura.GetItemString(ll_row_cap,"par_descrip")
        Else
                return
        END IF


        If isnull(is_partida)  or len(is_partida) = 0 Then
                MessageBox ("Precaución ", "Debe Introducir los Datos que se Piden. ",
Exclamation!)
                Return
        End If

        if len(is_partida) < 3 then
                        messagebox('Precaución','Partida no válida', Exclamation!)

                        return
        end if

        select par_partida into :ls_partida
                from partida
                        where par_partida = :is_partida;

        IF ls_partida = is_partida THEN
                MessageBox ("Precaución ", "Ya existe una partida con la misma clave. ",
Exclamation!)
                Return
        END IF

        If isnull(ls_descrip)  or len(ls_descrip) = 0 Then
                MessageBox ("Precaución ", "Debe Introducir los Datos que se Piden. ",
Exclamation!)
                Return
        End If
```

```
    li_resul_upd1 = dw_captura.Update()

            w_mdi_comp2.uo_sle_gral.sle_gral1.text = " Actualizando la BD ..."

            IF li_resul_upd1 = 1 THEN
                COMMIT using SQLCA     ;
            ELSE
                ROLLBACK using SQLCA ;
                MessageBox("Mensaje de Error", "Operación NO efectuada.
Informe al Administrador de la Base de Datos. Gracias.", StopSign!)
        return
            END IF

MessageBox("Atencion", "Operación efectuada exitosamente.", Exclamation!)

this.triggerevent("open")
this.triggerevent("insertar")

CASE 'M'

        if ii_modifica = 0 then

            if ii_verifica = 0 then
                is_partida = dw_captura.GetItemString(ll_row_cap,"par_partida")
          ii_verifica = 0
            end if

            If isnull(is_partida)  or len(is_partida) = 0 Then
                MessageBox ("Precaución ", "Debe Introducir los Datos que se
Piden. ", Exclamation!)
                Return
            End If

            ll_row_cap = dw_captura.Retrieve(is_partida)

            IF ll_row_cap < 1 THEN
                this.triggerevent("modificar")
                MessageBox("Error", "No existe información.", Exclamation!)

                RETURN
            end if

            wf_asigna_tab('M')
            ii_modifica = 1
            is_ver_partida = is_partida

else
```

```
        IF dw_captura.AcceptText() = 1 THEN
                is_partida = dw_captura.GetItemString(ll_row_cap,"par_partida")
                ls_descrip = dw_captura.GetItemString(ll_row_cap,"par_descrip")
        Else
                return
        END IF


        If isnull(ls_descrip)  or len(ls_descrip) = 0 Then
                MessageBox ("Precaución ", "Debe Introducir los Datos que se Piden. ",
Exclamation!)
                Return
        End If

        if is_partida <> is_ver_partida then

        select par_partida into :ls_partida
                from partida
                        where par_partida = :is_partida;

        IF ls_partida = is_partida THEN
                MessageBox ("Precaución ", "Ya existe una partida con la misma clave. ",
Exclamation!)
                Return
        END IF

        end if
    li_resul_upd1 = dw_captura.Update()

                w_mdi_comp2.uo_sle_gral.sle_gral1.text = " Actualizando la BD ..."

                IF li_resul_upd1 = 1  THEN
                        COMMIT using SQLCA      ;
                ELSE
                        ROLLBACK using SQLCA  ;
                        MessageBox("Mensaje de Error", "Operación NO efectuada.
Informe al Administrador de la Base de Datos. Gracias.", StopSign!)
      return
                END IF

MessageBox("Atencion", "Operación efectuada exitosamente.", Exclamation!)

this.triggerevent("open")
this.triggerevent("modificar")

end if
```

```
CASE 'B'

        cb_aceptar.text = "&Borrar"

        if ii_modifica = 0 then

                if ii_verifica = 0 then
                        is_partida = dw_captura.GetItemString(ll_row_cap,"par_partida")
          ii_verifica = 0
                end if

                If isnull(is_partida)  or len(is_partida) = 0 Then
                        MessageBox ("Precaución ", "Debe Introducir los Datos que se
Piden. ", Exclamation!)
                        Return
                End If

                ll_row_cap = dw_captura.Retrieve(is_partida)

                IF ll_row_cap < 1 THEN
                        this.triggerevent("modificar")
                        MessageBox("Error", "No existe información.", Exclamation!)

                        RETURN
                end if

                wf_asigna_tab('N')
                ii_modifica = 1

else

        IF dw_captura.AcceptText() = 1 THEN
                is_partida = dw_captura.GetItemString(ll_row_cap,"par_partida")
        Else
                return
        END IF

        If isnull(is_partida)  or len(is_partida) = 0 Then
                MessageBox ("Precaución ", "Debe Introducir los Datos que se Piden. ",
Exclamation!)
                Return
        End If
//      Display a message to confirm the delete operation.
li_response = MessageBox(this.Title, "En realidad deseas borrar esta partida" ,  Question!,
YesNo!,2)
```

```
// If "yes", delete the current row.
IF li_response = 1 THEN

        delete from partida
        where par_partida = :is_partida;

        w_mdi_comp2.uo_sle_gral.sle_gral1.text = " Actualizando la BD ..."

                if SQLCA.sqlcode <> 0 then
                        ROLLBACK using SQLCA ;
                        MessageBox("Mensaje de Error", "Operación NO efectuada.
Informe al Administrador de la Base de Datos. Gracias.", StopSign!)
      return
                ELSE
                        COMMIT using SQLCA      ;
                END IF

  MessageBox("Atencion", "Operación efectuada exitosamente.", Exclamation!)

End if

        this.triggerevent("open")
        this.triggerevent("borrar")

End if

END CHOOSE
```

En el script anterior, primeramente, son declaradas las variables que se utilizarán a lo largo de este código y en donde serán almacenados los valores tanto de las entradas y salidas, como los que sean calculados por algunas operaciones que pudieran encontrarse en el script. Cabe mencionar, que el evento aceptar, es un evento que denominamos de usuario, ya que fue creado por éste mismo y el cual es usado en múltiples ventanas y en donde por lo regular controlamos el flujo de las opciones por medio de un controlador "Chose Case", que tiene la siguiente sintaxis:

```
CHOOSE CASE "expression  a evaluar"
        CASE "lista expresiones" "bloque de código"{ .
        CASE "lista expresiones" "bloque de código" . . .

        CASE "lista expresiones" "bloque de código" }
        CASE ELSE  "bloque de código" }
```

END CHOOSE

En donde:

- "Expresión a evaluar". Es la expresión con la cual se basa la ejecución del script.
- "Lista expresiones". Puede ser una de las siguientes expresiones:
  1. Un simple valor.
  2. Una lista de valores separados por comas (tal como 2, 4, 6, 8).
  3. La cláusula TO (tal como 1 TO 30).
  4. IS seguido por un operador de comparación relacional de valores (tal como IS>5).
  5. Cualquier combinación de las expresiones anteriores (tal como 1, 3, 5, 7, 9, 27 TO 33, IS >42).
- "Bloque de código". Es el bloque de código que se desea ejecutar si la expresión a evaluar corresponde con alguna opción en la lista de expresiones.

De lo anterior se puede analizar el script del evento aceptar, en donde el "Chose Case" quedo armado de la siguiente forma:

```
CHOOSE CASE is_opcion
      CASE "A"                  //Altas
            "bloque de código"
      CASE "M"                  //Modificación
            "bloque de código" . . .

      CASE "B"                  //Borrar
            "bloque de código" ;
END CHOSE
```

El código anterior nos muestra en una manera muy general la forma en que el script del evento aceptar esta dividido por las diferentes opciones que normalmente contempla cualquier ventana del sistema de almacenes (Altas, Modificaciones o Cambios y Borrado) y que cada uno de estos esta en un bloque de código y así todo el conjunto le dan la funcionalidad que presenta la ventana a la vista del usuario del sistema.

Ahora bien, del código correspondiente a la opción de altas (inserción de registros) se puede observar, en las primeras líneas, múltiples validaciones

a los datos que el usuario ha capturado en la ventana, a fin de evitar errores en la integridad de la información en la base de datos. Estas validaciones son fáciles de identificar, ya que por lo regular se usa un controlador de condiciones que en este caso es el "IF...THEN", cuya sintaxis es la siguiente:

IF "condición" THEN "accion1" {ELSE "accion2"}

En donde:

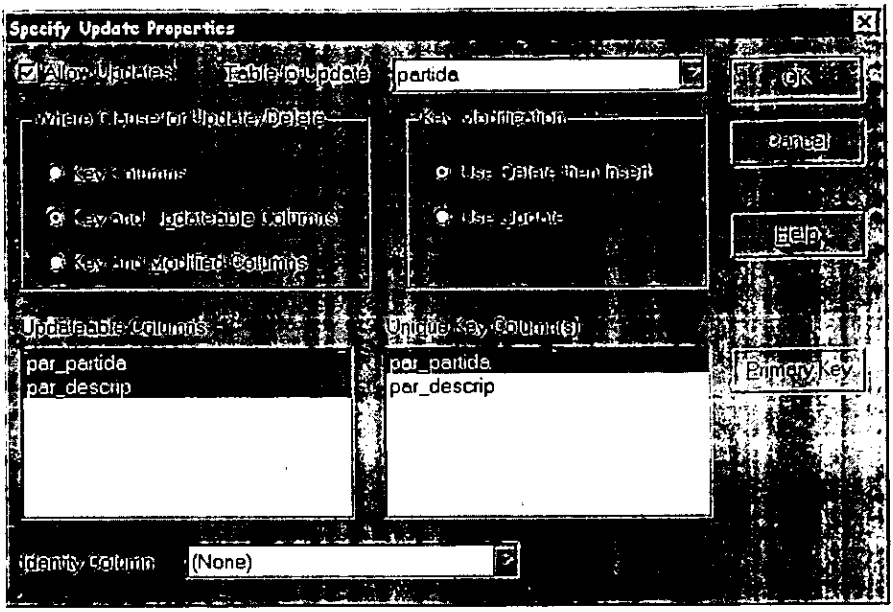
"Condición". Es la expresión condicional que se quiere emplear.
"accion1". Es la acción que se ejecutará si la condición resulta verdadera.
"accion2". Es la acción que se ejecutará si la condición resulta contraria.

Posteriormente, ya que se tiene la seguridad de que los datos introducidos son correctos, entonces se procede a la grabación de estos mismos a las tablas que le correspondan en la base de datos. En este caso, como se explico anteriormente, el datawindow, esta asociado directamente a una tabla que, para este ejemplo en particular, es la tabla de "partidas". A continuación se extraerá la porción de ese código a fin de analizarlo:

```
li_resul_upd1 = dw_captura.Update()

        IF li_resul_upd1 = 1  THEN
            COMMIT using SQLCA     ;
        ELSE
            ROLLBACK using SQLCA  ;
            MessageBox("Mensaje de Error", "Operación NO efectuada.
Informe al Administrador de la Base de Datos. Gracias.", StopSign!)
    return
        END IF
```

La primera línea nos muestra el uso de la función "Update()", la cual es la encargada de grabar lo que contiene el datawindow a la base de datos y es en las propiedades del datawindow donde se especifica a que tabla va a hacer referencia. A continuación se muestra la ventana de las opciones de PowerBuilder para manipular las propiedades del "update" del datawindow.



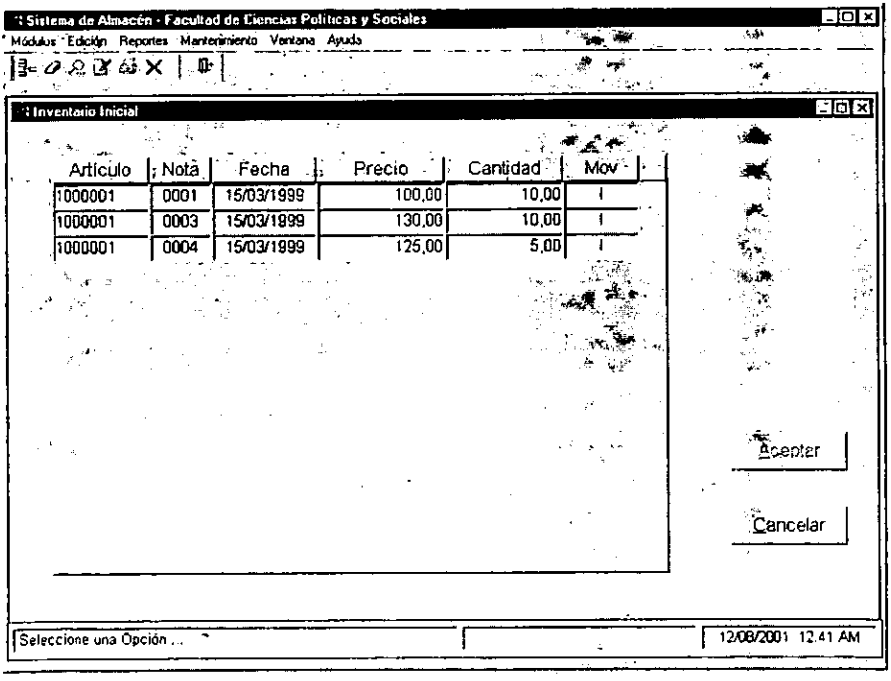
En esta pantalla se observan cuáles son las columnas que son grabablesde la tabla "partidas", así como cual es la llave primaria de ésta misma.

Continuando con el análisis del código, la función "update()" cuando es ejecutado en el código, ésta retorna un número "1" cuando la operación es exitosa y un "-1" cuando a ocurrido algún fallo. Este número es almacenado en la variable "li_resul_upd1" que se aprecia en el código. Posteriormente se observa una serie de validaciones en torno a esta misma variable y como es de suponerse la validación corresponde al éxito o fallo en la grabación de los datos en la base de datos, y es aquí donde aparecen dos líneas que corresponderían a código de Sybase incrustado en el script de PowerBuilder. Éstos son el "commit" y el "rollback", donde el primero graba todos los movimientos que se han hecho a la base de datos y el segundo hace todo lo contrario, es decir, deshace los cambios hechos en la base de datos, de ahí que estén dentro de la validación y en caso de que la operación de grabación de datos haya sido exitosa se utilice un "commit" y en caso contrario, se utilice un "rollback".

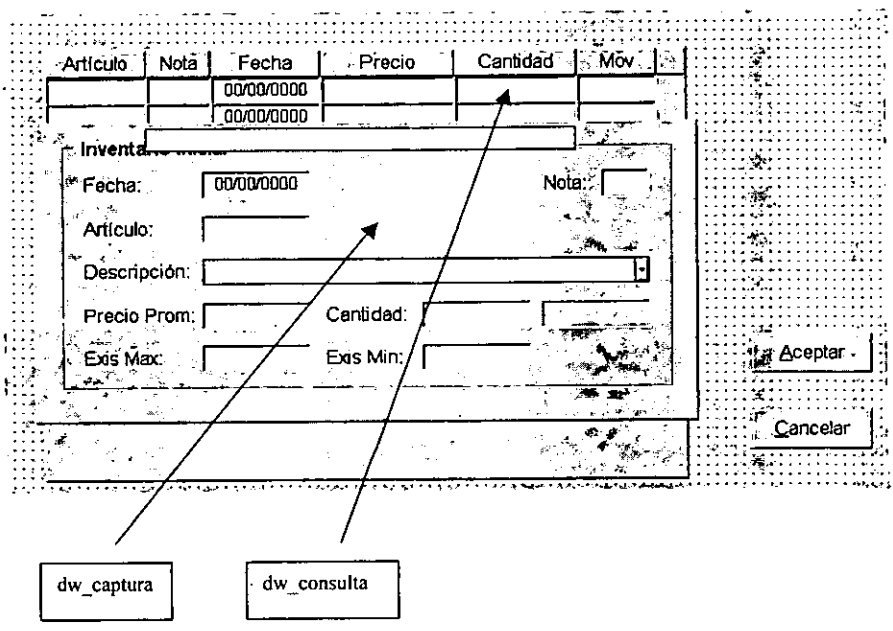
3.3 Actualización

Las ventanas correspondientes al módulo de actualización, a diferencia del módulo de catálogos, tienen las características singulares de que afectan a más de una tabla en la base de datos, por lo que en la codificación de éstas se emplearon métodos distintos a los presentados anteriormente. Esto es comprensible debido a que a diferencia del módulo de catálogos, en donde se emplearon datawindows como interfase de grabación, aquí no se utilizó solamente ese método sino que se emplearon opciones alternas, para así poder afectar a más de una tabla en cuanto a la grabación de la información, y es que los datawindows tienen la limitante de poder afectar únicamente a una sola tabla, en lo que se refiere al almacenamiento de datos.

En este inciso se analizará la ventana correspondiente al Inventario Inicial la cual, en el sistema, tiene el nombre "w_inv_ini", con la cual se hará una descripción del funcionamiento general de este tipo de ventanas que se incluyeron en el módulo de actualización y que en tiempo de ejecución del sistema de almacén se vería de la siguiente manera.

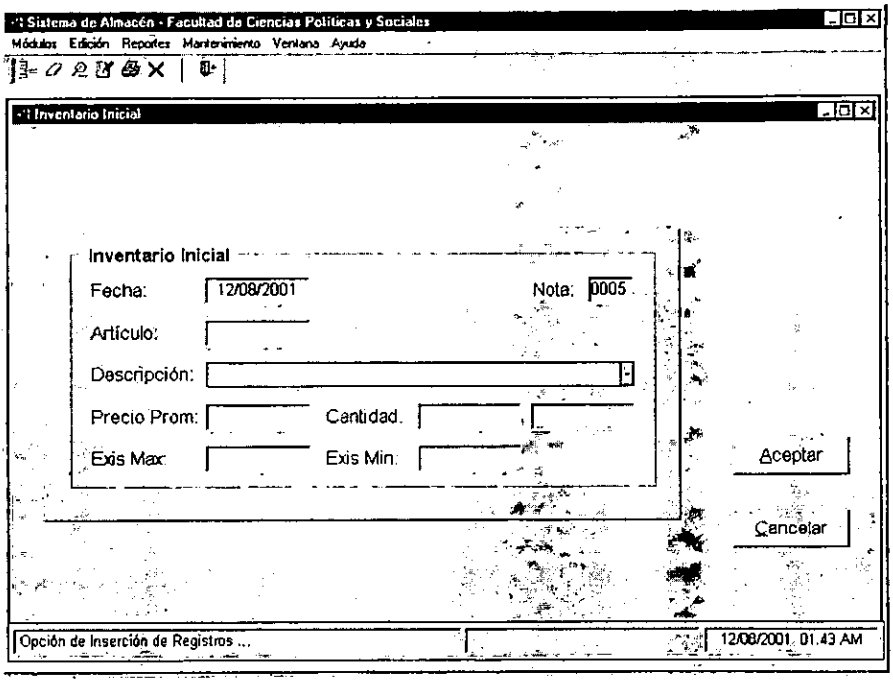


En la imagen se puede observar los objetos que componen a la ventana de inventario inicial, los cuales son: un objeto datawindow y dos botones. Sin embargo esta ventana es heredada de la ventana padre "w_catalogos" que se usa también en las ventanas correspondientes al módulo de catálogos y como se podrá recordar éstas contenían dos datawindows, sin embargo en esta ventana si existe otro objeto datawindow, pero no se muestra visible para los ojos del usuario sino hasta que se seleccione la opción correspondiente que hará visible a este objeto. Ahora bien, en la siguiente pantalla se verá la ventana en su etapa de construcción.



En la imagen anterior se pueden apreciar los dos datawindow de la ventana de inventario inicial, tanto el visible como el no visible, así como dos botones genéricos de la mayoría de las ventanas del sistema de almacenes. Como ya se habrá notado, cuando se abre la ventana de inventario inicial, únicamente aparece el datawindow "dw_consulta", esto es debido a que como primer paso, al presentarse la ventana, es mostrar los registros del inventario

inicial que existen en ese momento, es decir, la opción de consulta va implícita automáticamente al abrirse esta ventana. Y por otra parte, cuando se le da la opción de insertar un nuevo registro, la ventana se ve de la siguiente forma.



Aquí se aprecia como ahora el datawindow "dw_consulta" se ocultó y el datawindow "dw_captura" se hizo visible, y es aquí donde el usuario puede empezar a insertar los datos, y al finalizar se deberá presionar el botón "aceptar" el cual ejecutará el script del evento "aceptar" en forma similar que en las ventanas de catálogos. Dicho script es el siguiente:

```
long ll_row, ll_checa, ll_row_cap, ll_ver
date ld_fecha
string ls_nota, ls_cve_art, ls_error, ls_partida, ls_ver
dec{2} ldc_precio, ldc_entrada, ldc_exis_max, ldc_exis_min
int li_response, li_consec

/**************/
DECLARE sp_ins_inv_ini PROCEDURE FOR alm_ins_inv_ini
```

```
        @vl_cnt_fecha = :ld_fecha,
        @vl_cnt_nota  = :ls_nota,
        @vl_cve_art   = :ls_cve_art,
        @vl_precio    = :ldc_precio,
        @vl_entrada   = :ldc_entrada,
        @vl_exis_max  = :ldc_exis_max,
        @vl_exis_min  = :ldc_exis_min,
        @vl_partida   = :ls_partida,
        @vl_tipo          = 'I'
    USING SQLCA;

    DECLARE sp_del_inv_ini PROCEDURE FOR alm_del_inv_ini
        @vl_cve_art   = :ls_cve_art,
        @vl_precio    = :ldc_precio,
        @vl_entrada   = :ldc_entrada,
        @vl_partida   = :ls_partida,
        @vl_consec    = :li_consec,
        @vl_tipo      = 'I'
    USING SQLCA;

    ll_row= dw_captura.GetRow()

    IF dw_captura.AcceptText() = 1 THEN
            ld_fecha = today()
            ls_nota =  dw_captura.GetItemString(ll_row,"ent_nota")
            ls_cve_art =  dw_captura.GetItemString(ll_row,"alm_cve_art")
            ldc_precio = dw_captura.GetItemDecimal(ll_row,"ent_precio")
            ldc_entrada = dw_captura.GetItemDecimal(ll_row,"ent_cantidad")
            ldc_exis_max = dw_captura.GetItemDecimal(ll_row,"alm_exis_max")
            ldc_exis_min = dw_captura.GetItemDecimal(ll_row,"alm_exis_min")
            li_consec = dw_captura.GetItemNumber(ll_row,"ent_consec")
    else
            return
    end if

            ls_partida = left(ls_cve_art,3)

    CHOOSE CASE is_opcion


    CASE 'A'    // Altas


        if isnull(ls_cve_art) or len(ls_cve_art) = 0 or isnull(ldc_entrada) or
isnull(ldc_precio) &
          or isnull(ldc_exis_max) or isnull(ldc_exis_min) or isnull(ls_nota) or len(ls_nota)
= 0 then
```

```
            MessageBox('Precaución',"Introduzca los datos que se piden",
exclamation!)
            return
        end if

        w_mdi_comp2.uo_sle_gral.sle_gral1.text = " Actualizando la BD ..."

        EXECUTE sp_ins_inv_ini   ;
        IF SQLCA.SQLCode <> 0 THEN
            Messagebox(this.Title, SQLCA.SQLDBCode)
            CLOSE sp_ins_inv_ini      ;
            ROLLBACK using SQLCA;
            return
        END IF
        FETCH sp_ins_inv_ini INTO :ll_ver        ;
        CLOSE sp_ins_inv_ini      ;
        COMMIT using SQLCA ;

        ii_flag_nota = 1   //controla la secuencia del numero de nota

        //              this.triggerevent('open')
                        this.triggerevent('insertar')

        CASE 'B' //Bajas


            if ii_modifica = 0 then

              ll_row_cap = dw_captura.Retrieve(is_art, ii_consec)

                IF ll_row_cap < 1 THEN
                    this.triggerevent("borrar")
                    MessageBox("Error",     "No     existe     información.",
Exclamation!)
                    RETURN
                end if

                dw_captura.visible = true
                dw_consulta.visible = false
                cb_aceptar.text = '&Borrar'
                ii_modifica = 1

            else

            select max(ent_mov) into :ls_ver
             from entrada
```

```
                where ent_mov = 'E';

        if ls_ver = 'E' then
                MessageBox("Error", "Existen Notas de Entrada para este articulo.",
Exclamation!)
                RETURN
        end if
                //      Display a message to confirm the delete operation.
        li_response = MessageBox(this.Title, "En realidad desea borrar este
inventario inicial" , Question!, YesNo!,2)

        // If "yes", delete the current row.
        IF li_response = 1 THEN

        w_mdi_comp2.uo_sle_gral.sle_gral1.text = " Actualizando la BD ..."

                EXECUTE sp_del_inv_ini  ;
                        IF SQLCA.SQLCode <> 0 THEN
                                Messagebox(this.Title, SQLCA.SQLDBCode)
                                CLOSE sp_del_inv_ini      ;
                                ROLLBACK using SQLCA;
                                return
                        END IF
                FETCH sp_del_inv_ini INTO :ll_ver;
                        CLOSE sp_del_inv_ini      ;
                COMMIT using SQLCA ;


                cb_aceptar.text = '&Aceptar'
                ii_modifica = 0
                this.triggerevent('open')
                this.triggerevent('borrar')

        end if
end if

end choose
```

El script anterior se divide en dos bloques que corresponden a las opciones de insertar y borrar, los cuales están definidos por un controlador "CHOOSE CASE" que determina la secuencia a seguir del código.

En esta parte hay que hacer un gran paréntesis, debido a que en el script anterior se puede observar código que en determinado momento se podría

decir que es ajeno a PowerBuilder, sin embargo, se trata de código de Sybase (Base de Datos), que interactúa con el código de PowerBuilder y que se puede distinguir debido a que al término de su sintaxis se le agrega un punto y coma (;). El usar este tipo de código en los scripts de PowerBuilder presenta una ventaja, ya que se pueden hacer accesos directos a la base de datos y en este caso se está aprovechando de manera considerable. Ahora bien, en Sybase, es muy frecuente la programación de Store Procedures (SP's) que son como pequeños programas, por llamarles de alguna forma, que contienen instrucciones en código propio de Sybase que nos permiten manipular la información de la base de datos. Se puede decir que los SP's también se podrían considerar como objetos de una base de datos, tal como lo es una tabla, ya que también a éstos hay que asignarles permisos para usuarios de la base de datos, tal como se hace también con las tablas. Estos SP's pueden ser ejecutados en cualquier evento de PowerBuilder, y en el caso de la ventana de inventario inicial se aprovechan los beneficios de éstos. En el script que estamos revisando los SP's son declarados casi al principio de la siguiente manera:

```
DECLARE sp_ins_inv_ini PROCEDURE FOR alm_ins_inv_ini
      @vl_ent_fecha = :ld_fecha,
      @vl_ent_nota  = :ls_nota,
      @vl_cve_art   = :ls_cve_art,
      @vl_precio    = :ldc_precio,
      @vl_entrada   = :ldc_entrada,
      @vl_exis_max  = :ldc_exis_max,
      @vl_exis_min  = :ldc_exis_min,
      @vl_partida   = :ls_partida,
      @vl_tipo          = 'I'
USING SQLCA;

DECLARE sp_del_inv_ini PROCEDURE FOR alm_del_inv_ini
      @vl_cve_art   = :ls_cve_art,
      @vl_precio    = :ldc_precio,
      @vl_entrada   = :ldc_entrada,
      @vl_partida   = :ls_partida,
      @vl_consec    = :li_consec,
      @vl_tipo      = 'I'
USING SQLCA;
```

A los SP's, generalmente, se les asigna parámetros de entrada y en ocasiones también retornan valores, semejantes a una función y en el código anterior se observa la declaración de los SP's llamados "sp_ins_inv_ini" y

"sp_del_inv_ini", en donde el primero se utiliza en la opción de insertar y el segundo en la opción de borrado de registros.

Otra cosa que hay que mencionar que los nombres de los SP's pueden ser iguales o diferentes, es decir, por poner un ejemplo en las declaraciones anteriores en el script de PowerBuilder se tiene el nombre del SP como "sp_ins_inv_ini", sin embargo en sybase este mismo SP se llama "alm_ins_inv_ini", pero esto ya depende del programador si desea conservar el sp con el mismo nombre o bien, cambiárselo, como fue el caso del ejemplo.

A continuación se verá el código del SP "sp_ins_inv_ini" a fin de analizarlo posteriormente:

```
create procedure alm_ins_inv_ini(
        @vl_ent_fecha date,
        @vl_ent_nota  char(4),
        @vl_cve_art   char(7),
        @vl_precio    numeric(12,2),
        @vl_entrada   numeric(12,2),
        @vl_exis_max  numeric(12,2),
        @vl_exis_min  numeric(12,2),
        @vl_partida   char(3),
        @vl_tipo                char(1)) AS

declare
        @gast_actu decimal(12,2),
        @entrada   numeric(12,2),
        @consec    numeric(10,0),
        @prec_prom numeric(12,2),
        @cost_prom numeric(12,2)

if @vl_tipo = 'I' .
        Begin
           insert into almacen(alm_cve_art, alm_entrada, alm_salida, alm_exis_max,
                                                        alm_exis_min,  alm_prec_prom)
                values(@vl_cve_art, @vl_entrada, 0, @vl_exis_max, @vl_exis_min,
                                                                    @vl_precio)
                if @@error != 0
                        return -1
        end
else
        begin
```

```
            select IsNull(alm_entrada,0), IsNull(alm_prec_prom,0) into @entrada,
            @prec_prom
                        from almacen
                                where alm_cve_art = @vl_cve_art
                    if @@rowcount = 0
                            return -1

                    select @cost_prom = (@entrada * @prec_prom) + (@vl_entrada *
@vl_precio)
                    select @entrada = @vl_entrada + @entrada
                    select @prec_prom = @cost_prom / @entrada

            update almacen
                    set alm_entrada = @entrada,
                      alm_prec_prom = @prec_prom
                        where alm_cve_art = @vl_cve_art
            if @@error != 0
                    return -1

        end

            select @consec = 1 + (select IsNull(max(ent_consec),0)
                                                from entrada)


        insert into entrada(ent_consec, ent_cve_art, ent_nota, ent_fecha, ent_precio,
        ent_cantidad, ent_mov)
                values(@consec, @vl_cve_art, @vl_ent_nota, @vl_ent_fecha, @vl_precio,
                @vl_entrada, @vl_tipo)
                        if @@error != 0
                                return -1

        select @gast_actu = (select IsNull(par_gast_actu,0)
                from partida
                        where par_partida = @vl_partida)
        if @@rowcount = 0
                return -1

                select @gast_actu = (@gast_actu + (@vl_precio * @vl_entrada))

                update partida
                        set par_gast_actu = @gast_actu
                                where par_partida = @vl_partida
        if @@error != 0
                return -1

        select @vl_entrada
```

```
return;
```

El código anterior nos muestra la forma en que están estructurados los Store Procedures, que como se puede observar es un conjunto de querys de SQL pero con una secuencia lógica y además de que se pueden declarar variables dentro de éste. Así, al principio del código son declaradas las variables de entrada, que como se podrá recordar también fueron introducidas al momento de declarar el SP en el script del evento "aceptar" de la ventana de inventario inicial, por lo cual deben de coincidir tanto en el nombre como en el número, ya que de lo contrario habría un fallo al momento de la ejecución. Posteriormente son declaradas las variables que se utilizarán para almacenar datos de entrada y salida que se obtengan durante la ejecución del código.

Las principales operaciones que se hacen en el anterior SP son las de selección, inserción y actualización de datos en las tablas pertenecientes a la base de datos y cuya sintaxis es la siguiente respectivamente:

```
Select campo1, campo2,....campoN
    From "tabla1","tabla2",...."tablaN"
        Where "condición"
```

```
Insert into "nombre tabla"(campo1, campo2,.....campoN)
    Values(valor1, valor2,.....valorN)
```

```
Update "nombre tabla"
    Set "campo1" = "valor1",
        "campo2" = "valor2",
            :
            :
        "campoN" = "valorN"
            Where "condición"
```

En términos generales, el funcionamiento del SP se basa en seleccionar, insertar y actualizar en las tablas de la base de datos la información del sistema de almacén dependiendo de los datos de entrada que el usuario digitó

en la ventana, haciendo de este proceso una forma ágil de manipular la información.

Pero regresando al código del script del evento "aceptar" veremos como es la ejecución del Store Procedure, para lo cual tomaremos ese bloque de código para ejemplificarlo y posteriormente explicarlo:

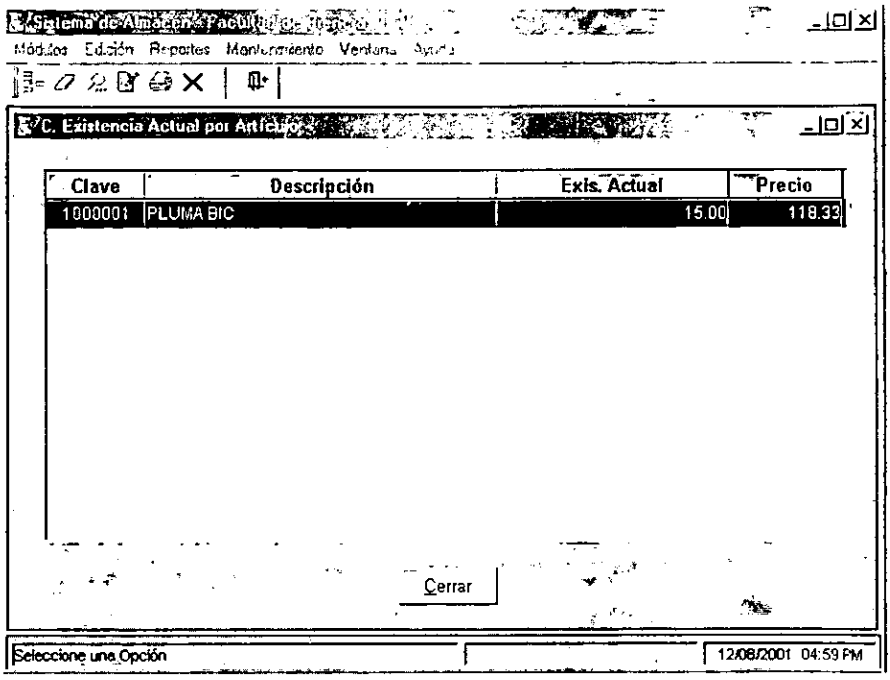
```
EXECUTE sp_ins_inv_ini   ;
IF SQLCA.SQLCode <> 0 THEN
        Messagebox(this.Title, SQLCA.SQLDBCode)
        CLOSE sp_ins_inv_ini        ;
        ROLLBACK using SQLCA;
        return
END IF
FETCH sp_ins_inv_ini INTO :ll_ver        ;
CLOSE sp_ins_inv_ini        ;
COMMIT using SQLCA ;
```

El comando "EXECUTE" es el que pone en ejecución al Store Procedure, pero es preciso que antes de llegar a esta línea del código, ya se hayan alimentado las variables que pasarán los parámetros al SP y que se pueden observar en la declaración de éste, porque de lo contrario, la variable de transacción SQLCA de PowerBuilder retornará un código de error, que como se puede observar es la siguiente línea después de la ejecución y que es una validación preguntado por el valor que se guardo en la variable "SQLCA.SQLcode" que en caso de retornar un "0" indicará que no hubo ningún problema en la ejecución del SP y de lo contrario regresará un número que corresponde a un código de error el cual puede ser variado dependiendo del problema que se haya presentado. Además también se usan los comandos "commit" o "rollback" para grabar o deshacer los datos, dependiendo de la validación anterior. Los SP's, en algunos casos, retornan un valor, el cual se puede recuperar mediante un comando llamado "FETCH", y en el ejemplo anterior el valor que retorna el "sp_ins_inv_ini" es depositado en la variable "ll_ver".

3.4 Consultas

En el sistema existen ventanas que su única función es mostrarle al usuario la información específica que éste requiere, es decir, son ventanas de consultas de la información contenida en la base de datos, y si por un lado

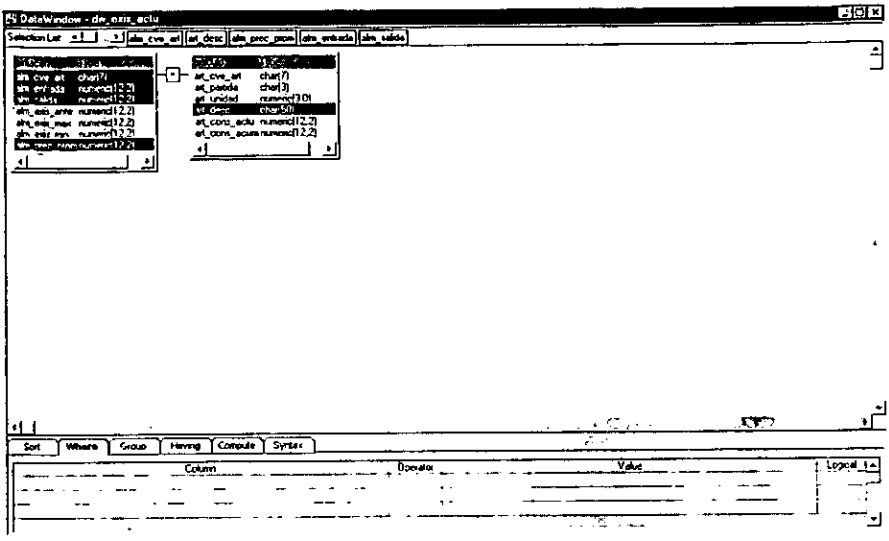
analizamos las ventajas que nos ofrecen los datawindows veremos que se pueden hacer consultas a las tablas de la base de dato con una facilidad que no sería necesario escribir gran número de líneas de código. Así para ejemplificar este inciso referente al módulo de consultas, tomaremos la ventana correspondiente a "Consulta de existencia por artículo" y cuyo nombre del objeto ventana en el sistema corresponde a "w_exis_actu", y que al correr el sistema de almacenes se vería de la siguiente forma.



En esta ventana se puede observar que en su conjunto, esta constituida por un objeto datawindow y un botón, y que vienen de una ventana ancestro llamada "w_consultas_grales", y tan como ya se ha mencionado esta ventana no necesitó de una amplia codificación, debido a que se aprovecharon las ventajas del datawindow que puede hacer una consulta directa a la base de datos y mostrar los datos al usuario.

Los datawindows, como se ha mencionado anteriormente, al momento de su construcción, se pueden asociar a una tabla y la forma en que esto se hace es mediante un query de SQL. Esto puede ser de forma gráfica o de

forma de texto, y a continuación mostraremos cómo se verían de ambas formas.



Forma Gráfica.

```
DataWindow - dw_exis_actu
    SELECT "almacen"."alm_cve_art",
           "articulo"."art_desc",
           "almacen"."alm_prec_prom",
           "almacen"."alm_entrada",
           "almacen"."alm_salida"
      FROM "almacen",
           "articulo"
     WHERE ( "articulo"."art_cve_art" = "almacen"."alm_cve_art" )
```

Forma de texto

Esta ventana, al abrirse, ejecuta automáticamente el evento "open" el cual contiene en su script la siguiente línea de código:

dw_consulta.retrieve()

La cual, lo único que hace es ejecutar la función retrieve() el cual trae los renglones de la base de datos al datawindow y cuya sintaxis es la siguiente:

"dwcontrol".Retrieve ({, argument, argument . . . })

donde:

- "dwcontrol". Es el nombre del datawindow con el cual se desea traer la información de la base de datos.
- argument (opcional). Son uno o más valores que pasan como argumentos al query asociado al datawindow.

De esta forma es como funcionan en términos generales las ventanas de consultas que al igual que los reportes lo único que se necesita es armar la consulta de la información que deseamos obtener y asociarla a un datawindow en forma de query o en forma gráfica.

4

Pruebas e Implantación.

En el siguiente capítulo se abordará sobre todo lo referente a la puesta en marcha del sistema de almacén, en donde se mencionarán diferentes aspectos en lo que concierne a las pruebas que se realizaron, así como a la implantación de dicho sistema en la dependencia de la U.N.A.M., que colaboró para tal fin.

4.1 Requerimientos Técnicos

Para el buen funcionamiento del Sistema de Inventario es necesario contar con el equipo y software adecuado, por lo cual a continuación se listan una serie de requerimientos que deben contemplarse para la instalación.

Sistemas Operativos
Microsoft Windows 95 ó superior
Microsoft Windows NT 4.0
Memoria
32 Mb de RAM
Recomendado: 64 Mb de RAM
Espacio en disco duro:
10 Mb Instalación Típica y una capacidad en el disco duro de 2 GB
Ratón de Microsoft o compatible.
PC con procesador Pentium ó superior.
(Recomendado Pentium III)
Tarjeta Gráfica VGA o superior
(Recomendado SGVA de 256 colores)
Unidad de Floppy de 3.5 pulgadas (alta densidad).
CD- ROM.(Opcional) solo es necesario para la instalación de los controladores.

4.2 Pruebas

El sistema de almacén fue sometido a pruebas extensas para asegurar la calidad del mismo, el cual fue probado y aprobado por el usuario que, junto con nosotros, se dedicó a apoyarnos en esta revisión. Dichas pruebas fueron realizadas a lo largo del desarrollo del sistema y no simplemente al final, y así, de esta forma, pudimos corroborar y corregir, al momento, cualquier problema que surgió en los programas, así como en sus interfaces, antes de que el sistema fuera usado. De igual forma, posteriormente, fue probado el sistema como un todo.

Cabe mencionar que se hicieron pruebas del sistema con datos, es decir, revisión de escritorio de los programas contenidos en cada módulo, luego posteriormente, se efectuaron pruebas de enlace con datos ficticios (para ver si los programas o módulos interrelacionados trabajaban como se planeó), revisión completa del sistema con datos de prueba (tomado como unidad completa), prueba completa del sistema con datos reales, esto después de que la prueba completa con datos ficticios resultó satisfactoria, ya que solo así se pudo probar el sistema con datos reales, los cuales han sido procesados satisfactoriamente con el sistema existente.

Además, es preciso comentar, que se realizaron pruebas especiales, tales como pruebas de tiempos (cuánto lleva realizar algo), prueba de capacidad de almacenamiento, prueba de carga máxima (fueron puestas en funcionamiento todas las terminales juntas para ver como el sistema soporta la información).

Durante el desarrollo del sistema las pruebas estuvieron siendo realizadas en paralelo con el sistema anterior, es decir, se comparaban datos de entrada y salida tanto del nuevo sistema como del anterior ya que teóricamente deberían arrojar el mismo resultado, lo cual finalmente nos sirvió de guía para validar la información con el sistema anterior.

4.3 Implantación y conocimientos previos

La implantación es el proceso de asegurar que el sistema de información es operacional y permitir que luego los usuarios tomen control de su operación y evaluación. El analista tiene varios enfoques para la implantación que deben ser considerados cuando se esta preparando el cambio al nuevo sistema, éstos son:

1) Establecimiento de un Centro de información: el papel principal del analista es supervisar la implantación, estimando el tiempo necesario y luego supervisando la instalación del equipo para los sistemas tradicionales, procesamiento distribuido, capacitación de usuarios y la conversión de archivos y bases de datos al nuevo sistema. Un centro de información dependiente de sistemas es una forma de hacer más fácil a los usuarios satisfacer sus necesidades de información a corto plazo. Deben dar un papel de soporte técnico a los usuarios en forma permanente.
2) Capacitación de usuarios: la capacitación de usuarios y de personal para interactuar con el sistema es una parte importante de la implantación, ya que ellos deben ser capaces de ejecutar el sistema sin intervención del analista. El analista necesita considerar quiénes necesitan ser capacitados, quién los capacitará, los objetivos de la capacitación, los métodos de capacitación a ser usados, los lugares adecuados de capacitación y los materiales comprensibles que se utilizarán para la misma.
3) Conversión: Hay distintas técnicas para la implantación y puesta en marcha de nuevo sistema, 5 estrategias de conversión física del antiguo sistema al nuevo:
   a) Conversión directa: Significa que a partir de una fecha específica, el sistema es desechado y el sistema nuevo es puesto en uso. Ventajas: no hay costo adicional que no sea del nuevo sistema, los usuarios no tienen posibilidad de usar el sistema antiguo, la adaptación es una necesidad. Desventaja: pueden darse largos retrasos si suceden errores, debido a que no hay forma alterna para lograr el procesamiento, si por eventualidad no funciona es un riesgo muy grande que puede afectar a terceros. Si el sistema no es crítico y la empresa puede funcionar si la información se puede hacer.
   b) Conversión en Paralelo: Se refiere a ejecutar el sistema antiguo y el sistema nuevo al mismo tiempo, en paralelo. La ventaja es que proporciona una seguridad a los usuarios, no están forzados a hacer un cambio abrupto. La desventaja es el costo de ejecutar dos sistemas al mismo tiempo y duplicar la carga de trabajo de los empleados durante la

conversión, además del que debe controlar las similitudes y diferencias entre los sistemas.

c) Conversión gradual: En este plan, el volumen de transacciones manejado por el nuevo sistema aumenta gradualmente conforme el sistema avanza en sus fases, se implementa el sistema por partes, la ventaja es que permite a los usuarios involucrarse con el sistema paulatinamente, la desventaja es que lleva demasiado tiempo poner el nuevo sistema en su lugar. En el momento del traspaso final debe estar presente auditoria para verificar los saldos al momento y las coincidencias.

d) Conversión por prototipos modulares: Conforme cada módulo es modificado y aceptado, es puesto en uso. Una ventaja es que cada módulo es probado a fondo antes de ser usado. El hecho de que la construcción de prototipos no sea factible tan frecuentemente elimina automáticamente este enfoque para muchas conversiones.

e) Conversión distribuida: Se realiza una conversión completa, con cualquiera de los enfoques anteriores en un lugar. Cuando esa conversión ha sido terminada satisfactoriamente se realizan las conversiones para otros sitios. La ventaja es que los problemas pueden ser detectados y detenidos, en vez de afectar simultáneamente a todos los sitios. Una desventaja es que aunque una conversión sea satisfactoria, cada sitio tendrá sus propias peculiaridades sobre las que hay que trabajar.

4.3.1 Seguridad

La seguridad de las instalaciones, los datos y la información generada es parte de una conversión satisfactoria. La seguridad tiene tres aspectos interrelacionados, física, lógica y de comportamiento. Los tres tienen que trabajar juntos si se pretende que la calidad de la seguridad permanezca alta.

Seguridad Física: se refiere a la seguridad de las instalaciones de computación, su equipo y software por medios físicos.

Seguridad Lógica: se refiere a los controles lógicos dentro del mismo software (contraseñas).

Seguridad de comportamiento: se refiere al comportamiento interno de los miembros de la organización, es crítico para el éxito de los esfuerzos de seguridad (políticas y procedimientos), que el sistema registre la cantidad de empleados autorizados, monitorea el ingreso indebido de algunos empleados no autorizados.

4.3.2 Evaluación

- Técnicas de Evaluación: muchas técnicas de evaluación incluyen el análisis costo - beneficio, modelos que estiman valores de decisiones, simulación de estadísticas, evaluaciones del usuario que enfatiza problemas de implantación y el involucramiento del usuario y los enfoques de utilidad de la información que examinan las propiedades de información.

- El enfoque de utilidad del sistema de información: Para la evaluación del sistema de información puede ser una técnica comprensiva y fructífera para la medición del éxito de un sistema desarrollado. Las utilidades de la información incluyen, posesión (quien debe recibir salida, ya que no tiene valor la información en manos de alguien que no la sabe usar), de forma (que tipo de salida se distribuye al tomador de decisiones, deben ser útiles en su formato y vocabulario), de lugar (de donde es distribuida la información, debe ser entregada en el lugar donde se tomen las decisiones), de tiempo (cuando es distribuida, debe llegar antes de que se tomen decisiones), de actualización (cómo es presentada y usada por el encargado de la decisión), y utilidad de objetivo que responde al porqué del sistema de información preguntando si la salida tiene valor para ayuda a que la organización obtenga los objetivos.
- Evaluación del Sistema: Un sistema puede ser evaluado como satisfactorio si posee las seis utilidades, si un módulo es juzgado como pobre en alguna de las utilidades, el módulo completo estará destinado al fracaso, un logro parcial o regular de una utilidad dará como resultado un módulo parcialmente satisfactorio, si es juzgado como bueno al proporcionar todas las utilidades, el módulo es un éxito.

La etapa de evaluación comienza a partir de que se implementa el sistema nuevo, se recogen opiniones de los usuarios, para ver la medida en que se ha modificado en la manera de trabajar o actuar. El sistema nuevo es más amigable, es bien recibido, se presentan situaciones no previstas, etc. Se hace con cuestionarios a los usuarios con preguntas concretas, Ej. Las decisiones que se toman son mejores?
Se debe llevar un registro de emergencias o accidentes y sacar conclusiones, periodicidad, si se repiten, cada cuanto, etc.

4.3.3 Instalación

La etapa de la instalación es la última en lo que a la programación se refiere por que esta etapa pone en marcha el sistema en los computadores de los usuarios, en donde a partir de este momento se empieza la realización de las pruebas y la reprogramación o retroalimentación para el uso del sistema y mejoramiento de éste.

4.3.4 Pruebas y retroalimentación

Esta etapa es la última de todas, ya que a partir de aquí depende únicamente de lo que el usuario pida y los errores que detecto, como también algunos nuevos comentarios que se tendrán en el mismo. A esta etapa de pruebas se le conoce como reprogramación (para los programadores) y retroalimentación (para los analistas). Aquí es donde todo el entorno desde el análisis hasta la implantación del sistema se ven nuevamente involucradas pero de una nueva manera o forma, ya que se vuelven a analizar los requerimientos pero en práctica únicamente.

4.4 La Instalación del sistema en una dependencia de la U.N.A.M..

Al instalarse el sistema inicialmente en la dependencia, se tuvo problemas por el equipo de cómputo que se tenía ya que en la facultad sólo contaban con computadoras con procesadores 386 ó 486, lo cual implicaba inicialmente una nula posibilidad de trabajar con la nueva versión del sistema de almacén, pero finalmente, lo único que había sucedido fue un retraso en el equipo nuevo, el cual contiene las características solicitadas previamente.

Lo anterior se menciona debido a que de instalarse el sistema en otra dependencia de la U.N.A.M., se puede correr el riesgo de que no funcione o en su caso no se pueda instalar y no tendría ningún caso que este sistema se instale en dichas dependencias.

Cuando se instaló el sistema en los equipos de cómputo surgió algo que no se tenía contemplado, debido a que el sistema se diseñó como una estructura cliente-servidor y para esto es necesario contar con una red interna y que la base de datos principal se encuentre en un servidor único dentro de la facultad. Pero debido a que se carece de todo esto, se tuvo que reprogramar y elaborar una versión local, en la cual la base de datos se instaló en el sistema

local y posteriormente por medio de un proceso simple se efectúa una copia mensual para enviarlo directamente a auditoria interna.

4.4.1 Primeros resultados

Inicialmente, al entrar en funcionamiento el sistema, se empezó a notar un problema que fue solucionado inmediatamente, éste era la captura del catálogo de artículos, ya que se cuenta con una existencia de 100,000 diferentes artículos y cada uno tiene una particularidad diferente a los demás por lo cual se elaboró un sistema de migración de datos (solo del catálogo) de las bases originales de DBase IV a Sybase (SQL Anywhere) obteniendo los resultados esperados, un catálogo 100% confiable y auditado por las autoridades de la U.N.A.M.

En estos primeros resultados se inicio por confirmar que los catálogos correspondían a los originales, así como también con la información que se contenían en éstos, lo cual fue un proceso lento y tedioso pero al finalizar se "cuadraron" sin ningún problema.

4.5 Captura, consulta y emisión de reportes

Después de validar la información de los catálogos se empezó a la captura de nuevos artículos, partidas, empleados y notas de entrada en el sistema a manera de ver cuál es la diferencia y la eficacia con el sistema original. Esto porque, como en todos los programas siempre se debe de empezar con la captura inicial de los catálogos y posteriormente con la captura de información (altas, bajas, cambios, en sí, movimientos en general). Al ver los resultados y la facilidad con la que pueden ser capturados los artículos y demás datos a los catálogos se optó por verificarlos nuevamente de forma de que realmente sean los datos capturados y que pudieran ser en un futuro validados por las autoridades de la dependencia y finalmente por las autoridades de la U.N.A.M. Durante este proceso se observaron algunos datos y nuevas requerimientos del sistema, como una medida especifica o estándar de los artículos (pieza, caja, kilo, etc.), la captura de un empleado directamente desde la opción de departamento o viceversa, la captura de un departamento desde la captura de un empleado, y así con esto se evita una redundancia de la información.

Al efectuar los ajustes correspondientes a esta etapa se volvió nuevamente a la captura de la información para lo cual ahora era una consulta

real de la información desde el sistema y no desde una serie de comandos efectuados a la base de datos. Con esto se validó la forma en que el sistema efectúa la consulta y nuevamente se vio la versatilidad y la facilidad de efectuar estas consultas de datos en el sistema.

Posteriormente, al término de esta etapa se continuó con la revisión de distintos reportes que genera el sistema, para así, de esta forma, poder checar la captura inicial de movimientos. En este momento se podía contemplar que se tenia una estructura bastante aceptable, lo que es la base de datos con todos los catálogos de la U.N.A.M. y los primeros movimientos de captura de notas de entrada, captura del inventario final del almacén del mes pasado y con esto se generó por primera vez, en el sistema, el inventario inicial, con el cual empezaríamos a trabajar con datos reales de información.

Se empezaron a capturar todos los datos de movimientos del almacén, lo que fueron las salidas de mercancía (vales de salida), los cuales efectúan los empleados asignados de cada departamento. Al término de esta captura de movimientos se continuó con la consulta en el sistema y la validación de los datos, en este momento no se tocó en lo absoluto ningún otro reporte que la única captura de datos y nuevamente se solicitaron nuevos cambios para una mayor eficiencia en el sistema, esta vez fue la captura de vales de salida desde cada departamento, sin la necesidad de tener que llevarlos personalmente al almacén, pero por la estructura de la red de la Facultad no se pudo efectuar esto, aunque el sistema esta abierto en esta parte para un futuro desarrollo y sin necesidad de efectuar muchos cambios. Esto es necesario resaltar debido a que en un futuro, la herramienta con la que se programó el sistema (Power Builder) tiene en desarrollo una versión que podrá trabajar los sistemas en línea, es decir la información podrá ser solicitada desde Internet, pero para esto es necesario una migración del sistema (solo lo programado o el código) a esta plataforma, y nuevamente reiterar que no tendrá ninguna complicación puesto que el sistema esta listo para trabajarse en el momento en que esto suceda.

Al término de la validación de la información, tanto de consulta en pantalla como en reportes, se continuó con la revisión de los datos por movimientos, por ejemplo:

las entradas + el inventario inicial - salidas = inventario real

Al validarse, fueron aumentando las solicitudes de nuevos reportes al llegar a un total de 41, los suficientes para que el sistema quede dado de alta entre los sistemas autorizados por auditoria de la U.N.A.M..

4.6 Validación de datos

En esta última etapa y como se mencionó con anterioridad el sistema ya esta listo para poder trabajarse con la autorización de auditoria de la U.N.A.M.. Lo que sucede a partir de este momento es que a cada una de las dependencias se les recomendará el uso de éste debido a todas las innovaciones tanto tecnológicas, así como de la eficiencia en el análisis de flujo de información.

Cabe mencionar también, que uno de los problemas que se evitó, en cuanto a información de cantidades de los artículos, es que ésta no se pierde, pero en cuanto a las cantidades monetarias, que con anterioridad se perdía bastante información, es evitada con el uso de este nuevo sistema, ya que el anterior sistema maneja un precio promedio por artículo, mientras que este nuevo sistema maneja las cantidades monetarias reales.

Debido a lo anterior se comprobó la eficiencia del sistema, por locual se espera la autorización para poder distribuirlo en las diferentes dependencias de la U.N.A.M., y no de manera opcional si no de forma obligatoria para que al momento de efectuar las auditorias a los diferentes almacenes, éstos muestren en ese momento los reportes reales y sin retrasos de tiempo que en ocasiones demoraban cerca de 1 semana. Ahora con este nuevo sistema no lleva más de 10 minutos el mostrarlo.

Conclusiones.

Anteriormente las dependencias de la U.N.A.M. utilizaban sistemas de información desarrollados por lenguajes tediosos y difíciles de programar, ante lo cual el proceso de información era difícil y de dudosa credibilidad en cuanto a los resultados obtenidos por éstos, aunque actualmente siguen vigentes en la mayoría de las dependencias de la U.N.A.M.. Por lo cual, al termino del desarrollo del nuevo sistema de almacén se mejoró en su mayoría todos estos problemas mencionados.

Así, mediante el empleo de una metodología que involucró la realización de una serie de reuniones y entrevistas con el usuario, se puso en marcha un sistema que permite a las dependencias proporcionar sus servicios en forma rápida y eficiente y al mismo tiempo hace que su información se mantenga actualizada. Para conseguirlo tomamos una de las metodologías existentes y la adaptamos a las necesidades del proyecto, pues en varias ocasiones las restricciones que establece la normatividad interna de la U.N.A.M. impedían seguir la teoría al pie de la letra.

Excluyendo aquellos procesos en los que los requerimientos dependen de las circunstancias y por lo tanto requieren intervención humana, por ejemplo, llenado de formularios especiales o procesamiento de solicitud de material, se diseñó y realizó un sistema que cubre todas las necesidades de los usuarios, preparándolos al mismo tiempo, para solventar ligeros ajustes en la forma de trabajo de las dependencias.

Sin embargo, si hubieran cambios mucho más drásticos en la estructura de las dependencias sería necesario alterar el código de programación. En este sentido, la modularidad en la que se baso el diseño, las consideraciones hechas en la definición de los archivos y la propia documentación harían posible cualquier modificación a ese respecto. Por tal motivo se consideró una etapa de entrenamiento para los administradores de la dependencia con la que trabajamos, con el fin de que ellos se hicieran cargo del mantenimiento y la actualización del mismo. Y así, del mismo modo, a partir de la liberación del sistema, se seguirá efectuando un seguimiento para remediar los posibles problemas que llegaran a ocurrir durante el periodo anual y no previstos durante la etapa de diseño.

Cabe destacar que para que los usuarios tuvieran una mayor aceptación del sistema tratamos al máximo de no alejarnos de la forma a la que están acostumbrados a manejar su información, por ejemplo, el sistema es muy parecido al anterior en cuanto a captura de información así como la emisión de reportes ya establecidos, mas los nuevos formatos con información, previamente avalados por la dependencia, etc. Consideramos que es importante que las personas que van a utilizar el sistema y nunca han utilizado una computadora no se sientan intimidados por ésta, por el contrario al encontrar cierta familiaridad con lo que ya saben hacer pueden sentirse mucho más motivados a utilizarlo.

Por otra parte el uso de un lenguaje de cuarta generación en el desarrollo de la aplicación trae como ventaja que las dependencias hagan uso de las tecnologías de información mas recientes, dando como resultado un sistema robusto sin tener las complejidades inherentes a la utilización de una base de datos. Siendo otra ventaja el hecho de no depender de un tipo de Hardware y, por lo tanto, poder migrar su aplicación a otra plataforma en caso de un eventual crecimiento de ésta.

También queda establecido que un ingeniero en computación esta capacitado no solo para resolver problemas técnicos relacionados con su área, si no para crear verdaderas soluciones que involucren cualquier disciplina y ramo de la industria. Además queda comprobado que un ingeniero en computación puede manejar a un nivel muy alto cualquier lenguaje de un área especifica por ajeno que parezca.

El logro principal de este proyecto fue ver creado un sistema que simplifica la realización de las actividades de una organización real. Además la experiencia adquirida durante la realización del sistema así como el trato directo con los usuarios, fueron aspecto de gran valor que podemos usar en futuros proyectos o en situaciones similares que puedan presentarse en la practica profesional.

Apéndice A

Manual de usuario.

CAPTURA DE CATÁLOGOS

Para que pueda comenzar rápidamente a hacer uso del sistema, hay ciertas consideraciones que se deben tener en cuenta al instalarse por primera vez el sistema, esto es, se debe llevar a cabo la definición de catálogos (partidas, artículos, departamentos y empleados) que la dependencia usuaria utiliza, así como los niveles de control.

Partidas

Anualmente, la universidad emite un catálogo presupuestal mostrando las diferentes partidas, bajo las cuales se afectan las cuentas, este número formado de tres dígitos, es un elemento presupuestario en el cual se dividen los subgrupos de gasto y que clasifican las distribuciones de acuerdo con el objeto especifico del gasto.

Artículos

El artículo es un elemento primario sobre el cual se debe de controlar todos los gastos y consumos de un almacén, por lo tanto este artículo le será asociado un número de artículo, que debe de cumplir con la regla de pertenecer a una partida (la cual debe de ser dada de alta anteriormente) y un número consecutivo, de acuerdo a las necesidades de la dependencia, donde este número deberá constar de 5 dígitos.

Departamentos

Después de haber definido los catálogos de partidas y artículos el siguiente paso es definir los departamentos de la dependencia usuaria, los

cuales deben de tener una clave y una descripción propia que son indispensables para llevar un control de consumos de los departamentos autorizados, dicha clave debe ser numérica.

Empleados

La manipulación de catálogos de empleados, consiste en poder realizar altas, bajas y cambios de todas aquellas personas autorizadas para poder obtener los artículos del almacén, que se encuentran en la dependencia en un determinado departamento.

Para poder dar de alta un empleado, es necesario contar con su RFC y con el departamento en el cual se encuentra laborando. Esto se hace con el fin de poder determinar los consumos que genera un departamento en un periodo dado.

MOVIMIENTOS

La actualización de información, es la opción más importante de cualquier sistema administrativo, ya que es donde se empiezan a realizar movimientos que serán reflejados en cada uno de los catálogos anteriormente definidos, por lo tanto, se dará a continuación una explicación detallada de cada una de las opciones que comprenden esta sección.

Inventario inicial

Es el punto de partida para poder manejar el sistema, ya que en este proceso se definen las condiciones iniciales con las que parte el almacén de la dependencia usuaria, en cuanto a los artículos y las cantidades que de ellos se dispone.

Hay que tomar en cuenta que dar un inventario inicial implica una entrada al almacén, por lo tanto para comodidad, seguridad y control del

responsable del almacén se puede expedir un comprobante impreso,el cual es una nota de entrada que contiene la fecha , el número de nota, el nombre de la dependencia, la clave y descripción de el(los) artículo(s), la cantidad y el precio unitario. Si por algún motivo usted no imprimió la nota y necesita dicho comprobante, en el menú de reportes detallados se tiene una opción dedicada a obtener reportes de acuerdo al número de nota que usted desee.

Nota de entrada

Una de las características principales de un almacén es la constante entrada y salida de artículos, por lo tanto el sistema cuenta con una opción que le permite dar acceso a nuevos artículos de consumo con los cuales habrá usted generado una entrada al almacén. Al mismo tiempo de ingresar una cantidad de artículos, el sistema recalcula el precio promedio de los artículos de nuevo ingreso con respecto a los que se encuentran actualmente.

Al realizar usted una entrada al almacén, podrá obtener una nota de entrada generada por el mismo sistema el cual es un documento de respaldo y control que contiene información al detalle de la operación efectuada.

Vales de salida

En esta opción del sistema, se realiza una captura de datos, la cual sirve para poder realizar cálculos sobre los artículos que el usuario vaya a adquirir del almacén, donde se registrarán más tarde en un vale de salida que el sistema genera, dicho vale de salida cuenta con un número consecutivo que es asignado automáticamente por el sistema para poder llevar un control interno.

Los datos que se requieren para poder solicitar artículos al almacén son: fecha de solicitud, RFC de la persona que solicita, clave(s) de (los) artículo(s) solicitados, así como la cantidad de éstos. Al terminar la captura de los datos, usted podrá obtener un reporte impreso de todos los movimientos registrados.

Actualización de máximos y mínimos

La actualización de máximos y mínimos, consiste en poder volver asignar un número de artículos que pueden ser almacenados y el número mínimo de éstos que son indispensables en el almacén para poder satisfacer las necesidades de la dependencia.

Para dicha actualización, basta con saber la clave del artículos que tendrá el cambio.

CONSULTA

La función de esta opción es poder observar los movimientos en el almacén, así como poder hacer una estimación de los recursos que se emplearán y necesitarán para poder brindar un buen servicio dentro de la dependencia.

Se cuenta con un menú de opciones, en donde encontrará los tipos de consulta más usuales dentro de un sistema de almacén. Normalmente, usted tendrá una tabla de información la cual es manipulable a través de las flechas de navegación, mouse, o teclas especiales con las que podrá buscar un tipo de información específica de su interés.

Catálogos

Con esta opción usted podrá observar toda la información de los catálogos que han sido capturados hasta la fecha. Una de las aportaciones que realiza esta consulta es poder ver también los movimientos que ha tenido cada uno de los catálogos de una manera independiente.

Partidas

Permite realizar una consulta de todas las partidas que han sido definidas, la cual es presentada por medio de una tabla que muestra la información de los movimientos como son: número de la partida, descripción

de la partida, gasto actual, consumo actual, navegar a través de la información libremente.

Artículos

La consulta que se realiza en este módulo, es para poder observar en detalle cual es la situación actual de algún artículo en específico, la información que se puede obtener de esta consulta es la clave del artículo, su descripción el consumo actual y el consumo acumulado.

Departamentos

En la consulta por departamentos, únicamente se mostrará la información de todos los departamentos registrados, así como los consumos y gastos actuales de cada uno de ellos.

Empleados

Una de las consultas más importantes, sin lugar a duda dentro de un almacén, es la consulta de las personas que están autorizadas para obtener los productos. En esta consulta se puede observar, información relevante de todos los empleados como es: el RFC, nombre completo del empleado, departamento en el cual se encuentra laborando, consumo en el periodo actual y el consumo acumulado.

Notas de entrada

Es importante poder comprobar con una nota de entrada impresa, que actividades se han desarrollado en el almacén en cuanto al ingreso de nuevos productos; pero también es conveniente que el responsable pueda tener absoluto control sobre el manejo del almacén, por lo tanto usted tendrá la capacidad de poder observar con detalle cada una de las notas generadas por el sistema, a través del monitor.

Vales de salida

Un documento de suma importancia, es el vale de salida, ya que cumple con la función de amparar tanto a la persona encargada del sistema, como a la persona que obtuvo artículos del almacén.

Cada vez que se realiza una solicitud de artículos al almacén y ésta es autorizada el sistema registra todos los movimientos, así como realiza todas las operaciones de cálculo para un mejor control.

Artículo con existencia <= al máximo

En esta sección de la consulta, usted puede obtener información acerca de los artículos que estén por debajo de su existencia necesaria para poder brindar un servicio adecuado en el almacén.

Artículo existencia actual

La ayuda que puede ofrecer esta consulta, es poder saber la cantidadde artículos con la que se cuenta en el almacén, al mismo tiempo de ofrecer a usted cual es el precio unitario del artículo.

Detalle de entrada /salida de artículos

Esta consulta ofrece la oportunidad de ver todos los movimientos registrados por el artículo en cuestión, usted observará las entradas registradas, donde se podrá verificar todas las notas en donde el artículo esta involucrado. La fecha de ingreso al almacén, el precio unitario del artículo, la cantidad y el tipo de movimiento que puede ser de nota de entrada o bien de inventario inicial.

Detalle de salidas por empleado

En esta consulta usted podrá examinar con detalle cada uno de los artículos que ha adquirido cualquier empleado, esta consulta es de gran utilidad ya que proporciona en que fecha fue la obtención de los artículos, el número de vale de salida con que fue realizada la operación, el precio unitario de los artículos, el número de vale de salida con que fue realizada la operación, el precio unitario de los artículos y la cantidad que el empleado haya recibido.

Apéndice B

Manual Técnico.

Este manual técnico se compone de dos partes esenciales, lo referente a la aplicación (sistema) y lo referente a la Base de Datos, los cuales interactuan en conjunto, sin embargo, su diseño y construcción fue por separado. Así pues en la primera parte se hará mención a los objetos de la aplicación y posteriormente a lo referente a la base de datos.

1) Aplicación

El sistema de almacén se compone por siete librerías fuentes, las cuales por haberse desarrollado con el lenguaje Power Builder, tienen la extensión pbl (*.pbl.).

A continuación se muestra un listado con el nombre de las librerías, así como una breve descripción.

Nombre	Descripción
sis_alm01.pbl	Librería Principal
sis_alm02.pbl	Librería con objetos de uso general
sis_alm03.pbl	Librería que contiene todo lo referente al módulo de Catálogos.
sis_alm04.pbl	Librería que contiene los objetos del módulo de Actualización.
sis_alm05.pbl	Librería que contiene los objetos del módulo de Catálogos.
sis_alm06.pbl	Librería que contiene los objetos de cierre de mes y año.
sis_alm07.pbl	Librería de reportes.

Enseguida se listarán los diferentes objetos contenidos en cada librería, así como una breve descripción de cada uno de ellos.

sis_alm01.pbl

almacén	Objeto aplicación del sistema de almacén.
m_almacen	Objeto menú del sistema de almacén
w_logo_inicial	Ventana que presenta el fondo con el logotipo de la U.N.A.M.
w_mdi_comp2	Ventana principal base del Sistema.

sis_alm02.pbl

f_nota_vale	Función global para obtener número de notas y vales.
u_sle_gral	Objeto de Usuario para enviar comentarios, filtros, fecha, hora, etc.
uo_toolbar_contrato	Objeto de usuario para barra de herramientas.
w_about2	Ventana que muestra información acerca del sistema.
w_catalogos	Ventana ancestro.
w_consultas_grales	Ventana ancestro para consultas generales.

sis_alm03.pbl

dw_a_articulo	Objeto datawindow para las altas y cambios de artículos.
dw_a_depto	Objeto datawindow para las altas y cambios de deptos.
dw_a_empleado	Objeto datawindow para las altas y cambios de empleados.
dw_a_partida	Objeto datawindow para las altas y cambios de partidas.
dw_c_articulo	Objeto datawindow para consultas de artículos.
dw_c_depto	Objeto datawindow para consulta de artículos que se encuentran en deptos.
dw_c_empleado	Objeto datawindow para consultas de empleados.
dw_c_partida	Objeto datawindow para consultas de partidas.
dw_rep_articulo	Objeto datawindow para reportes de artículos.
dw_rep_depto	Objeto datawindow para reportes de deptos
dw_rep_empleado	Objeto datawindow para reportes de empleados.
dw_rep_partida	Objeto datawindow para reportes de partidas.
dw_unidad	Objeto datawindow para listar el tipo de unidades.
dwc_deptos	Objeto datawindowchild para listar a los deptos.
dwc_unidad	Objeto datawindowchild para listar las unidades.
w_articulos	Ventana de catálogos de artículos.

w_cat_unidad	Ventana de auxiliar de catálogos de unidades.
w_deptos	Ventana de catálogos de departamentos.
w_empleados	Ventana de catálogos de empleados.
w_partida	Ventana de catálogos de partidas.

sis_alm04.pbl

dw_a_inv_ini	Objeto datawindow de captura de inventario inicial.
dw_a_max_min	Objeto datawindow de alta de existencia máxima y mínima.
dw_a_vales	Objeto datawindow de captura de los vales de salida
dw_c_inv_ini	Objeto datawindow de consulta del inventario inicial
dw_c_max_min	Objeto datawindow de consulta de existencia máxima y mínima
dw_c_vales	Objeto datawindow de consulta de los vales de salida.
dwc_descrip	Objeto datawindowchild para listado de la descripción de artículos.
dwc_nombres	Objeto datawindowchild para listado de nombres.
w_inv_ini	Ventana de inventario inicial.
w_max_min	Ventana para actualización de máximos y mínimos
w_not_ent	Ventana de notas de entrada.
w_vales_sal	Ventana de vales de salida.

sis_alm05.pbl

dw_c_ent_sal	Objeto datawindow cabecera de detalle de entrada/salida de artículos.
dw_d_ent	Objeto datawindow del detalle de entradas e inventario de artículos.
dw_d_sal	Objeto datawindow del detalle de salidas e inventario de artículos.
dw_exis_min	Objeto datawindow de la existencia mínima.
dw_exis_actu	Objeto datawindow de la existencia actual.
dw_sal_emp	Objeto datawindow de detalle de salidas por empleado.
w_det_ent_sal	Ventana de detalle de entrada/salida de artículos.
w_exis_actu	Ventana de la existencia actual.
w_exis_min	Ventana de existencia de mínimos.

w_sal_emp	Ventana del detalle de salidas por empleado.

sis_alm06.pbl

d_calendar	Objeto datawindow para calendario.
dddw_meses	Objeto datawindow para los meses.
dw_almacen_import	Objeto datawindow para exportar e importar tabla almacén.
dw_articulos_import	Objeto datawindow para exportar e importar tabla artículos.
dw_cat_unidad_import	Objeto datawindow para exportar e importar tabla cat_unidad.
dw_control_import	Objeto datawindow para exportar e importar tabla control.
dw_depto_import	Objeto datawindow para exportar e importar tabla depto.
dw_empleados_import	Objeto datawindow para exportar e importar tabla empleados.
dw_entradas_import	Objeto datawindow para exportar e importar tabla entradas.
dw_mes_import	Objeto datawindow para exportar e importar tabla mes.
dw_meses	Objeto datawindow que despliega los meses.
dw_partidas_import	Objeto datawindow para exportar e importar tabla partidas.
dw_rep_alm_cierre	Objeto datawindow para reporte general del almacén para el cierre.
dw_rep_depto_cierre	Objeto datawindow para el reporte de deptos. para el cierre.
dw_rep_emp_cierre	Objeto datawindow para el reporte de empleados para el cierre.
dw_rep_inv_cierre	Objeto datawindow para el reporte de inventario inicial del cierre.
dw_rep_par_cieere	Objeto datawindow para el reporte de partidas del cierre.

dw_salidas_import	Objeto datawindow para exportar e importar tabla de salidas.
f_dia_mes	Función global para calcular los días de un mes.
u_ddcal	Objeto de usuario para el calendario.
u_dw_gral	Objeto de usuario datawindow de uso general.
w_cierre_mes2	Ventana del cierre de mes.
w_proteccion	Ventana de protección y resguardo de archivos.

sis_alm07.pbl

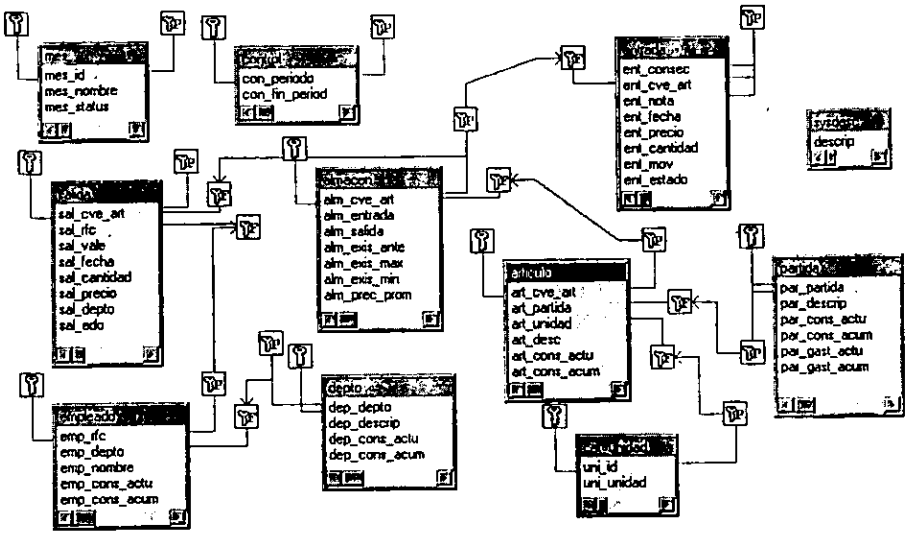
w_imprimir	Ventana que permite establecer las opciones de impresión de un datawindow.
w_preview	Ventana madre para mostrar reportes.
w_rep_partidas2	Ventana de reportes de partidas.

2) Base de datos

La base de datos que utiliza el sistema de almacén se compone por las siguientes tablas:

- almacén
- articulo
- cat_unidad
- control
- depto
- empleado
- entrada
- mes
- partida
- salida
- sysdesc

La siguiente gráfica o diagrama entidad/relación, que fué generada desde el "database" del lenguaje de programación Power Builder, muestra la integridad referencial que guardan entre si las tablas utilizadas por el sistema:



Store Procedures (Procedimientos)

Dentro de la base de datos "sis_alm", que es el nombre que se determinó para la base de datos del sistema de almacén, se encuentran contenidos algunos store procedures (sp´s) que son utilizados en ventanas de la aplicación, dichos sp's son los siguientes:

- alm_del_inv_ini
- alm_del_val_sal
- alm_ins_inv_ini
- alm_ins_val_sal
- sp_cierre_anual
- sp_cierre_mes

Bibliografía

Libros

Titulo: PowerBuilder 6
Autor: William B. Heys
Editorial: Prentice Hall

Titulo: Análisis y diseño de Sistemas de Información
Autor: James A. Senn
Editorial: McGraw-Hill

Titulo: PowerBuilder
Autor: Luis Barrientos
Editorial: Arcos

Titulo: Ingeniería del software
Autor: Richard Farley
Editorial: Limusa

Titulo: Ingeniería del software: un enfoque practico
Autor: Roger Pressman
Editorial: McGraw-Hill

Internet

Pagina de programación orientada a objetos:

http://personal.redestb.es/magabaldon/
http://www.microsoft.com/us/op/indexp.html
http://www.microsoft.com/us/op/index.asp
http://www.lawebdelprogramador.com
http://www.hckrs.com/pwbl/

Pagina de conceptos generales de computación

http://www.monografias.com/Computacion/General/
http://www.lawebdelprogramador.com
http://www.misprogramas.com.ar
http://www.unisys.com
http://www.pb.net
http://www.creativeclick.org