11
25j

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

APLICACION DE LOS PRINCIPIOS DE BASES DE DATOS PARA LA RECUPERACION DE CREDITOS EN EL INFONAVIT

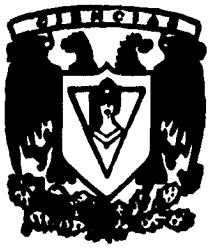
T E S I S

QUE PARA OBTENER EL TITULO DE:

M A T E M A T I C O

P R E S E N T A :

CARMEN GUTIERREZ RODRIGUEZ



DIRECTOR M. en C. [illegible] HUGO GUROVICH

MEXICO, D. F. 1996

VNIVERSIDAD NACIONAL
AVFNºMA DE
MEXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
P r e s e n t e

Comunicamos a usted que hemos revisado el trabajo de Tesis:

"APLICACION DE LOS PRINCIPIOS DE BASES DE DATOS PARA LA RECUPERACION DE CREDITOS EN EL INFONAVIT"

realizado por CARMEN GUTIERREZ RODRIGUEZ

con número de cuenta 7155530-6 , pasante de la carrera de MATEMATICAS

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis
Propietario M. EN C. ELISA VISO GUROVICH

Propietario DRA. CARMEN LOPEZ LAISECA

Propietario M. EN C. VIRGINIA ABRIN BATULE

Suplente M. EN D. ALEJANDRO MINA VALDEZ

Suplente DR. FERNANDO BRAMBILA PAZ

Consejo Departamental de Matemáticas

M. EN C. ALEJANDRO BRAVO MOJICA

A mis padres Adrián Sergio y Ma. del Carmen, agradeciéndoles su amor y el ejemplo de su trabajo y honradez que son las bases de mi formación. Que este trabajo, de alguna manera, retribuya todos sus esfuerzos.

A mis hermanos Sergio y Carlos con cariño, en especial a Sergio por su apoyo en los momentos más difíciles.

A mi esposo Jorge por todo su amor y comprensión.

A mis hijas Georgina y Elizabeth por todo lo que significan para mí.

Mi más profundo y sincero agradecimiento a la
M. en C. Elisa Viso Gurovich
por su valiosa dirección en la elaboración de este trabajo.

A las siguientes personas :
M. en C. Virginia Abrín Batule
Dra. Ma. del Carmen López Lozano
M. en D. Alejandro Mina Valdés
Dr. Fernando Brambila Paz
por la orientación y apoyo en la realización del mismo.

Al Dr. Gabriel Rivero González, quién hizo posible la realización del seminario de tesis.

Mi más sincero agradecimiento a mis amigos y familiares por la motivación y apoyo que me brindaron.

ÍNDICE

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

CAPÍTULO 1

ANTECEDENTES

El Instituto del Fondo Nacional de la Vivienda para los Trabajadores (INFONAVIT) fue creado el 1o. de mayo de 1972 con el objetivo de proporcionar créditos a los trabajadores afiliados al IMSS. Para esto se crearon 5 líneas de crédito:

I. Adquisición de viviendas financiadas por el propio Instituto.
II. Para la compra de vivienda propiedad de terceros.
III. Para construcción en terreno propiedad del trabajador.
IV. Para ampliación, mejoramiento o reparación de su vivienda.
V. Para el pago de pasivos.

En el Instituto se distinguen tres períodos básicos para el otorgamiento y recuperación de los créditos (ver tabla 1.1) :

Primer período.- de 1972 hasta julio de 1987 en que los créditos se expresaban en términos de un monto de dinero y se recuperaban a valores históricos, en un plazo de 20 años para su amortización.

Segundo período.- a partir de agosto de 1987, los créditos se expresan en veces el salario mínimo (VSM), con una tasa de interés del 4%, con un plazo de amortización de 20 años;

Tercer período.- a partir de febrero de 1992, los créditos se mantienen expresados en VSM, pero con una tasa de interés del 4% al 8% (6% en promedio) sobre el saldo insoluto de los créditos, con un plazo de 30 años para su amortización.

Durante el primer período los créditos se recuperaban, en términos reales, en un promedio del 10%, no obstante la inflación el valor de los mismos se mantenía en valores monetarios rígidos. Los pagos de los créditos se hacían con el descuento de una cantidad fija al salario de los trabajadores acreditados, o creciente al 3% para los trabajadores con ingreso entre 1 y 1.25 VSM, para que alcanzaran montos de crédito suficientes.

Dado el aumento en los precios de las viviendas, fue necesario crear un nuevo mecanismo para actualizar el monto de los créditos, el cual consistió en expresar el monto del crédito en un número de veces el salario mínimo del Distrito Federal,

de la fecha en que sea otorgado. Además, el descuento que se hace al acreditado, para la amortización de su crédito, se hará con un porcentaje fijo a su salario, que sustituye al descuento de cantidades fijas en monetario como se hacía anteriormente.

La situación anterior se corrigió en buena medida para los créditos que fueron otorgados en el segundo período, al quedar sus saldos indizados a la variación de los salarios mínimos. Con esa medida se perseguía recuperar el 100% de los créditos en términos reales. Sin embargo, para los trabajadores con ingresos entre 1 y 1.9 VSM, la recuperación era parcial, dado que se les otorgaba un monto de crédito por encima de su capacidad de pago.

PRINCIPALES CONDICIONES FINANCIERAS EN EL OTORGAMIENTO DE LOS CREDITOS				
PERIODO	TASA DE DE INTERES	PAGOS	MONTO DEL CREDITO	PLAZO DE AMORTIZACION
Hasta julio de 1987.	4% sobre saldos insolutos expresados en monetario.	Hasta septiembre de 1974: Cuotas fijas y cuotas crecientes al 3% anual. A partir de octubre de 1974 : Porcentaje fijo del salario (14%, 16% y 18%).	En monetario sin indización	Hasta 20 años.
A partir de agosto de 1987	4% incluida en la indización del saldo insoluto.	Porcentaje fijo del salario 20%	En VSM (Indizado con el salario mensual del D. F.)	Hasta 20 años.
A partir de febrero de 1992	6 % sobre saldos insolutos indizados (1)	Porcentaje fijo del salario (20% y 25%)	En VSM (Indizado con el salario mensual del D. F.)	Hasta 30 años.

(1) A partir de las modificaciones en las Reglas de Crédito, de octubre de 1993, la tasa varía entre el 4% y el 8% anual (6% en promedio).

Tabla 1.1.

En este segundo período para los créditos Línea I, de promociones de vivienda se les otorgó a los trabajadores de salario mínimo, un monto de crédito equivalente al precio de venta al costo de producción de la vivienda, siempre y cuando ésta fuera vivienda mínima[1] en cuanto a superficie y acabados. Dado que casi la totalidad de las viviendas de Línea I caían en esta definición, esto provocó un serio desajuste entre el monto de los créditos que se otorgaban a los trabajadores, particularmente a los de salario mínimo o cercano a éste y la cantidad que en realidad podrían amortizar durante el plazo del crédito.

Para el tercer período, se estableció como condición básica la recuperación total de los créditos en términos reales, estableciéndose lo siguiente:

- Se ampliaría el plazo de amortización del crédito a 30 años.
- Dependiendo de la relación edad-salario se otorgaría un monto máximo de crédito acorde con la capacidad de pago del trabajador.
- El financiamiento a la construcción de viviendas se realizaría por convenio y a precio alzado [2] a fin de garantizar un precio de venta máximo desde el inicio acorde con la capacidad de pago del trabajador.

Para 1994, el Instituto había otorgado 1'118,895 créditos, de los cuales 594,558 han sido amortizados, quedando vigentes 524,337.

El Instituto recupera los créditos mediante los pagos de las amortizaciones efectuadas a través de la empresa, donde trabaja el acreditado o directamente por el trabajador, haciendo el pago en el Instituto o en el banco.

---

[1] *Vivienda mínima*, es la construcción en una superficie de 55 metros cuadrados.

[2] *Precio alzado*, es el precio de la vivienda incluyendo el terreno, el material y los acabados.

RECUPERACIÓN DE LOS CRÉDITOS

Existen dos formas para el pago de las amortizaciones a los créditos otorgados:

- Régimen Ordinario de Amortización (ROA), en el cual el patrón efectúa el descuento al salario de los trabajadores acreditados; y
- Régimen Especial de Amortización (REA), es cuando el trabajador pierde su relación laboral, por lo que efectúa en forma directa sus pagos.

Existen casos muy especiales, en que el acreditado ya no puede amortizar el crédito, es entonces cuando el autoseguro paga al Instituto lo que resta del crédito, esto es aplicable sólo cuando el acreditado fallezca, tenga invalidez definitiva, o incapacidad total o permanente.

Para el caso de Línea I, si la vivienda otorgada por el Instituto sufre daños, entonces el autoseguro paga al Instituto el resto del monto que debe el acreditado liberándolo así de la deuda contraída; en el caso en que el Instituto sea el responsable de los daños de la vivienda, y si así lo amerita , el Instituto se encargará de reponerla su vivienda al acreditado.

Para 1994, se encuentran inscritos 406,487 trabajadores acreditados en ROA y 117,850 en REA. De los acreditados inscritos en ROA, existe un número no determinado, sin relación laboral, que no ha dado el aviso correspondiente a su empresa.

La recuperación de los créditos se inicia con el aviso que se hace a los patrones para que le descuente a los trabajadores acreditados que tiene en su nómina, el porcentaje de su salario, establecido por el Instituto para el pago de los créditos otorgados.

A esa fecha se ha encontrado que el 29% de los acreditados inscritos en el ROA tienen al menos una omisión de pago; para los inscritos en el REA, este porcentaje se eleva al 44%, ver tabla 1.2. Se han llevado a cabo diversas acciones aisladas, en diferentes épocas, sin la continuidad necesaria que garantice la eficiencia en la recuperación de los créditos.

DINAMICA DE LA RECUPERACION DE LA CARTERA HIPOTECARIA
CON RELACION AL INGRESO POR APORTACIONES
MILLONES DE NUEVOS PESOS

AÑO	A) APORTACION	B) RECUPERACION DE CREDITOS	B/A %
1985	182	37	20.3
1986	331	58	17.5
1987	738	129	17.5
1988	1621	261	16.1
1989	2050	336	16.4
1990	3121	438	14.0
1991	3871	638	16.5
1992	4811	861	17.9
1993	5304	1187	22.4
1994	5708	1676	29.0

Fuente: Informe Anual de Actividades 1985-1992.
Programa de Labores y Financiamientos 1993-1994.

Tabla 1.2.

SITUACIÓN ACTUAL

Para el requerimiento de pagos de abonos de crédito tanto a los patrones como a los trabajadores acreditados se encontraron los siguientes problemas:

a) Dificultad para localizar al patrón y/o trabajador, en los domicilios registrados en el Instituto.

b) Una vez localizados, dificultad para que el deudor cubra los abonos vencidos que se han acumulado con el paso del tiempo, al no ser detectado y requerido su pago oportunamente por el Instituto.

Se observa que al trabajador acreditado, si no se le requiere cuando pierde o cambia de trabajo, difícilmente toma la iniciativa para continuar pagando su crédito, al darse cuenta de que el Instituto no realiza ninguna acción (tabla 1.3).

SITUACION DEL TRABAJADOR		CAUSA DE OMISION
El acreditado mantiene la misma relación laboral original del momento del otorgamiento del crédito.		- No se efectúan los descuentos. - Se efectúan los descuentos pero no se enteran al Instituto.
El acreditado no conserva la relación laboral original del otorgamiento del crédito.	- No tiene una nueva relación laboral.	- No acude a solicitar prórroga. - No solicita ingreso al REA.
	- Obtiene una nueva relación laboral.	- No dá aviso al Instituto.
El acreditado a través del patrón, en caso del ROA, o en forma directa en el REA, efectúa los pagos de amortización.		- No se efectúan los pagos en las cantidades correctas.

Tabla 1.3.

Se requiere establecer nuevos programas de cobranza para recuperar la cartera vencida y acciones específicas para mejorar la situación de la cartera en lo general, que permitan un mayor control de depuración de la misma y que contemplen facilidades a los acreditados que deseen regularizarse.

OBJETIVO DEL SISTEMA DE FACILIDADES

Para poder contar con un mejor sistema de recuperación es necesario primero identificar las omisiones que existen de cada trabajador, determinar su cuantía y requerir el pago que corresponda, o bien, requerir aquellas cantidades que no se estén pagando en la cantidad correcta.

En este sentido se plantean las siguientes medidas:

a) Establecer un mecanismo automatizado que mensualmente genere los requerimientos para el cobro de los adeudos vencidos, una vez que en forma masiva se depure la morosidad existente en la cartera. Para este propósito se

emitirían los requerimientos de pago dirigidos simultáneamente al último patrón registrado y al trabajador acreditado .

b) Localizar a los trabajadores acreditados con omisiones en las nuevas empresas en que estén prestando sus servicios, con base en la información del SAR[3] proporcionada por la Banca y notificar a estas empresas la obligación de descontarles las cantidades que correspondan para la amortización de los créditos concedidos, sin menoscabo de requerir al trabajador, en su caso, por conducto de la misma empresa, para que cubra los adeudos vencidos de su crédito.

c) Establecer una política de convenios de pago (automatizado), que facilite a los trabajadores que han omitido pagos, la regularización de sus adeudos, tomando en cuenta su cuantía y el nivel de ingresos que perciben actualmente de acuerdo a la tabla 1.4 .

d) Determinar las medidas a seguir con los trabajadores sujetos a novación[4], esto es, a los que se les otorgó un crédito en el periodo de 1984 a 1987 y cuentan con más de doce omisiones. Los saldos de estos créditos son inferiores en la mayoría de los casos a N$1,500, por lo que se les podría aceptar la liquidación total de su adeudo, evitando así la discresionalidad existente para dispensar el tratamiento de la novación, atendiendo a situaciones de orden socioeconómico, de difícil valoración objetiva.

e) Facilitar la liquidación anticipada de todos aquellos créditos con saldos insolutos inferiores a determinada cantidad (N$1,000 o N$2,000), mediante el ofrecimiento de descuentos atractivos para los deudores. Esta medida liberaría al Instituto de la administración de un buen número de créditos de reducido valor.

Los cambios en cuanto a equipo y metodología de trabajo en sistemas implica la necesidad de migrar bases de datos y procedimientos a sistemas abiertos, ya que se van a hacer cambios se deben completar los sistemas para tener acceso más fácil, evitar las redundancias en los datos, etc. Estos nuevos cambios deben incluir nuevos procedimientos que apoyen de manera más efectiva la localización de los acreditados morosos.

---

[3] *SAR*, Sistema de Ahorro para el Retiro, que comienza en el 2° bimestre de 1992 .

[4] *Novación*, mecanismo por el cual se redocumentan los créditos de Línea I en el valor actual del avalúo que se practique a la vivienda del acreditado y que haya incurrido en más de doce omisiones mensuales consecutivas en el pago de su crédito, y que éste se haya otorgado en el periodo de 1984 a 1987.

Para alcanzar estos objetivos el nuevo sistema deberá incluir, al menos módulos de:

- Control de gestión frente a morosos antiguos, seguimiento de las acciones que se han llevado a cabo en la dirección de:

  a) recuperar el monto.
  b) recuperar el contacto con el acreditado (conocer su situación).

- Para el caso de morosos antiguos, al recuperar el contacto, cálculo automático de las opciones disponibles para regularizar la situación.

Para esto se va a diseñar y utilizar un Sistema de Base de Datos que facilite, entre otras, estas tareas. Procederemos a establecer un marco de referencia para el desarrollo de sistemas, enmarcados en la Ingeniería de Software. A continuación trataremos los aspectos teóricos importantes de lo que es un Sistema de Base de Datos, sus propiedades, aplicaciones y distintas implementaciones, para terminar con el diseño y el análisis del sistema propuesto.

PROGRAMA DE FACILIDADES A LOS TRABAJADORES ACREDITADOS			
SITUACION DEL ACREDITADO	PERIODO DE REGULARIZACION	CREDITOS OTORGADOS EXPRESADOS	TIPO DE FACILIDADES
Trabajadores con mora en el pago de amortización de su crédito.	PROGRAMA VOLUNTARIO	En monetario $	- Liquidación total del adeudo. - Condonación de intereses moratorios y de gastos de cobranza.
		En VSM	- Establecimiento de convenios para el pago en parcialidades, sin cobros por concepto de financiamiento. - Condonación de intereses moratorios y de gastos de cobranza. - Recepción, durante la vigencia del convenio, de los pagos regulares de amortización.
	PROGRAMA COERCITIVO	En monetario $	- Liquidación total del adeudo con cargo de intereses moratorios. - Disminución al 10% de gastos de cobranza.
		En VSM	- Establecimiento de convenios para el pago de parcialidades, sin cobros por concepto de financiamiento. - Aplicación de intereses moratorios y disminución al 10% de gastos de cobranza. - Recepción, durante la vigencia del convenio de los pagos regulares de amortización.
Trabajadores al corriente en el pago de amortización de su crédito.		En monetario $	- Liquidación total del adeudo con descuento del 10% sobre dicho saldo. - Liberación de las escrituras de su vivienda.

Se excluyen de este Programa de Facilidades :
- Los acreditados que no hayan realizado pago alguno para la amortización de créditos expresados en monetario.
- Los acreditados de Línea I cuyas viviendas presenten irregularidades en su ocupación y se hayan iniciado las acciones legales correspondientes.

Tabla I 4.

CAPÍTULO 2
SISTEMAS BASADOS EN COMPUTADORA

Un *sistema basado en computadora* es un "conjunto u ordenación de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento de información". [1]

Los elementos genéricos (figura 2.1) de un sistema basado en computadora son los siguientes :

Software. Son los programas de la computadora, los datos y la documentación, que sirven para realizar el método lógico, el procedimiento o control requerido.

Hardware. Son los dispositivos electrónicos (p. ej.: el CPU, la memoria) que proporcionan la capacidad de cómputo y los dispositivos electromecánicos (por ej.: sensores, motores, bombas) que proporcionan las funciones del mundo exterior.

Gente. Son los usuarios y operadores del software y del hardware.

Bases de Datos. Es una colección grande y organizada de información a la que se accede mediante el software y que es una parte integral del funcionamiento del sistema.

Documentación. Los manuales y otra información descriptiva que explica el uso y/o la operación del sistema.

Procedimientos. Los pasos que definen el uso específico de cada elemento del sistema o el contexto de procedimientos en que reside el sistema.

Estos elementos se combinan de diversas formas para transformar la información. Una característica de los sistemas basados en computadoras es que los elementos que componen un sistema también pueden representar un macroelemento de un sistema aún mayor. Un macroelemento es un sistema basado en computadora que forma parte de un sistema basado en una computadora mayor. El analista de sistemas (o ingeniero de sistemas) definirá los elementos de un sistema basado en computadora dentro del contexto de la jerarquía de sistemas (macroelementos).

---

[1] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

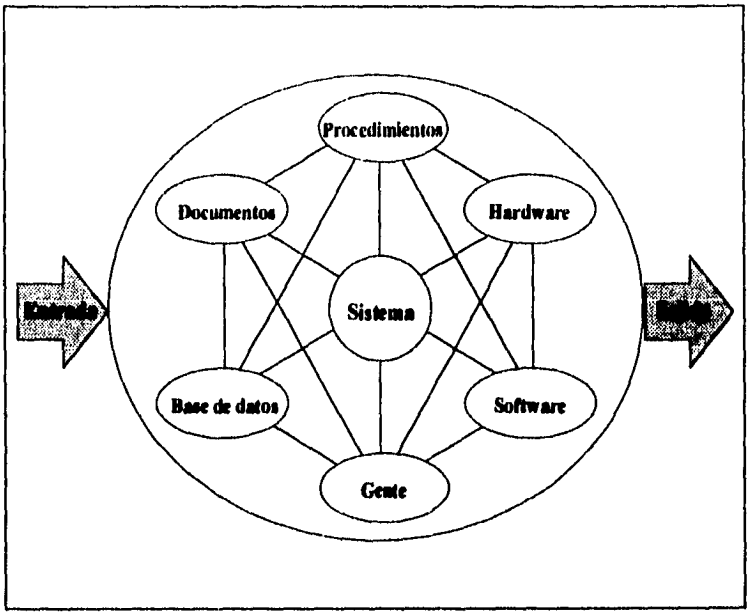


Figura 2.1. Elementos del sistema. [2]

INGENIERÍA DE SISTEMAS DE COMPUTADORA

La *ingeniería de sistemas de computadora* es una actividad de resolución de problemas. El analista se basa en los objetivos y las restricciones que el usuario le define para desarrollar una representación de la función, del rendimiento, de las interfaces, de las restricciones de diseño y de la estructura de la información, los cuales asocia a cada uno de los elementos genéricos anteriormente descritos.

Un sistema basado en computadora lo podemos representar como un modelo de entrada-salida, donde el software tiene un papel en cada aspecto del modelo. El software se usa para adquirir información que puede ser suministrada de

[2] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A. Capítulo 5.

alguna fuente externa o por otro elemento del sistema, e implementa los algoritmos de procesamiento requeridos para realizar las funciones del sistema. Un algoritmo de procesamiento transforma los datos de entrada, produciendo información o control como salida para otro elemento del sistema.

La ingeniería del software es una disciplina para el desarrollo del software de alta calidad para sistemas basados en computadora. Las figuras[3] 2.2a, b y c ilustran los pasos genéricos del proceso de ingeniería del software. Aquí se muestran los pasos que se deben de llevar a cabo y las distintas representaciones del software que se derivan según se evoluciona desde el concepto a la realización.

En todo proyecto de desarrollo de sistemas, se debe seguir alguna metodología tanto en el planteamiento como en el desarrollo del mismo. Esta metodología nos la ofrece la Ingeniería de Software.

[3] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

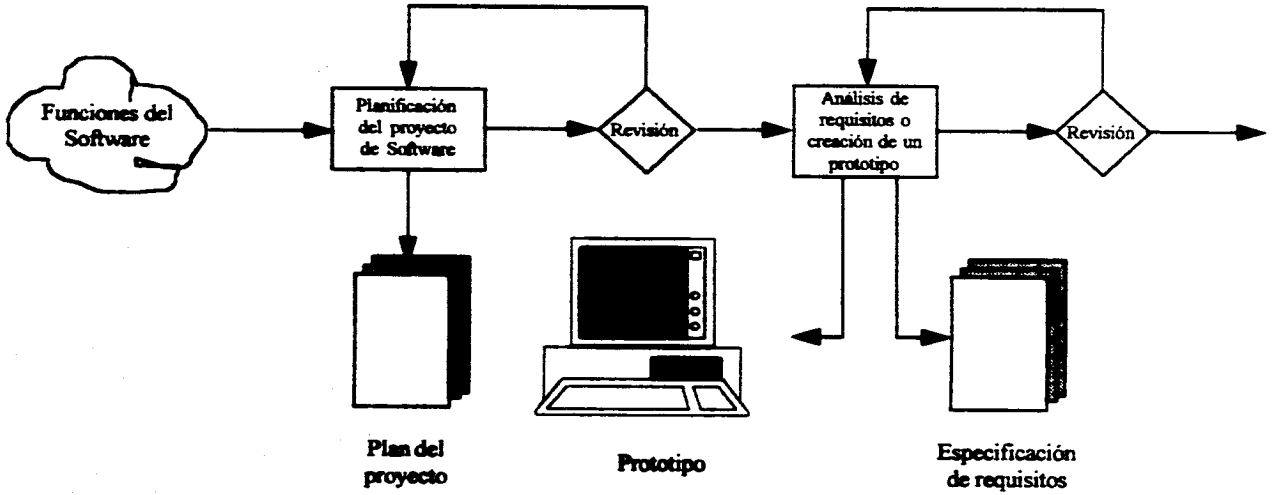


Figura 2.2 (a). Ingeniería del software. Fase de definición.

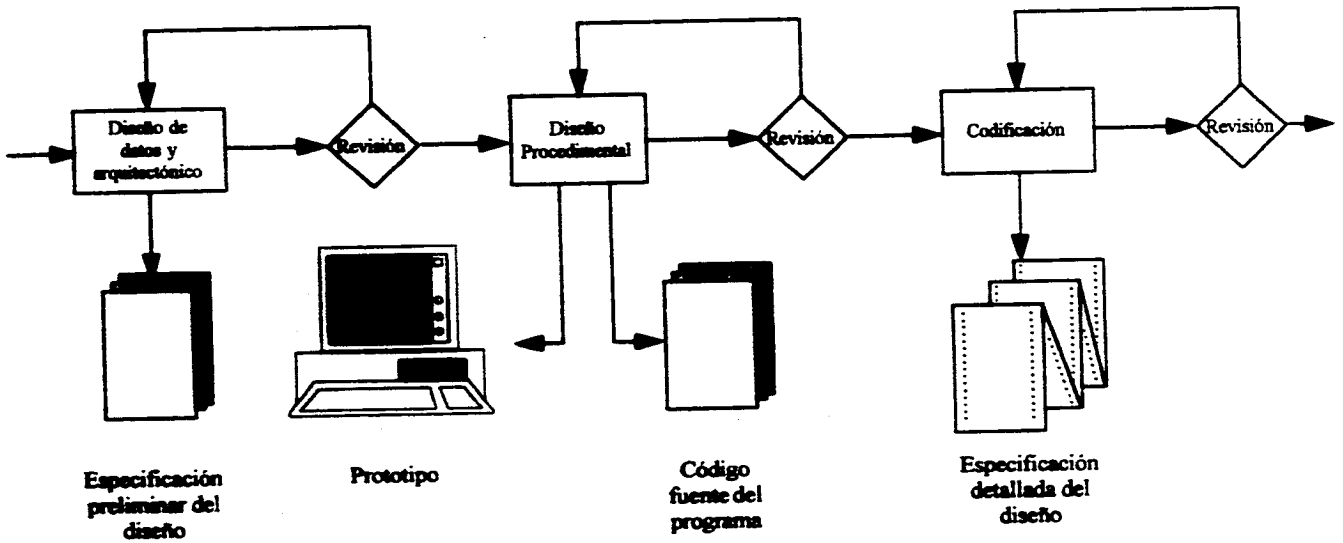


Figura 2.2(b). Ingeniería del software. Fase de desarrollo.

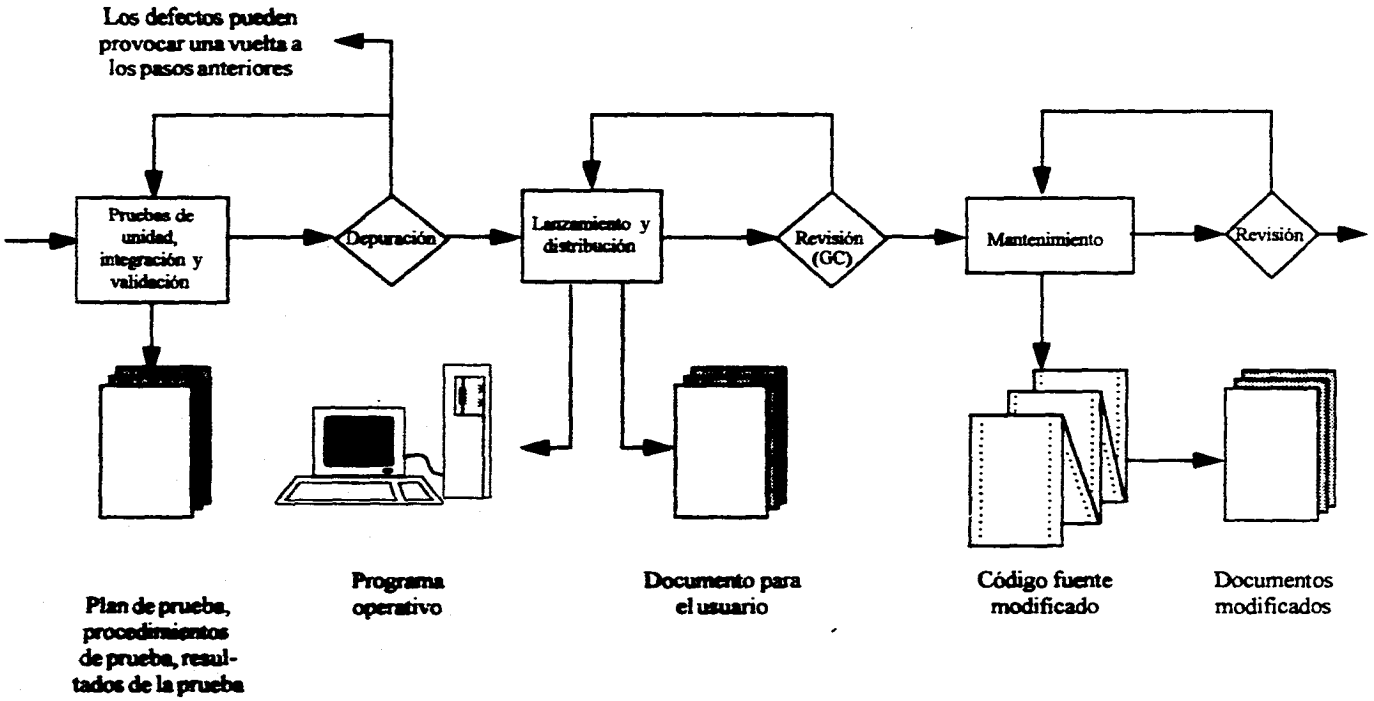


Figura 2.2 (c). **Ingeniería del software. Fase de verificación**, lanzamiento y mantenimiento.

INGENIERÍA DE SOFTWARE

A partir de la ingeniería de sistemas y de hardware surge la ingeniería de software, la cual consta de tres elementos: *métodos*, *herramientas* y *procedimientos*. Estos elementos son las bases para controlar el proceso de desarrollo del software de alta calidad de una manera productiva, por lo que podemos decir que la ingeniería del software es una disciplina para el desarrollo del software. A continuación describiremos brevemente los elementos de la ingeniería del software.

Los *métodos* nos dicen "cómo" construir técnicamente el software, y tienen una notación especial orientada a un lenguaje o una gráfica, y un conjunto de criterios para la calidad del software. Estos métodos son:

- planificación y estimación de proyectos
- análisis de los requisitos del sistema y del software
- diseño de estructuras de datos
- arquitectura de programas y procedimientos algorítmicos
- codificación
- prueba
- mantenimiento

Las *herramientas* de la ingeniería de software dan un soporte automático o semiautomático para cada uno de los métodos anteriormente mencionados. Por ejemplo CASE (ingeniería del software asistida por computadora) combina software, hardware y bases de datos para generar un entorno de ingeniería de software.

Los *procedimientos* de la ingeniería de software unen los métodos y las herramientas, facilitando un desarrollo racional y oportuno del software de computadora. Los procedimientos definen la secuencia en la que se aplican los métodos, las entregas (documentos, informes, formas, etc.) que se requieren, los controles que ayudan a asegurar la calidad y coordinar los cambios y las directrices que ayudan a evaluar el progreso del software.

La ingeniería de software está compuesta por una serie de pasos que contienen a los métodos, a las herramientas y a los procedimientos, anteriormente mencionados. A estos pasos se les llama frecuentemente *paradigmas de la ingeniería del software*. Los paradigmas son cuatro : el ciclo de vida clásico, la construcción de prototipos, el modelo en espiral y las técnicas de cuarta generación. A continuación los describiremos brevemente.

El *ciclo de vida clásico* o "modelo en cascada" exige un enfoque sistemático y secuencial del desarrollo del software y que consta de los siguientes pasos: análisis, diseño, codificación, prueba y mantenimiento.

La *construcción de prototipos* facilita al programador la creación de un modelo de software a construir; esto se puede hacer cuando el cliente, aunque defina un conjunto de objetivos generales para el software, no logra identificar los requisitos detallados de entrada, proceso o salida. En otros casos el programador no puede estar seguro de la eficiencia de algún algoritmo, de la adaptabilidad de un sistema operativo; en estas y en otras ocasiones se requiere construir un prototipo (figura 2.3).

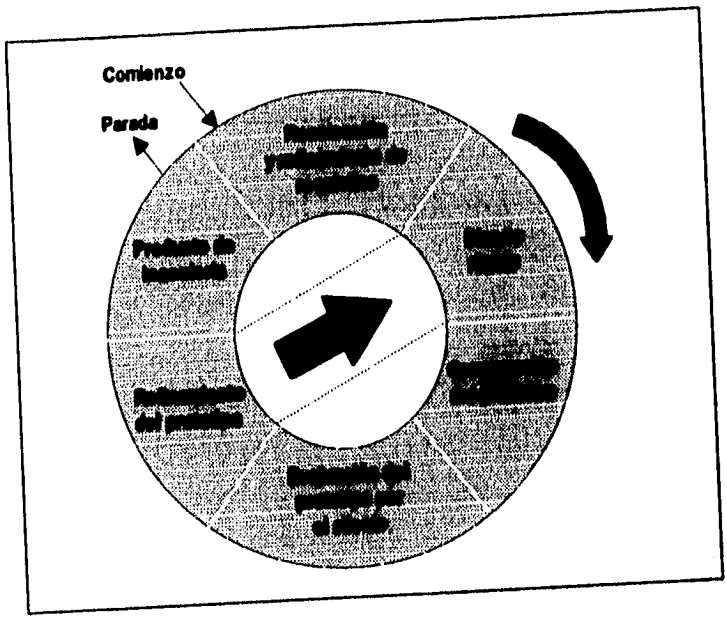


Figura 2.3. Creación de prototipos[4].

[4] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

En la figura 2.3, podemos apreciar que se comienza con la recolección de los requisitos, reuniendo y definiendo los objetivos globales para el software, se identifican todos los requisitos conocidos y se definen las áreas donde será necesaria una mayor definición. Posteriormente se produce un "diseño rápido", el cual se enfoca sobre la representación de los aspectos del software que son visibles al usuario, como por ejemplo, los métodos de entrada y los formatos de salida.

Este diseño rápido conduce a la construcción de un prototipo, el cual es evaluado por el cliente/usuario y se usa para refinar los requisitos del software a desarrollar. Aquí se produce un proceso iterativo en el que el prototipo se va afinando para que se satisfagan las necesidades del cliente y facilitando al desarrollador una mejor comprensión de lo que hay que hacer . Es importante tener en cuenta que el prototipo que se haga se va a desechar, ya que, el primer sistema construido apenas es utilizable dado que puede ser lento, demasiado grande, difícil de usar o las tres cosas.

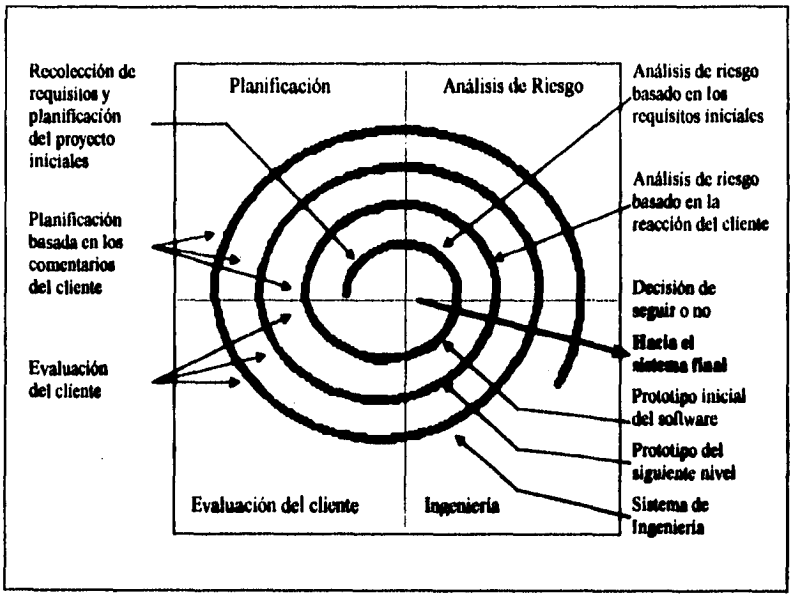


Figura 2.4. El modelo en espiral. [5]

[5] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

El *modelo en espiral* cubre las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos, pero le añade el análisis de riesgo que le falta a esos paradigmas. El modelo en espiral consta de cuatro actividades: planificación, análisis de riesgo, ingeniería y evaluación del cliente (figura 2.4).

Como se puede ver en la figura 2.4, se empieza del centro definiendo los objetivos, las alternativas y las restricciones, y se analizan e identifican los riesgos. Si el análisis de riesgo indica que hay incertidumbre en los requisitos, se puede usar la creación de prototipos en el cuadrante de ingeniería para dar asistencia al cliente así como al encargado del desarrollo. Se pueden utilizar simulaciones y otros modelos para definir más el problema y refinar los requisitos.

Cuando el cliente evalúa el trabajo de ingeniería (cuadrante de evaluación del cliente) sugiere modificaciones, y en base a estos comentarios se produce la siguiente fase de planificación y de análisis de riesgo. En cada vuelta alrededor de la espiral, la culminación del análisis de riesgo resulta en una decisión de 'seguir' o 'no seguir', si los riegos son demasiado grandes, se puede dar por terminado el proyecto. Si se decide por seguir, entonces se avanzará alrededor del camino de la espiral el cual llevará a los desarrolladores hacia afuera, hacia un modelo más completo del sistema, y finalmente, al propio sistema operacional.

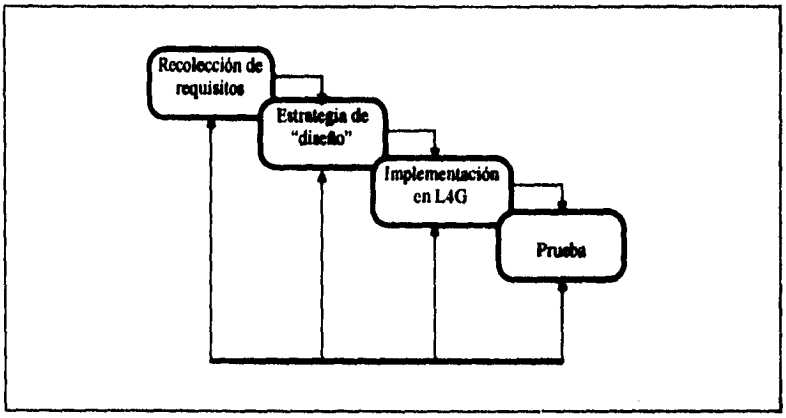


Figura 2.5. Técnicas de cuarta generación. [6]

[6] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

Las *técnicas de cuarta generación* (T4G) facilitan la especificación de algunas características del software a alto nivel, es decir, a un nivel más próximo al lenguaje natural o en una notación que proporcione funciones significativas. El paradigma consta de los siguientes pasos: *recolección de requisitos, estrategia de "diseño", implementación en lenguaje de cuarta generación (L4G) y prueba* (figura 2.5).

Como se puede ver en la figura 2.5 , T4G comienza con la recolección de requisitos, aquí es muy importante el dialogo cliente-desarrollador, ya que las herramientas actuales de T4G no son muy sofisticadas para entender el lenguaje natural, además de que el cliente puede no estar muy seguro de lo que necesita.

Se puede usar un *lenguaje de cuarta generación* no procedimental (L4G), para aplicaciones pequeñas al ir directamente desde el paso de recolección de requisitos al paso de implementación. Para grandes proyectos, el uso de T4G sin diseño causará poca calidad, mantenimiento pobre y mala aceptación por el cliente. La implementación mediante un L4G permite al desarrollador de software, centrarse en la representación de los resultados deseados, lo que se traducirá automáticamente en código fuente produciendo dichos resultados. Deberá existir una estructura de datos con información relevante, en la cual el L4G pueda acceder rápidamente.

Para transformar una implementación T4G en un producto, el desarrollador debe de dirigir una prueba completa, hacer una documentación y ejecutar el resto de las actividades de "transición" requeridas en los otros paradigmas de la ingeniería del software. Además, el software que se desarrolla con T4G deberá ser construido de forma que facilite la realización del mantenimiento de forma rápida.

La elección de uno de estos paradigmas depende de la naturaleza del proyecto y de la aplicación, así como de los métodos y herramientas a usar y los controles y entregas requeridos.

Es de notarse que estos métodos son complementarios, en muchos casos, estos paradigmas pueden combinarse de tal manera que puedan utilizarse las ventajas de cada uno de ellos en un solo proyecto, como lo podemos observar en la figura 2.6 [7], en todos los casos se comienza con la *recolección preliminar de requisitos*, es decir, determinando los objetivos, alternativas y restricciones. Después se puede elegir cualquier camino cómo se indica en la figura 2.6.

---

[7] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

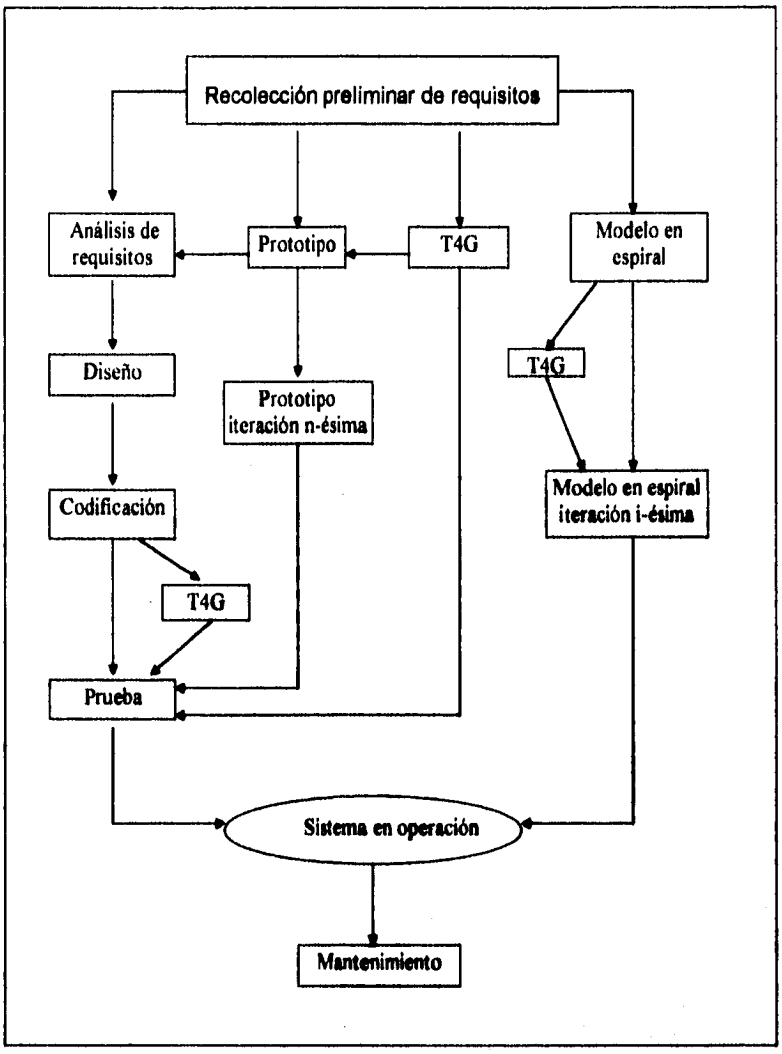


Figura 2.6. Combinación de paradigmas.

CICLO DE VIDA DEL SOFTWARE

El ciclo de vida para la ingeniería del software es el "*modelo en cascada*", ilustrado en la figura 2.7[6] . Este ciclo de vida exige un enfoque sistemático y secuencial del desarrollo del software, comenzando en el nivel del sistema y continúa a través del análisis, diseño, codificación, prueba y mantenimiento. De acuerdo a la figura del ciclo de vida, describiremos las actividades que lo componen:

Ingeniería y análisis del sistema. Se establecen los requisitos de todos los elementos del sistema. Posteriormente se asigna algún subconjunto de estos requisitos al software, para saber cuándo el software se interrelaciona con otros elementos, tales como el hardware, las personas y las bases de datos. La ingeniería y el análisis del sistema comprende los requisitos globales a nivel del sistema con poca cantidad de análisis y diseño a un nivel superior.

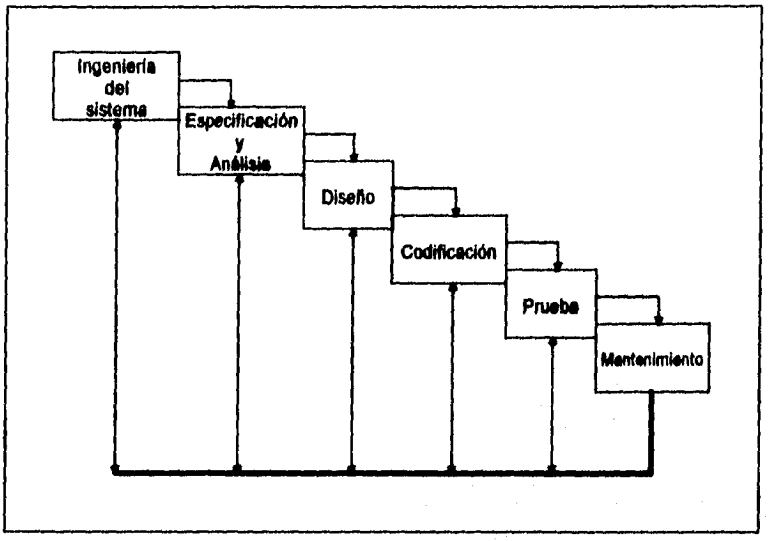


Figura 2.7. Ciclo de vida clásico.

[6] Pressman, Roger S., INGENIERÍA DEL SOFTWARE. Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

Análisis de los requisitos del software. Dado que se requiere comprender la naturaleza de los programas que hay que construir, el analista (ingeniero de software) debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridos. Los requisitos del sistema y del software se deberán documentar y serán revisados con el cliente.

Diseño. Los requisitos se traducen en una representación del software para que se obtenga la calidad requerida antes de la codificación. El diseño consta de cuatro atributos : la estructura de los datos, la arquitectura del software, el detalle de los procedimientos y la caracterización de la interface.

Codificación. En este paso el diseño se traduce en una forma legible para la máquina. Si el diseño se hizo detalladamente, la codificación se hace mecánicamente.

Prueba. Después que se ha generado el código se hace la prueba del programa. Comprobando si la lógica interna del software cumple con los requisitos establecidos para que la entrada definida produzca los resultados que se requieren.

Mantenimiento. Después que el software se ha entregado al cliente, pueden ocurrir cambios , debido a que se hayan encontrado errores, o que el software deba adaptarse a cambios del entorno externo (p. ej. : un nuevo sistema operativo o dispositivo periférico) o que el cliente requiera alguna ampliación. El mantenimiento del software aplica cada uno de los pasos anteriormente descritos, del ciclo de vida, a un programa ya existente en lugar de a uno nuevo.

Este ciclo de vida es el método más antiguo usado en la ingeniería del software, que con el paso de los años se ha visto que no se aplica a todas las situaciones, pero que sin embargo es mejor tener una guía que nos ayude a desarrollar el software, que estar sin ella y por lo tanto tener más problemas. Pasaremos a detallar cada una de las etapas.

ESPECIFICACIÓN DEL SISTEMA

En la *especificación del sistema* se describe la función, el rendimiento y las restricciones de un sistema; por lo que se delimita a cada uno de los elementos del sistema, también se describe la información (control y datos) que sirve de entrada y salida al sistema.

Después de haber elaborado el documento de la *especificación del sistema*, el siguiente paso será efectuar su revisión , la cuál será realizada por el analista y el cliente, para evaluar la definición contenida . Aquí será necesario tomar en cuenta lo siguiente :

1. Si se ha delineado adecuadamente el ámbito del proyecto.
2. Si se ha definido correctamente la funcionalidad, el rendimiento y las interfaces.
3. Si el enálisis del entorno y del riesgo del desarrollo justifican el sistema.
4. Si el analista y el cliente tienen la misma percepción de los objetivos del sistema.

Esta revisión se realiza en dos partes, primeramente se realiza desde el punto de vista de gestión, es decir, se deben de haber planteado las siguientes preguntas y éstas deben de haber sido respondidas:

- ¿Se ha establecido una necesidad en firme? ¿Tiene sentido la justificación del sistema?
- ¿Necesita el entorno especificado (o el mercado) el sistema anteriormente descrito?
- ¿Qué alternativas se han considerado?
- ¿Cuál es el riesgo de desarrollo de cada elemento del sistema?
- ¿Están disponibles los recursos necesarios para el desarrollo?
- ¿Tienen sentido las restricciones de los costos y de agenda?

Estas preguntas, que durante la etapa del análisis deben de haberse planteado y respondido, en la etapa de la revisión se vuelven a examinar.

Posteriormente se realiza una evaluación técnica de los elementos y de las funciones del sistema. El nivel de detalle de la revisión varía de acuerdo con el nivel de detalle que fue considerado durante la etapa de asignación. En esta etapa de revisión se deben de cubrir los siguientes aspectos :

- ¿La complejidad funcional del sistema corresponde con el riesgo del desarrollo, los costos y la agenda?
- ¿Está definida la asignación de funciones con suficiente detalle?
- ¿Se han definido con suficiente detelle las interfaces entre los elementos del sistema y del entorno?
- ¿Se han planteado los aspectos del rendimiento, la fiabilidad y facilidad de mantenimiento en la especificación?

- ¿Proporciona la especificación del sistema una base suficiente para que se puedan realizar los siguientes pasos de ingeniería del software y del hardware?

Después de haber terminado la revisión de la especificación del sistema, se continuará con los siguientes pasos de la ingeniería del software. En la ingeniería de sistemas es importante mencionar que la comunicación entre el cliente y el analista debe ser intensa, ya que el cliente debe de comprender los objetivos del sistema y deberá ser capaz de exponerlos claramente al analista. De la misma manera, el analista deberá saber qué preguntas hacer, que soluciones (o consejos) dar y qué investigación realizar. Si la comunicación no es adecuada o se rompe en esta fase, el éxito del proyecto estará en peligro.

I. Introducción.
   A. Ámbito y propósito del documento.
   B. Visión general.
      1. Objetivos.
      2. Restricciones.
II. Descripción funcional y de datos.
   A. Arquitectura del sistema.
      1. Diagrama de contexto de la arquitectura (DCA).
      2. Descripción del DCA.
III. Descripción de los subsistemas.
   A. Especificación del diagrama de arquitectura para el subsistema n.
      1. Diagrama de flujo de la arquitectura.
      2. Narrativa del módulo del sistema.
      3. Aspecto de rendimiento.
      4. Restricciones de diseño.
      5. Asignación de componentes del sistema.
   B. Diccionario de la arquitectura.
   C. Diagramas y descripción de la interconexión de la arquitectura.
IV. Resultados de la modelización y la simulación del sistema.
   A. Modelo del sistema utilizado para la simulación.
   B. Resultados de la simulación.
   C. Aspectos especiales del rendimiento.
V. Aspectos del proyecto.
   A. Costos de desarrollo proyectados.
   B. Agenda proyectada.
VI. Apéndices.

Tabla 2.1. Esquema de especificación del sistema.

Resumiendo, en la tabla 2.1 se tiene un esquema donde de presentan los pasos a seguir para hacer el documento de la especificación del sistema. El orden en el documento debe coincidir con el orden en el que se desarrolle el proceso de análisis.

ANÁLISIS DEL SISTEMA

El *análisis del sistema* contiene la mayoría de las tareas de la ingeniería de sistemas basados en computadora. Los objetivos del análisis del sistema son:

1. Identificar las necesidades del cliente.
2. Evaluar la viabilidad del sistema.
3. Realizar un análisis técnico y económico.
4. Asignar funciones al software, al hardware, a la gente, a la base de datos y a otros elementos del sistema.
5. Establecer restricciones de los costos y del tiempo.
6. Crear una definición del sistema que sea la base para todo el trabajo posterior de ingeniería.

Para que se alcancen estos objetivos, el analista deberá tener experiencia en hardware, software, ingeniería humana y en bases de datos, y deberá trabajar con el personal técnico y el administrativo. Se ha visto que es necesario dedicar al análisis del sistema del 10, al 20% de todo el esfuerzo de desarrollo.

ANÁLISIS Y DISEÑO ESTRUCTURADO

El análisis estructurado es una metodología para el análisis de los requerimientos de un sistema, que utiliza herramientas estructuradas, permitiendo definir tanto los requerimientos de información como los de procesamiento del sistema.

El análisis estructurado tiene como objetivo producir un documento de la especificación de los requerimientos de un sistema que cumpla con las siguientes características: que sea mantenible, gráfico, lógico, particionado y entendible. Para que se pueda realizar esto se necesitan usar las siguientes herramientas:

- Diagrama de flujo de datos (DFD). Permite representar un sistema en forma particionada, enfatizando los procesos que son componentes del mismo y las interfaces entre ellos.
- Diccionario de datos, sirve para declarar las interfaces entre los elementos del DFD.
- Miniespecificación. Se usan dentro de una DFD para describir los procesos.
- Diagrama entidad-relación. Permite la especificación de los elementos y estructuras de datos, que son independientes ya que la notación que se utiliza es estándar.

Es importante señalar las ventajas que el análisis estructurado tiene, entre ellas está la eliminación de los problemas en la redacción de un texto, ya que el uso de las figuras evita esto (una figura dice más que mil palabras). A continuación ampliaremos estas herramientas.

DIAGRAMA DE FLUJO DE DATOS

Un diagrama de flujo de datos (DFD) es una técnica gráfica que representa el flujo de la información y las transformaciones que se aplican a los datos al moverse de la entrada hasta la salida. El DFD se usa a cualquier nivel de abstracción, empezando en el nivel cero, el cual es el *modelo fundamental del sistema* o *modelo de contexto*, y que representa al elemento de software completo como una sola burbuja con datos de entrada y salida representados por flechas de entrada y salida respectivamente. El nivel 1 de un DFD, por ejemplo puede tener cuatro o cinco burbujas interconectadas con flechas, donde cada una de las burbujas (procesos) representados en este nivel es una subfunción del sistema general en el modelo de contexto.

La figura 2.8 ilustra la notación básica que se usa para un DFD. El rectángulo se usa para representar una entidad externa, es decir un elemento del sistema (p. ej.: un programa, hardware, una persona). Un proceso es un transformador de información que se aplica a los datos y los cambia de alguna forma. El proceso se representa por medio de un círculo en donde aparece el nombre del proceso y su número. Una flecha representa un elemento de datos que entra o sale de un proceso, las flechas deben ser etiquetadas. Un almacén de datos, representado por líneas paralelas, es una colección organizada de datos.

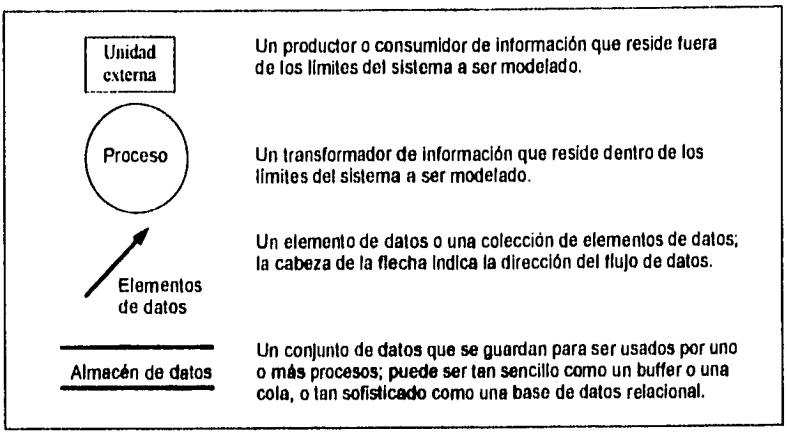


Figura 2.8. Notación DFD básica. [9]

Es importante hacer notar, que el DFD no tiene ninguna indicación explícita de la secuencia lógica de procesamiento, ya que el procedimiento o la secuencia puede estar implícitamente en el diagrama. Por lo que no hay que confundir un diagrama de flujo de datos con un diagrama de flujo, siendo este último una representación explícita de la lógica de procedimientos y el DFD representa el flujo de información haciendo énfasis en el flujo de datos y poco énfasis en el flujo de control.

Se puede refinar, como dijimos anteriormente, cada una de las burbujas en los distintos niveles para tener un mayor detalle. La figura[10] 2.9 nos muestra esto. Aquí tenemos que F es el modelo fundamental del sistema, cuya entrada principal es *A* y su salida final es *B*. Al refinar el modelo F obtenemos las transformaciones f_1 a f_7. Nótese que se debe de mantener la continuidad del flujo de información, es decir, que la entrada y la salida de cada refinamiento debe de ser la misma. Este concepto, llamado *equilibrado*, es fundamental para el desarrollo de modelos consistentes. Un mayor refinamiento de f4 nos muestra más detalle en sus transformaciones f_{41} a f_{43}. Donde la entrada (*X*, *Y*) y la salida (*Z*) permanecen sin alteración, es decir, está equilibrado.

[9] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

[10] Idém.

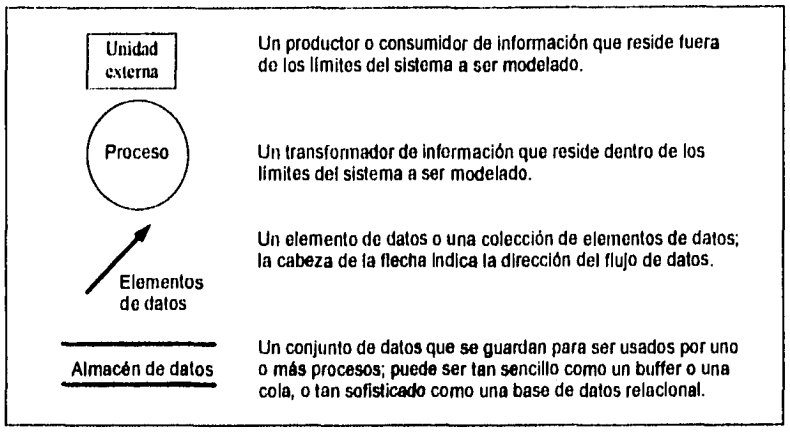


Figura 2.8. Notación DFD básica. [9]

Es importante hacer notar, que el DFD no tiene ninguna indicación explícita de la secuencia lógica de procesamiento, ya que el procedimiento o la secuencia puede estar implícitamente en el diagrama. Por lo que no hay que confundir un diagrama de flujo de datos con un diagrama de flujo, siendo este último una representación explícita de la lógica de procedimientos y el DFD representa el flujo de información haciendo énfasis en el flujo de datos y poco énfasis en el flujo de control.

Se puede refinar, como dijimos anteriormente, cada una de las burbujas en los distintos niveles para tener un mayor detalle. La figura[10] 2.9 nos muestra esto. Aquí tenemos que F es el modelo fundamental del sistema, cuya entrada principal es *A* y su salida final es *B*. Al refinar el modelo F obtenemos las transformaciones f_1 a f_7. Nótese que se debe de mantener la continuidad del flujo de información, es decir, que la entrada y la salida de cada refinamiento debe de ser la misma. Este concepto, llamado *equilibrado*, es fundamental para el desarrollo de modelos consistentes. Un mayor refinamiento de f4 nos muestra más detalle en sus transformaciones f_{41} a f_{45}. Donde la entrada (*X*,*Y*) y la salida (*Z*) permanecen sin alteración, es decir, está equilibrado.

[9] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

[10] Idém.

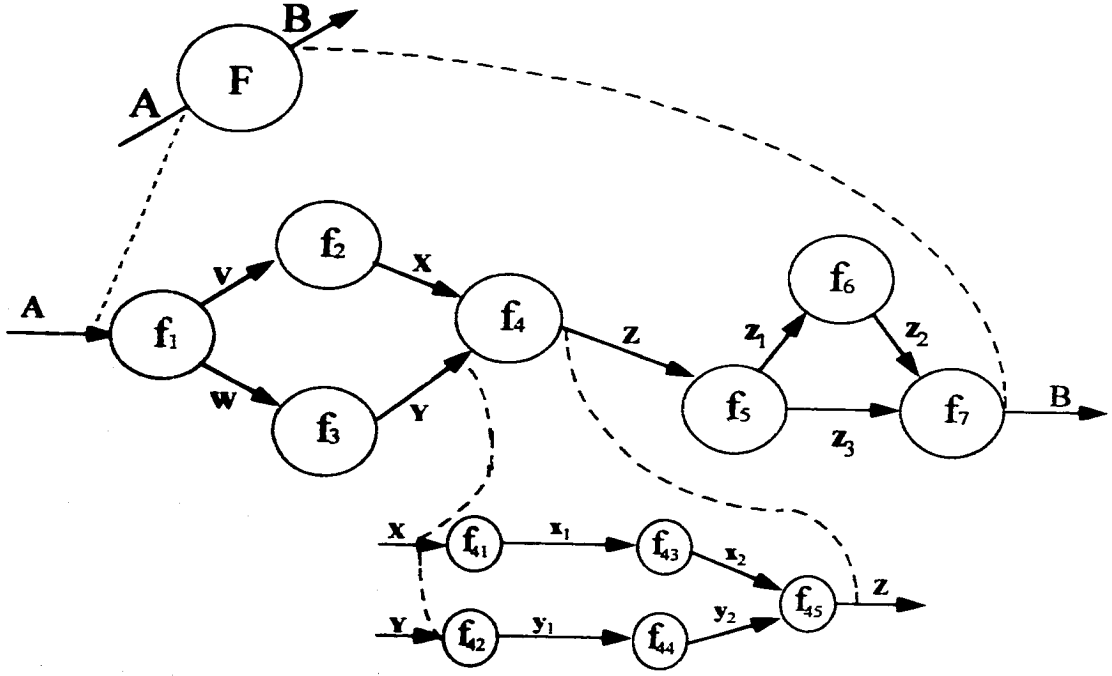


Figura 2.9. Refinamiento de información.

La notación gráfica representada en la figura 2.9 debe ser ampliada con un texto descriptivo. Para esto se puede usar un párrafo que describa una burbuja de procesamiento, para especificar los detalles que tiene una burbuja del DFD. Este párrafo describirá la entrada a la burbuja, el algoritmo que se aplica a esa entrada y la salida que se produce; además indicará las restricciones y limitaciones que deberá tener el proceso, las características de rendimiento relevantes al proceso y las restricciones de diseño que puedan tener influencia en la forma de implementar el proceso.

DICCIONARIO DE DATOS

Otro componente de la notación básica del análisis estructurado es el *diccionario de datos* o *diccionario de requisitos*, el cual describe el contenido de los datos implicados en las flechas o en el conjunto de datos. Veamos la siguiente definición:

> "El diccionario de datos es un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que permiten que el usuario y el analista del sistema tengan una misma comprensión de las entradas, salidas, de los componentes de los conjunto de datos y también de los cálculos intermedios." [11]

En la herramienta CASE de análisis y diseño estructurado, casi siempre se encuentra implementado el diccionario de datos. Aunque el formato de estos diccionarios varía en las distintas herramientas, la mayoría contiene la siguiente información:

- *Nombre*.- el nombre principal del elemento de datos o de control, del almacén de datos o de una entidad externa.
- *Alias*.- otros nombres usados para la entrada.
- *Dónde se usa/cómo se usa*.- un listado de los procesos que usan el elemento de datos o de control y cómo lo usan (p. ej.: como entrada al proceso, como salida del proceso, como almacén de datos, como entidad externa).
- *Descripción del contenido*.- el contenido representado mediante una notación.

[11] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

La notación gráfica representada en la figura 2.9 debe ser ampliada con un texto descriptivo. Para esto se puede usar un párrafo que describa una burbuja de procesamiento, para especificar los detalles que tiene una burbuja del DFD. Este párrafo describirá la entrada a la burbuja, el algoritmo que se aplica a esa entrada y la salida que se produce; además indicará las restricciones y limitaciones que deberá tener el proceso, las características de rendimiento relevantes al proceso y las restricciones de diseño que puedan tener influencia en la forma de implementar el proceso.

DICCIONARIO DE DATOS

Otro componente de la notación básica del análisis estructurado es el *diccionario de datos* o *diccionario de requisitos*, el cual describe el contenido de los datos implicados en las flechas o en el conjunto de datos. Veamos la siguiente definición:

> "El diccionario de datos es un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que permiten que el usuario y el analista del sistema tengan una misma comprensión de las entradas, salidas, de los componentes de los conjunto de datos y también de los cálculos intermedios." [11]

En la herramienta CASE de análisis y diseño estructurado, casi siempre se encuentra implementado el diccionario de datos. Aunque el formato de estos diccionarios varía en las distintas herramientas, la mayoría contiene la siguiente información:

- *Nombre.*- el nombre principal del elemento de datos o de control, del almacén de datos o de una entidad externa.
- *Alias.*- otros nombres usados para la entrada.
- *Dónde se usa/cómo se usa.*- un listado de los procesos que usan el elemento de datos o de control y cómo lo usan (p. ej.: como entrada al proceso, como salida del proceso, como almacén de datos, como entidad externa).
- *Descripción del contenido.*- el contenido representado mediante una notación.

[11] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

- Información adicional.- otra información sobre los tipos de datos, los valores implícitos (si se conocen), las restricciones o limitaciones, etc.

La figura 2.10 nos muestra la notación usada para desarrollar una descripción del contenido, representando los datos compuestos en una de tres formas fundamentales de construcción.

Construcción del dato	Notación	Significado
	=	está compuesto de
Secuencia	+	y
Selección	[|]	o bien-o
Repetición	{ }	n repeticiones de
	( )	datos opcionales
	* *	delimita comentarios

Figura 2.10. Notación de descripción de contenido para un diccionario de datos.[12]

Para ilustrar esto, veamos por ejemplo, el número de crédito cómo se define en el diccionario de requisitos:

nombre: número de crédito

alias: ninguno

dónde se usa/cómo se usa: (entrada)

descripción:

número de crédito = [últimos dos dígitos del año de asignación del crédito]
+ [000 | 001 | 002 | ... | 999] + [número secuencial]
+ [dígito verificador]

número secuencial = [00000 | 00001 | ... | 99999]

Esta descripción significa que el *número de crédito* está compuesto de los dos últimos dígitos del año en que se asignó el crédito, es decir, el año en que se generó el número de crédito, y de un número de tres dígitos, y de un número secuencial de 5 dígitos, y de un dígito verificador.

[12] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

Para grandes sistemas basados en computadora, el diccionario de datos crece rápidamente en tamaño y en complejidad, por lo que es difícil darle mantenimiento manual. Por esta razón, es preferible usar herramientas CASE.

ESPECIFICACIÓN DE PROCESAMIENTO

En el nivel final del refinamiento, la especificación de procesamiento(PSPEC) se usa para describir los procesos del modelo de flujo. La PSPEC se encuentra en cada burbuja del modelo de flujo y es una narración textual, puede ser una descripción del algoritmo del proceso, una ecuaciones matemáticas, tablas, diagramas o gráficos; esto constituye una guía para el diseño del componente de programa que implementará el proceso.

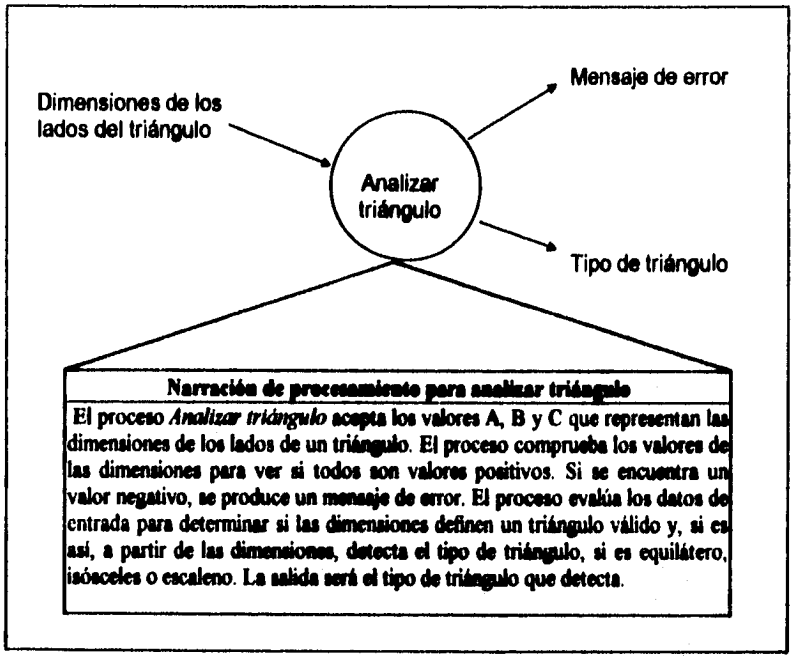


Figura 2.11. Especificación de procesamiento para un proceso de un DFD.

En la figura[13] 2.11, vemos un ejemplo, en la que se analiza un triángulo, éste forma parte de un refinamiento del diagrama de flujo de una aplicación de software, en la que se analizan las dimensiones de varios objetos geométricos para identificar la forma del objeto. Aquí ya nos encontramos en el nivel 2, y se incluye una narración de la PSPEC de analizar un triángulo.

[13] Pressman, Roger S., INGENIERÍA DEL SOFTWARE, Un enfoque práctico. 3ª edición. 1993. McGraw-Hill/Interamericana de España, S. A.

CAPÍTULO 3

SISTEMAS DE BASES DE DATOS

Una **base de datos** es una colección de datos interrelacionados, almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es la de servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir datos nuevos y para modificar o extraer los datos almacenados.

Antes de seguir adelante daremos algunas definiciones que nos van a ser de utilidad :

Campo de datos es el elemento básico, el más pequeño grupo de datos , también llamado ítem o elemento de datos.

Registro es un conjunto de campos o ítems de datos .

Archivo es un conjunto de registros .

Visto desde estas definiciones una **Base de datos** es un conjunto de archivos relacionados entre sí , y un **Sistema de bases de datos** es una colección de bases de datos , cuyos contenidos son totalmente independientes y disjuntos, desde el punto de vista estructural.

Características de una base de datos :

- En la base de datos se guarda toda la información necesaria para el ejercicio de las funciones propias de una organización; esto permite la lectura de los datos almacenados, por muy diversos usuarios, así como su posible modificación cuando sea necesaria para el control de las operaciones.

- En la base de datos puede haber procesamiento por lotes, en tiempo real o en línea (en el que las transacciones se procesan de una por una hasta completarse, pero sin los severos requisitos de tiempo propios de los sistemas de tiempo real) .

- Antes de las bases de datos, existían datos que se hallaban simultáneamente almacenados en diferentes lugares con distintas finalidades y también con diferentes fechas de actualización. En las bases de datos se pretende eliminar esta redundancia, pero en ocasiones es necesario tener cierta redundancia con el objeto de reducir tiempos de acceso o simplificar los métodos de

direccionamiento. Por ello es necesario que en una base de datos se hable de redundancia controlada o redundancia mínima en lugar de no redundancia como criterio de diseño. Es decir, en una base bien diseñada se evita la redundancia perjudicial o innecesaria.

Es importante que la redundancia esté controlada, ya que de no ser así, el costo del almacenamiento de copias múltiples de los datos es mayor; además, lo que es más serio, para actualizar esas copias es necesario hacer múltiples operaciones de actualización, así como inclusión de datos nuevos y la eliminación de los ya obsoletos que deberán hacerse en todas las copias que se tengan; de no hacer esto, las distintas copias estarán en diferentes estados de actualización y entonces el sistema proporcionará información contradictoria .

- Las bases de datos deben prestarse a una fácil reestructuración cuando sea necesario agregarle nuevos tipos de datos o utilizarla para nuevas aplicaciones. Esta reestructuración no debe originar la necesidad de volver a escribir los programas de aplicación y no debe originar trastornos.

- Por **independencia de datos**, se entiende que los datos y los programas de aplicación son mutuamente independientes, de tal manera que unos u otros pueden ser modificados sin afectar a los restantes. De esta manera el programador de aplicaciones no debe ser afectado por los cambios que se introduzcan en los datos, en su organización, o en los dispositivos físicos donde se almacenan. Así como anteriormente se dijo que los datos son pocas veces totalmente no redundantes, así también ellos son pocas veces completamente independientes. La independencia de los datos es uno de los argumentos más valiosos en pro de los sistemas de bases de datos .

- La seguridad de los datos es la protección de éstos contra el acceso accidental o intencional por parte de personas no autorizadas y contra su indebida destrucción o alteración.

- La reserva de los datos se refiere al derecho de las personas y organismos para determinar por sí mismos cuándo, cómo y en qué medida se permitirá la transmisión a terceros de la información que les concierne.

- La base de datos está organizada de tal manera que los datos que contiene pueden ser alcanzados de muchas maneras diferentes y ser utilizados para responder a una diversidad de interrogantes .

COMPONENTES PRINCIPALES DE UN SISTEMA BASE DE DATOS

Los componentes principales de un *sistema de base de datos* son : los **datos**, el **hardware**, el **software** y los **usuarios**.

Los **datos** se encuentran almacenados en una o más bases de datos.

El **hardware** se compone de volúmenes de almacenamiento secundario (discos, cintas, etc.), donde reside la base de datos, pues se requiere de un medio de almacenamiento donde la información perdure más allá del tiempo de proceso. Generalmente, las bases de datos son muy grandes para poderse almacenar en la memoria; es más, aunque fueran de tamaño pequeño, la memoria es un componente volátil.

Entre la base de datos física en sí (es decir, el almacenamiento real de los datos) y los usuarios del sistema existe el **software**, llamado a menudo Sistema de Administración de Bases de Datos (DBMS). Este maneja todas las solicitudes de acceso a la base de datos efectuadas por los usuarios.

Se consideran tres clases de **usuarios** :

- El **programador de aplicaciones**, que es el encargado de escribir los programas de aplicación que utilicen las bases de datos.

- El **usuario final** , que acceda la base de datos desde una terminal a través de un programa de aplicación o puede emplear un lenguaje de consulta proporcionado como parte integral del software .

- El **administrador de la base de datos** (DBA) , es el responsable de la seguridad y control de los datos. Si se requiere crear un nuevo tipo de registro, o modificar un registro antiguo para incluir nuevos campos de datos o aumentar el tamaño de un ítem existente, el administrador tomará las medidas necesarias para modificar la base de datos de la manera que él crea más conveniente. El administrador de la base de datos en realidad es un departamento o un grupo de personas que tienen la completa comprensión de la base de datos , su organización , su economía, sus criterios de diseño, y los requerimientos de muchos usuarios. Estas actividades generalmente son llevadas a cabo por un equipo, ya que resultan ser demasiado para una sola persona.

INDEPENDENCIA DE DATOS, LÓGICA Y FÍSICA

En muchas bases de datos los cambios de la estructura lógica general de los datos son continuos, por lo que se necesitan dos niveles de independencia da datos : la independencia lógica y la independencia física.

La **independencia lógica** de los datos es la modificación en la estructura lógica general y no afecta a los programas de aplicación; este cambio no debe eliminar ninguno de los datos que cada programa necesita.

La **independencia física** de los datos es cuando se puede modificar la distribución y organización físicas de los datos sin afectar ni a la estructura lógica general ni a los programas de aplicación.

El programador trabaja con una vista lógica, que es una parte de los datos de la base, y el sistema lo debe transformar a la representación física que consiste en la relación que hay en los datos.

TIPOS DE ORGANIZACIÓN

Para la organización de los datos existen tres aspectos :

- **Organización de archivos**, que es la que se ocupa de la vista de los datos tal como los percibe el programador, ya que éste describe su propia vista de los archivos en su programa de aplicación. El programador puede ver los archivos como un registro maestro con registros de detalle subordinados.

- **Organización lógica global de los datos**, se ocupa de la organización general de la base de datos de la cual pueden derivarse múltiples organizaciones de archivos. Este tipo de organización es diferente de la organización física del almacenamiento.

- **Organización física del almacenamiento**, tiene por objeto la representación, organización y distribución física de los datos en las unidades de almacenamiento.

ESTRUCTURAS DE DATOS

Al usuario se le presenta sólo una vista externa de los datos, en la que se omiten los detalles de almacenamiento; en la mayoría de los casos la vista externa y la vista conceptual son muy semejantes, casi idénticas.

Existen tres estructuras de bases de datos : jerárquico, red y relacional, las cuales describiremos brevemente a continuación.

JERÁRQUICO. Tiene como principal característica que, las relaciones entre las entidades son del tipo 1-1 (uno a uno; un padre-un hijo), o también 1-M (uno a M; un padre-varios hijos). es muy importante observar que las relaciones M-M (muchos a muchos; varios padres-varios hijos) no está permitida. La terminología para las estructuras jerárquicas es :

ÁRBOL : se compone de nodos.
RAÍZ : un árbol contiene un sólo nodo, llamado raíz.
HOJAS : son aquellos nodos que no tienen ningún subordinado.
PADRE e HIJO : cada nodo está conectado con un sólo nodo en el nivel superior, excepto el nodo raíz. El nodo de nivel más alto se llama padre y el subordinado se llama hijo.

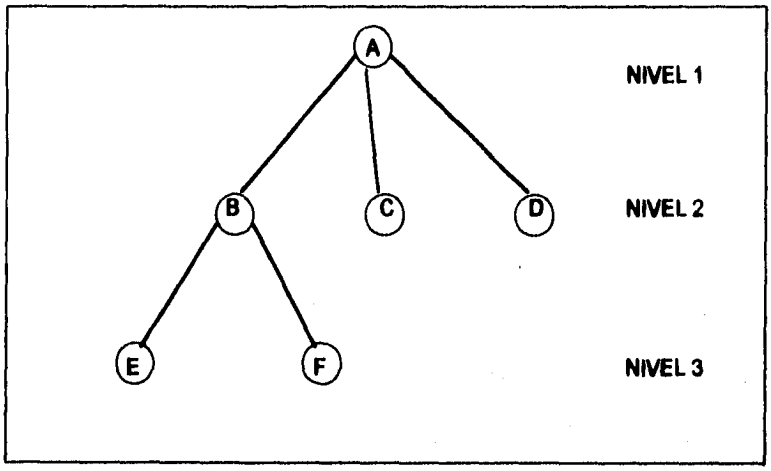


Figura 3.1. Modelo Jerárquico.

En la figura 3.1 observamos que el nodo A es el nodo raíz, B es el nodo padre de E y F, y los nodos C, D, E y F son las hojas, ya que no tienen ningún subordinado.

RED. Aquí se permiten relaciones más complejas entre entidades que en el jerárquico, como las del tipo M-M (muchos a muchos; varios padres-varios hijos), es decir un hijo puede tener varios padres.

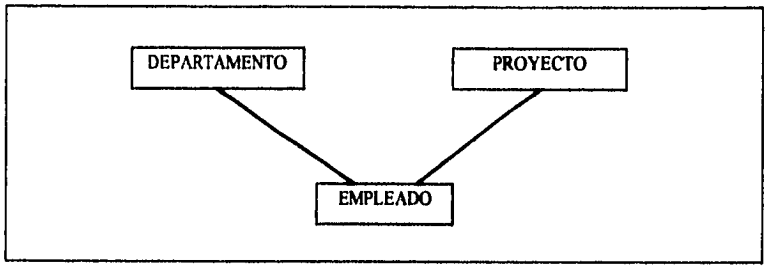


Figura 3.2. Modelo de Red.

Como se puede observar en la figura 3.2 , DEPARTAMENTO y PROYECTO son padres de EMPLEADO. En la figura 3.3 tenemos una ocurrencia de este modelo de red, donde DEPARTAMENTOA tiene a los hijos EMPLEADO1, EMPLEADO3 y EMPLEADO8, mientras que PROYECTOC tiene a los hijos EMPLEADO8, EMPLEADO5 y EMPLEADO7.

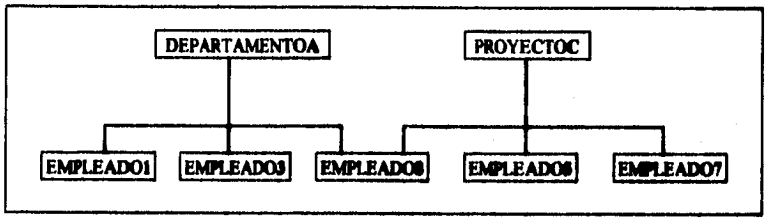


Figura 3.3 . Una ocurrencia en una base de datos de red

RELACIONAL. Organiza lógicamente a los datos en tablas, donde una tabla es una relación. Aquí existen relaciones entre los registros (renglones) de los archivos (tablas) que componen la base de datos. Cada columna de la relación es un atributo, y cada renglón es un tuplo. Se llama relacional por la semejanza de las relaciones que están definidas en matemáticas, las cuales obedecen ciertas restricciones. Más adelante ampliaremos esta parte dado que nos abocaremos a este tipo de base de datos.

ACREDITADO

NUMERO DE CREDITO	NOMBRE	DIRECCION
9308237856	PEREZ CRUZ JUAN	CALLE 9 ·Nº 324
9378192867	REYES LIMA ELISA	AV. CINCO Nº 40
9402365850	ALBA ROSAS SOFIA	PIRINEOS Nº 452
9409253674	BARRON PEREZ JOSEFINA	PIRINEOS Nº 15

Figura 3.4. Tabla de un Modelo Relacional.

La figura 3.4 nos muestra una tabla (archivo) cuyo nombre es ACREDITADO y sus atributos son : NUMERO-DE-CREDITO, NOMBRE y DIRECCION. Un tuplo de esta relación es (9308237856, PEREZ CRUZ JUAN, CALLE 9 Nº 324).

La estructura de la base de datos determina el tipo de análisis a efectuar para una cierta aplicación. Las bases de datos relacionales son hoy en día las más usuales, dado que presentan mayor versatilidad que la jerárquica ó de red y se encuentran en el mercado muchos productos para explotarlas, que han demostrado su eficiencia y exactitud. Por ello para el desarrollo de nuestra aplicación utilizaremos el modelo de bases de datos relacional.

ESTRUCTURA DE DATOS RELACIONAL

Una de las partes más importantes en las estructuras de datos relacionales o modelo relacional es el concepto de tener operadores lógicos para la manipulación de los datos.

Una *entidad* es algo que existe y se distingue, es decir, podemos distinguir una entidad de otra. Por ejemplo, cada silla es una entidad, así como cada persona o cada coche. Podríamos decir que una hormiga es una entidad si tuviéramos una manera de distinguir una hormiga de la otra; fuera de eso no podemos decir que una hormiga es una entidad. También podemos identificar

entidades en conceptos de alto nivel. Por ejemplo, en una base de datos de biología, los términos Arácnido, Roedor y Planta podrían ser entidades.

Un grupo de entidades similares forma un **conjunto entidad** . Ejemplos de conjunto entidad son :

1. todas las personas
2. todas las personas vivas
3. todos los automóviles
4. todas las emociones

Nótese en los ejemplos (1) y (2), personas y personas vivas, que el término "entidades similares" no está precisamente definido, y podemos establecer un número infinito de diferentes propiedades por las cuales definimos un conjunto entidad.

ATRIBUTOS Y LLAVES

Las entidades tienen propiedades, llamadas atributos, los cuales asocian un valor desde un dominio de valores para ese atributo con cada entidad en un conjunto entidad. Usualmente, el dominio para un atributo será un conjunto de enteros, números reales, ó cadenas de caracteres. Por ejemplo, las entidades en el conjunto entidad de personas se puede decir que tiene atributos tales como nombre (una cadena de caracteres), estatura (un número real), y así sucesivamente.

La selección de atributos relevantes para conjuntos entidad es otro paso crítico en el diseño de un modelo del mundo real. Un atributo o conjunto de atributos cuyos valores identifican unívocamente a cada entidad en un conjunto entidad se llama llave para ese conjunto entidad. En principio, cada conjunto entidad tiene una llave, estamos asumiendo que cada entidad se distingue de las demás. Pero, si para un conjunto entidad, no escogemos una colección de atributos que incluyan una llave, entonces no seríamos capaces de distinguir una entidad de otra en el conjunto. Por ejemplo, en un conjunto entidad podríamos usar como llave el RFC ó el número del IMSS, si estos son atributos del conjunto entidad.

La llave es un valor usado para identificar un registro. Una llave puede ser un campo ó varios campos, en cuyo caso se le conoce como una llave compuesta. Una llave no puede ser modificada, ya que no podría identificar al registro. Hay varios tipos de llaves :

La **llave primaria** es un valor que identifica univocamente a un registro dentro de un archivo. Cada registro en una base de datos normalizada debe tener una llave primaria, ningún valor de esta llave puede ser nulo.

La **llave primaria compuesta** contiene varios campos.

La **llaves candidatas** consiste de llaves en un registro, cada una de las cuales identifica ese registro, de entre estas llaves candidatas se escoge la que será la llave primaria y una vez elegida no se podrá cambiar, ya que implicaría un gran costo en el sistema.

Las **llaves secundarias** son las llaves candidatas que quedan después de haber seleccionado la llave primaria.

La **llave externa** es una llave que no es primaria en un archivo y además es una llave primaria en otro archivo. Dado que se necesitan mantener relaciones, las llaves externas son los únicos datos redundantes encontrados en una base de datos completamente normalizada.

MODELO RELACIONAL

La estructura básica del modelo relacional es la "relación" según se define en matemáticas. Es una correspondencia entre dos conjuntos, que asocia a cada elemento del primer conjunto algún elemento del segundo conjunto. La relación es un subconjunto del producto cartesiano de una lista de dominios, siendo un dominio un conjunto de valores. Por ejemplo, el conjunto de enteros es un dominio, también lo son, el conjunto de cadenas de caracteres, el conjunto de caracteres de longitud 30, los números reales, el conjunto { 0, 1 }, etc.

Al producto Cartesiano de dominios $D_1, D_2, \ldots, D_n$, escrito $D_1 \times D_2 \times \ldots \times D_n$, es el conjunto de todos los *n*-tuplos ó *n*-adas $(v_1, v_2, \ldots, v_n)$ tales que $v_1 \in D_1$, $v_2 \in D_2, \ldots, v_n \in D_n$.
Por ejemplo, si n=2 y tenemos los dominios $D_1 = \{ 0, 1 \}$ y $D_2 = \{ a, b, c \}$, entonces el producto cartesiano :

$$D_1 \times D_2 = \{ (0,a), (0,b), (0,c), (1,a), (1,b), (1,c) \}$$

Una relación es cualquier subconjunto del producto Cartesiano de uno o más dominios. Por ejemplo, el conjunto $\{ (0, a), (0, c), (1, b) \}$ es una relación y es un subconjunto de $D_1 \times D_2$, como lo definimos en el ejemplo anterior. El conjunto vacío es una relación. El conjunto de parejas $(0, x)$ con x en D_2 es una relación en $D_1 \times D_2$.

Los elementos de una relación son llamados tuplos o n-adas. Se dice que la relación tiene grado n si es un subconjunto del producto cartesiano $D_1xD_2x \ldots xD_n$. Un tuplo ó n-ada (v_1 , v_2 , ... , v_n) tiene n componentes, donde el i-ésimo componente es v_i . El tuplo (v_1 , v_2 , ... , v_n) también se puede escribir como $v_1 v_2 \ldots v_n$.

La relación la podemos ver como una tabla, donde cada renglón es un tuplo y cada columna corresponde a un componente. Muchas veces a las columnas se les dan nombres, llamados atributos. El esquema de relación es el conjunto de nombres de atributo de una relación. Si a una relación la llamamos REL, y su esquema de relación tiene los atributos A_1 , A_2 , ... , A_n , podemos escribir el esquema de relación como REL (A_1 , A_2 , ... , A_n) .

La colección de esquemas de relación usados para representar información se llama un **esquema de base de datos relacional** y los valores mismos de las relaciones correspondientes constituyen la **base de datos relacional** .
En el ejemplo que vimos al principio de este capítulo, en la relación

ACREDITADO(NUMERO-DE-CREDITO, NOMBRE, DIRECCION)

la llave primaria es NUMERO-DE-CREDITO, siendo (9409253874 , BARRON PEREZ JOSEFINA, PIRINEOS Nº 15, COL. PORTALES) un tuplo de la relación ACREDITADO.

ACREDITADO

NUMERO DE CREDITO	NOMBRE	DIRECCION
9308237888	PEREZ CRUZ JUAN	CALLE 8 Nº 324
9378192887	REYES LIMA ELISA	AV. CINCO Nº 40
9402388880	ALBA ROSAS SOFIA	PIRINEOS Nº 482
9409253874	BARRON PEREZ JOSEFINA	PIRINEOS Nº 15

Figura 3.5. Tabla relacional ACREDITADO.

ÁLGEBRA RELACIONAL

Una de las ventajas de la manipulación de los datos con el modelo relacional es que se puede trabajar con conjuntos de registros y no con registros individuales, como tradicionalmente se hacía cuando se usaban archivos convencionales .

Cuando dos entidades están definidas en los mismos dominios, se pueden considerar como conjuntos del mismo universo : el conjunto de todos los posibles tuplos en la relación resultante del producto cartesiano de esos dominios. De manera que si R y S son relaciones de grado n tales que el dominio D_1 de R es igual al dominio D_1 de S , para toda i = { 1, 2, ..., n } , les podemos aplicar las operaciones booleanas unión, intersección y diferencia, tradicionales, además de las de producto cartesiano, proyección y selección.

OPERACIONES BOOLEANAS SOBRE RELACIONES

1.- **Unión.** La unión de las relaciones R y S, denotado por $R \cup S$, es el conjunto de tuplos que están en R ó en S, ó en ambos .

2.- **Diferencia.** La diferencia de las relaciones R y S, denotado por R - S , es el conjunto de tuplos que están en R pero no están en S .

3.- **Producto Cartesiano.** Sean R y S relaciones de grado n_1 y n_2 , respectivamente. Entonces el producto Cartesiano de R y S, RxS ,es el conjunto de ($n_1 + n_2$) tuplos cuyos primeros n_1 componentes forman un tuplo en R y cuyos últimos n_2 componentes forman un tuplo en S .

4.- **Proyección.** Si R es una relación de grado k, denotamos $\pi_{i1,i2,...,im}$ (R), donde las i_j son enteros distintos entre 1 y k, a la proyección R sobre los componentes $i_1 , i_2 , ... , i_m$, es decir , el conjunto de m-tuplos $a_1 a_2 ... a_m$ tal que hay algún k-tuplo $b_1 b_2 ... b_k$ en R para lo cual $a_j = b_{ij}$,para j=1,2,...,m. Por ejemplo, $\pi_{3,1}$ (R) se calcula tomando cada tuplo t en R y formando una pareja del tercer y primer componentes de t, en ese orden. Si R tiene a sus atributos etiquetando sus columnas, entonces podemos substituir los nombres de los atributos por números del componente, y podemos usar los mismos nombres de los atributos en la relación proyectada.
Por ejemplo, si la relación R es R(A, B, C, D) , entonces $\pi_{C,A}$ (R) es la misma que $\pi_{3,1}$ (R) , y la relación resultante tiene como nombre de su primera columna al atributo C y el nombre de su segunda columna al atributo A .

5.- **Selección**. Sea F una fórmula que tiene :

i) operandos que son números constantes o componentes ,
ii) los operadores de comparación < , = , > , $\leq$, $\neq$ y $\geq$, y
iii) los operadores lógicos $\wedge$ (y) , $\vee$ (o) , y la $\neg$ (negación) .

Entonces σ_F (R) es el conjunto de tuplos t en R, tales que, para toda i , cuando substituimos el i-ésimo componente de t para cualquiera ocurrencias del número i en la fórmula F , la fórmula F es verdadera.
Por ejemplo , $\sigma_{2>3}$ (R) es el conjunto de tuplos en R cuyo segundo componente es mayor a su tercer componente , mientras $\sigma_{1='BAEZ' \vee 1='REYES'}$ (R) es el conjunto de tuplos en R cuyo primer componente tiene el valor 'BAEZ' o el valor 'REYES'. Igual que en la proyección, si en una relación las columnas tienen nombres, entonces la fórmula en una selección puede referirse a las columnas por nombre en lugar de su número. Nótese también que aún las constantes numéricas deben ser citadas en fórmulas, para distinguirlas de los números o nombres de las columnas.

A	B	C
a	b	e
d	a	f
c	b	d

Relación R

D	E	F
b	g	a
d	a	f

Relación S

Figura 3.6. Relación R y S.

Ahora pasaremos a ver los ejemplos : Sean R y S dos relaciones las cuales están en la figura 3.6 . En la figura 3.7 inciso a) y b) , tenemos la relación R $\cup$ S y R - S . Nótese que podemos hacer uniones y diferencias aún cuando las columnas de las dos relaciones tengan diferentes nombres, así como las relaciones tengan el mismo número de componentes; pero la relación resultante no tiene nombres para sus columnas. La figura 3.8 nos muestra la relación RxS. Dado que R y S tienen conjuntos de atributos disjuntos, podemos poner los nombres de las columnas en R x S . Si R y S tienen un nombre de columna en común , por ejemplo G, podríamos distinguir a las dos columnas llamándolas R.G y S.G. En la figura 3.9 incisos a) y b) vemos la proyección $\pi_{A,C}$ (R), y la selección $\sigma_{B=b}$ (R).

5.- **Selección**. Sea F una fórmula que tiene :

i) operandos que son números constantes o componentes ,
ii) los operadores de comparación $<$, $=$, $>$, $\leq$, $\neq$ y $\geq$, y
iii) los operadores lógicos $\wedge$ (y) , $\vee$ (o) , y la $\neg$ (negación) .

Entonces σ_F (R) es el conjunto de tuplos t en R, tales que, para toda i , cuando substituimos el i-ésimo componente de t para cualquiera ocurrencias del número i en la fórmula F , la fórmula F es verdadera.
Por ejemplo , $\sigma_{2>3}$ (R) es el conjunto de tuplos en R cuyo segundo componente es mayor a su tercer componente , mientras $\sigma_{1='BAEZ' \vee 1='REYES'}$ (R) es el conjunto de tuplos en R cuyo primer componente tiene el valor 'BAEZ' o el valor 'REYES'. Igual que en la proyección, si en una relación las columnas tienen nombres, entonces la fórmula en una selección puede referirse a las columnas por nombre en lugar de su número. Nótese también que aún las constantes numéricas deben ser citadas en fórmulas, para distinguirlas de los números o nombres de las columnas.

A	B	C
a	b	e
d	a	f
c	b	d

Relación R

D	E	F
b	g	a
d	a	f

Relación S

Figura 3.6. Relación R y S.

Ahora pasaremos a ver los ejemplos : Sean R y S dos relaciones las cuales están en la figura 3.6 . En la figura 3.7 inciso a) y b) , tenemos la relación $R \cup S$ y R - S . Nótese que podemos hacer uniones y diferencias aún cuando las columnas de las dos relaciones tengan diferentes nombres, así como las relaciones tengan el mismo número de componentes; pero la relación resultante no tiene nombres para sus columnas. La figura 3.8 nos muestra la relación RxS. Dado que R y S tienen conjuntos de atributos disjuntos, podemos poner los nombres de las columnas en R x S . Si R y S tienen un nombre de columna en común , por ejemplo G, podríamos distinguir a las dos columnas llamándolas R.G y S.G. En la figura 3.9 incisos a) y b) vemos la proyección $\pi_{A,C}$ (R), y la selección $\sigma_{B=b}$ (R).

a	b	c
d	a	f
c	d	d
b	g	a

a) $R \cup S$

a	b	c
c	b	d

b) $R - S$

Figura 3.7. Unión y Diferencia de las relaciones R y S.

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

Figura 3.8. Relación RxS

A	C
a	c
d	f
c	d

a) $\pi_{AC}(R)$

A	B	C
a	b	c
c	b	d

b) $\sigma_{a=b}(R)$

Figura 3.9. Proyección y Selección de la relación R.

A partir de esta operaciones básicas del álgebra relacional podemos obtener otras operaciones. Así tenemos :

1.- **Intersección** . La intersección de las relaciones R y S, $R \cap S$, es equivalente a $R - (R - S)$.

2.- **Cociente** . Sean R y S relaciones de grado r y s, respectivamente, donde r>s, y S es diferente del vacío. Entonces el cociente de R y S , $R \div S$, es el conjunto de (r-s)-tuplos $a_1, \ldots, a_{r-s}$ tales que para todos los s-tuplos

a_{r-s+1}, ..., a_r están en S, el tuplo a_1, ..., a_r está en R. Para expresar R ÷ S usando las cinco operaciones básicas, sea T para $\pi_{1,2,...,r-s}$ (R). Entonces (T x S) - R es el conjunto de r -tuplos que no están en R, pero se forman al tomar los primeros (r - s) componentes de un tuplo en R y seguido de un tuplo en S. Entonces sea

$$V = \pi_{1,2,...,r-s} ((T \times S) - R)$$

donde V es el conjunto de (r - s) -tuplos t que los primeros (r - s) componentes de un tuplo en R tal que para algún s-tuplo *u* en S, *tu* no está en R. Por consiguiente T - V es R ÷ S. Podemos escribir R ÷ S de la siguiente manera:

$$R \div S = \pi_{1,2,...,r-s} (R) - \pi_{1,2,...,r-s} ((\pi_{1,2,...,r-s} (R) \times S) - R)$$

En la figura 3.10, tenemos en los incisos a) y b), las relaciones R y S. Al efectuar el cociente el resultado se muestra en el inciso c). Vemos que el tuplo *ab* está en el cociente R ÷ S, ya que *abcd* y *abef* están en la relación R, y además el tuplo *de* está en R ÷ S por que *edcd* y *edef* están en la relación R.

a	b	c	d
a	b	e	f
b	b	e	f
e	d	c	d
e	d	e	f
a	b	c	e

a) Relación R.

c	d
e	f

b) Relación S.

a	b
e	d

c) R ÷ S

Figura 3.10. Cociente de las relaciones R y S.

3.- **Join**. El θ–join de R y S de las columnas i y j, $R \triangleright\triangleleft_{i\,\theta\,j} S$, donde θ es un operador de comparación (=, <, >, etc.), es decir, $\sigma_{i\theta(r+j)}$ (R x S), si R es de grado r. Esto es, el θ–join de R y S son aquellos tuplos en el producto Cartesiano de R y S tales que el i-ésimo componente de R permanece en la relación θ para el j-ésimo componente de S. En ocasiones la operación se llama **equijoin** cuando θ es el signo = (igual).

En la figura 3.11, tenemos las relaciones R y S, en los incisos a) y b) respectivamente. Entonces $R \triangleright\!\lhd_{B<D} S$ se lo tenemos en el inciso c) de la misma figura. Como podemos ver, las columnas tienen nombres por lo que la operación de join la podemos expresar, en este caso, con el nombre de sus columnas , y así tenemos $R \triangleright\!\lhd_{B<D} S$ o bien, por el número de sus columnas $R \triangleright\!\lhd_{2<1} S$.

A	B	C
1	2	3
4	5	6
7	8	9

a) Relación R.

D	E
3	1
6	2

b) Relación S.

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

c) $R \triangleright\!\lhd_{B<D} S$

Figura 3.11. Ejemplo de la operación Join con el operador <.

4.- **Join Natural** . El Join Natural , $R \triangleright\!\lhd S$, se aplica solamente cuando R y S tienen columnas con nombres de atributos. Para calcular $R \triangleright\!\lhd S$, se procede como sigue :

a) Calcular R x S

b) Para cada atributo A , que es el nombre de una columna de R y otra columna en S, seleccionada de R x S, en cuyos tuplos corresponden valores en las columnas R.A y S.A . Decimos que R.A es el nombre de la columna de R x S que corresponde a la columna A de R, y S.A se define análogamente.

c) Para cada atributo A mencionado anteriormente , se proyecta la columna S.A .

Formalmente, tenemos que , si A_1 , A_2 , ... A_k son todos los nombres de los atributos usados tanto para R como para S, $R \triangleright\!\lhd S$ es entonces

$$\pi_{i_1,i_2,\ldots,i_m}\, \sigma_{R.A1=S.A1\wedge\ldots\wedge R.Ak=S.Ak}(R \times S)$$

donde i_1 , i_2 , ... , i_m es la lista de todos los componentes de R x S , en el orden , excepto los componentes $S.A_1$, ... , $S.A_k$.

En la figura 3.11, tenemos las relaciones R y S, en los incisos a) y b) respectivamente. Entonces $R \triangleright\!\!\triangleleft_{B<D} S$ se lo tenemos en el inciso c) de la misma figura. Como podemos ver, las columnas tienen nombres por lo que la operación de join la podemos expresar, en este caso, con el nombre de sus columnas , y así tenemos $R \triangleright\!\!\triangleleft_{B<D} S$ o bien, por el número de sus columnas $R \triangleright\!\!\triangleleft_{2<1} S$.

A	B	C
1	2	3
4	5	6
7	8	9

a) Relación R.

D	E
3	1
6	2

b) Relación S.

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

c) $R \triangleright\!\!\triangleleft_{B<D} S$

Figura 3.11. Ejemplo de la operación Join con el operador <.

4.- **Join Natural** . El Join Natural , $R \triangleright\!\!\triangleleft S$, se aplica solamente cuando R y S tienen columnas con nombres de atributos. Para calcular $R \triangleright\!\!\triangleleft S$, se procede como sigue :

a) Calcular R x S

b) Para cada atributo A , que es el nombre de una columna de R y otra columna en S, seleccionada de R x S, en cuyos tuplos corresponden valores en las columnas R.A y S.A . Decimos que R.A es el nombre de la columna de R x S que corresponde a la columna A de R, y S.A se define análogamente.

c) Para cada atributo A mencionado anteriormente , se proyecta la columna S.A .

Formalmente, tenemos que , si $A_1 , A_2 , \ldots A_k$ son todos los nombres de los atributos usados tanto para R como para S, $R \triangleright\!\!\triangleleft S$ es entonces

$$\pi_{i1,i2,\ldots,im}\ \sigma_{R.A1 = S.A1 \wedge \ldots \wedge R.Ak = S.Ak} (R \times S)$$

donde $i_1 , i_2 , \ldots , i_m$ es la lista de todos los componentes de R x S , en el orden , excepto los componentes $S.A_1 , \ldots , S.A_k$.

Veamos ahora los ejemplos. En la figura 3.11, tenemos las relaciones R y S. Entonces R ⊳⊲ S permanece para $\pi_{A,R.B,R.C,D}\ \sigma_{R.B=S.B\ \wedge\ R.C=S.C}$ (R X S) . Para construir R ⊳⊲ S, consideramos cada tuplo en R para ver cuál tuplo de S corresponde en ambas columnas B y C. Por ejemplo, *abc* en R corresponde con *bcd* y *bce* en S, así obtenemos *abcd* y *abce* en R ⊳⊲ S. De la misma manera, *dbc* nos da *dbcd* y *dbce* para R ⊳⊲ S. El tuplo *ddf* no corresponde con ningún tuplo de S en las columnas B y C, así que no obtenemos ningún tuplo en R⊳⊲ S que empiece con *bbf*. Finalmente, *cad* coteja con *adb* , así obtenemos el tuplo *cadb*.

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

a) Relación R.

B	C	D
b	c	d
b	c	e
a	d	b

b) Relación S.

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

c) R ⊳⊲ S

Figura 3.12. Operación Join Natural de las relaciones R y S.

NORMALIZACIÓN

Los campos de datos pueden ser agrupados en registros (tuplos) . Existen muchos caminos diferentes por los cuales cientos o miles de campos pueden ser agrupados, y algunos caminos son mejores que otros.

La mayoría de las bases de datos cambian constantemente, frecuentemente se agregan nuevos campos y nuevas asociaciones entre los campos y surgen nuevos patrones de uso. A medida que cambiemos la base de datos, debemos preservar para el usuario antiguo la vista de los datos, así como evitar que se tengan que reescribir los programas. Sin embargo hay ciertos cambios en las asociaciones o uso de los datos los cuales podrían forzar la modificación de los programas. Por ejemplo nosotros podemos dividir un registro en dos, o cambiar la llave que se usa para ciertos campos; tales cambios pueden ser extremadamente desorganizadores. Si el agrupamiento de los campos de datos y llaves se piensa bien originalmente, es improbable que podamos hacer tal desorganización.

La *normalización* es un método de organización de los datos en una base de datos, donde se descomponen los datos en unidades relacionadas no redundantes pequeñas y simples, las cuales contienen datos funcionalmente relacionados. Es importante hacer notar que la normalización describe la representación lógica de los datos, no la física. Hay muchas maneras de implementación física de los datos.

La normalización es una vía formal, la cual examina al mismo tiempo datos y grupos de campos en una forma que es la más apta para acomodar futuros cambios de la compañía, y para minimizar el impacto de ese cambio en los sistemas de aplicación. Las ventajas que resultan de la normalización de los datos son :

- El almacenamiento de la base de datos requiere de un mínimo de espacio, ya que la eliminación de datos redundantes elimina también la necesidad de espacio que los datos redundantes pudieran haber requerido.

- El mantenimiento de los datos es rápido y fácil, ya que los campos cuando son actualizados necesitan ser actualizados solamente en un lugar, no en muchos, debido a la ausencia de redundancia.

- La inconsistencia de los datos se evita, porque en la mayoría de los casos hay solamente una copia de los datos, dado que no existen copias múltiples de los datos con valores que difieran .

- La circulación de los datos está asegurada; como no hay copias múltiples de datos inconsistentes, no hay duda que la copia es la única en uso.

- El manejo de la base de datos se facilita , ya que el agrupamiento funcional de datos iguales en un lugar, más bien que en varios lugares , elimina la reestructuración principal de la base de datos cuando se agregan o suprimen los campos o archivos.

Los datos existen en la vida real como grupos de campos : existen en facturas, en formas de declaración de impuesto, en licencias de manejo, en recibos, etc. Estos agrupamientos a menudo no están en una forma normalizada, por lo que nos pueden llevar a diversos problemas sutiles en el futuro.

Llamamos *forma normal* a una relación que satisface cierto conjunto de restricciones; por ejemplo, se dice que una relación está en primera forma normal , abreviada 1FN, si y sólo si contiene valores atómicos, es decir , no contiene valores repetidos.

Existen hasta cinco formas normales : 1FN, 2FN, 3FN, 4FN, FNBC (forma normal de Boyce/Codd ó 3FN nueva) y FN/RN (forma normal de proyección-reunión ó 5FN). En la siguiente figura podemos apreciar que todas las relaciones en FN/PR (5FN) están en 4FN, todas las relaciones en 4FN están en 3FN, todas las relaciones en 3FN están en 2FN, y así sucesivamente; por lo que tenemos que, si hay una relación en 3FN, debe de estar en 2FN y 1FN .

Como se puede observar en la figura 3.13, las primeras formas formales contienen a la siguiente; donde cada una de ellas es un "refinamiento" particularización de la anterior.

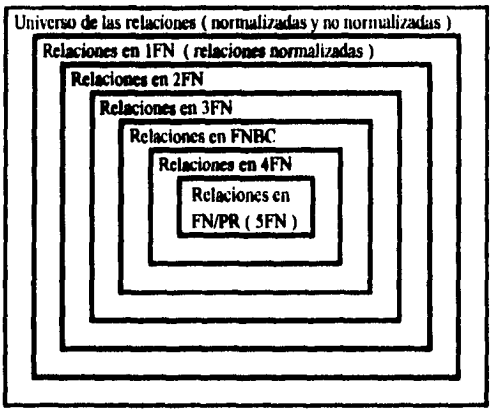


Figura 3.13. Formas Normales.[1]

La experiencia ha demostrado que cuando los datos que ya están organizados en la tercera forma normal, resultan datos estructurados que son más estables y fáciles de cambiar. Cada atributo se relaciona con su propia entidad y no se mezcla con atributos relacionados a diferentes entidades. Las acciones que crean y actualizan los datos pueden entonces ser aplicadas con un diseño estructurado simple, una sola vez, en un registro normalizado.

La dependencia funcional trata con la identificación del valor de un campo con otro campo o llave. El campo B es funcionalmente dependiente del campo A si y solamente si para cualquier valor A existe solamente un valor B .

[1] DATE, C. J. *Introducción a los Sistemas de Bases de Datos.* Addison Wesley Iberoamericana, S. A. 1989.

El campo A también es conocido como el determinante, ya que la llave primaria identifica unívocamente un registro, la llave primaria es el determinante de todos los campos en este registro.

El propósito es llevar a la base de datos a la tercera forma normal. Explicamos a continuación las características de las formas normales que nos interesan.

PRIMERA FORMA NORMAL

La primera forma normal se refiere a una colección de datos organizados en registros los cuales no tiene grupos repetidos de campos de datos dentro del registro. En general un archivo no plano en la primera formalización se convierte en 2 o más archivos planos o matrices de dos dimensiones. El archivo plano puede pensarse como una tabla de dos dimensiones, conteniendo miles de registros.

TRABAJADOR

RFC-TRABAJADOR	NOMBRE-TRABAJADOR	IMSS-TRABAJADOR	EXPEDIENTE-EMPRESA	NOMBRE-EMPRESA	DIRECCION-EMPRESA	CLAVE-ENTIDAD	NOMBRE-ENTIDAD	FECHA ALTA	FECHA BAJA
RFC1	PEREZ	IMSS1	EXPED1	NOMB1	DIR1	COD1	EDO1	FA1	FB1
			EXPED2	NOMB2	DIR2	COD2	EDO2	FA2	FB2
			EXPED3	NOMB3	DIR3	COD3	EDO3	FA3	FB3
			EXPED4	NOMB4	DIR4	COD4	EDO4	FA4	FB4

Figura 3.14. Tabla de la relación TRABAJADOR, con campos repetidos.

En la figura 3.14, tenemos al trabajador PEREZ que ha trabajado en cuatro empresas por lo que hay cuatro registros en el archivo o tabla TRABAJADOR, como se puede observar, al efectuar la primera forma normal obtenemos los archivos (tablas) planos TRABAJADOR y EMPRESA-TRABAJADOR, como se puede observar en la figura 3.15, los rectángulos sombreados son las llaves primarias (simples o compuestas) de los archivos y las flechas señalan la dependencia funcional. El archivo TRABAJADOR tiene como llave primaria a RFC-TRABAJADOR , y ésta determina a NOMBRE-TRABAJADOR. En el archivo EMPRESA-TRABAJADOR la llave primaria compuesta tiene los campos RFC-TRABAJADOR y EXPEDIENTE-EMPRESA, la cual determina a NOMBRE-EMPRESA, DIRECCION-EMPRESA, CODIGO-ESTADO, NOMBRE-ESTADO,

FECHA-ALTA y FECHA-BAJA, además, el campo CODIGO-ESTADO determina a NOMBRE-ESTADO.

TRABAJADOR

EMPRESA-TRABAJADOR

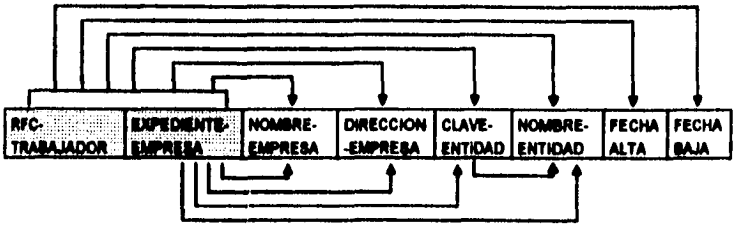


Figura 3.15. Primera Forma Normal, sin campos repetidos.

SEGUNDA FORMA NORMAL

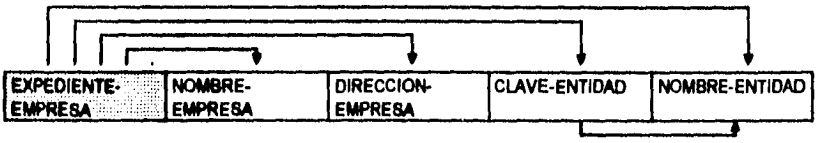
La dependencia funcional trata con la identificación del valor de un campo con otro campo o llave. Sea R un registro, A y B campos de R. Se dice que B es funcionalmente dependiente de A si y sólo si para cada valor de A existe un valor B.

Sea B un campo o una colección de campos del registro r, y A otra colección de campos del registro r , se dice que B es funcionalmente dependiente por completo de A si y sólo si B es funcionalmente dependiente de A pero no de un subconjunto de A.

Un registro r está en 2FN si y sólo si está en 1FN y cada campo , que no es llave de R es funcionalmente dependiente por completo de la llave candidata de R.

Es decir , que cuando se pasa de la 1FN a 2FN, las relaciones se transforman en relaciones más pequeñas para eliminar los atributos no dependientes de la llave primaria compuesta, es decir aquellos atributos que no dependen totalmente de la llave. En la figura 3.15, vemos que EMPRESA-TRABAJADOR tiene campos que son funcionalmente dependientes de un campo que forma parte de la llave primaria compuesta, esto se eliminará al realizar la 2FN, como se puede apreciar en la figura 3.16, al separa ese archivo en dos archivos EMPRESA y EMPRESA-TRABAJADOR.

EMPRESA



EMPRESA-TRABAJADOR

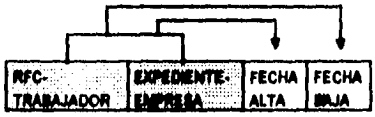


Figura 3.16. Segunda Forma Normal. Eliminación de los atributos que no dependen de una llave primaria compuesta.

TERCERA FORMA NORMAL

La tercera forma normal es una agrupación simple, relativamente estable de campos de datos en registros o tuplos. Su propósito es encontrar aquellos agrupamientos de campos de datos destinados, en menor medida, a dar problemas de mantenimiento, los menos adecuados para cambiar de manera que fuerce a reescribir el programa de aplicación.

La simplicidad básica de los datos en la tercera forma normal hace a los registros fáciles de entender y fáciles de cambiar, lo que no sucede en otras

maneras de organización de una base de datos. Evita algunas anomalías que surgen con otras estructuras de datos. Usualmente, esta forma de diseño es mejor en términos de requerimientos de la máquina así como en la estructura lógica, pero este no es siempre el caso. Algunas veces el diseñador físico encuentra deseable desviarse de la tercera forma normal . ¿ Qué es preferible : un poco mejor funcionamiento de la máquina o mejor protección de costos de mantenimiento ? Usualmente, el potencial de costos de mantenimiento son mas costosos.

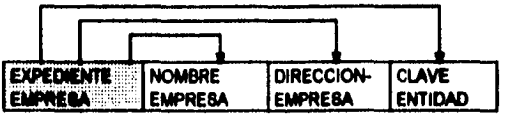
Para poner los datos en la tercera forma normal se debe de haber efectuado la primera y la segunda forma normal.

Dependencia transitiva. Sean A, B y C campos ó colecciones de campos del registro r. Si A $\rightarrow$ B y B $\rightarrow$ C entonces A $\rightarrow$ C, C es transitivamente dependiente de A.

Un registro que está en 2FN puede tener un campo que no es llave pero que identifica a otros campos. Esta dependencia transitiva puede causar problemas. El paso de colocar los datos en 3FN remueve la dependencia transitiva.

Un registro R está en 3FN si está en 2FN y cada campo que no es llave de R, no es transitivamente dependiente en cada llave candidata de R. Es decir, en la 3FN se elimina la dependencia transitiva, como se puede observar en la figura 3.17, donde el archivo EMPRESA tiene una dependencia transitiva que es el campo de ENTIDAD, por lo que el archivo quedará dividido en dos archivos EMPRESA y ENTIDAD.

EMPRESA



ENTIDAD

Figura 13.17. Tercera Forma Normal. Eliminación de la dependencia transitiva.

CAPÍTULO 4

ANÁLISIS Y DISEÑO DE LA INFORMACIÓN

El registro de los créditos se hace en el momento en que es aceptada la solicitud del trabajador, y el registro de los abonos se hace cuando se recibe la información de parte de la empresa o cuando el acreditado paga personalmente.

A continuación tenemos la descripción de la información contenida en los abonos, para la amortización del crédito otorgado al trabajador .

Existen diferentes tipos de abonos para el pago efectuado a través de la empresa (ROA); el período de los abonos está dado por bimestre-año:

- ANEXO-B .- Contiene el período, la fecha de pago, y el monto del abono . Este tipo de abono abarca del año 1973 al 1er bimestre de 1982.

- TALÓN BIMESTRAL .- Contiene el período, la fecha de pago y el monto del abono. Este tipo de abono abarca del 2do bimestre de 1982 al 6o. bimestre de 1992.

- TALÓN DE LIBERACIÓN A LA RETENCIÓN .- Con este talón la empresa avisa a INFONAVIT que el acreditado ya no tiene ninguna relación con la empresa por alguna razón (baja, fallecimiento, invalidez temporal o permanente, etc.) , por lo que ya la empresa no tiene ninguna obligación con INFONAVIT de la retención del abono de ese acreditado.

Cuando el pago de los abonos los efectúa personalmente el acreditado (REA) , el período está dado en mes-año , y los tipos de abonos son :

- PAGO MENSUAL .- Contiene el período, la fecha de pago y el monto del abono.

- TALÓN MENSUAL .- Contiene el período, la fecha de pago, el monto del abono , la duración (número de meses que abarca un pago múltiple) y pago extraordinario , este último sólo cuando se hace un pago extraordinario, de liquidación, atrasado ó múltiple.

Cuando el trabajador deja de laborar en una empresa y tiene un crédito otorgado por INFONAVIT, él tiene la obligación de notificar su baja ante éste último, y de requerirlo el trabajador puede solicitar una prórroga, para no amortizar su crédito hasta por 12 meses. Esto puede hacerse una sola vez, durante el período que dure el crédito.

PRORROGA .- Es la que el trabajador tiene cuando no tiene trabajo , y lo solicita ante INFONAVIT.

Cuando esta información requiera de algún cambio, únicamente se actualiza la información y se pone una marca que indique que ya hubo una modificación en la información. Para efectuar una baja se pone una marca de baja, únicamente para conservar por un tiempo razonable esta información, y al término de la misma darla de baja físicamente; no es conveniente tener información, que ya está marcada como baja, almacenada durante toda la vida del sistema , ya que estaría ocupando área que se podría usar en otras cosas.

Como solución para abatir la cartera vencida, el Instituto propuso, como se mencionó en el primer capítulo, que a los derechohabientes con créditos otorgados en monetario desde 1973 a 1987 se les pediría la liquidación total de su adeudo. Para los créditos en vsm se establecería un convenio para el pago en parcialidades de los adeudos vencidos. Para los créditos que actualmente se están otorgando es necesario detectar mensualmente si hay omisión de pago durante dos meses (un bimestre en ROA, dos meses en REA), de ser así se deberá emitir un aviso , así como su estado de cuenta al trabajador, notificándolo de su retraso. Para poder identificar cada uno de estos casos y hacer llegar la propuesta al derechohabiente, debemos contar con una base de datos que contenga al menos, la siguiente información :

Para los *trabajadores* lo que se requiere es :

RFC del Trabajador	: como un identificador
Nombre del Trabajador	: en el orden apellido paterno, apellido materno y nombre (s)
Número del IMSS	: adicional para su identificación

y la información de la *empresa* donde trabaja es :

Expediente de la empresa	: clave interna de la Empresa, proporcionada por INFONAVIT
Nombre de la empresa	: nombre o razón social de la empresa
Dirección de la empresa	
Código ó número de Estado	: clave de la entidad federativa
Estado	: estado donde se ubica la empresa
Giro de la empresa	
RFC de la empresa	: Registro Federal de Causantes
Fecha de alta del trabajador	: fecha en que se dio de alta el trabajador
Fecha de baja del trabajador	: fecha en que se dio de baja el trabajador

La información del *crédito otorgado* es :

Número de crédito	: asignado en el momento de otorgar el crédito
RFC del trabajador	: Registro Federal de Causantes del trabajador
Expediente de la emprese	: expediente de la empresa cuando le fue otorgedo el crédito al trabajador
Monto del crédito en VSM	: monto expresado en veces salerio mínimo del D.F.
Monto del crédito en moneterio	: monto expresado en nuevos pesos
Impuestos	: impuestos del crédito
Fecha del crédito	: fecha en que se otorgó el crédito
SMGM	: salario mínimo general mensual del D.F. e la fecha de otorgado el crédito, dado que éste es variable .

Le información de los *abonos* es :

Número de crédito	: es número que le fué asignado en su crédito
Expediente de la empresa	: clave de la empresa que efectúa el abono (sólo en caso de pago bimestral)
Monto del abono	: monto de dinero para amortizar el crédito
Período del abono	: año y bimestre (o mes, si es mensual)
Marca del abono	: para indicar si es una baja o hubo modificación

NORMALIZACIÓN

La información para el trabajador la podemos poner en forma de una tabla, la cual llamaremos **TRABAJADOR**, como se muestra en la figura 4.1:

TRABAJADOR

RFC-TRABAJADOR	NOMBRE-TRABAJADOR	MSS-TRABAJADOR	EXPEDIENTE-EMPRESA	NOMBRE-EMPRESA	DIRECCION-EMPRESA	CLAVE-ENTIDAD	NOMBRE-ENTIDAD	FECHA ALTA	FECHA BAJA

Figura 4.1. Tabla de la relación **TRABAJADOR**.

El trabajador CRUZ trabajó en las empresas E1, E2, E3, E4 en un lapso de 4 años, por lo que tenemos la siguiente información , ver figura 4.2:

RFC1	CRUZ	EXP1	E1	DIR1	01	HERMOSILLO	010287	310887
		EXP2	E2	DIR2	01	HERMOSILLO	010987	300688
		EXP3	E3	DIR3	01	HERMOSILLO	010788	310789
		EXP4	E4	DIR4	01	HERMOSILLO	100889	310791

Figura 4.2. Registro sin formalizar, con campos repetidos.

Podemos observar que el trabajador CRUZ ha estado en cuatro empresas diferentes. La información como está, presenta redundancia y repeticiones. Para cumplir con las características de un sistema de bases de datos , debemos buscar la no redundancia, por lo que esta tabla la debemos convertir en una tabla plana empleando la primera formalización (1FN), para quitar los campos repetidos.

Al emplear la primera Forma Normal el archivo TRABAJADOR se convierte en dos archivos planos, TRABAJADOR y EMPRESA-TRABAJADOR, como lo muestra la figura 4.3.

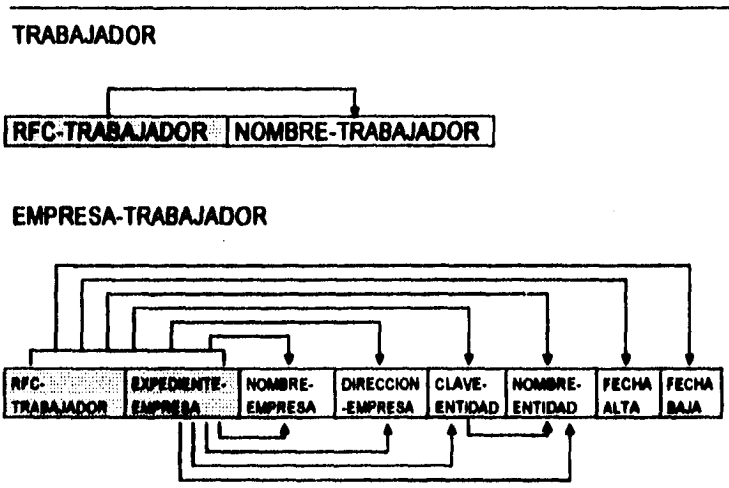


Figura 4.3. Primera Forma Normal, eliminación de campos repetidos.

Podemos observar aquí la dependencia funcional de algunos campos, en el registro de TRABAJADOR :

RFC-TRABAJADOR	depende de	NOMBRE-TRABAJADOR
NOMBRE-TRABAJADOR	depende de	RFC-TRABAJADOR

Para el registro EMPRESA-TRABAJADOR la dependencia funcional es la siguiente :

NOMBRE-EMPRESA	depende de	RFC-TRABAJADOR+EXPEDIENTE-EMPRESA o EXPEDIENTE-EMPRESA
DIRECCION-EMPRESA	depende de	RFC-TRABAJADOR+EXPEDIENTE-EMPRESA o EXPEDIENTE-EMPRESA
CLAVE-ENTIDAD	depende de	RFC-TRABAJADOR+EXPEDIENTE-EMPRESA o EXPEDIENTE-EMPRESA
NOMBRE-ENTIDAD	depende de	RFC-TRABAJADOR+EXPEDIENTE-EMPRESA o EXPEDIENTE-EMPRESA
FECHA-ALTA	depende de	RFC-TRABAJADOR+EXPEDIENTE-EMPRESA o EXPEDIENTE-EMPRESA
FECHA-BAJA	depende de	RFC-TRABAJADOR+EXPEDIENTE-EMPRESA o EXPEDIENTE-EMPRESA

Aquí se puede ver que la llave compuesta es

RFC-TRABAJADOR+EXPEDIENTE-EMPRESA .

En general un registro en la 1ª Forma Normal se convierte en dos ó más registros planos.

Para poder saber el nombre de la empresa, la fecha de alta, la fecha de baja y la entidad, necesitamos conocer tanto la clave de trabajador como el expediente de la empresa, éstos dos campos forman la llave compuesta para identificar la información del trabajador en una empresa dada.

Para la 2ª Forma Normal, debemos de quitar los atributos que no dependen de una llave compuesta, como en el registro **EMPRESA-TRABAJADOR** anterior. Únicamente **FECHA-ALTA** y **FECHA-BAJA** son funcionalmente dependientes por completo de la llave compuesta **RFC-TRABAJADOR + EXPEDIENTE-EMPRESA**, por lo que las separamos del registro, y los campos **NOMBRE-EMPRESA**, **DIRECCION-EMPRESA**, **CLAVE-ENTIDAD** y **NOMBRE-ENTIDAD** dependen de un solo campo de la llave, **EXPEDIENTE-EMPRESA**. En la figura 4.4, tenemos el resultado de la segunda Forma Normal.

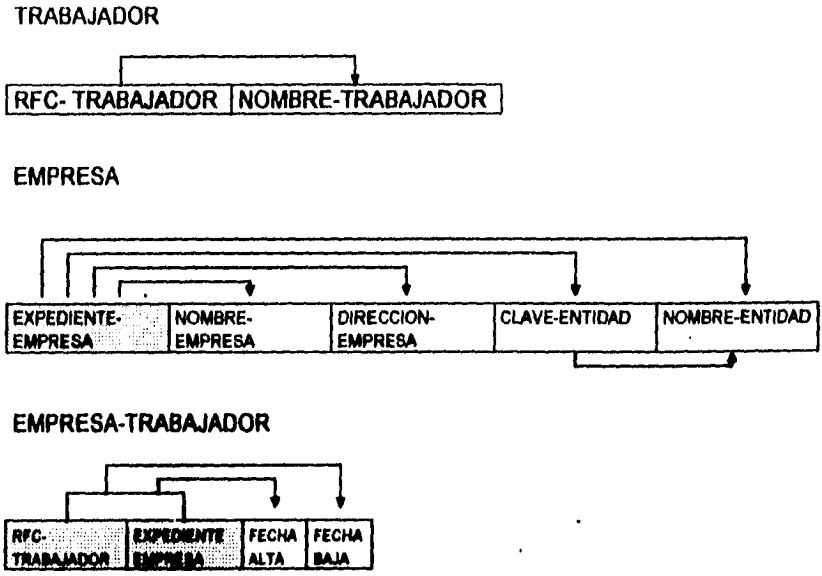
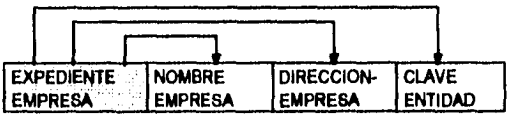


Figura 4.4. Segunda Forma Normal. Eliminación de los atributos que no dependen de una llave primaria compuesta.

Cuando tenemos que no es una llave, pero que identifica a otros campos, entonces estamos hablando de dependencia transitiva. Dado que esto nos crea problemas entonces debemos emplear la 3ª Forma Normal para quitar esta dependencia transitiva. Esta dependencia transitiva la encontramos en el registro EMPRESA , porque el campo NOMBRE-ENTIDAD depende del campo CLAVE-ENTIDAD. Entonces al efectuarse la 3ª Forma Normal el registro EMPRESA lo separamos en dos registros que son EMPRESA y ENTIDAD , como se puede ver en la figura 4.5.

TRABAJADOR

EMPRESA



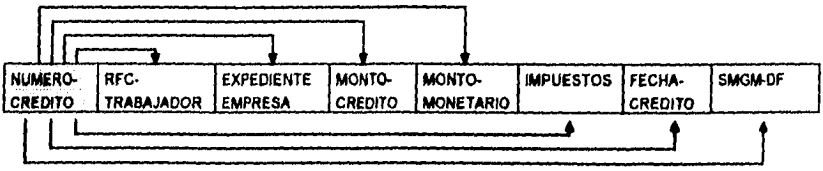
EMPRESA-TRABAJADOR

ENTIDAD

Figura 4.5. Tercera Forma Normal. Eliminación de la dependencia transitiva.

En la figura 4.5 tenemos cuatro archivos que son parte de nuestra base de datos , los cuales requiriero de la normalización, en la figura 4.6 vemos los otros dos archivos CREDITO y ABONOS, que no requirieron de la normalización. De esta manera tenemos seis archivos para nuestra base de datos del Sistema de Facilidades : TRABAJADOR, EMPRESA, EMPRESA-TRABAJADOR, ENTIDAD CREDITO Y ABONOS, las lleves se encuentran con los rectángulos sombreados.

CREDITO



ABONOS

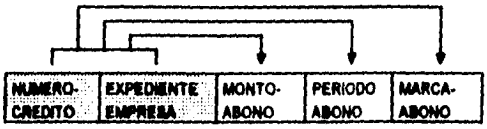


Figura 4.6. Archivos de CREDITO y ABONOS, los cuales complementan nuestra base de datos.

DIAGRAMAS

DIAGRAMA DE CONTEXTO
(DIAGRAMA 0)

El objetivo del Sistema de Facilidades es proporcionar información de los acreditados morosos en su omisión de pagos desde 2 meses o más con respecto de su último pago.

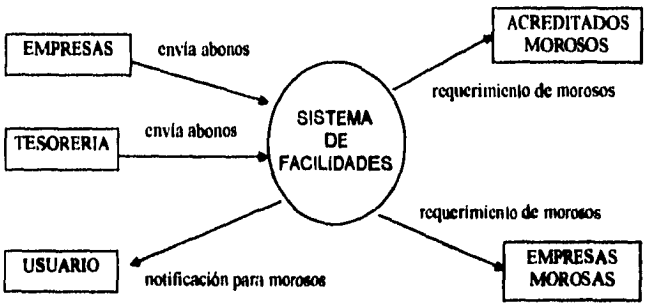
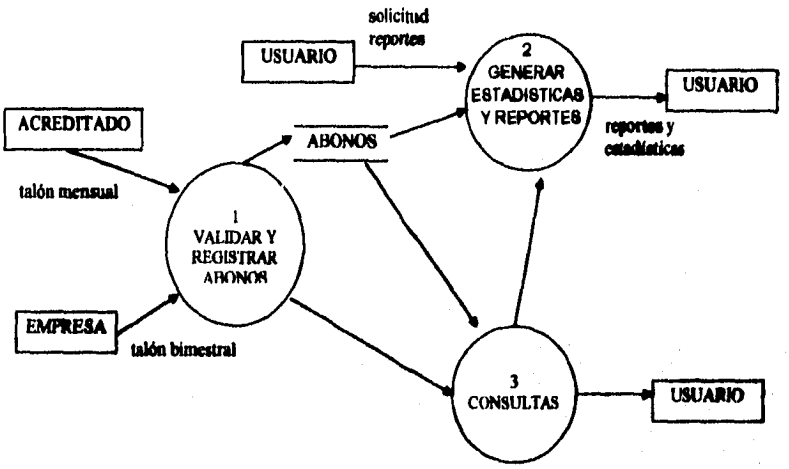
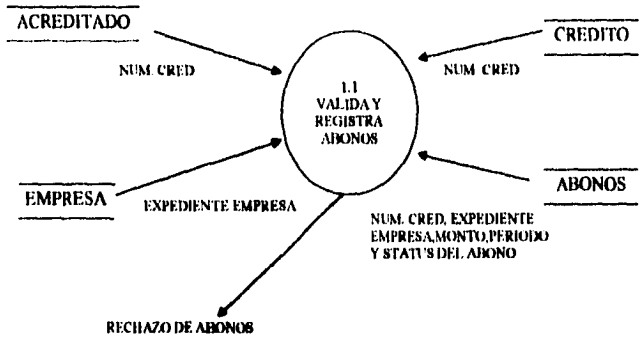


DIAGRAMA UNO

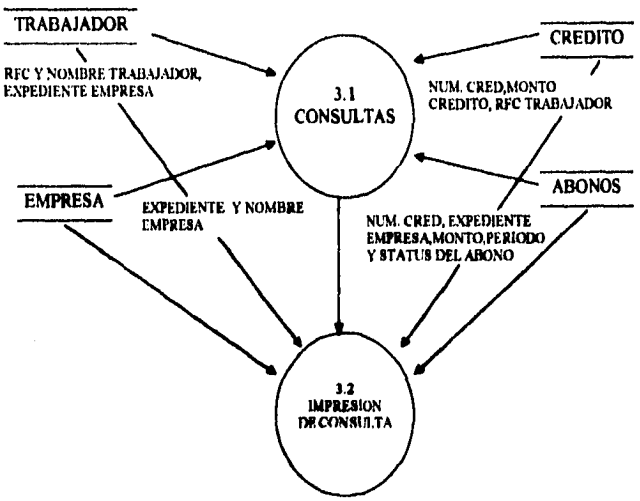


VALIDAR Y REGISTRAR ABONOS



GENERAR ESTADISTICAS Y REPORTES

CONSULTAS


TRABAJADOR
CREDITO
RFC Y NOMBRE TRABAJADOR, EXPEDIENTE EMPRESA
3.1 CONSULTAS
NUM. CRED,MONTO CREDITO, RFC TRABAJADOR
EMPRESA
ABONOS
EXPEDIENTE Y NOMBRE EMPRESA
NUM. CRED, EXPEDIENTE EMPRESA,MONTO,PERIODO Y STATUS DEL ABONO
3.2 IMPRESION DE CONSULTA

CONCLUSIONES

Si bien el sistema desarrollado no es muy complicado, su análisis y diseño se convirtió en una tarea difícil en el ambiente de desarrollo del Instituto. Actualmente está en un período en que el hardware es muy sofisticado, mientras que el software no ha evolucionado al igual que el hardware para poder explotar todo su potencial. Esto es debido a que nuestra capacidad para construir software de calidad se ve limitada por diversos factores a nivel de programadores, ejecutivos y usuarios, los cuales son:

- Una falta de capacitación al personal que hace los sistemas, así como también a los que elaboran los programas. Esta capacitación es tanto a nivel de lenguajes de programación como en el análisis y diseño de sistemas.

Y aunque hubiere algún tipo de capacitación, ésta no se hace con un plan definido, en donde se tengan objetivos y metas a alcanzar, a corto, mediano y largo plazo. No existe un equipo de trabajo organizado y capacitado para hacer el análisis, diseño e implantación del sistema o de los sistemas.

- Se cree erróneamente que únicamente con tener el último modelo de computadora o de PC, se puede desarrollar software de calidad. Las herramientas de ingeniería de software asistida por computadora (CASE) son más importantes que el hardware más sofisticado que exista.

- Se piensa que si no se hizo un buen diseño, se pueden añadir más programadores para que no se atrase más el proyecto. Pero no se toma en cuenta que la gente nueva que se incorpore deberá aprender acerca de la información que se está manejando y deberá tener una buena comunicación con su equipo de trabajo, lo cual requiere de tiempo.

- El usuario al solicitar el desarrollo de un sistema o el requerimiento de cierta información , piensa que con unas cuantas especificaciones a nivel global, dadas por él, se podrá atender su solicitud. Pero es importante que el usuario sepa que para que se desarrolle el software es fundamental hacer una descripción formal y detallada del ámbito de la información, así como las funciones, el rendimiento, las interfaces y los criterios de validación. Además, esto requiere tiempo. Al existir las presiones por parte del usuario para la realización de su proyecto, se cae en errores por falta de una planeación, lo que repercute en la deficiencia del análisis y el diseño.

Dado que el sistema cambia continuamente, el cliente cree que si existe algún cambio éste se puede acomodar fácilmente, esto únicamente se logra si desde el principio se tuvo cuidado en la definición del sistema . Cuando los cambios se

CONCLUSIONES

Si bien el sistema desarrollado no es muy complicado, su análisis y diseño se convirtió en una tarea difícil en el ambiente de desarrollo del Instituto. Actualmente está en un período en que el hardware es muy sofisticado, mientras que el software no ha evolucionado al igual que el hardware para poder explotar todo su potencial. Esto es debido a que nuestra capacidad para construir software de calidad se ve limitada por diversos factores a nivel de programadores, ejecutivos y usuarios, los cuales son:

- Una falta de capacitación al personal que hace los sistemas, así como también a los que elaboran los programas. Esta capacitación es tanto a nivel de lenguajes de programación como en el análisis y diseño de sistemas.

Y aunque hubiere algún tipo de capacitación, ésta no se hace con un plan definido, en donde se tengan objetivos y metas a alcanzar, a corto, mediano y largo plazo. No existe un equipo de trabajo organizado y capacitado para hacer el análisis, diseño e implantación del sistema o de los sistemas.

- Se cree erróneamente que únicamente con tener el último modelo de computadora o de PC, se puede desarrollar software de calidad. Las herramientas de ingeniería de software asistida por computadora (CASE) son más importantes que el hardware más sofisticado que exista.

- Se piensa que si no se hizo un buen diseño, se pueden añadir más programadores para que no se atrase más el proyecto. Pero no se toma en cuenta que la gente nueva que se incorpore deberá aprender acerca de la información que se está manejando y deberá tener una buena comunicación con su equipo de trabajo, lo cual requiere de tiempo.

- El usuario al solicitar el desarrollo de un sistema o el requerimiento de cierta información , piensa que con unas cuantas especificaciones a nivel global, dadas por él, se podrá atender su solicitud. Pero es importante que el usuario sepa que para que se desarrolle el software es fundamental hacer una descripción formal y detallada del ámbito de la información, así como las funciones, el rendimiento, las interfaces y los criterios de validación. Además, esto requiere tiempo. Al existir las presiones por parte del usuario para la realización de su proyecto, se cae en errores por falta de una planeación, lo que repercute en la deficiencia del análisis y el diseño.

Dado que el sistema cambia continuamente, el cliente cree que si existe algún cambio éste se puede acomodar fácilmente, esto únicamente se logra si desde el principio se tuvo cuidado en la definición del sistema . Cuando los cambios se

solicitan durante el diseño del software se pueden producir muchos trastornos, lo que significa dinero y tiempo adicionales.

Una buena calidad del software se adquiere mediante un buen análisis y diseño. Además la documentación es la base de un buen desarrollo ya que proporciona las guías para su posterior mantenimiento.

Existen malos hábitos que dan como resultado una pobre calidad y mantenimiento del software. Pero la gente encargada del desarrollo del software normalmente se resiste al cambio comenzando por los ejecutivos, posiblemente se deba a las altas presiones a las que éstos están sometidos para que a los usuarios se les entregue lo antes posible sus requerimientos.

Es importante que las personas que realizan los programas, así como los que hacen el análisis y a los usuarios se les informe que existen métodos y herramientas para que se haga un buen desarrollo del software, y lo más importante es que se ponga en práctica.

BIBLIOGRAFÍA

1. ADAD Rubén, MEDINA Miguel A., CAREAGA Alfredo , *Fundamentos de las Estructuras de Datos Relacionales.* Editorial Megabyte. Grupo Noriega Editores. 1992.

2. CAPRON, H. L. , System Analysis and Design. The Benjamin/Cummings Publishing Company, Inc. 1986.

3. DATE, C. J. , *Introducción a los Sistemas de Bases de datos.* Addison-Wesley Iberoamericana. 3ª Edición. 1989.

4. GILLENSON, Mark L. , *Introducción a las Bases de Datos.* Mc Graw Hill. 1985.

5. HOWE, D. R. , *Data Analysis for Data Base Design.* Edward Arnold (Publishers) Ltd. 1983.

6. MEDINA Miguel A., ADAD Rubén, CAREAGA Alfredo , *Fundamentos de las Estructuras de Datos Relacionales.* Editorial Megabyte. Grupo Noriega Editores. 1992.

7. MARTIN, James , *Organización de las Bases de Datos.* Prentice-Hall Hispanoamericana, S. A. 1ª Edición en español, México. 1985.

8. PRESSMAN, Roger S. , *Ingeniería del Software. Un enfoque práctico.* Mc Graw Hill. 3ª Edición en español.1993.

9. TALAVERA, J. Carlos, *Bases de Datos, concepto, diseño e implantación.* México. ITAM. 1993.

10. TSAI, Alice Y. H. , *Sistemas de Base de Datos: Administración y uso.* Prentice-Hall Hispanoamericana, S. A. 1ª Edición en español, México. 1990.

11. ULLMAN, Jeffrey D. *Principles of Database Systems.* Computer Software Enginneering Series. Computer Science Press, Inc. 2ª Edición. 1982.

12. ULLMAN, Jeffrey D. *Principles of Database and Knowledge - Base Systems. Volume I.* Principles of Computer Science Series. Computer Science Press, Inc. .

13. WIEDERHOLD, Gio , *Diseño de Bases de Datos*. Mc Graw Hill. Traducido de la segunda edición en inglés DATA BASE DESIGN. 1987.