BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE COMPUTACION

DIPLOMADO EN BASES DE DATOS

CONTROL DE INGRESOS Y EGRESOS DE UNA DEPENDENCIA DE LA BUAP

TESINA QUE PRESENTA:

EPIGMENIO GONZALEZ NIETO

PARA OBTENER EL TITULO DE:

LICENCIADO EN COMPUTACION

ASESOR:

MARCO ANTONIO SORIANO OLLUA

CONTENIDO

INTRODUCCION

A lo largo de estas dos últimas décadas, la ingeniería de software a mejorado sustancialmente. Desde la década de los 70's se ha dirigido más la atención a la tecnología de desarrollo de software. Conforme los sistemas de cómputo se multiplican, se hacen más complejos y penetran con mayor profundidad en la sociedad moderna, se puede observar la necesidad de enfoques sistemáticos para el desarrollo de software, así como para su mantenimiento.

Actualmente en la mayoría de las dependencias se realiza manualmente el control de ingresos o egresos, por lo que la Dirección de Contabilidad solicita su información impresa para capturarla después en un sistema que la integra.

Pero como cada dependencia utiliza una manera diferente de contabilizar sus gastos estos no se pueden estandarizar, por lo tanto y si no se corrige este problema nunca se podrá calcular un presupuesto desglosado, estados financieros etc. como la SEP lo está requiriendo ni sacar indicadores verídicos que reflejen el comportamiento de la BUAP.

Actualmente se cuenta con herramientas para el control y manipulación de grandes cantidades de información como los DBMS (Sistemas manejadores de bases de datos). Se plantea la creación del sistema SIE que lleve el control de los ingresos y egresos de cualquier dependencia de la Universidad Autónoma

El objetivo general es el análisis, diseño e implantación del un sistema SIE para el control de ingresos y egresos de cualquier dependencia de la BUAP.

Otros de los objetivos particulares son :

a) Determinar las necesidades que tiene una dependencia de la BUAP en particular las que no cuentan con accesos a Banner.
b) Diseñar un software que satisfaga los requerimientos para controlar ingresos y egresos de cualquier dependencia de la BUAP.
c) Implantar el sistema en una dependencia de la BUAP.
d) Elaboración de un manual de usuario, I

Este trabajo pretende construir una herramienta para satisfacer las necesidades de cualquier dependencia de la BUAP. Tal herramienta traerá como consecuencias:

- Responder en un tiempo mínimo a las demandas de información que estos requiera tal dependencia.
- Tener de manera desglosada y resumir toda la información respecto a sus movimientos contables.

Con el sistema Control de Ingresos y Egresos (SIE) se plantea la automatización de los diferentes movimientos financieros de cada una de las dependencias, que les será de gran utilidad para su control interno, en lugar de solicitar la información impresa se solicitarán las diferentes bases de datos, que con ayuda de un módulo de integración se unirán para formar así la base de datos general que será controlada por las diferentes dependencias y al existir una estandarización de los movimientos financieros estas podrán integrarse con la parte manejada por el sistema Banner. De esta forma se podrán conocer los estados financieros de cada una de las dependencias (Solicitado por la SEP en el Presupuesto de cada una de las dependencias y que no se cuenta con ellos actualmente) que al unirlas nos darán el estado general en cuanto a las dependencias. Ya que toda la información será estandarizada y será por tanto cuestión de minutos

llevarlo a cabo, además se reducirán los errores tan comunes en el proceso de captura.

CAPITULO I

INGENIERIA DE SOFTWARE

1.1 INGENIERIA DE SOFTWARE.

1.1.1 Definiciones.

El modelo de ingeniería de software que se utilizara para desarrollar el sistema "Control de ingresos y egresos para una dependencia de la BUAP" será el de espiral, el cual consiste en cubrir las mejores características tanto del ciclo de vida clásico como de la creación de prototipos, añadiendo un nuevo elemento: el análisis de riesgos. El modelo representado mediante la espiral, el cual define cuatro actividades principales, representadas por los cuatro cuadrantes de la figura 1.

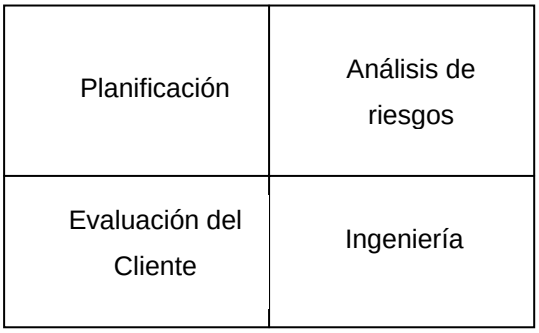


Figura 1. Los cuadrantes del modelo en espiral

Planificación: determinación de objetivos alternativas y restricciones.
Análisis de riesgo: Análisis de alternativas e identificación y resolución de riesgos.
Ingeniería: Desarrollo del producto del siguiente nivel.
Evaluación del cliente: Valoración de los resultados de la ingeniería.

Es importante mencionar que el modelo en espiral, se hace evidente cuando se considera la dimensión radial, con cada iteración alrededor del espiral, comenzando en el centro y siguiendo hacia el exterior, se construyen sucesivas versiones del software cada vez más complejas.

La ingeniería de software se define como la disciplina tecnológica preocupada de la producción sistemática y mantenimiento de los productos de software que son desarrollados y modificados en tiempo y dentro de un presupuesto definido.

En ciertas ocasiones se ha dicho que los conceptos de la ingeniería de software son aplicados solamente a proyectos grandes y de larga duración; es cierto que en ciertos proyectos grandes son esenciales las prácticas estándar y los procedimientos formales, y que algunas notaciones, herramientas y técnicas de ingeniería de software se han desarrollado para tales casos. En otro caso un proyecto pequeño puede ser más sencillo; sin embargo, los principios fundamentales del análisis sistemático, diseño, instrumentación, pruebas y modificaciones permanecen constantes no importando si éste es demasiado grande o muy pequeño.

1.1.2 Introducción a la ingeniería de software.

El desarrollo y mantenimiento de productos de programación requieren de un enfoque más sistemático que es necesario en el desarrollo de programas para uso personal.

Para desarrollar productos de programación, las necesidades y limitaciones del usuario deben estar determinadas y explícitamente establecidas; el producto debe de diseñarse considerando tanto a los instrumentadores como a los usuarios, y a quienes les den mantenimiento; el código fuente

debe instrumentarse con cuidado y se puede probar tomando en cuenta documentación de apoyo, principios de operación, manual de usuario, instrucciones de instalación, guías de entrenamiento y manuales de documentación. La tarea de mantenimiento de software incluye solicitudes de análisis de cambio, rediseño y modificación del código objeto, pruebas completas de la versión modificada, actualización de la documentación para mostrar dichos cambios, así como la distribución de la nueva versión en los lugares apropiados.

Las necesidades de enfoques sistemáticos para el desarrollo y mantenimiento de estos productos se patentizó en los años 60's. Durante esta década aparecieron las computadoras de la tercera generación y se desarrollaron técnicas de programación como la multiprogramación y el tiempo compartido. Estas nuevas herramientas aportaron la tecnología necesaria para el establecimiento de sistemas computacionales interactivos de multiusuario, en línea y en tiempo real, así surgieron nuevas aplicaciones para la computación, como las reservaciones aéreas, bancos de información médica, bancos de información bibliográfica, tiempo compartido para diversas aplicaciones, control de procesos, guías de navegación y control de dirección de equipo militares, etc.

Desde 1968, se diversificó la utilización de las computadoras y se ha hecho más compleja y crítica para la sociedad moderna, como resultado de esto, el campo de la ingeniería de productos de programación, ha evolucionado para convertirse en una disciplina tecnológica de importancia considerable.

De acuerdo con Boehm, la ingeniería de software, incluye la aplicación práctica de conocimientos científicos en el diseño y construcción de programas para computadoras y la documentación asociada requerida para desarrollarlos, operarlos y mantenerlos. El término diseño debe ser interpretado ampliamente para incluir actividades como el análisis de

requisitos y el diseño durante la modificación de un sistema. En el libro Standard Glossary of Enginnering Terminology se define a la ingeniería de software como " El enfoque sistemático para el desarrollo, operación, mantenimiento y eliminación de software", donde software se define como " "Aquellos programas, procedimientos, reglas y documentación posible asociada con la computación, así como los datos pertenecientes a la operación de un sistema de computo, en otras palabras, es la disciplina tecnológica y administrativa dedicada a la producción sistemática de productos de programación, que son desarrollados, y modificados a tiempo y dentro de un presupuesto definido.

Las metas principales de esta nueva disciplina tecnológica son mejorar la calidad de estos productos y aumentar la productividad y satisfacción profesional de los ingenieros de esta disciplina.

La ciencia de la administración proporciona los fundamentos para la administración del proyecto. Los sistemas computacionales deben ser desarrollados y mantenidos en tiempo y dentro de un régimen de estimación de costos; la economía por su parte, brinda los fundamentos para la estimación de recursos y el costo del control. El quehacer de la ingeniería de software se desarrolla dentro del contexto organizacional de una empresa, por lo que se necesita un alto grado de comunicación entre los clientes, administración, ingenieros de programación, ingenieros de computación y demás técnicos; así, resulta importante que el ingeniero de programación cuente con habilidades de expresión oral y escrita, además de comunicación interpersonal. Ya que la ingeniería de software se preocupa del desarrollo y mantenimiento de productos de la tecnología moderna, es necesario utilizar técnicas de resolución de problemas comunes a todas la ramas de la ingeniería, estas técnicas sientan las bases de la planeación y administración de proyectos, análisis, diseño metódico, fabricación cuidadosa, validación profunda y mantenimiento continuo del

producto. Para efectuar esto se requiere de la aplicación de una notación adecuada, así como de herramientas y técnicas en cada área además los ingeniero deben equilibrar en forma práctica los principios básicos con los aspectos económicos y las preocupaciones sociales cuando resuelven problemas y desarrollan productos tecnológicos.

Durante la década de los 70's sucedieron avances significativos en todas las áreas de la ingeniería de software, se desarrollaron técnicas de análisis para determinar los requisitos proliferando los enfoques metodológicos para el diseño de programas, y las notaciones diversas, han nacido nuevas técnicas de instrumentación, así como nuevos lenguajes de programación, se han examinado técnicas para la validación de programas y se han instituido controles de calidad, se han desarrollado técnicas formales para la verificación de programas y finalmente se han mejorado los procedimientos de mantenimiento de programas. Las técnicas de administración se han adaptado especialmente para la ingeniería de software, además se han explotado técnicas de dinámica de grupos y comunicación de proyectos, también han evolucionado las técnicas cuantitativas para la estimación de costos y la confiabilidad de un producto, se han descubierto los principios fundamentales de análisis, diseño, instrumentación y pruebas. Se han desarrollado herramientas automáticas de desarrollo para mejorar la calidad de los programas, la productividad, el control y la administración de un proyecto.

Todo esto no implica que los problemas de la ingeniería de software estén resueltos, al contrario, el nivel de actividad de esta rama es la gran cantidad de problemas por resolver. Cada tecnología crece por medio de faces predecibles de descubrimientos apropiados, desarrollo sistemático de procedimientos y por último la culminación de un enfoque en forma de manual de la disciplina efectuada en forma rutinaria.

1.1.3 Software e Ingeniería de Software.

Actualmente, el software ha superado al hardware como la clave del éxito de muchos sistemas basados en computadoras. Tanto si se utiliza la computadora para llevar un negocio, controlar un producto o capacitar un sistema, el software es el factor que marca la diferencia. Lo que diferencia a una compañía de su competidora es la suficiencia y oportunidad de la información dada por el software. El diseño de un producto amigable a los humanos lo diferencia de los productos competidores que tengan unas funciones similares. La inteligencia y función que proporciona el software empotrado distingue normalmente dos productos industriales o de consumo similar. El software marca la diferencia.

1.1.4 importancia del software.

Durante las tres primeras décadas de la información, el principal desafío era el desarrollo del hardware de las computadoras, de forma que se redujera el costo de procesamiento y almacenamiento de datos. A lo largo de la década de los ochenta, los avances en microelectrónica han dado como resultado una mayor potencia de cálculo a la vez que una reducción del costo. Hoy, el problema es diferente. El principal desafío es mejorar la calidad de las soluciones basadas en computadoras, soluciones que se implementan con el software.

1.1.5 evolución del software

en el contexto en el que ha desarrollado el software está fuertemente ligado a las casi cinco décadas de evolución de los sistemas informáticos, un mejor rendimiento del hardware, una reducción del tamaño y un costo más bajo, han dado lugar a sistemas informáticos más sofisticados, hemos pasado de los procesadores con válvulas de vacío a los dispositivos microelectrónicos que son capaces de procesar 200 millones por segundo.

- Primera era: orientación por lotes (1959 – 1960)
- Segunda era: Multiusuarios (1960 – 1970)
- Tercera era: sistemas distribuidos (1970 – 1980)
- Cuarta era: Tecnología orientado a objetos

Durante los primeros años de desarrollo de las computadoras, el hardware sufrió continuos cambios, mientras que el software se contemplaba simplemente como un añadido, la programación de computadora era un arte, para el cual existían pocos métodos sistemáticos. El desarrollo de software se realizaba virtualmente sin ninguna aplicación.

1.1.6 Construcción de prototipos.

Normalmente un cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos el programador, puede no estar seguro de la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma en que debe realizarse la interacción hombre-máquina. En estas y muchas otras situaciones, puede ser mejor método de ingeniería de software la construcción de un prototipo.

La construcción de prototipos es un proceso que facilita al programador la creación de un modelo de software a construir, el modelo tomará una de las tres siguientes formas:

1. Un prototipo en papel o un modelo basado en computadora persona, que describa la interacción hombre máquina, de forma que facilite al usuario la compresión de cómo se producirá tal iteración.
2. un prototipo que implemente algunos subconjuntos de la función requerida del programa deseado.
3. Un programa existente que ejecute parte o toda la función deseada pero que tenga otras características que deben ser mejoradas en el nuevo trabajo de desarrollo.

La figura 1.1 muestra la secuencia de sucesos del paradigma de construcción de prototipos. Como en todos los métodos de desarrollo de software, la construcción de prototipos comienza con la recolección de requisitos. El técnico y el cliente se reúnen y definen los objetivos globales para el software, identifican todos los requisitos conocidos y perfilan las áreas donde será necesario una mayor definición. Luego se produce un diseño rápido. El diseño rápido se enfoca sobre la representación de los aspectos del software visibles al usuario (por ejemplo, métodos de entrada y formatos de salida). El diseño rápido conduce a la construcción de un prototipo, este prototipo es evaluado por el cliente – usuario y se utiliza para refinar los requisitos del software a desarrollar.

Se produce un proceso interactivo en el que el prototipo es afinado para que satisfaga las necesidades del cliente, al mismos tiempo que facilita al que desarrolla una mejor compresión de lo que hay que hacer.

Idealmente el prototipo sirve como mecanismo para identificar los requisitos del software. Si se va a construir un prototipo que funcione, el realizar intenta hacer uso de fragmentos de programas existentes o aplica herramientas, tales como: generadores de informes, generadores de ventanas y otros, que faciliten la rápida generación de programas que funcionen.

El prototipo puede servir como "primer sistema", pero este puede ser ua visión idealizada al igual que en el ciclo de vida clásico. La construcción de prototipos como paradigma para la ingeniería del software, puede ser problemática por las siguientes razones:

1. El cliente ve funcionando lo que parece ser una primera versión del software ignorando que por las prisas de hacer que funcione, no hemos considera los aspectos de calidad o de mantenimiento del software a largo plazo. Cuando se le informa de que el software debe ser reconstruido, el cliente se vuelve loco y solicita que se le aplica cuantas mejoras sean necesarias para hacer del prototipo un producto final que funcione. El gestor del desarrollo del software cede demasiado a menudo.
2. El técnico de desarrollo frecuentemente impone ciertos compromisos de implementación con el fin de obtener un prototipo que funcione rápidamente, puede que utilice un sistema operativo o un lenguaje de programación inapropiados, simplemente porque ya está disponibles y son conocidos, puede que implemente ineficientemente un algoritmo sencillamente para demostrar su capacidad, después de algún tiempo el técnico puede haberse familiarizado con esas elecciones y haber olvidado las razones por las que eran inapropiadas. La elección menos ideal forma ahora parte integral del sistema.

Fig. 1.1 Creación de prototipos

Aunque pueden aparecer problemas, en la construcción del prototipo es un paradigma efectivo para la ingeniería del software. la clave está en definir al comienzo las reglas del juego; esto es, el cliente y el técnico deben de estar de acuerdo en que el prototipo se construya para servir solo como un mecanismo de definición de los requisitos. Posteriormente ha de ser descartado y debe construirse el software real con el pensamiento en la calidad y en el mantenimiento.

1.1.7 Modelo en espiral.

El modelo en espiral para la ingeniería de software ha sido desarrollado para cubrir las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos añadiendo al mismo tiempo un nuevo elemento, el análisis de riesgo, que falta en esos paradigmas. El modelo en espiral esta representado en la figura 1.2 en donde se definen las cuatro actividades principales.

Planificación: Determinación de Objetivos, alternativas y restricciones.

Análisis de riesgo: Análisis de alternativas e identificación /resolución de riesgos.

Ingeniería: Desarrollo del producto de siguiente nivel.

Evaluación del cliente: Valoración de los resultados de la ingeniería.

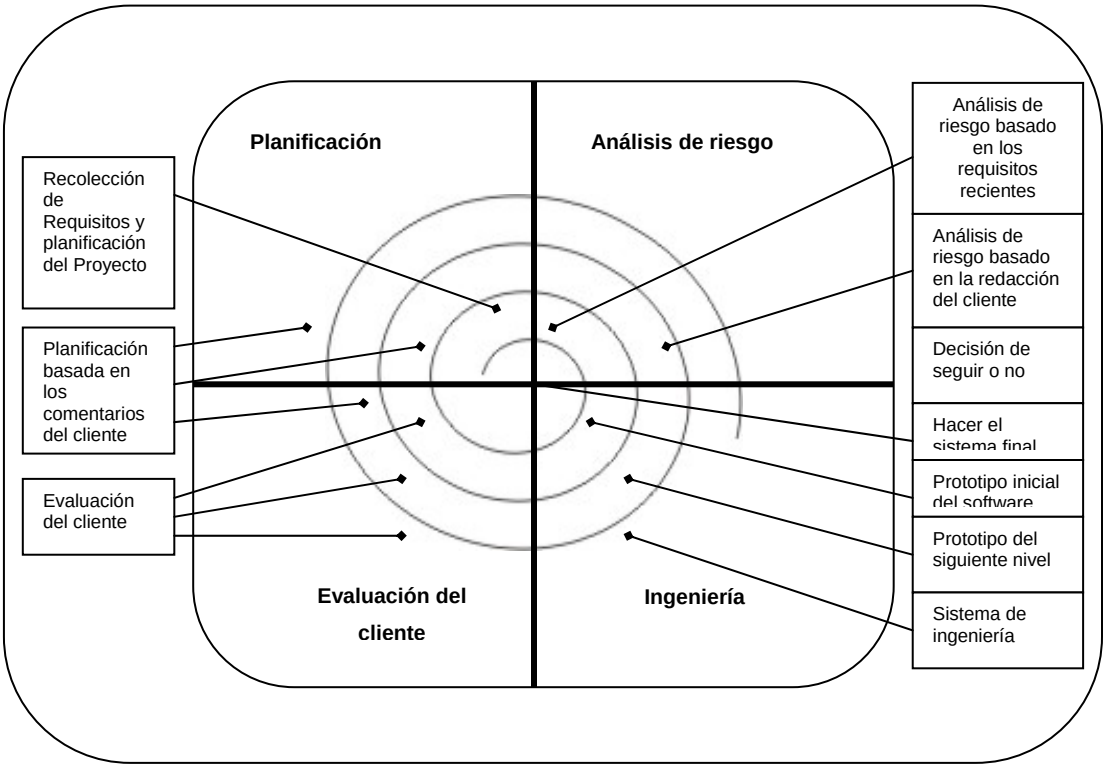


Figura 1.2 El modelo en espiral

Un modelo integrante del modelo en espiral se hace evidente cuando consideramos la dimensión radial representada en la figura 1.2. Con cada iteración alrededor (iniciando en el centro y siguiendo hacia el exterior), se construyen sucesivas versiones del software, cada vez más completas. Durante la primera vuelta alrededor del espiral se definen los objetivos, las alternativas y las restricciones, y se analizan e identifican los riesgos. Si el analista de riesgos indica que existe incertidumbre en los requisitos, para usar la creación de prototipos en el cuadrante de ingeniería para dar asistencia tanto al encargado del desarrollo como al cliente. Se pueden usar

simulaciones y otros modelos para definir más el problema y refinar los requisitos.

El cliente evalúa el trabajo de ingeniería y sugiere modificaciones. En base a los comentarios del cliente se produce la siguiente fase de planificación y de análisis de riesgo. En cada bucle alrededor del espiral, la culminación del análisis de riesgo resulta en una decisión de continuar a no, si los riesgos son demasiado grandes, se puede dar por terminado el proyecto.

Sin embargo en la mayoría de los casos se sigue avanzando alrededor del camino del espiral, y ese camino lleva a los desarrolladores hacia afuera, hacia un modelo más completo del sistema, y al final al propio sistema operacional, cada vuelta alrededor del espiral requiere ingeniería, que se puede llevar a cabo mediante el enfoque del ciclo de vida clásico o de la creación de prototipos. Debe tenerse en cuenta que el número de actividades de desarrollo que ocurren en el cuadrante inferior derecho aumenta al alejarse del centro del espiral.

El paradigma del modelo en espiral para la ingeniería de software es actualmente el enfoque más realista para el desarrollo de software y de sistemas a gran escala utiliza un enfoque evolutivo para la ingeniería de software, permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo, utiliza la creación de prototipos como un mecanismo de reducción de riesgos, pero lo más importante es que permite a quien lo desarrolla, aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución del producto. Mantiene el enfoque sistemático correspondiente a los pasos sugeridos por el ciclo de vida clásico, pero incorporándolo dentro de un marco de trabajo interactivo que refleja de forma más realista el mundo real. El modelo en espiral demanda una consideración directa de riegos técnicos en todas las etapas del

proyecto y si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemas.

Pero al igual que otros paradigmas, el modelo en espiral no es la panacea. Puede ser difícil convencer a grandes clientes, de que el enfoque evolutivo es controlable. Requiere una considerable habilidad para la valoración del riesgo para tener éxito. Si no se descubre un riesgo importante, indudablemente surgirán problemas. Por último el modelo en sí mismo es relativamente nuevo y no se ha utilizado tanto como el ciclo de vida clásico o la creación de prototipos. Pasaran unos cuantos años antes de que se pueda determinar con absoluta certeza la eficacia de este importante paradigma.

1.2 BASES DE DATOS

1.2.1 introducción a las bases de datos.

Un sistema de gestión de bases de datos (DBMS database management system) consiste en una colección de datos interrelacionados y u conjunto de programas para acceder a esos datos. La colección de datos, normalmente denominada base de datos, contiene información acerca de una empresa determinada. El objetivo primordial de un DBMS es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos.

Los sistemas de bases de datos están diseñados para gestionar grandes bloques de información. La gestión de datos implica tanto la definición de estructuras para el almacenamiento de información como la provisión de mecanismos para la gestión de la información. Además, los sistemas de bases de datos deben mantener la seguridad de la información almacenada,

pese a caídas del sistema o intentos de acceso no autorizados. Si los datos van a ser compartidos por varios usuarios, el sistema debe evitar posibles resultados anómalos.

1.2.2 Objetivo de los sistemas de bases de datos.

El típico sistema de procesamiento de archivos, es apoyado por un sistema operativo convencional. Los registros permanentes se almacenan en varios archivos, y se escribe un número de diferentes programas de aplicación para extraer y añadir registros a archivos apropiados. Este sistema tiene un número de desventajas importantes.

1.2.2.1 Redundancia e inconsistencia de los datos.

Puesto que los archivos y los programas de aplicación son creados por distintos programas durante un periodo largo de tiempo, es probable que los archivos tengan diferentes formatos y los programas pueden estar duplicados en varios sitios. Por ejemplo el sueldo y la categoría de un trabajador pueden estar almacenados en la tabla de prestaciones y en la tabla de estímulos. Esta redundancia aumenta los costos de almacenamiento y acceso, además puede llevar a inconsistencia de los datos, esto es, las diversas copias de los datos no concuerdan entre si. Por ejemplo la categoría de un empleado puede estar actualizado en la base prestaciones pero puede ser que no este actualizado en la base de datos plantilla. El resultado es una inconsistencia de los datos.

1.2.2.2 Aislamiento de los datos.

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

1.2.2.3 Anomalías de accesos concurrente.

Para mejorar el funcionamiento global del sistema y obtener un tiempo de respuesta más rápido, muchos sistemas permiten que múltiples usuarios actualicen los datos simultáneamente. En un entorno así la interacción de actualizaciones concurrentes puede dar por resultados datos inconsistentes.

1.2.2.4 Problemas de seguridad.

No todos los usuarios del sistema de bases de datos deben poder acceder a todos los datos. Por ejemplo, en una nómina, solo el personal encargado de ella podrá acceder e esa información, de lo contrario todos se pondrán el sueldo que quieran y se llega a tener una base de datos incosistente.

1.2.2.5 Problemas de integridad.

Los valores de datos almacenados en la base de datos deben satisfacer ciertos tipos de restricciones de consistencia. Por ejemplo el saldo en una cuenta bancaria nunca debe ser por debajo de una cantidad establecida (digamos 50, 500, 1000 etc.). Estas restricciones se hacen cumplir en el sistema añadiendo códigos apropiados en los diversos programas de aplicación.

1.2.3 Abstracción de datos.

Un sistema de gestión de archivos de bases de datos es una colección de archivos interrelacionados y conjunto de programas que permiten a los usuario a acceder y modificar esos archivos. Un objetivo importante de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos, sin embargo para que el sistema sea manejable, los datos se deben extraer eficientemente. Este requerimiento a llevado al diseño de estructuras de datos complejas para la

representación de datos en las bases de datos. Puesto que muchos usuarios de sistemas de bases de datos no tienen mucha experiencia, se les esconde la complejidad a través de diversos niveles de abstracción para simplificar su interacción con el sistema.

1.2.3.1 Nivel Físico

es el nivel más bajo de abstracción, describe como se almacenan realmente los datos. En el nivel físico se describen en detalle las estructuras de datos complejas del nivel bajo.

1.2.3.2 Nivel conceptual.

El siguiente nivel más alto de abstracción describe sólo parte de la base de datos completa. A pesar del uso de estructuras más sencillas en el nivel conceptual, permanece algo de complejidad debido al gran tamaño de la base de datos.

La interrelación entre estos tres niveles se ilustran en la figura 1.3

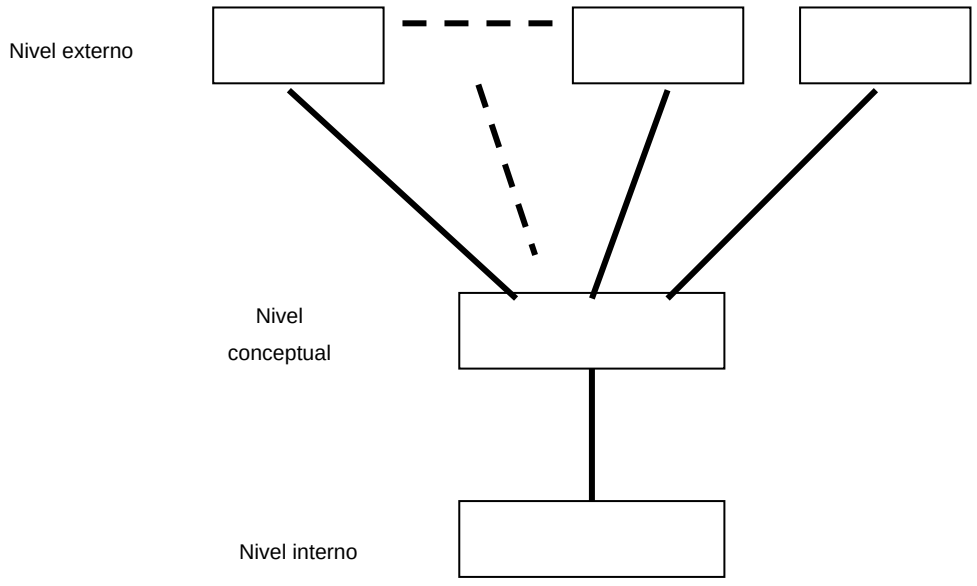


Fig. 1.3 Los tres niveles de la arquitectura

Una analogía con el concepto de tipos de datos en los lenguajes de programación puede clarificar la distinción entre niveles de abstracción. La mayoría de los lenguajes de programación de alto nivel soporta la noción de un tipo de registro. Por ejemplo en un lenguaje como pascal podemos declarar un registro como sigue:

```
Type alumno =record
        Nombre:String
        Calle:String
        Ciudad:String
        End.
```

Esto define un registro nuevo llamado cliente con tres campos. Cada campo tiene un nombre y un tipo asociado a el. En un instituto puede haber varios tipos de registro, como este:
carga. Con los campos num_cont, materia etc.
kardex. Con los campos num_cont, materia etc.

en el nivel físico, un registro de alumno, carga y kardex puede describirse como un bloque de posiciones de memoria consecutivas. En el nivel conceptual, cada uno de estos registros se describe por medio de una definición de tipo, y se define la interrelación entre estos tipos de registro. Finalmente en el nivel de visión se definen varias visiones de la base de datos.
La siguiente figura 1.4 muestra los tres niveles de un DBMS

Arquitectura de un DBMS.

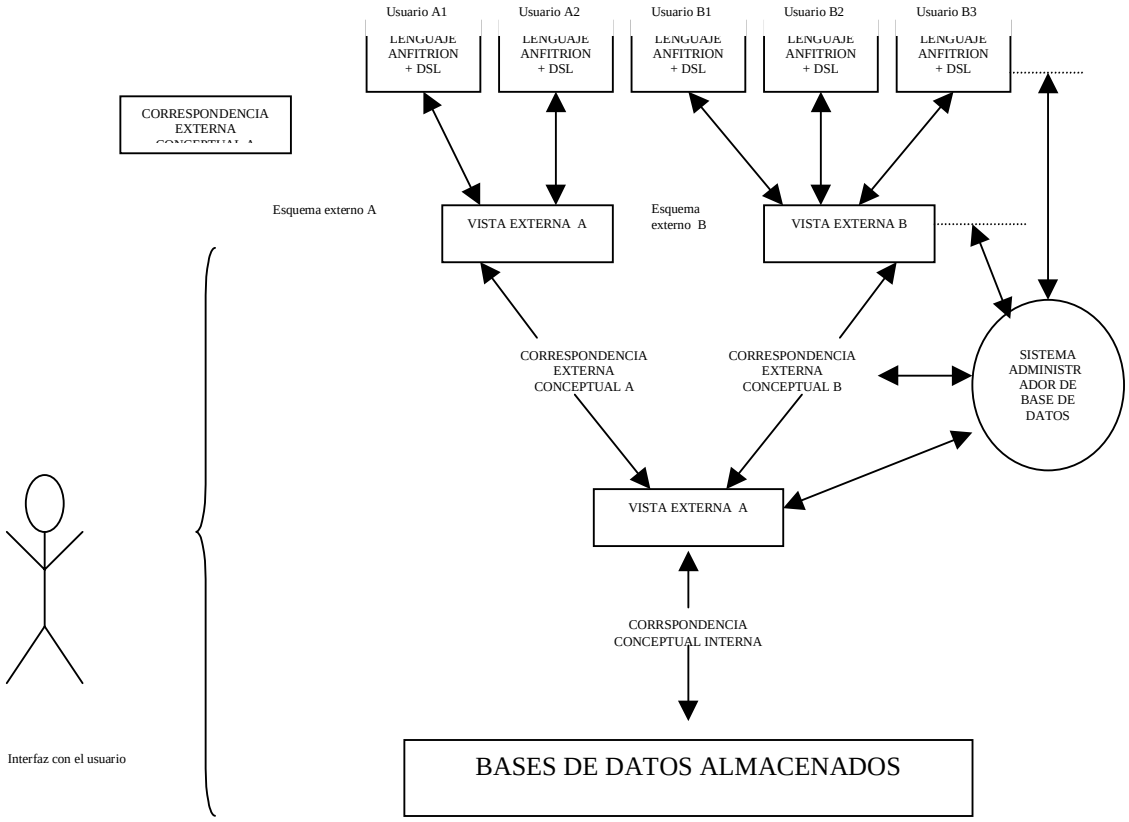


Fig. 1.4 Arquitectura detallada del sistema de bases de datos

1.2.4 Modelo de datos.

Para describir la estructura de una base de datos es necesario definir el concepto de modelo de base de datos, una colección de herramientas conceptuales para describir datos, relacionados entre ellos, se mantica asociada entre a los datos y restricciones de consistencia. Los diversos modelos de datos que se han propuesto se dividen en tres grupos, los cuales son: Modelos lógicos basados en objetos, Modelos lógicos basados en registros y Modelos físicos de datos.

1.2.4.1 Modelos lógicos basados en objetos.

Los modelos lógicos basados en objetos se usan para describir datos en los niveles conceptual y de visión, se caracterizan por el hecho de que proporcionan capacidad de estructuración flexible y permiten especificar restricciones de datos explícitamente. Existen muchos modelos diferentes, algunos de los más extensamente conocidos son:

- El modelo entidad relación.
- El modelo orientado a objetos.
- El modelo binario.
- El modelo semántico de datos.
- El modelo infológico.
- El modelo funcional de datos.

El modelo entidad relación (E-R).

El modelo entidad relación se basa en una percepción de un mundo real que consiste en una colección de objetos básicos llamados entidades y relaciones entre estos objetos. Una entidad es un objeto que es distinguible de otros objetos por medio de un conjunto específico de atributos. Una relación es una asociación entre varias entidades.

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

La estructura lógica global de una base de datos puede expresarse gráficamente mediante un diagrama E-R, que consta de los siguientes componentes:

- Rectángulos, que representas conjuntos de entidades.
- Elipses, que representan atributos.
- Rombos, que representan relaciones entre conjuntos de entidades.
- Líneas que conectan atributos a conjuntos de entidades y conjuntos de entidades a relaciones.

Cada componente se etiqueta con la entidad o relación que representa un diagrama E-R, se ilustra en la figura 1.5

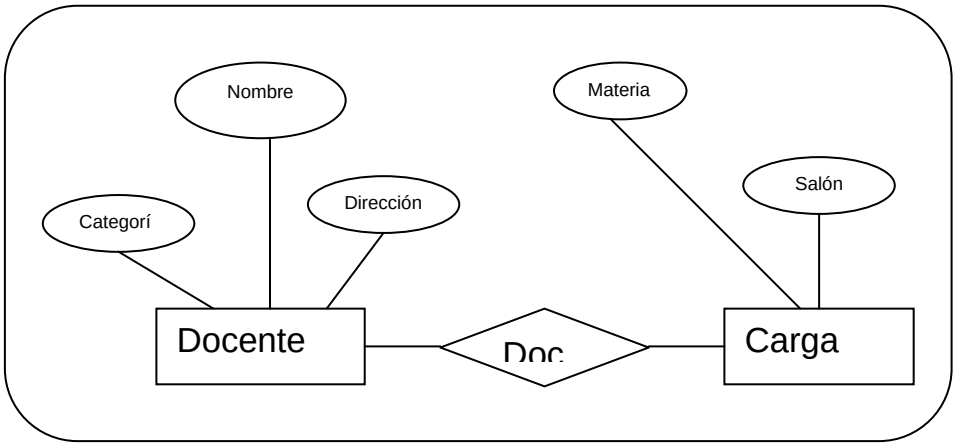


Figura 1.5 Ejemplo de un diagrama de Entidad - Relación

1.2.4.2 El modelo orientado a objetos.

Al igual que el modelo E-R, el modelo orientado a objetos se basa en una colección de objetos. Un objeto contiene valores almacenados en variables instancia dentro del objeto. A diferencia de los modelos orientados a registros, estos valores son objetos por sí mismos, así los objetos contienen objetos a nivel de anidamiento de profundidad arbitraria. Un objeto también

contiene partes de código que operan sobre el objeto. Estas partes se llaman métodos.

Los objetos que contienen los mismos tipos de valores y los mismos métodos se agrupan en clases. Una clase puede ser vista como una definición de tipo para objetos. Esta combinación de datos y código en una definición de tipo es parecida al concepto de tipos de datos abstractos en los lenguajes de programación.

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

Para ilustrar el concepto, considere un objeto que representa una cuenta bancaria. Dicho objeto contiene las variables de instancia número y saldo, que representa el número de cuenta y el saldo de cuenta. Contiene un método interés de pago, que añade interés al saldo. Supóngase que el banco había estado pagando el 6% de interés en todas las cuentas, pero ahora está cambiando su política a pagar el 5% si el saldo es menor de 1000 pesos o el 6% si el saldo es 1000 pesos o mayor. Bajo la mayoría de los modelos de datos, esto implicaría cambiar de código de uno o más programas de aplicación. Bajo este modelo, solo se hace un cambio dentro del método interés de pago y la interfaz externa del objeto permanece sin cambios.

1.2.4.3 Modelos lógicos basados en registros.

Los modelos lógicos basados en registros se utilizan para describir datos en los modelos conceptual y físico. A diferencia de los modelos de datos

basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

Los modelos basados en registros se llaman así porque la base de datos está estructurada en registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos, o atributos, y cada campo normalmente es de longitud fija.

Los modelos de datos basados en registros no incluyen un mecanismo para la representación directa de código en la base de datos. En cambio hay lenguajes separados que asocian con el modelo para expresar consultas y actualizaciones de la base de datos. Los tres modelos de datos más ampliamente aceptados son los modelos relacional, de red y jerárquico.

1.2.4.4 Modelo Relacional.

El modelo relacional representa los datos y las relaciones entre los datos mediante una colección de tablas, cada de las cuales tiene un número de columnas con nombres únicos. Ejemplo de una base de datos relacional se muestra en la figura 2.3.

A.cat	descripción
174	PROGRAMADOR C
175	PROGRAMADOR B
176	PROGRAMADOR A
177	AUXILIAR DE PROGRAMADOR

B.cat	Pago
174	234.99
175	232.65
176	456.89
177	567.88
187	678.9
189	446.99

Figura 1.6 ejemplo de una base de datos relacional.

1.2.4.5 Modelo de Red

Los datos en el modelo de red se representan mediante colecciones de registros y las relaciones entre los datos se representan mediante enlaces, los cuales pueden verse como punteros. Los registros en la base de datos se organizan como colecciones de gráfos arbitrarios. En la Figura 2.4 se representa una base de datos en el modelo de red.

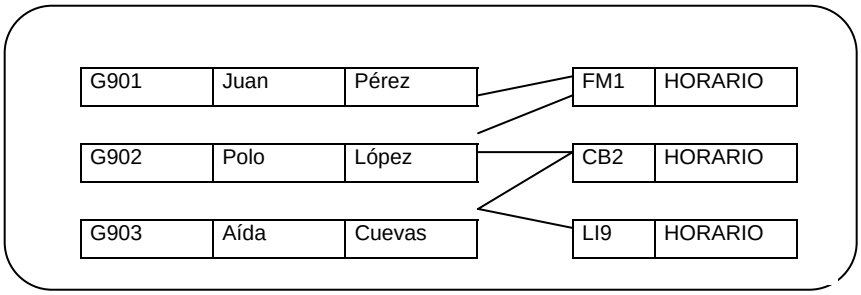
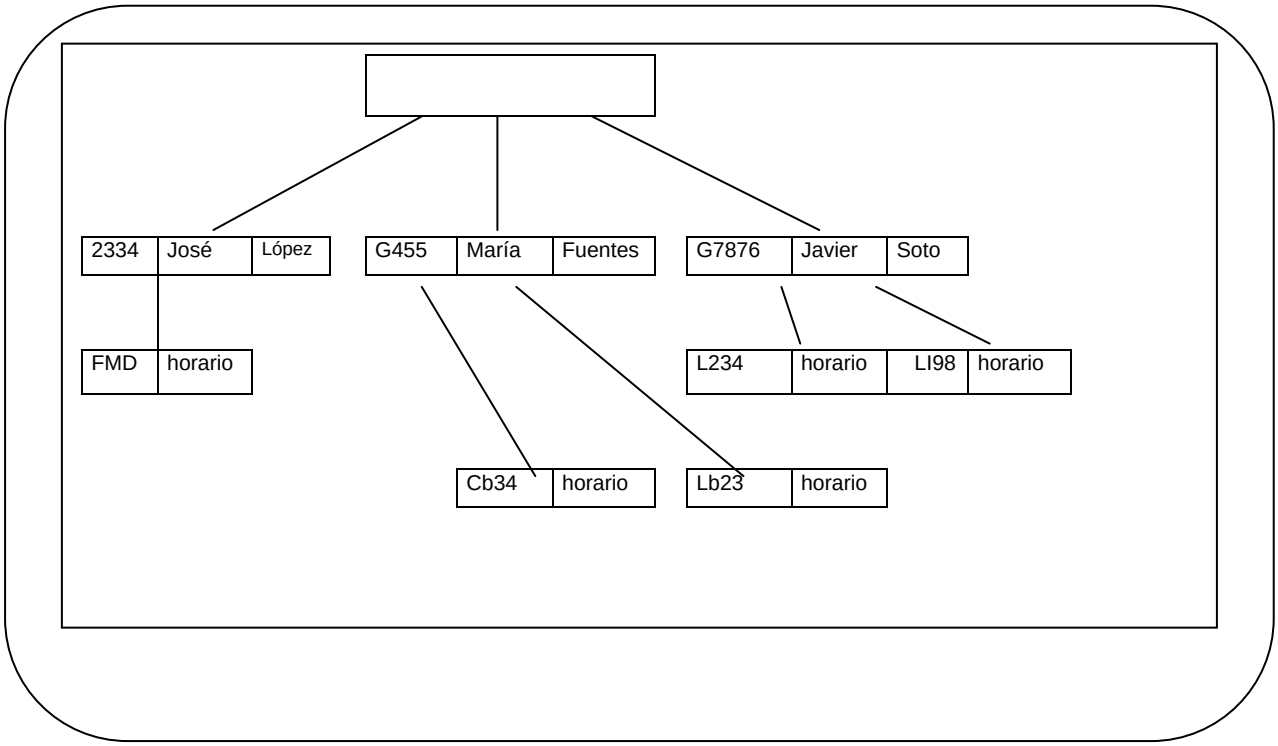


Figura 1.7 Base de datos en modelo de red

1.2.4.6 Modelo jerárquico

El modelo jerárquico es similar al modelo de red en el sentido de que los datos se representan mediante registros y enlaces, respectivamente. Se diferencian del modelo de red en que los registros están organizados como colecciones de árboles en vez de grafos arbitrarios. La figura 2.5 presenta un ejemplo de la base de datos jerárquica.



1.2.5 Instancias y esquemas.

Las bases de datos cambian a lo largo del tiempo según se añade y se suprime información. La colección de información almacenada en la base de datos, en un determinado momento del tiempo, se llama una instancia de la base de datos. El diseño global de la base de datos se llama esquema de la base de datos. Los esquemas se cambian muy raras veces o nunca.

El concepto de esquema de la base de datos corresponde a la noción de definición de tipo en el lenguaje de programación. Una variable de un tipo

dado tiene un valor determinado en un instante de tiempo dado. Así el concepto del valor de una variable en los lenguajes de programación corresponde al concepto de una instancia de un esquema de la base de datos.

1.2.6 Independencia de datos.

En la sección 1.2.3 se definió los tres niveles de abstracción en los que puede verse la base de datos. La capacidad de modificar una definición de un esquema en un nivel sin afectar la definición de un esquema en el nivel superior siguiente se llama independencia de datos. Hay dos niveles de independencia de datos.

- Independencia física de datos. Es la capacidad de modificar el esquema físico sin provocar que se vuelva a escribir los programas de aplicación. En algunas ocasiones son necesarias las modificaciones en el nivel físico para mejorar el funcionamiento.
- Independencia lógica de datos. Es la capacidad de modificar el esquema conceptual sin provocar que se vuelva a escribir los programas de aplicación. Las modificaciones en el nivel conceptual son necesarias siempre que se altera la estructura lógica de la base de datos.

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

1.2.7 Lenguaje de definición de datos.

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

Un diccionario de datos es un archivo que contiene metadatos, es decir datos entre datos. Este archivo se consulta antes de leer o modificar los datos reales en el sistema de bases de datos. La estructura de almacenamiento y los métodos de acceso usados por los sistemas de bases de datos se especifican por medio de un conjunto de definiciones en un tipo especial de DDL llamado lenguaje de almacenamiento y definición de datos. El resultado de la compilación de estas definiciones es un conjunto de instrucciones que especifican los detalles de implementación de los esquemas de bases de datos que normalmente se esconde a los usuarios.

1.2.8 Lenguaje de manipulación de datos.

Los noveles de abstracción que mencionamos en la sección 1.2.3, se aplican no solo a la definición o estructura de datos sino también a la manipulación de datos, por manipulación de datos queremos decir.

- La recuperación de información almacenada en la base de datos
- La inserción de información nueva en la base de datos.
- La supresión de información en la base de datos.
- La modificación de datos almacenada en la base de datos.

A nivel físico, debemos definir algoritmos que permitan acceso eficiente a los datos. El objetivo es proporcionar una interacción eficiente entre las personas y el sistema. Un lenguaje de manipulación de datos (DML) es un lenguaje que capacita a los usuarios a acceder o manipular datos según estén organizados por el modelo de datos adecuado. Existen básicamente dos tipos:

- Procedurales. Los DML requieren que el usuario especifique que datos se necesitan y como obtenerlos.
- No procedurales. Los DML requieren que el usuario especifique qué datos se necesitan sin especificar como obtenerlos.

Los DML no procedurales normalmente son más sencillos de aprender y usar que los procedurales. Sin embargo, puesto que el usuario no tiene que especificar como conseguir los datos, estos lenguajes pueden generar código que no sea tan eficiente como el producido por los lenguajes procedurales.

1.2.9 Gestor de bases de datos.

Un gestor de bases de datos, es un módulo de programas que proporciona la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y consulta hechos al sistema. El gestor de bases de datos es responsable de las siguientes tareas:

- Interacción con el gestor de archivos. Los datos sin procesar se almacenan en el disco usando el sistema de archivos que normalmente es proporcionado por un sistema operativo convencional.

- Implantación de la integridad. Los valores de los datos que se almacenan en la base de datos deben satisfacer ciertos tipos de restricciones de consistencia.
- Implantación de la seguridad. Como se discutió anteriormente, no todos los usuarios de la base de datos, necesitan tener acceso a todo su contenido, es trabajo del gestor de la base de datos hacer que se cumplan estos requisitos de seguridad.
- Copia de seguridad y recuperación. Un sistema informático, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallas, las causas de las fallas incluyen roturas de discos, problemas del suministro de energía y errores de software. en cada uno de estos casos, se pierde información referente a las bases de datos. Es responsabilidad del gestor de la base de datos detectar tales fallos y restaurar la base de datos al estado que tenía antes de ocurrir el fallo.
- Control de concurrencia. Cuando varios usuarios actualizan la base de datos concurrentemente, es posible que no se conserve la consistencia de los datos. Controlar la interacción entre los usuarios concurrentes es otra responsabilidad del gestor de la base de datos.

1.2.10 Administrador de la base de datos.

Una de las razone principales para tener sistemas de gestión de bases de datos es tener control central de datos y de los programas que acceden a estos datos. La persona que tiene dicho control central sobre el sistema se llama administrador de la base de datos (DBA) las funciones de del DBA son:

- Definición de esquemas. El esquema original de la base de datos se crea escribiendo un conjunto de definiciones que son traducidas por el

compilador de DDL a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.

- Definición de la estructura de almacenamiento y del método de acceso. Es también encargado de definir las estructuras de almacenamiento y los métodos de accesos de la base de datos.
- Modificación del esquema y de la organización física. Las modificaciones tanto al esquema de la base de datos como a la descripción de la organización física de almacenamiento.
- Concesión de autorización para el acceso de datos. Da concesión y autorización para acceder a la base de datos, regulado las partes que pueden ser accedidas por diferentes usuarios.
- Especificación de las restricciones de integridad. Debe vigilar la modificaciones a la base de datos, para que este se mantenga íntegra.

1.3 MODELO RELACIONAL.

1.3.1 Introducción.

Desde una perspectiva histórica el modelo de datos relacional es relativamente nuevo. Los primeros sistemas de bases de datos estaban basados en el modelo de red o en el modelo de datos jerárquico. Estos dos primeros modelos están más íntimamente ligados a la implementación física de la base de datos que el modelo relacional.

En los años siguientes a la introducción del modelo relacional se ha desarrollado una teoría esencial para las bases de datos relacionales, esta teoría ayuda al diseño de base de datos relacionales y al procesamiento eficiente de solicitudes de información a la base de datos por parte de los usuarios. El modelo relacional se ha establecido como el principal modelo de datos para aplicaciones comerciales de procesamiento de datos, su

éxito en este campo ha llevado a su aplicación fuera del procesamiento de datos en sistemas para diseño asistido por computadora y otros entornos.

1.3.2 Estructura de la base de datos relacional.

Una base de datos relacional consiste en una colección de tablas, a cada una de las cuales se asigna un nombre único. Cada tabla tiene una estructura similar a la representada anteriormente. Una fila de una tabla representada una relación entre conjunto de valores. Puesto que una tabla es una colección de dichas relacionales, hay una estrecha correspondencia entre el concepto de tabla y el concepto matemático de relación, del cual se toma su nombre el modelo de datos relacional.

1.3.3 Esquema de la base de datos.

Cuando hablamos de una base de datos debemos diferenciar entre el esquema de la base de datos o el diseño lógico de la base de datos, y una instancia de la base de datos, que son los datos en un instante de tiempo dado.

El concepto de esquema de una relación correspondiente a la noción de definición de tipo en los lenguajes de programación. Una variable de un tipo en los lenguajes de programación. Una variable de un tipo dado tiene un valor determinado en un instante de tiempo dado. Así, una variable en los lenguajes de programación correspondiente al concepto de una instancia de una relación. A continuación, se muestra un ejemplo de esquema de una relación:

Esquema - Alumnos = (No de control, Nombre, Apepat, Apemat, etc.)

1.3.4 Lenguaje de consulta.

Un lenguaje de consulta es un lenguaje en el que un usuario solicita información de la base de datos. Estos lenguajes son normalmente de más alto nivel que los lenguajes estándar de programación. Los lenguajes de consulta pueden clasificarse en lenguajes procedimentales o no procedimentales. En un lenguaje procedimental, el usuario da instrucciones al sistema para que realice una secuencia de operaciones en la base de datos para calcular el resultado deseado. En un lenguaje no procedimental, el usuario describe la información deseada sin dar un procedimiento específico para obtener esa información.

La mayor parte de los sistemas comerciales de bases de datos relacionales ofrecen un lenguaje de consulta que incluyen elementos de los dos enfoques: procedimental y no procedimental. El álgebra relacional es procedimental, mientras que el cálculo relaciona de tuplas y el cálculo relacional de dominios son no procedimentales. Estos lenguajes de consulta son concisos y formales pero ilustran las técnicas fundamentales para extraer datos de la base de datos. Un lenguaje de manipulación de datos complejo incluye no solo un lenguaje de consulta, sino también un lenguaje para la modificación de la base de datos. Dichos lenguajes incluyen órdenes para insertar y borrar tuplas así como órdenes para modificar partes de tuplas existentes.

1.3.5 Algebra relacional.

El álgebra relacional es un lenguaje de consulta procedimental, consta de un conjunto de operaciones que toman una o más relaciones como entradas y producen una nueva relación como resultado. Las operaciones fundamentales en el álgebra relacional son: seleccionar, proyectar, producto cartesiano, renombrar y diferencia de conjuntos. Además de las

operaciones fundamentales en el álgebra relacional existen otras operaciones, a saber, intersección de conjuntos, producto natural, división y asignación, estas operaciones se definen en términos de las operaciones fundamentales.

1.3.5.1 Operaciones fundamentales.

La manipulación del álgebra relacional se divide en dos partes:

1. Un conjunto de operadores que forman lo que se le llama álgebra relacional.
2. Una operación de asignación que asigna el valor de alguna expresión arbitraria del álgebra a una relación nombrada.

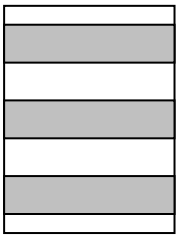
En si el álgebra relacional consiste en un conjunto de operadores que operan sobre relaciones, los cuales toman como entrada una o dos relaciones y produce otra nueva relación como salida.

En seguida se presentan los ocho operadores originales presentados por Codd.

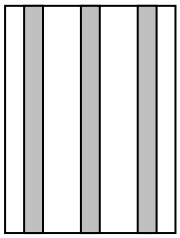
- *Restricción*. Extrae las tuplas especificadas de una relación dada.
- *Proyección*. Extrae los atributos de una relación especificada.
- *Producto*. A partir de dos relaciones especificadas, construye una relación que contiene todas las combinaciones posibles de tuplas, una de cada una de las dos relaciones.
- *Unión*. A partir de dos relaciones especificadas, construye una relación que contiene todas las tuplas que aparecen en cualquiera de las dos relaciones.
- *Intersección*. Construye una relación formada por las tuplas que aparecen en las dos relaciones especificadas.

- *Diferencia*. Construye una relación formada por todas las tuplas que aparecen en la primera relación, pero que no se encuentran en la segunda relación.
- *Reunión.* Construye una relación formada por todas las combinaciones posibles de tuplas, una de cada una de las dos relaciones, tales que las dos tuplas participantes en una combinación dada satisfagan una condición especificada.
- *División.* Toma una relación binaria y otra unaria, y construye una relación formada por todos los valores de un atributo de la relación binaria que concuerdan con todos los valores de la relación unaria.

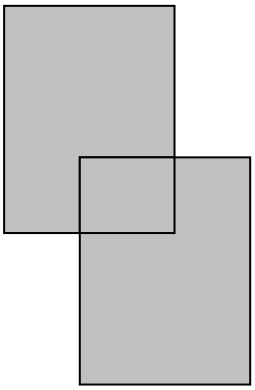
Restricción



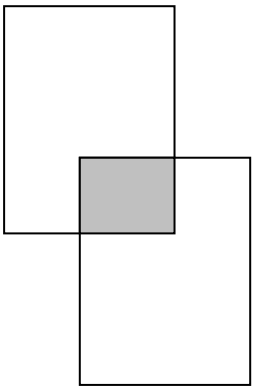
Proyección



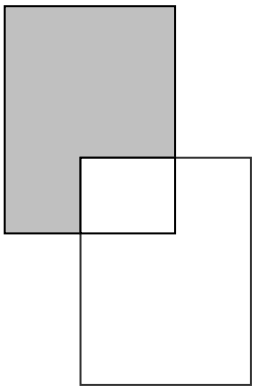
Unión



Intersecció



Diferenci



Produc

a
b

x

=

a	x
a	x
b	y

Reunión

a	b
1	1

b	c
1	1

a	b	c
1	1	1

División

a	x
a	y
a	z

x

a

Como se observa las operaciones cumplen con la propiedad de cerradura ya que el resultado de cada una de las operaciones es otra relación.

1.3.6 Operaciones tradicionales de conjuntos

E matemáticas las operaciones tradicionales de los conjuntos son unión, intersección, diferencia y producto. En el álgebra relacional como en términos informales una relación es un conjunto, es obvio que estas operaciones también se puedan realizar.

Empecemos con la unión, esta en el álgebra relacional es el conjunto de todas las tuplas que se encuentra en una relación original o ambas. Pero aunque el resultado es otro conjunto, es puede que no sea una relación ya que una relación no acepta una mezcla de tuplas de diferente tipo. Para que el resultado sea otra relación y conservar la propiedad de cerradura las tuplas de las relaciones deben ser homogéneas, en términos precisos deben cumplir lo siguiente:

- Tener el mismo conjunto de nombres de atributos
- Los atributos se deben definir sobre los mismos dominios

Lo anterior es necesario para realizar las operaciones de unión, intersección y diferencia en el álgebra relacional.

Veamos un ejemplo para mostrar las operaciones tradicionales con las dos relaciones, la primera muestra todos los fondos de donde se pueden obtener recursos y la segunda los fondos de donde una escuela X obtuvo recursos en el año de 1998.

Relación Recursos

Año	Recursos
1998	FOMES
1998	PROPIOS
1998	GENERICO
1998	FONDO FIJO
1998	PROADU
1998	CAPFCE
1998	ESTIMULOS
1998	CONACYT

Relación Gastos

Año	Recursos
1998	FOMES
1998	PROPIOS
1998	GENERICO
1998	FONDO FIJO

Veamos ahora el resultado de la unión de estas dos relaciones.

Como se verá la unión es la relación Recursos.

Resultado de la unión Recursos con Gastos

Año	Recursos
1998	FOMES
1998	PROPIOS

1998	GENERICO
1998	FONDO FIJO
1998	PROADU
1998	CAPFCE
1998	ESTIMULOS
1998	CONACYT

El resultado nos dice todos los fondos de donde una escuela puede tener recursos.

Ahora realicemos la intersección de las relaciones anteriores.

Resultado de la intersección de Recursos con Gastos

Año	**Recursos**
1998	FOMES
1998	PROPIOS
1998	GENERICO
1998	FONDO FIJO

La interpretación es que la escuela X obtuvo recursos solamente de los fondos que se muestran en el resultado.

Veamos ahora la resta de Recursos menos Gastos
El resultado es el siguiente:

Resultado de Recursos menos Gastos

Año	**Recursos**
1998	PROADU
1998	CAPFCE

1998	ESTIMULOS
1998	CONACYT

Como se observa el resultado nos dice que a esta escuela le falta conseguir recursos de los fondos mostrados en el resultado.

CAPITULO II

PLANIFICACION

2.1 PLANIFICACION

En esta etapa recolectaremos todos los requisitos y planificaremos el proyecto inicial basándonos en la estructura anteriormente descrita.

Dicha planeación será conforma a los comentarios realizados por el cliente, en este caso contabilidad que es el principal solicitante de información a cada una de las dependencias de la BUAP.

A continuación se describen las funciones de la dependencia que se desean incluirse en el sistema:

2.1.1 Dirección de contabilidad.

Los requerimientos y características del sistema después de hacer un análisis con distintas dependencias fueron las siguientes:

1. Ambiente windows 95 a 32 bits o posterior
2. Un programa que lo instale (Setup)
3. 80 por ciento visual
4. Almacenar todo tipo de ingreso a la dependencia
5. Almacenar todo tipo de egreso de la dependencia
6. Clasificar los ingresos por función
7. Clasificar los egresos por función
8. Modificar cualquier dato en caso de error
9. Emitir un reporte de ingreso por función
10. Emitir un reporte de egreso por función
11. Emitir un reporte de ingreso por fondo
12. Emitir un reporte de egreso por fondo

13. Diferenciar las bases de datos con la clave de la dependencia

Ambiente windows 95 a 32 bits.
El programa se ejecutará en windows 95 y debe utilizar la capacidad de 32 bits de los microprocesadores para ejecutar sus operaciones con mayor rapidez.

Un programa que lo instale (Setup)
Para facilitar su instalación en diferentes máquinas, contará con un programa que lo instale, el cual debe facilitar elegir el lugar donde será instalado, así como el cambio de discos si está en mas de uno.

80 por ciento visual.
El ambiente del programa no debe ser menos de 80 % visual, esto es, a través de menús y de otras herramientas de diferentes tipos se evitará que el usuario introduzca muchos datos por el teclado.

Almacenar todo tipo de ingreso a la dependencia.
Los diferentes tipos de ingresos se agruparan de la siguiente manera:

Ingresos propios
Fondo fijo
Conacyt
otros

Ingresos propios.
Solo estos se desglosarán de la siguiente manera:

SERVICIOS ESCOLARES

PRODUCTOS
MAESTRIAS
DIPLOMADOS
CURSOS ESPECIALES
CURSOS DE VERANO
SEMINARIOS DE TITULACION
COPIAS
VENTA DE PAPELERIA
VENTA DE LIBROS
REALZACION DE EVENTOS
PRODUCTOS FINANCIEROS
CONACYT
SEP
INGRESOS POR COBRAR
OTROS

Almacenar todo tipo de egreso de la dependencia.
Estos se desglosarán en los siguientes rubros.

HONORARIOS
HONORARIOS ASIMILABLES A SUELDOS
VIATICOS GASTOS Y PASAJES P DOCENTES
VIATICOS Y PASAJES P INVESTIGACION
VIATICOS Y PASAJES P ADMON.
PUBLICACIONES E IMPRESIONES
SERVICIOS COMERCIALES
MANTENIMIENTO
SERVICIOS GENERALES
ARTS. Y MATERIALES PARA DOCENCIA

ARTS. Y MATERIALES PARA INVESTIGACION
ARTS Y MATERIALES PARA ADMINSTRACION
ARTS. Y MATERIALES PARA DIFUSION CULT.
ART. Y MATERIALES PARA MANTENIMIENTO
GASTOS FINANCIEROS
FONDO COMUN 30%
OTROS

Clasificar los ingreso por función.
Estos se desglosarán en las siguientes funciones.

Función Docencia
Función Investigación
Función Extensión
Función apoyo

Clasificar los egresos por función
Al igual que los ingresos, quedarán clasificados en las diferentes funciones.
Función Docencia
Función Investigación
Función Extensión
Función apoyo

Modificar cualquier dato en caso de error
En caso de registrar un dato incorrecto el sistema será capaz de proporcionar un medio de modificación a esos datos erróneos.

Emitir un reporte de ingreso por función.

Este reporte debe cumplir con el formato siguiente:

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

Ingresos

DEPENDENCIA: NOMBRE DE DEPENDENCIA

PERIODO: 01 De Enero 1998 a 30 De Enero de 1998.

Función	**Ingreso**
Docencia	323.00
Investigación	989.00
Extensión	565.00
Apoyo	0.0
TOTAL:	34,455.00

Emitir un reporte de egreso por función

Es te reporte será el siguiente.

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

Egresos

DEPENDENCIA: NOMBRE DE DEPENDENCIA

PERIODO: 01 De Enero 1998 a 30 De Enero de 1998.

Función	**Egreso**
Docencia	323.00
Investigación	989.00
Extensión	565.00
Apoyo	0.0
TOTAL:	34,455.00

Emitir un reporte de ingreso por fondo

El reporte debe ser en una hoja completa como a continuación se muestra

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

Ingresos

DEPENDENCIA: NOMBRE DE DEPENDENCIA

PERIODO: 01 De Enero 1998 a 30 De Enero de 1998.

Función	**Ingreso**
SERVICIOS ESCOLARES	33.00
PRODUCTOS	655.00
MAESTRIAS	34.00
DIPLOMADOS	455.00
CURSOS ESPECIALES	455.00
CURSOS DE VERANO	3344.00
SEMINARIO DE TITULACION	344.00
COPIAS	2434.00
VENTA DE PAPELERIA	233.00
VENTA DE LIBROS	332.99
REALZACION DE EVENTOS	333.00
PRODUCTOS FINANCIEROS	12332.02
CONACYT	343.00
SEP	334.00
INGRESOS POR COBRAR	242.00
OTROS	2333.00
TOTAL:	34,455.00

Emitir un reporte de egreso por fondo

También este debe ser de hoja completa.

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

Egresos

DEPENDENCIA: NOMBRE DE DEPENDENCIA

PERIODO: 01 De Enero 1998 a 30 De Enero de 1998.

Función	**Egreso**
HONORARIOS	34.00
HONORARIOS ASIMILABLES A SUELDOS	3454.00
VIATICOS GASTOS Y PASAJES P DOCENTES	345.00
VIATICOS Y PASAJES P INVESTIGACION	355.88
VIATICOS Y PASAJES P ADMON.	234.00
PUBLICACIONES E IMPRESIONES	123.98
SERVICIOS COMERCIALES	234.00
MANTENIMIENTO	234.99
SERVICIOS GENERALES	345.99
ARTS. Y MATERIALES PARA DOCENCIA	234.99
ARTS. Y MATERIALES PARA INVESTIGACION	3445.99
ARTS Y MATERIALES PARA ADMINSTRACION	2343.99
ARTS. Y MATERIALES PARA DIFUSION CULT.	3454.00
ART. Y MATERIALES PARA MANTENIMIENTO	4556.99
GASTOS FINANCIEROS	344.99
FONDO COMUN 30%	345.99
OTROS	2344.00
TOTAL:	**34,455.00**

2.1.2 Objetivos

Entrevista con el cliente para determinar las necesidades, basándose en la información proporcionada por este.

Investigación sobre los diferentes lenguajes de programación para elegir el más adecuado para satisfacer las necesidades de información de la dirección de contabilidad.

Investigación sobre el funcionamiento del lenguaje seleccionado.

2.2 ANALISIS DE RIESGO.

2.2.1 Análisis de alternativas e identificación / resolución de riesgos.

- De acuerdo con las necesidades manifestadas por el cliente es necesario construir un sistema que permita: guardar información, actualizar información, eliminar información de tal forma que pueda satisfacer todas las necesidades del cliente.
- El programa será utilizado por un solo usuario, en cada una de las diferentes dependencias si así se requiere.
- El sistema contará con contraseñas en los diferentes módulos para evitar que personas ajenas puedan afectar los registros almacenados.
- Es necesario contar en un espacio mínimo de 5 megas de espacio en disco duro para el buen funcionamiento del sistema.

CAPITULO III

INGENIERIA

3.1 INGENIERIA.

Se habla de sistemas políticos, educativos, bancarios, etc. cuyo significado según el diccionario Webster es:

Uno conjunto de ordenación de cosas relacionadas de tal manera que forman una unidad o un todo orgánico.

Una definición de computadora basado en la definición anterior es:

Un conjunto u organización de elementos organizados para llevar a cabo un método, procedimiento o control mediante el procesamiento de información.

En la figura 3.1 se muestra los elementos de un sistema de computadora.

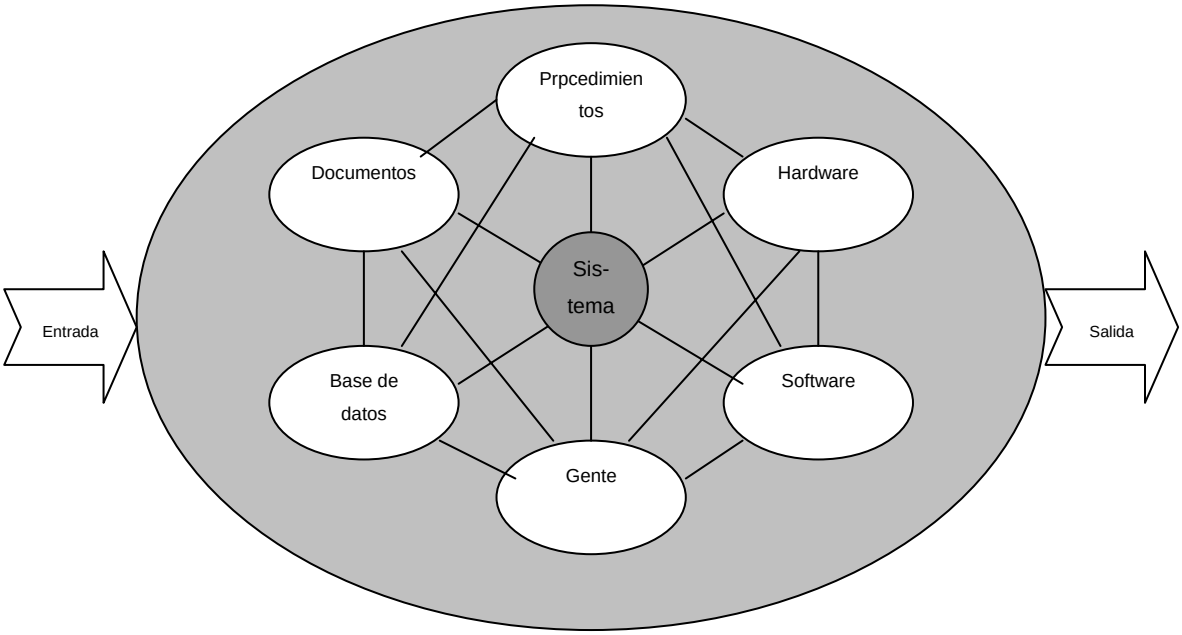


Figura 3.1 Elementos de un sistema de computadora

La ingeniería de sistemas de cómputo es una actividad de resolución de problemas. Las funciones que se desea que el sistema realice son descubiertas, analizadas y asignadas a elementos individuales del sistema, partiendo de los objetivos y de las restricciones definidas por el usuario y se desarrolla una representación de la función, del rendimiento, de las interfaces, de las restricciones de diseño y de la estructura de la información.

3.1.1 Etapas de desarrollo del software.

Primera etapa (prototipo inicial de software)

Esta primera etapa se realizo y fue estructurada en base a los procesos manuales que se efectúan en la dirección de contabilidad en cuanto a la requisición de información a las diferentes dependencias, en esta parte se planteo cambiar un poco el procesos manual para facilitar la integración de la información una vez obtenidas las bases de datos.

Segunda etapa (prototipo de siguiente nivel)

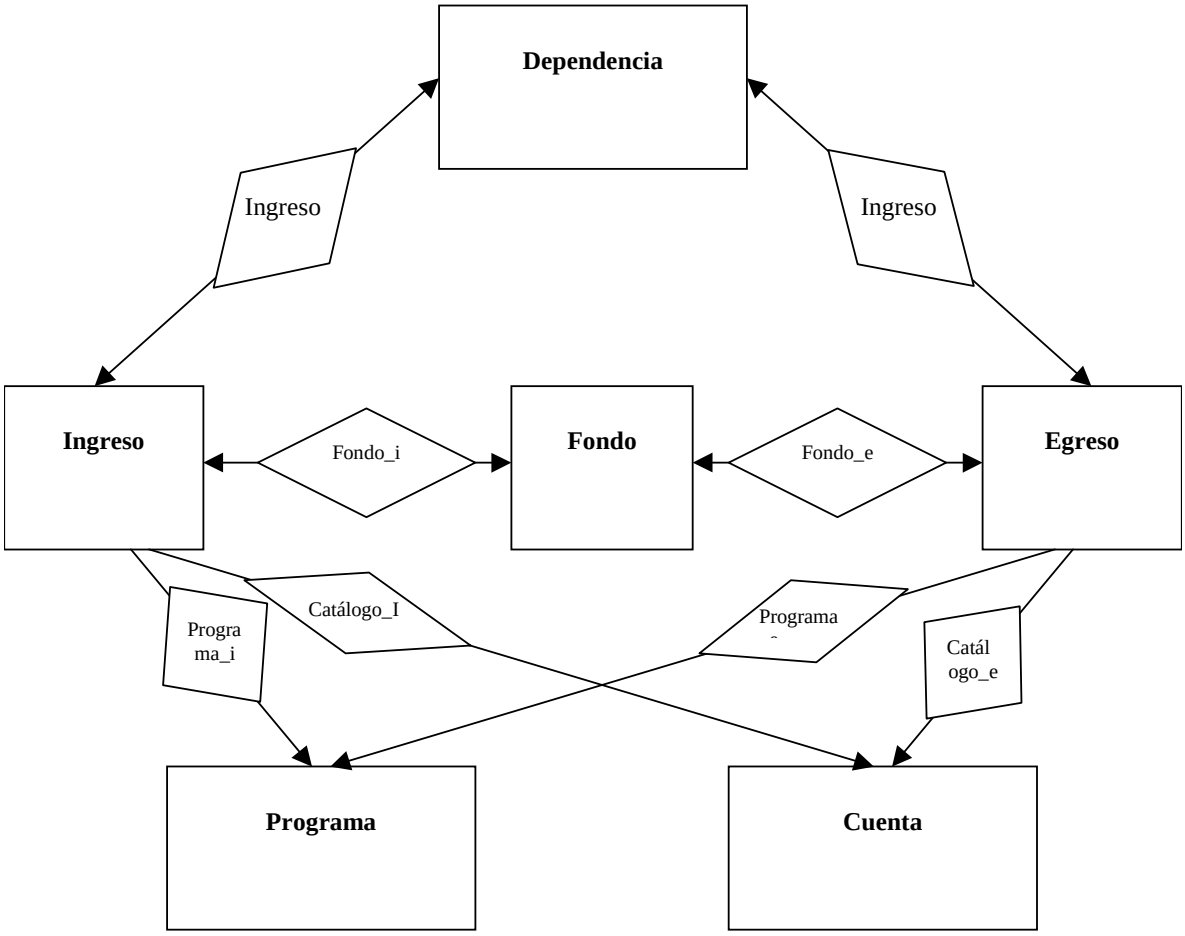
En esta etapa se trató de estar más en contacto con el cliente para sugerir y tratar de abarcar y comprender todo lo que anteriormente el cliente ya había solicitado en los requerimientos. Se hicieron modificaciones en base a lo observado y dialogado conjuntamente con el cliente.

Tercera etapa (sistema de ingeniería)

Después de varios ajustes y acuerdos con el cliente, se llegó al siguiente sistema el cual puede ser utilizado en:

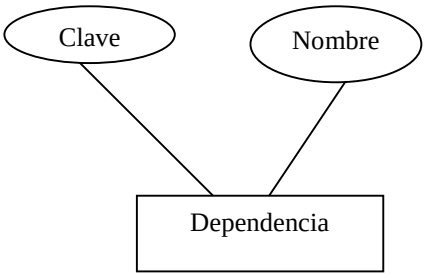
Dependencias de la BUAP.

3.2 DIAGRAMA CONCEPTUAL DEL SISTEMA



Entidad Dependencia

Diagrama

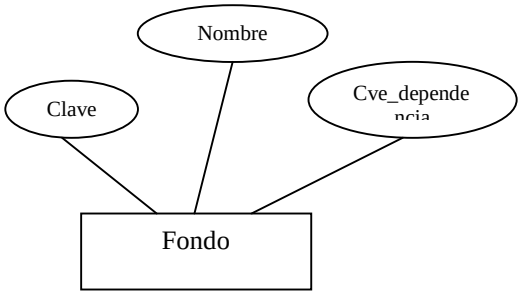


Ejemplo.

Clave	Nombre
101	RECTORIA
102	SECRETARIA PARTICULAR DE RECTORIA
103	UNIDAD DE ASESORES DE RECTORIA
122	DIRECCION DEL SIU
123	DIR. DE PROYECTOS Y SUP. DE OBRAS
	ETC...

Entidad Fondo

Diagrama

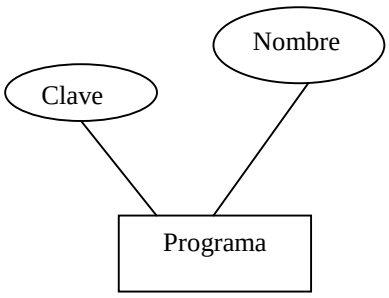


Ejemplo

CLAVE	NOMBRE	CLAV_DEPENDENCIA
10	FONDO GENERICO	701
20	FOMES	701
30	CONACITY	701
40	INGRESOS PROPIOS	701
50	OTROS	701
	ETC...	

Entidad Programa

Diagrama



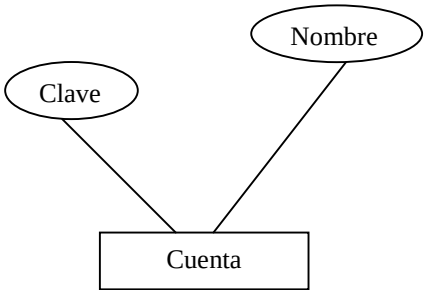
Ejemplo

CLAVE	NOMBRE
10	EDUCACION MEDIA SUPERIOR
20	PREPARATORIAS
30	EDUCACION SUPERIOR
40	TECNICO EN ENFERMERIA
50	TECNICO EN MUSICA
60	TECNICO EN INGLES
70	TECNICO EN FRANCES
80	TECNICO EN ITALIANO

90	TECNICO EN DANZA FOLKLORICA
100	TECNICO INSTRUCTOR EN DANZA
	ETC...

Entidad Cuenta

Diagrama



Ejemplo

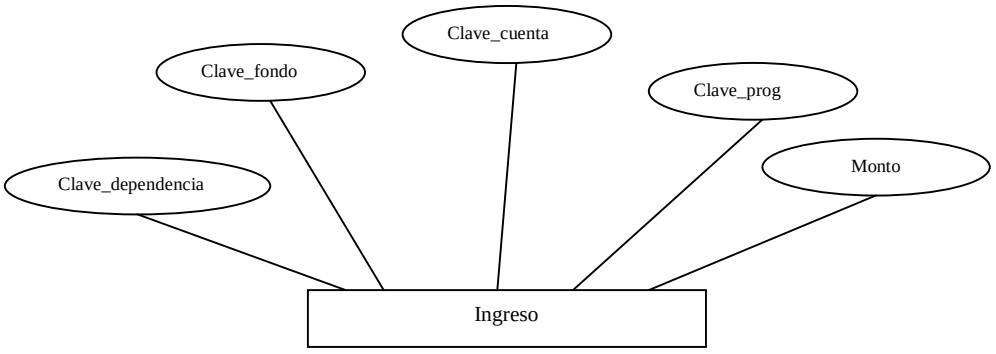
CLAVE	NOMBRE
5000	HONORARIOS
5001	HONORARIOS ASIMILABLES A SUELDOS
5002	VIATICOS GASTOS Y PASAJES P DOCENTES
5003	VIATICOS Y PASAJES P INVESTIGACION
5004	VIATICOS Y PASAJES P ADMON.
5005	PUBLICACIONES E IMPRESIONES
5006	SERVICIOS COMERCIALES
5007	MANTENIMIENTO
5008	SERVICIOS GENERALES
5009	ARTS. Y MATERIALES PARA DOCENCIA
5010	ARTS. Y MATERIALES PARA INVESTIGACION
5011	ARTS Y MATERIALES PARA ADMINSTRACION

5012	ARTS. Y MATERIALES PARA DIFUSION CULT.
5013	ART. Y MATERIALES PARA MANTENIMIENTO
	ETC

La entidad ingreso es en si una relación entre las entidades: Dependencia, Fondo, Cuenta, Programa cuya relación se muestra a continuación.

Entidad Ingreso

Diagrama



Ejemplo

CLAVE_DEPENDENCIA	CLAVE_FONDO	CLAVE_CUENTA	CLAVE_PROGRAMA	MONTO
701	10	5000	10	1000.00
701	20	5000	20	3000.21
701	10	5000	20	26566.2
701	10	5000	10	23655.3
701	10	5000	20	2545.36
701	20	5001	10	1455.36
701	20	5001	20	1525.32
701	20	5001	10	9255.32

3.2 DEPENDENCIA FUNCIONAL

Dada una relación R, el atributo Y de R depende funcionalmente del atributo X si y solo si un solo valor de Y en R está asociado a cada valor X en R. Los atributos X y Y pueden ser compuestos.

R.X ⟶ R.Y

Por lo tanto de nuestras tablas anteriores tenemos las siguientes dependencias funcionales

Dependencia.clave# ⟶ Dependencia.nombre
Cuenta.clave# ⟶ Cuenta.nombre
Fondo.clave# ⟶ Fondo.nombre
Programa.clave# ⟶ Programa.nombre

Ya que para cada valor de Dependencia.clave#, Cuenta.clave#, Fondo.clave, Programa.clave existe sólo un valor de Dependencia.nombre, Cuenta.nombre, Fondo.nombre, y Programa.nombre correspondientemente.

Nótese sin embrago que la dependencia funcional no requiere que X sea una clave candidata de R; en otras palabras, no es obligatorio que un valor de X aparezca en una sola tupla de R. Veamos otra definición de dependencia funcional.

Dada una relación R, el atributo Y de R depende funcionalmente del atributo X de R si y sólo si, siempre que dos tuplas de R concuerden en su valor de X, deben concordar también en su valor de Y.

Ejemplo:

Documento	Descripción
FF3434	CANCELACIÓN DEL CHEQUE 2000
FF3434	CANCELACIÓN DEL CHEQUE 2000

Para terminar se debe entender que la dependencia funcional es un concepto semántico. Reconocer las dependencias funcionales es parte del proceso de entender que significan los datos. El hecho de que Dependencia.clave determine funcionalmente a Dependencia.nombre significa que todas las dependencias tienen asignado una sola clave.

Normalización de la base de datos

1ra. Forma Normal

La teoría de la normalización tiene como fundamento el concepto de formas normales. Se dice que una relación está normalizada si satisface un cierto conjunto de restricciones. Veamos la primera forma normal.

1. Una relación está en primera forma normal (1NF) si y sólo si todos los dominios simples subyacentes contienen sólo valores atómicos.

Esta definición declara que cualquier relación normalizada está en 1NF, y por lo tanto todas las relaciones anteriores están en primera forma normal ya que todos sus valores simples subyacentes contienen solo valores atómicos.

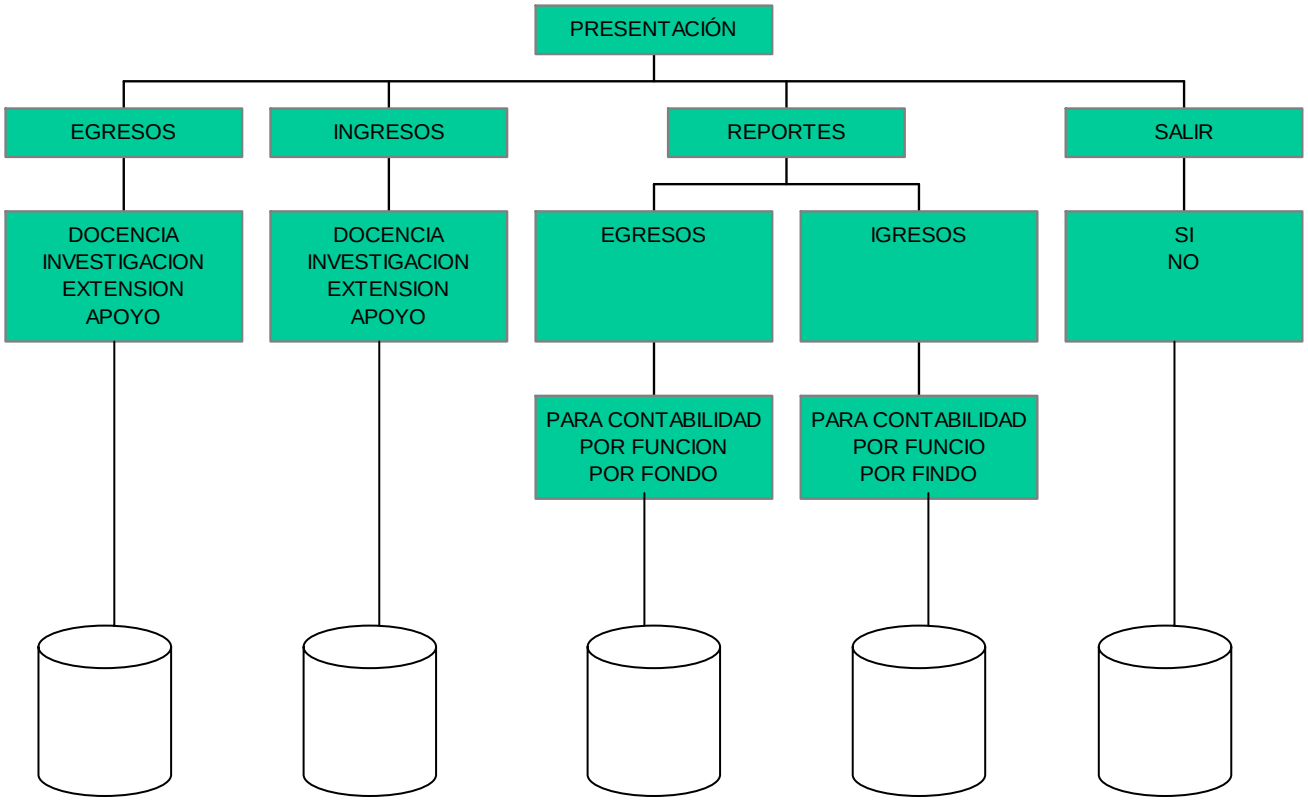
2. Una relación está en segunda forma normal (2NF) si y sólo si esta en 1NF y todos los atributos no clave dependen por completo de la clave primaria.

3. Una relación esta en 3NF si y sólo si los atributos no clave son:

- ✓ Mutuamente independientes
- ✓ Dependientes por competo de la clave primaria

Un atributo no clave es cualquier atributo que no participa en la clave primaria de la relación en cuestión.
Dos o mas atributos son mutuamente independientes si ninguno de ellos depende funcionalmente de cualquier combinación de los otros, esto implica que cualquiera de esos atributos puede actualizarse sin tomar en cuenta a los demás.

3.1.2 Estructura general del sistema



3.1.2.1 Descripción de módulos

El sistema en general está formado por varios módulos y submódulos los cuales describiremos a continuación, con el siguiente orden:

- Descripción del módulo o submódulo.
- Estructura de la base de datos usado en este módulo o submódulo.
- Algoritmo

Modulo: Presentación.

- Este módulo es el responsable de presentar el proyecto de una forma amena al cliente, así como también se encarga de inicializar variables globales para su uso en los diferentes módulos.
- No usa base de datos.

Módulo: Egresos.

- Este módulo se encarga de hacer posible la captura de todos los egresos efectuados en alguna dependencia.
- Estructura de la tabla

Nombre del Campo	**Tipo y longitud del campo**
Clave	Automático
Nombre_Egreso	Texto(50)
Fondo	Texto(2)
Dependencia	Texto(3)
Función	Texto(1)
Fecha	Fecha
Monto	Numérico (15,2)

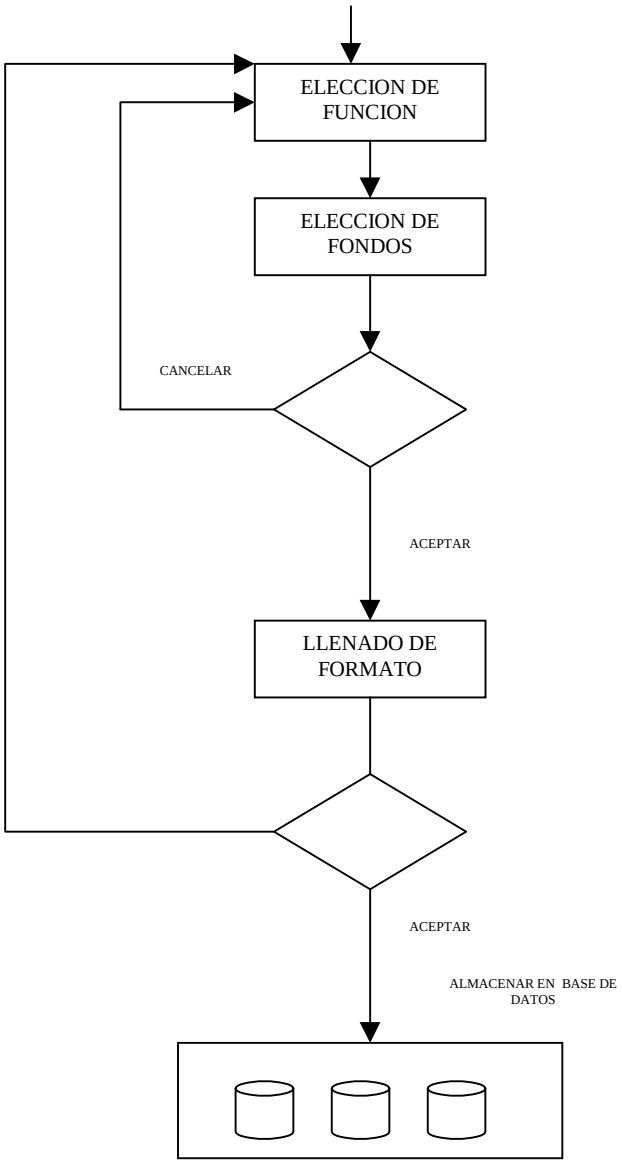
En esta tabla se almacena todo los datos necesarios de un egreso realizado para su procesamiento posterior.

Clave: es la clave única para identificar cada uno de los egresos realizados.
Nombre_Egreso: Descripción breve de la causa del egreso correspondiente.
Fondo: Fondo del cual se tomó el monto gastado.
Dependencia: Dependencia al que se le hace responsable de el monto gastado.
Función: Función a la cual se aplicó el monto gastado.

Fecha: fecha en que se realizó la transacción.

Monto: Cantidad solicitada.

Algoritmo del módulo Egresos



Módulo: Ingresos.

En esta tabla se almacena todos los datos correspondientes a un ingreso realizado.

Estructura de la tabla:

Tabla: ingresos

Indice: ingresos

Clave principal: clave

Nombre del Campo	**Tipo y longitud del campo**
Clave	Automático
Nombre_Ingreso	Texto(50)
Fondo	Texto(2)
Dependencia	Texto(3)
Función	Texto(1)
Fecha	Fecha
Monto	Numérico (15,2)

Clave: clave única para identificar cada uno de los ingresos realizados.

Nombre_Ingreso: Descripción breve del motivo por la cual se lleva a cabo un ingreso.

Fondo: Fondo en donde se almacena el monto ingresado.

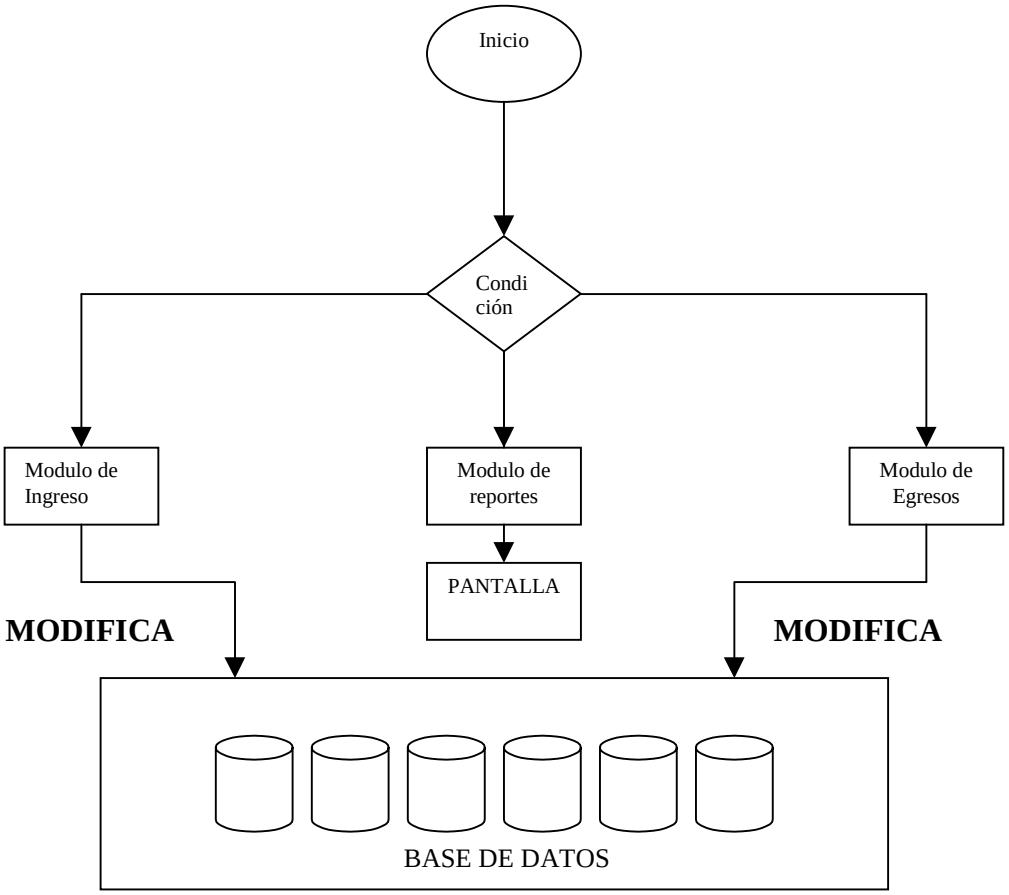
Dependencia: Dependencia que recibe el monto ingresado.

Función: Función por medio del cual se recibe el monto ingresado.

Fecha: Fecha en el que se registra este monto ingresado.

Monto: Cantidad que ingresa.

Algoritmo del módulo Egresos



CONCLUSIONES

Y PERSPECTIVAS

5.1 CONCLUSIONES

La gran importancia que tiene el realizar un buen diseño del sistema a desarrollar, se muestra en la fase de programación del sistema diseñado, ahorro de tiempo, disminución de costos y aumento en la probabilidad de hacer menos cambios a la hora de entregar el producto al cliente. También reduce el tiempo de programación y se tiene una fácil comprensión de los diagramas de flujo, así como del código resultante.

5.2 PERSPECTIVAS

Con la implementación del software en diferentes Unidades Académicas, será necesario la realización de un programa integrador de las bases de datos, que junte las diferentes bases de datos en una sola base, y que sea capaz de realizar los mismos reportes que realiza el sistema, posteriormente se puede utilizar para obtener los estados financieros de todas las dependencias y se puede utilizar para proporcionar varios formatos que solicita la SEP. Esta debe analizarse ya que la información solicitada varía año con año y puede que sea necesario un reporte del cual se pueda obtener cualquier tipo de información para el que se debe tener en cuenta toda la información almacenada en la base de datos y tal vez sea necesario solo la creación de un nuevo reporte, o creación de archivos para luego solo importarlos al sistema que se requiera.

5.3 PROMEDIO DE VIDA DEL SIE

Con respecto al tiempo de vida del software, señalaremos lo siguiente:

Software

El sistema fue desarrollado en Visual Basic 5.0 por lo que podrá ser ejecutado en máquinas que tengan Windows 95 o posterior.
Sus principales instrucciones está en SQL embebido en el lenguaje de BASIC, por lo que serán necesarios cambios mínimos para ejecutado en un SQL Server.

Wardware

Con respecto al wardware, puede ser ejecutado desde una máquina con las siguientes características.

Microprocesador 486 DX2 como mínimo
Windows 95 o posterior
10 megas de espacio en disco duro.
32 megas de RAM

El espacio requerido para que el sistema funcione es la cantidad mencionada anteriormente, pero como es un sistema de almacenamiento, el espacio estará determinado por la cantidad de información que el usuario registre en el sistema.

BIBLIOGRAFIA

Introducción a los Sistemas bases de datos

C.J. Date

Adison Wesley Longman

Anexo B

B.1 MANUAL DE USUARIO

Los resultados de la ingeniería de software fueron presentados ante el director de contabilidad que es el responsable de toda la información financiera, y fue probado en una máquina pentium mmx a 66 mhz. En donde se probo minuciosamente el funcionamiento de dicho sistema en los diferentes submódulos.

El sistema (SIE) se ejecutará de la siguiente manera:

1. Insertar disco 1/6 en la unidad a.
2. Ir a botón inicio de Windows.
3. Seleccionar opción ejecutar.
4. Escribir a:\instalar.

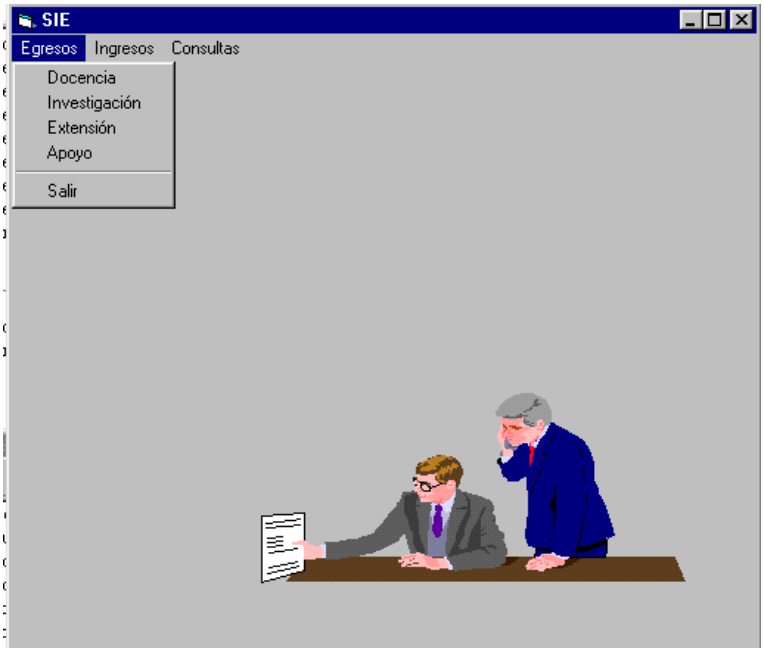
El programa de instalación le dará la ruta donde quedara instalado, si quiere modificarlo seleccione el botón cambiar y elija la ruta donde desee la instalación. El sistema le solicitará los siguientes discos.
Después de instalarlo, se crea en programas un icono en programas llamado SIE, que será utilizado para ejecutar el sistema.

Ejecute el sistema utilizando este acceso directo, la primera pantalla que aparecerá de inicio es la siguiente:

Pantalla 1 del módulo de egresos

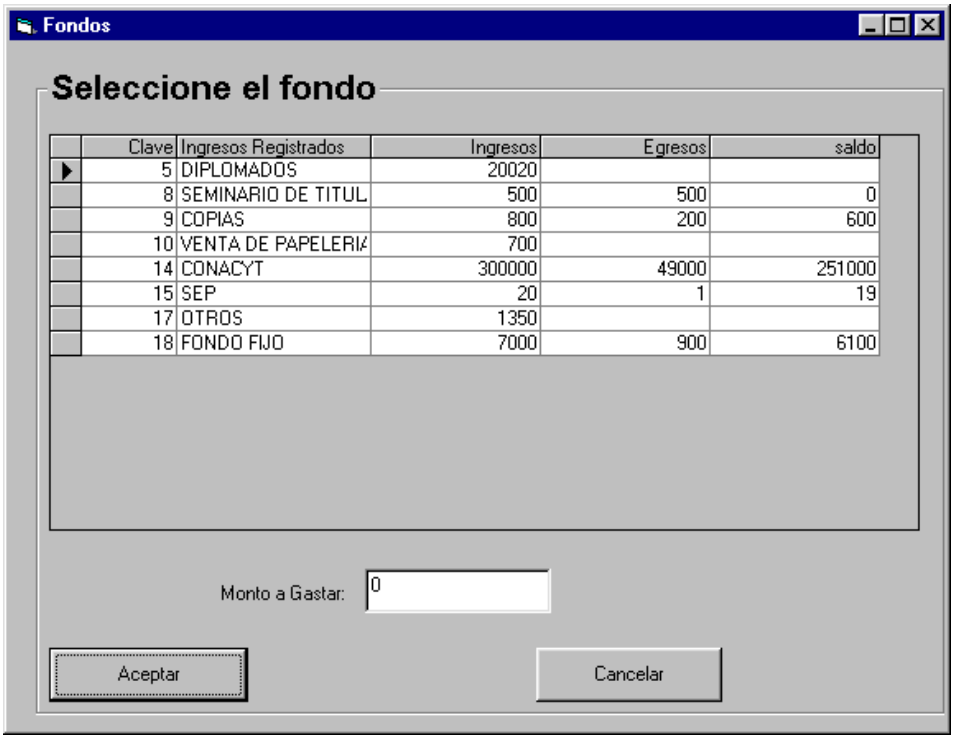
En esta pantalla se le presenta al usuario todas las opciones y el usuario debe seleccionar el adecuado dependiendo de la actividad de desee realizar.

Supongamos que deseamos registrar un egreso en la función Docencia



Pantalla 2 del módulo de egresos

Al seleccionar esta opción aparece la siguiente pantalla:



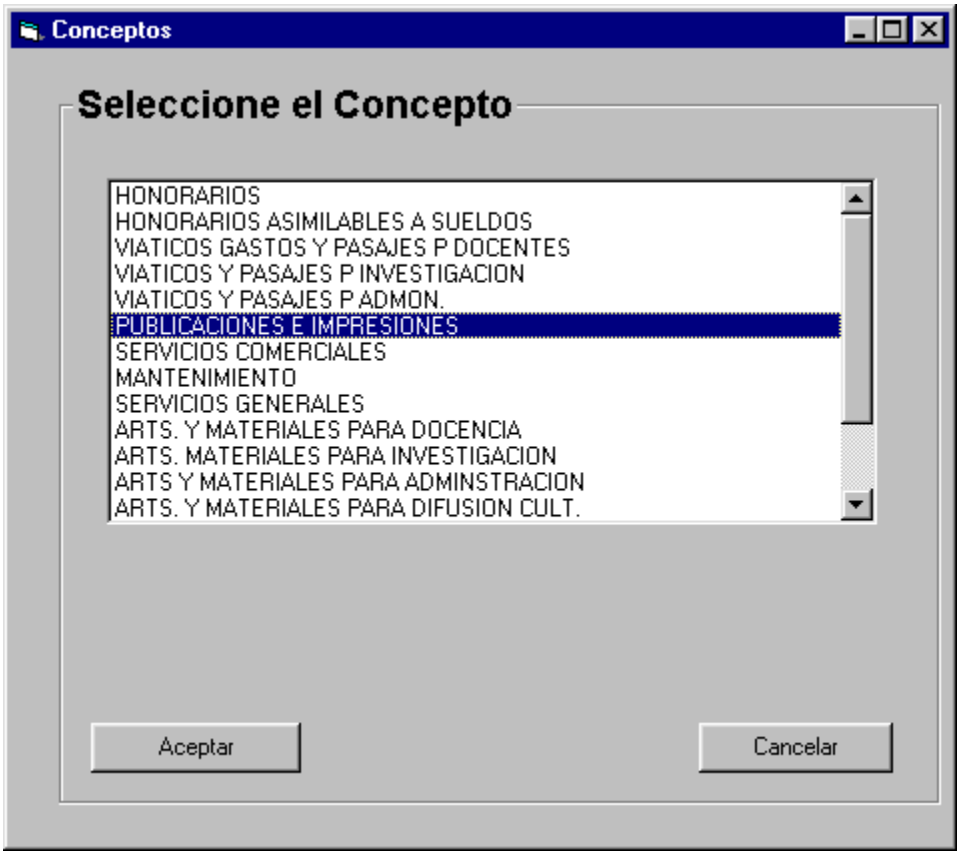
Pantalla 3 del módulo de egresos

Aquí se le presentan al usuario todos los fondos y la cantidad disponible correspondiente, de donde se puede realizar el egreso, así como se le pide también el monto de la operación. Es importante señalar que en caso de solicitar un monto mayor al que dispone el fondo el programa no permitirá que se lleve a cabo dicha operación.

Para señalar un fondo simplemente seleccione con el mouse el renglón correspondiente y asegúrese que el señalador de renglón este indicando el fondo deseado. Posteriormente presione aceptar o si desea regresar a la pantalla anterior presione cancelar.

Los fondos que aparecen en la pantalla son tablas que se deben llenar con todos los fondos con que cuenta la dependencia correspondiente, para posteriormente realizar los montos correspondientes por medio de módulo ingresos.

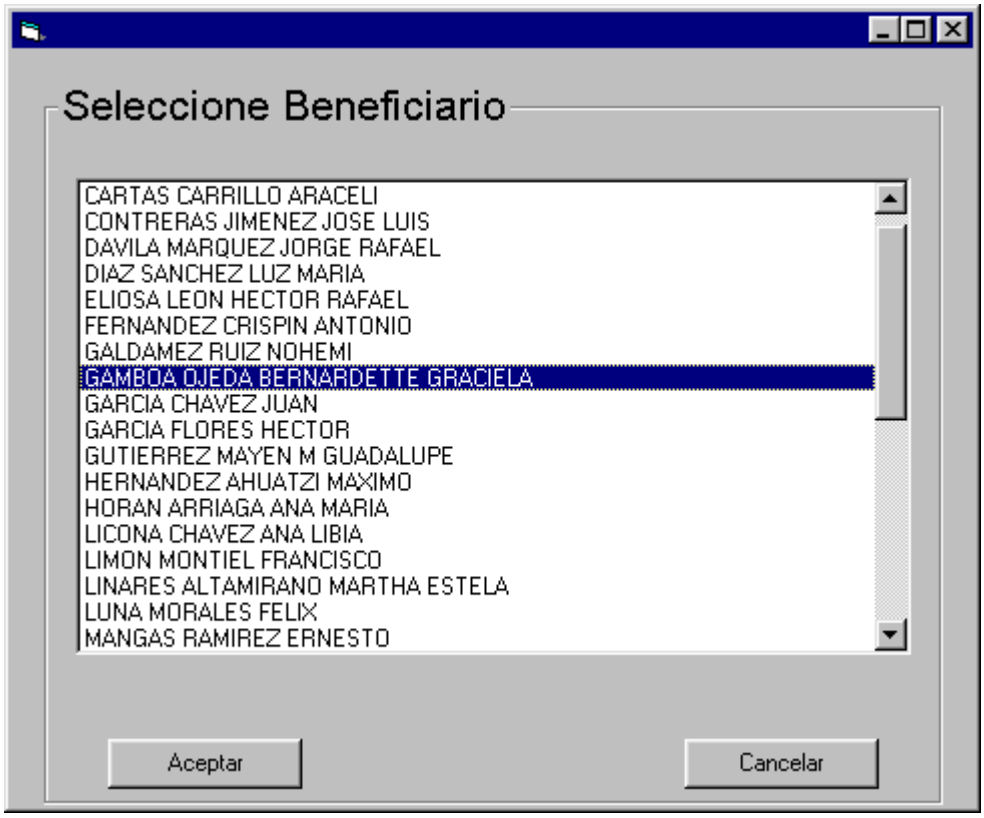
La siguiente pantalla después de aceptar el formato anterior es la siguiente:



Pantalla 4 del módulo de egresos

En esta pantalla se le pide al usuario seleccionar el concepto del egreso en la lista presentada. El concepto es una pequeña descripción de la causa por la que se realizará el egresos correspondiente. Solamente haga click en el concepto correspondiente y presione aceptar.

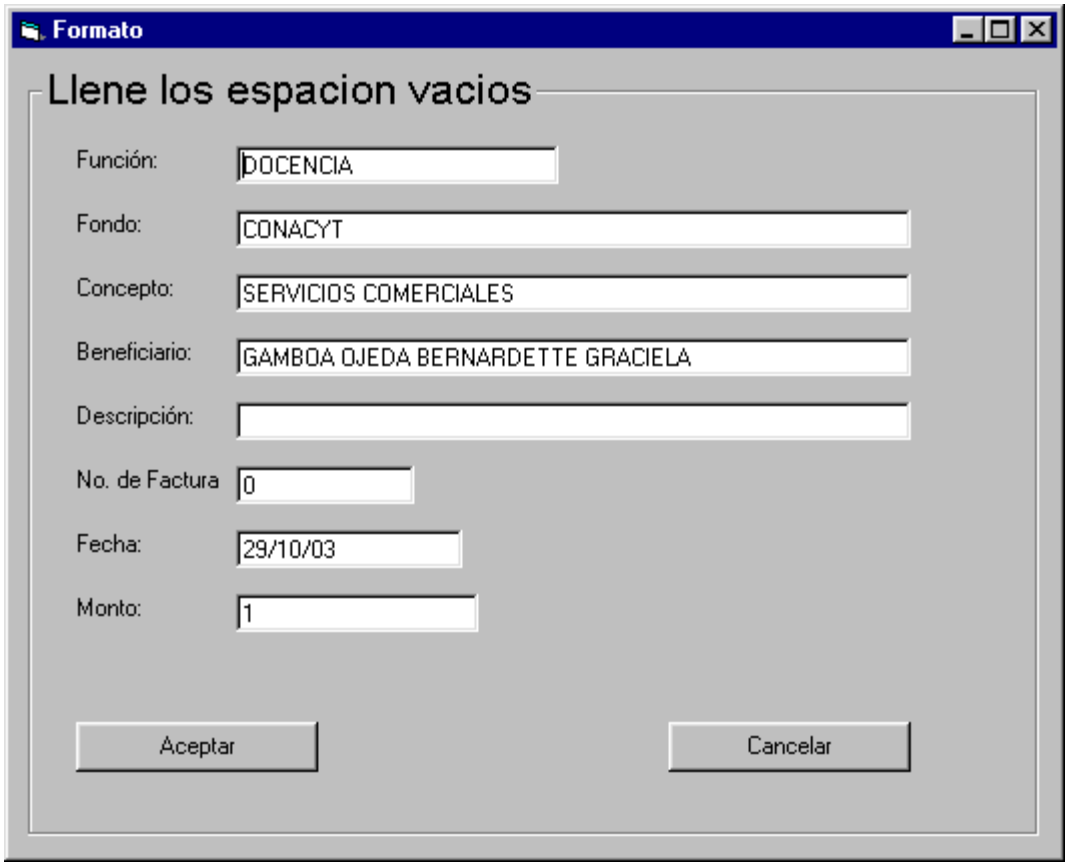
La siguiente pantalla se muestra a continuación:



Pantalla 5 del módulo de egresos

En esta pantalla se le pide al usuario seleccionar al beneficiario. En esta parte el beneficiario es a quién se le cargará como responsable del gasto, puede ser desde una persona hasta un edificio, pasando por una mesa una silla o una computadora.

Acepte el formato para ver la pantalla que se muestra a continuación.



Pantalla 6 del módulo de egresos

Esta es la pantalla final donde se presenta un formato con todas las opciones que se eligieron anteriormente, se piden tres datos opcionales más para completar el formato: Descripción, No. de factura y Fecha. A continuación se explica cada uno de ellos.

Descripción: Información adicional de esta operación.

No. de factura: número de factura con la cual se comprueba el gasto.
Fecha: Fecha en la cual se realiza la operación.

So los datos está correctos presione aceptar en caso contrario cancele la operación y vuelva a empezar.