40

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
A R A G O N

DISEÑO DE UNA BASE DE DATOS RELACIONAL CON PROGRAMACION ORIENTADA A OBJETOS EN PARADOX PARA WINDOWS PARA EL CONTROL ESCOLAR DE UNA ESCUELA DE COMPUTACION

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N:

EUSEBIO MARTINEZ PEÑA
GERARDO GONZALEZ HERNANDEZ

ASESOR: ING. ROBERTO BLANCO BAUTISTA

México, D.F. 1996

TESIS CON
FALLA DE ORIGEN

Diseño de una Base de Datos Relacional con Programación Orientada a Objetos en Paradox para Windows para el Control Escolar de una Escuela de Computación

Objetivo General:

Crear un sistema de computación con programación orientada a objetos, para el control escolar de una escuela de computación, con el fin de reutilizar el código realizado, aplicando una técnica de análisis y diseño orientada a objetos.

AGRADECIMIENTOS

A nuestros padres y hermanos por su apoyo siempre constante.

INTRODUCCIÓN

El presente trabajo, contiene información acerca de la Orientación a Objetos (OO), un tema actual dentro de lo que es la computación en México. Se explican conceptos teóricos para la comprensión del mismo, y trata también, el desarrollo de un sistema con programación OO, en base a la utilización de una técnica de análisis y diseño OO.

Con dicho trabajo, se pretende, dar a conocer los conceptos fundamentales de la Orientación a Objetos, así como, aplicar estos conceptos en forma práctica, desarrollando un Sistema de Control Escolar (información general del alumno y pagos de colegiatura del mismo) de una Escuela de Computación.

El aplicar dicha técnica orientada a objetos, permitirá crear un programa no secuencial, sino fragmentado (en partes), conteniendo cada una de ellas su propio código, lo que dará lugar a que no se dependa de un módulo del programa y que este, en caso de ser removido o modificado, no afecte a los demás. Asimismo, se podrá crear código reutilizable, es decir, que dichos códigos, podrán ser utilizados nuevamente, cuando se requieran. Por lo tanto, no existirá redundancia en la programación y será mucho más fácil poder realizar un nuevo programa, tomando partes del anterior.

El capítulo uno, contiene lo referente a los tipos de programación, como son la programación clásica o convencional, programación modular, programación estructurada y programación orientada a objetos, ventajas y desventajas. Se explican y detallan, por medio de pseudocódigos y diagramas, las estructuras de programación de cada uno de dichos tipos. Los conceptos básicos para entender lo que es una base de datos, sus características, objetivos y elementos se describen en el segundo capítulo.

En el capítulo tres, se describen los modelos de bases de datos que se usan con más frecuencia para el análisis de un sistema, algunos de éstos modelos son: Modelo lógico basado en registros, modelo jerárquico, modelo de red, modelo relacional y modelo entidad-

relación. Se explican, sus componentes, características y terminología, así como también, la forma de representar gráficamente (diagramas) su estructura.

El capítulo IV, trata sobre conceptos fundamentales de la orientación a objetos, como son: objetos, métodos, mensajes, clases, herencia, encapsulado. Se describen, las características de las técnicas orientadas a objetos y sus beneficios. También se encuentra una breve descripción de lo que son las bases de datos orientadas a objetos, su funcionalidad, elementos básicos y ventajas.

En el capítulo V, se explica detalladamente, una técnica de análisis y diseño orientado a objetos, conteniendo diagramas y ejemplos de los diferentes procesos. Cabe mencionar, que dicho capítulo, es la parte medular de la orientación a objetos, ya que aquí, se detalla y explica, todo el proceso a desarrollar, para analizar y diseñar un sistema. Dicha técnica, está compuesta de cuatro partes fundamentales, las cuales son: Análisis de la estructura y comportamiento de objetos, así como, diseño de la estructura y comportamiento de objetos.

Finalmente en el capítulo seis se lleva acabo el análisis y diseño orientado a objetos de una aplicación real, aplicando los conceptos del capítulo anterior.

ÍNDICE

CAPÍTULO I

CONCEPTOS DE PROGRAMACIÓN

Objetivo:

Investigar los tipos de programación existentes, para conocer sus características, ventajas y desventajas, describiendo cada uno de ellos.

I-. CONCEPTOS DE PROGRAMACIÓN

El realizar un programa requiere la utilización de un término llamado programación, éste significa indicar a la computadora una serie de instrucciones que debe ejecutar a través de un lenguaje de programación determinado para resolver una tarea específica.

Estas instrucciones surgieron a través de la necesidad de enfrentar problemas reales que se querían resolver por medio de la computadora con el fin de manejar la información en forma ordenada, rápida y precisa. Cada una de estas instrucciones se fueron desarrollando a través de las diversas etapas de programación que han surgido, dichas etapas son las siguientes :

- Programación Clásica o Convencional.
- Programación Modular.
- Programación Estructurada.
- Programación Orientada a Objetos.

1.1 PROGRAMACIÓN CLÁSICA O CONVENCIONAL

Dicha programación consiste en ejecutar una secuencia de instrucciones y de ramificaciones, usando elementos como :

- Constantes, variables, operadores.

Una Constante es un valor fijo que no cambia, una Variable es un valor cambiante, es decir, en un cierto momento tiene un valor y en otro uno distinto y un Operador es el símbolo que nos permite realizar operaciones matemáticas o lógicas como sumas, restas, comparaciones, etc...

- Expresiones.

Es el conjunto de constantes, variables y operadores que definen algún cálculo matemático o lógico y en éstos se pueden usar paréntesis para poder distinguir alguna operación respecto a otra.

- Sentencias de Asignación.

Son aquéllas expresiones a las cuales se les asigna una variable.

- Sentencias de Entrada y Salida.

Permite comunicarnos con el exterior (Usuarios, periféricos y la memoria principal de la computadora).

- Sentencias de Control.

Permiten efectuar una ruptura de secuencia de instrucciones en el programa, dentro de éstas se tienen las siguientes :

- Sentencia Condicional "If".

Sentencia condicional, transfiere el control del programa en varias direcciones.

- Sentencia Incondicional "Go To".

Transfiere el control del programa a otra parte de él.

- Sentencia Parada de un programa :

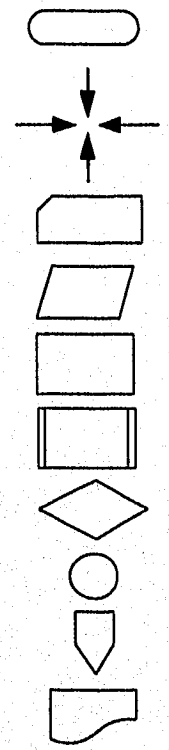
Permite detener momentáneamente la ejecución del programa o finalizarlo.

- Sentencia de Especificación.

Define la declaración de "arrays" y la asignación de tipos y variables.

Una forma de representar gráficamente el conjunto de instrucciones que se le indican a la computadora por medio del lenguaje de programación es a través del uso de Diagramas de Flujo, los cuales contienen una serie de símbolos universales para representar cierto tipo de operaciones en un programa, algunos de éstos son :

Simbología de Diagramas de Flujo



Inicio / Fin

Líneas de Flujo

Entrada de Datos

Presentación en Pantalla

Operación / Procedimiento

Procedimiento / Subrutina

Condicional

Conector misma página

Conector a otra página

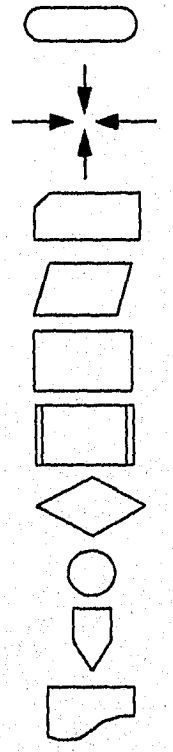
Imprimir

- Sentencia de Especificación.

Define la declaración de "arrays" y la asignación de tipos y variables.

Una forma de representar gráficamente el conjunto de instrucciones que se le indican a la computadora por medio del lenguaje de programación es a través del uso de Diagramas de Flujo, los cuales contienen una serie de símbolos universales para representar cierto tipo de operaciones en un programa, algunos de éstos son :

Simbología
de
Diagramas de Flujo



Inicio / Fin

Líneas de Flujo

Entrada de Datos

Presentación en Pantalla

Operación / Procedimiento

Procedimiento / Subrutina

Condicional

Conector misma página

Conector a otra página

Imprimir

1.2 PROGRAMACIÓN MODULAR

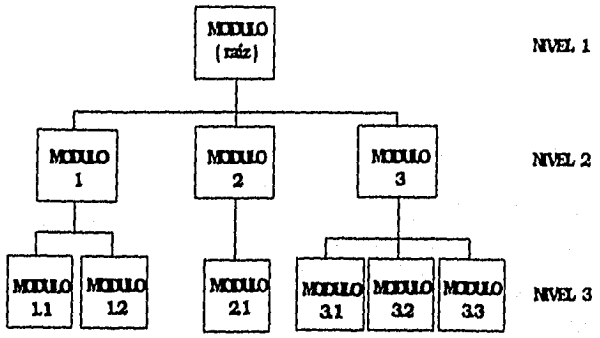
Durante la Programación Clásica se programaba secuencialmente y se utilizaba la expresión GO TO lo cual hacía que de una instrucción del programa se saltara a otra parte diferente de él pudiendo o no regresar a la instrucción que dio el salto, esto hacía que los programas no tuvieran un orden lógico de operación.

En esta etapa de la Programación se pueden realizar programas con mayor claridad y orden en el proceso de realización de rutinas para resolver un problema o situación real. Ya que este tipo de programación surge precisamente de la necesidad de estandarizar un programa que un cierto autor haya realizado y que cuando se requiera se pueda modificar por otra persona diferente sin que ésta última tenga problemas acerca de la lógica del mismo.

El programar en forma Modular significa dividir un programa en varios Módulos cada uno de los cuales es un procedimiento independiente que resuelve una parte de todo el algoritmo completo del problema. Cabe destacar que en este tipo de programación se requiere siempre de un Módulo Principal el cual mandará llamar a los demás , es decir, parte de una raíz y se extiende hacia abajo en forma de árbol .

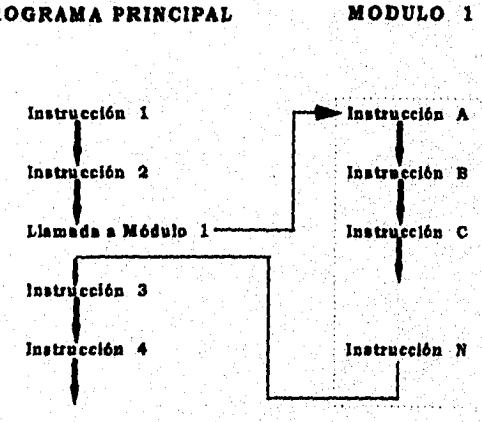
Un módulo se define como un conjunto formado por una o varias instrucciones. A cada módulo se le asigna un nombre para poder identificarlo. Cuando en alguna parte del programa se hace una llamada a un módulo, el programa cede el control para que se ejecuten todas sus instrucciones, una vez que ha terminado, el control se devuelve al lugar donde se llamó al módulo, y se continúa con la ejecución de la instrucción siguiente a la que realizó la llamada.

A continuación se muestra una representación gráfica del orden de ejecución :



Para el caso de que un módulo fuera demasiado grande se puede subdividir éste en submódulos siguiendo una secuencia de mantener éstos en el mismo nivel de ejecución. Si los módulos que componen el programa son llamados constantemente, se les llama rutinas, procedimientos o funciones. Y si fueran programas independientes llamados por otros programas se les denomina subprogramas.

En el siguiente diagrama ejemplifica el uso de módulos en un programa



La programación modular se basa en utilizar la técnica TOP-DOWN (Arriba - Abajo) que significa partir de lo general para llegar a lo particular, es decir, primero se realiza un análisis del programa indicando cada una de sus partes (módulos) de los que se compondrá y después se desarrollan cada uno de ellos.

Dicha técnica permite tener una mayor claridad en cada instrucción o módulo a ejecutar y también el poder tomar en cuenta módulos que se desarrollarán en un futuro, es decir, si dentro de un módulo se tiene una función cuyo desarrollo completo queremos posponer entonces lo dejamos indicado con el nombre de dicha función y después se desarrollaría.

Esta técnica en realidad se utiliza constantemente en cualquier actividad. Un ejemplo de ello es precisamente el desarrollo de una Tesis en la cual se construye un capitulado y después se desarrollan sus capítulos.

No existe una serie pasos a seguir para programar modularmente, sin embargo, se deben tomar en cuenta las siguientes normas generalizadas :

- Cada módulo sólo puede tener un punto de entrada y otro de salida.
- El módulo principal debe ser claro y debe indicar los módulos que lo componen así como su relación, es decir, indicar la solución completa del problema.
- Los módulos deben tener la máxima independencia entre ellos.

- Un módulo debe resolver una parte específica del problema con una estructura lógica bien definida.

1.3 PROGRAMACIÓN ESTRUCTURADA

La Programación Estructurada surgió debido a la necesidad de tener un mejor orden de seguimiento y análisis en la programación, para facilitar de esta forma el entendimiento, mantenimiento y elaboración de programas cosa que con la programación clásica y la modular no se tenia.

No se sabe con exactitud quien fue el precursor de esta técnica, alguna personas señalan a Allan Keller, mientras que los especialistas en el tema señalan a Dijkstra y próximo a él De Mills.

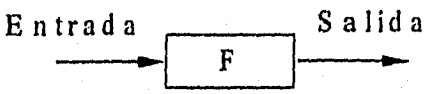
la Programación Estructurada esta basada en la programación modular y en el ***Teorema de la Estructura*** de Bohm y Jacopini el cual dice lo siguiente:

" Todo programa propio, es decir con un solo punto de entrada y un solo punto de salida, puede ser escrito utilizando únicamente tres tipos de estructuras de control: ***Estructura secuencial, Estructura condicional y Estructura repetitiva"***

La programación Estructurada se basa en los tres símbolos de diagramas que abajo se mencionan:

a) Tratamiento o "Procedimiento puro"

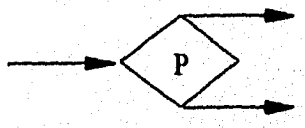
Permite representar una sentencia de asignación, una secuencia de sentencias, llamada a un subprograma, el cual solo tiene una entrada y una salida.



F = Función

b) Predicado o "Condición", o "Expresión Booleana"

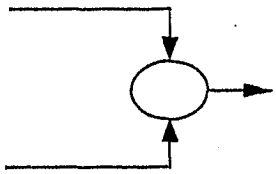
Nos da una alternativa de salida dependiendo del valor (Verdadero o falso) del dato de entrada.



P = Predicado

C) Reagrupamiento

Permite unir en una sola salida, diversas salidas de tratamientos y/o predicados.



1.3.1 DIAGRAMAS PRIVILEGIADOS

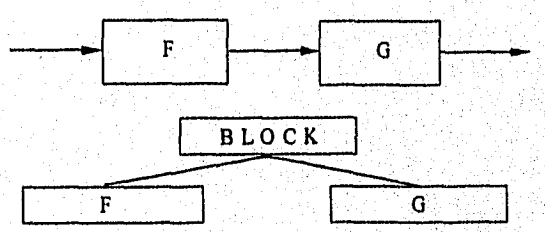
Se le da ese nombre a tres grupos de diagramas, los cuales permiten programar cualquier tipo de programa, los cuales se mencionan enseguida:

1- la secuencia.

2- La condición

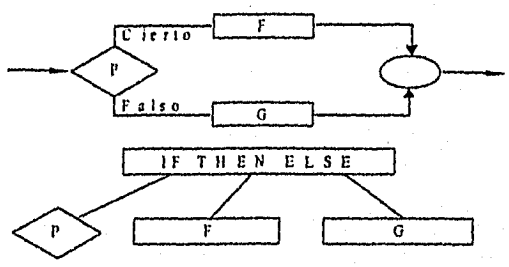
3.- La repetición

1.- El bloque secuencial o "secuencia" o "encadenamiento"



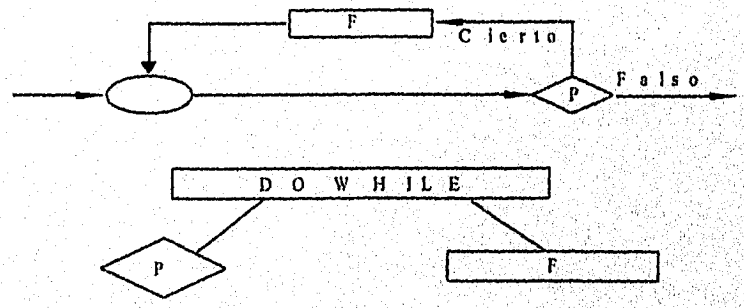
F y G se definen como tratamientos.

2.- El bloque condicional o "condición", o "alternativa"



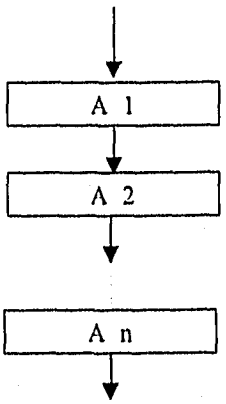
P=Predicado F,G=Tratamientos

3.- El bloque de repetición o "bucle", o "iteración"



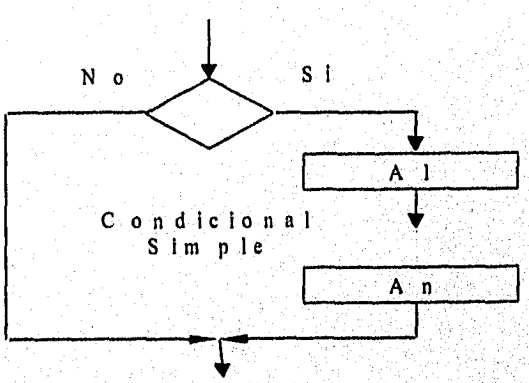
Los correspondientes ordinogramas estructurados (diagramas de flujo de proceso) de los diagramas privilegiados son los siguientes:

Estructura secuencial.

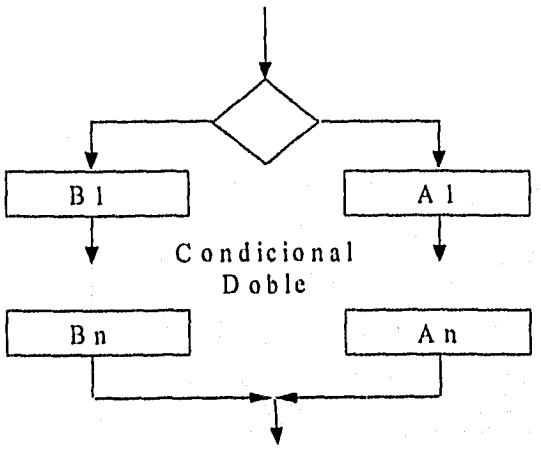


Siendo A 1, A 2, ... A n
las n diferentes acciones.

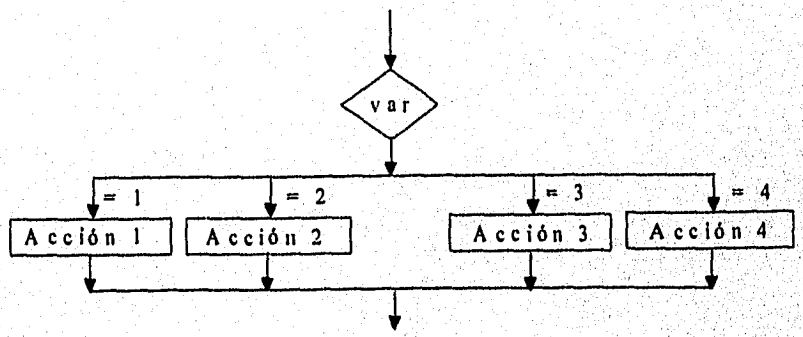
Estructura Condicional Simple



Estructura condicional doble

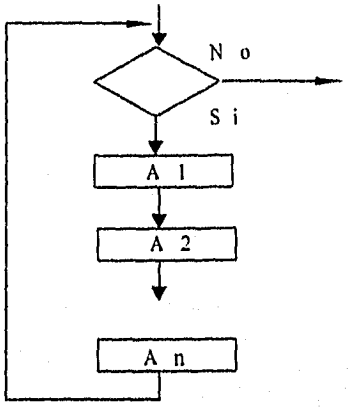


Estructura Condicional Múltiple



Estructura Repetitiva o Bucle Tipo Mientras

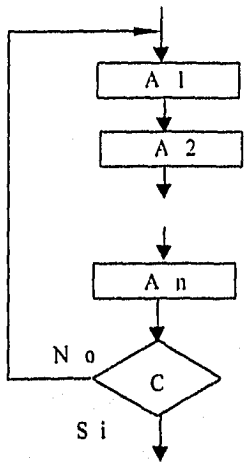
La característica principal es que la condición de salida del bucle esta al principio de este. Si la condición de salida es inicialmente falsa el bucle nunca se llegara a ejecutar.



Estructura Repetitiva o Bucle Tipo Hasta

Su característica principal es que la condición que controla la ejecución de las acciones del bucle esta al final de este mismo.

Este tipo de bucle se llega a ejecutar al menos una vez ya que la condición se encuentra al final de este.

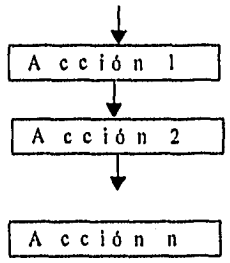


1.3.2 DESCRIPCIÓN DE LAS ESTRUCTURAS DE CONTROL EN PSEUDOCÓDIGO

Paralelamente al surgimiento de la programación estructurada surgió una nueva forma de descripción de algoritmos llamada ***pseudocódigo,*** debido a la necesidad que se presentaba de representar procedimientos cada vez más complejos y que con los diagramas de flujo era muy difícil de representar debido a que las líneas de conexión de estos se complicaban cada vez más para su representación.

Estructura Secuencial.

Ordinograma

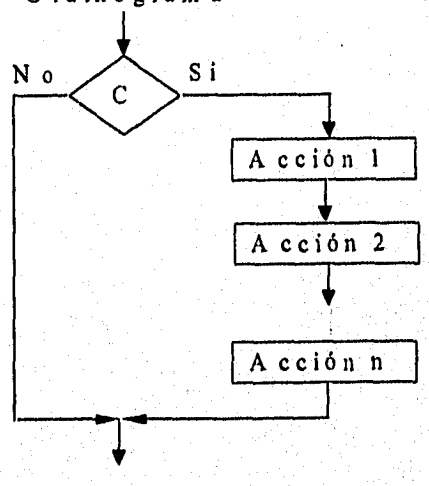


Pseudocódigo

```
Acción 1
Acción 2
Acción n
```

Estructura Condicional Simple

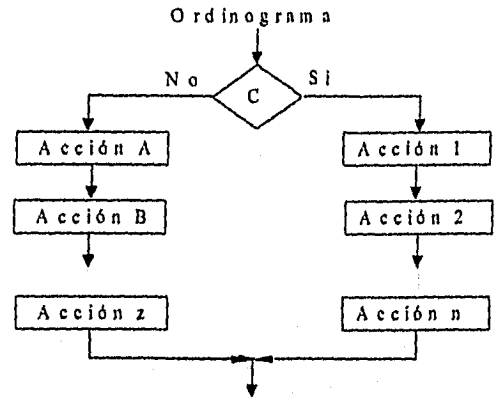
Ordinograma



Pseudocódigo

```
Si Condición
   Entonces
      Acción 1
      Acción 2
      Acción n
Fin Si
```

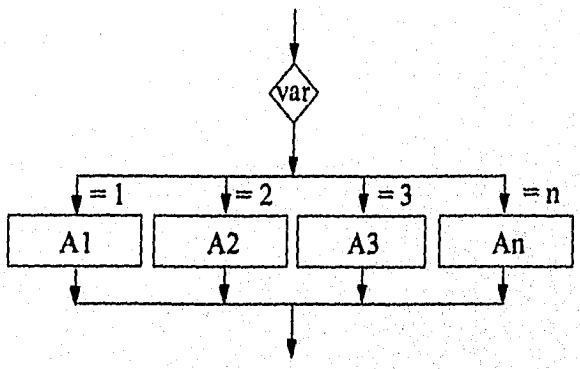
Estructura Condicional Doble



Pseudocódigo

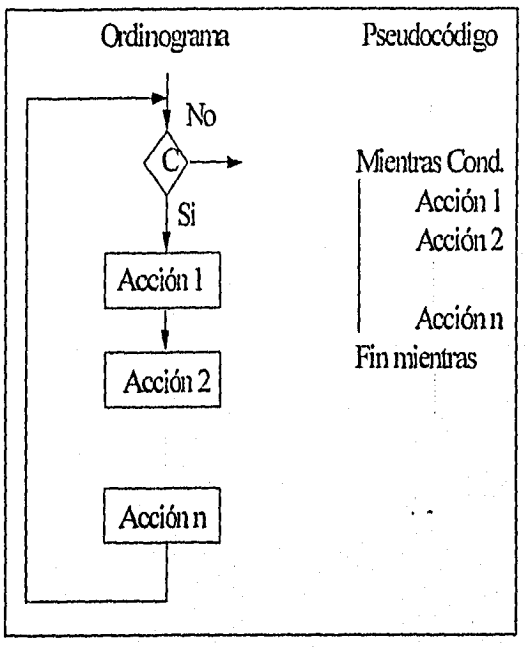
```
Si Condición
   Entonces
      Acción 1
      Acción 2
      Acción n
   Sino
      Acción A
      Acción B
      Acción z
Fin Si
```

Estructura Condicional Múltiple



Estructura Repetitiva o Bucle Tipo Mientras

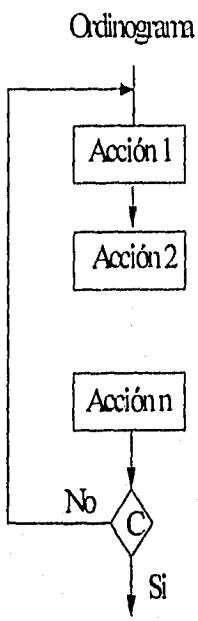
La característica principal es que la condición de salida del bucle esta al principio de este. Si la condición de salida es inicialmente falsa el bucle nunca se llegara a ejecutar.



Estructura Repetitiva o Bucle Tipo Hasta

Su característica principal es que la condición que controla la ejecución de las acciones del bucle esta al final de este mismo.

Este tipo de bucle se llega a ejecutar al menos una vez ya que la condición se encuentra al final de este.



Pseudocódigo

```
Repetir
    Acción 1
    Acción 2

    Acción n
Hasta cond.
```

1.3.3 PROGRAMAS ESTRUCTURADOS

Se dice que un programa esta estructurado si se puede representar exclusivamente con los diagramas E

E= BLOCK, IF THEN, IF THEN ELSE, CASE OF, DO WHILE, DO UNTIL

Programar de forma estructurada consiste en analizar el problema como si el lenguaje de programación no tuviera la orden GO TO. es decir, tener la idea de programas sin GO TO. ya que se considera incongruente la utilización de dicha sentencia.

Obliga a analizar el problema de "arriba a abajo" (Top-Down), en una descomposición de elementos más simple.

1.3.4 TEOREMAS DE LA PROGRAMACIÓN ESTRUCTURADA

Existen cuatro teoremas en los que se basa la programación estructurada.

T1. Teorema de Estructura (existencia).

Todo programa propio es equivalente a un programa en el que todas sus ramas resultan únicamente del empleo de diagramas de los tipos.

BLOCK, IF THEN ELSE, DO WHILE (Y/O DO UNTIL)

T2. Corolario de arriba a abajo (Top-down)

Todo programa propio es equivalente a un diagrama de una de las siguientes formas: "BLOCK (f,g); IF THEN ELSE(p,g,f); DO WHILE(p,f) y/o DO UNTIL(f,g)".

Donde:

p = Es un predicado del programa original

f,g = Son programas más reducidos, o subprogramas a los que se referencia desde el programa principal.

T3. Teorema de corrección (o validación).

"La validación de un programa estructurado se puede efectuar por pasos sucesivos, examinando cada uno de su estructura arborecente y probando localmente la validación de la descomposición funcional realizada".

T4. Teorema de descomposición.

Este teorema es útil ya que nos sirve para guiar la descomposición funcional y arborecente, es decir la programación.

1.3.5 LA PROGRAMACIÓN ESTRUCTURADA EN DIFERENTES LENGUAJES DE PROGRAMACIÓN

FORTRAN.- Aparece en la década de los 50 el cual era muy pobre en los diagramas privilegiados.

PASCAL(ADA).- Nace en los principios de la década de los 70 que han sido diseñados totalmente dirigidos a estas estructuras.

ALGOL.- Este lenguaje aparece en el año 1960 (ALGO 60) cuando todavía no se hablaba de la programación estructurada .De el arrancan los lenguajes estructurados como el PASCAL y el ADA que son los pilares de este tipo de programación.

COBOL.- El lenguaje cobol aparece también en la década de los 50 antes del movimiento estructurado y por esa razón dispone de pocos diagramas privilegiados. No obstante, su amplia gramática permite escribir fácilmente programas estructurados (simularlos).

FORTRAN IV.-(ANSI 1966).-Dispone de pocas instrucciones que se asocien a diagramas privilegiados por lo que estos se deben simular.

FORRAN 77.- producido según normas ANSI 1975, incorpora algunas sentencias estructuradas más que configuran diagramas privilegiados como son:

IF THEN, IF THEN ELSE

pero sigue sin disponer de los diagramas DO WHILE Y DO UNTIL.

1.3.6 PROGRAMACIÓN ESTRUCTURADA: MÉTODO JACKSON

La metodología de Jackson se Basa en los tres diagramas privilegiados y en la Estructura lógica de la programación modular. Sus Puntos esenciales son los siguientes:

1.- El programa, en la metodología Jackson, se subdivide en módulos atendiendo a la estructura del programa según una descomposición jerárquica por niveles. La estructura de los componentes o módulos después de esta descomposición, será solamente de los tres tipos siguientes (diagramas privilegiados)

a) Secuencial.

b) Selección o alternativa

c) Repetitiva

2.- Los criterios de descomposición del programa serán únicamente función de las estructuras de datos que intervengan. La estructura del problema viene definida por la estructura de los datos. En este sentido, la estructura de los datos constituye un modelo del problema y de su entorno. Realmente esta es la característica fundamental del método Jackson, pues en vez de estructurar el programa, define la estructura de los datos y así queda definida la estructura del programa.

3.- El procedimiento general de diseño de un programa, en esta metodología será:

a) Definir sus estructuras de datos

b) Crear las estructuras del programa a partir de las estructuras de los datos

c) Expresar cada componente del programa en operaciones ejecutables, situando cada una en la componente apropiada que se conoce normalmente como "obtención de pseudocódigo".

4.- La estructura del programa se obtendrá solo a partir de las estructuras de los datos, los datos de entrada pueden estructurarse constituyendo un único flujo de datos de entrada; ello predetermina una estructura del programa a la entrada. Los datos de salida pueden

estructurarse en general en varias estructuras de datos de salida (tantos como salidas distintas tenga) , por lo que cada una determina la estructura de un modulo para dicha salida.

1.3.7 PROGRAMACIÓN ESTRUCTURADA: MÉTODO DE BERTINI

El método Bertini y Tallineau parte del hecho sencillo de que el programa en general y cada nivel en particular deberán expresarse en un lenguaje de programación.

El programa deberá estructurarse por niveles, el cual tendrá un principio, una serie de tratamientos y un fin, esos tratamientos se aplicaran sobre unos determinados objetos, por lo que es preciso escoger si cada tratamiento se aplicara a dichos objetos para todos igual indiscriminadamente, o bien de forma discriminada .

En el primer caso se tratara de encontrar la condición que haga que dicho tratamiento deje de aplicarse al conjunto de dichos objetos y en el segundo caso, se trata de establecer que condición cumple un determinado objeto para que dicho tratamiento se lo aplique o no.

El Método Bertini se basa en los tres bloques de la programación estructurada el recurso gráfico que utiliza es el árbol programático.

1.4 PROGRAMACIÓN ORIENTADA A OBJETOS

La constante búsqueda en el campo de la computación por desarrollar cosas nuevas, más poderosas , sencillas, confiables, rápidas, realistas y eficientes, ha permitido llegar a desarrollar la técnica más poderosa de programación que es la **Programación Orientada a**

objetos.

Esto se debe a la urgente necesidad del Software de tener un salto en complejidad, capacidad de diseño, flexibilidad, rapidez de desarrollo, facilidad de modificación y confiabilidad.

Las técnicas orientadas a objetos permiten que el software sea creado a partir de objetos de comportamiento especifico, donde los objetos se pueden construir a partir de otros objetos ya existentes, así de esta forma hasta poder llegar a crear objetos sumamente complejos. Los cuales se pueden unir con otros para crear verdaderamente sistemas complejos.

Debido a los grandes avances que se han dado actualmente en la computación es probable que se pueda llegar a cambiar totalmente la forma de pensar en el análisis y desarrollo de Software ya que existen las técnicas adecuadas para poder llevar a cabo este cambio total que transformara la industria de la computación lo que se conocerá con el nombre de **Revolución Industrial de Software** en donde el software será compilado a partir de componentes de objetos reutilizables, con lo cual se creara una enorme biblioteca de componentes y donde el software se ensamblará a partir de componentes y paquetes de muchos proveedores, lo que desencadenara en el verdadero surgimiento de la **Ingeniería del Software.**

Para poder llevar a cabo este salto se requiere que se integren todas las tecnologías más poderosas para el desarrollo de software, bajo una misma base (análisis y diseño orientado a objetos) junto con la programación orientada a objetos, sólo así de esta forma se lograra la revolución industrial del software, estas tecnologías son las siguientes:

- CASE e I-CASE

- Programación visual
- Generadores de código
- Depósitos y coordinadores de depósitos
- Metodologías basadas en depósitos
- Ingeniería de la información
- Bases de datos orientadas a objetos
- lenguajes no por procedimientos
- Motores de inferencia
- Tecnología cliente-despachador
- Bibliotecas de clases que maximicen la reutilización
- Análisis y diseño orientado a objetos..

La integración de estas técnicas permitirá que el software, al igual que las máquinas se construya a partir de componentes reutilizables, creados por medio de la programación orientada a objetos, por lo que algunos objetos serán extremadamente complejos, debido a que estarán compuestos a partir de otros objetos..

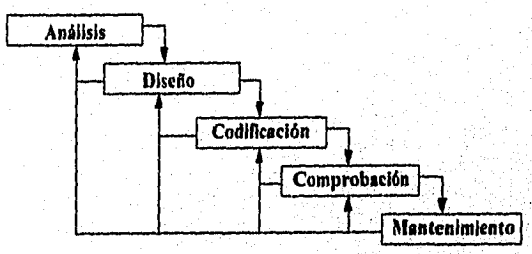
Las técnicas orientadas a objetos modificaran:

- Toda la industria del software
- La forma en que se venden los programas de aplicación
- La forma de uso de las computadoras
- La forma de uso de las redes
- La forma de analizar los sistemas
- La forma de diseñar sistemas

- La forma de utilizar las herramientas CASE
- La forma de Planear las empresas
- El trabajo de todos los profesionales de los sistemas de información.

La programación Orientada a Objetos al igual que los demás tipos de programación pasa por los mismos pasos para llevar a cabo el desarrollo de un sistema, estos pasos son los siguientes:

- **Análisis**
- **Diseño**
- **Codificación**
- **Comprobación**
- **Mantenimiento**



Desde el punto de vista de un programador tradicional, las técnicas Orientadas a Objetos parecen ser conceptos tradicionales pero con nombres diferentes, como se muestra en la siguiente tabla:

TÉCNICAS ORIENTADAS A OBJETOS	TÉCNICAS TRADICIONALES
Método	Procedimientos, funciones o subrutinas
Variable Modelo	Datos
Mensajes	Llamadas a procedimientos o funciones
Clases	Tipos abstractos de datos
Herencia	(No existe Técnica similar)
Llamada bajo control de sistema	Llamada bajo control del programa

En la programación Orientada a Objetos la mayoría de sus componentes son objetos, los cuales pueden mandar o recibir mensajes de o hacia otros objetos, los objetos al recibir un mensaje ejecutan un determinado método, el cual manipula los datos internos del objeto para llevar a cabo una tarea especifica del sistema.

La programación Orientada a Objetos aunque a simple vista parece ser en conceptos muy similar a los demás tipos de programación, no lo es así. Tanto en la programación clásica o convencional, programación modular y programación estructurada, el código del programa estaba en uno o varios programas o procedimientos y siempre que se ejecutaba un salto de control del programa al termino de este devolvía el control al programa principal para continuar ejecutando el programa, cosa que ahora en la programación Orientada a Objetos no sucede así, ya que el código del programa se encuentra fragmentado en los métodos de cada objeto y cuando un objeto recibe un mensaje y ejecuta su método este no regresa el control al sistema, si no que envía el control a algún otro objeto mediante un mensaje.

1.5 VENTAJAS Y DESVENTAJAS DE LOS TIPOS DE PROGRAMACIÓN

Podemos notar ciertas características distintivas entre un tipo de programación con respecto a otro lo que propicia el conocer las ventajas entre cada uno de ellos.

La Programación Clásica fue una de las primeras etapas para programar, esta programación como se ha visto, era en forma de instrucciones en secuencia y rutinas que hacían saltar el programa de un lugar a otro, se usaba la instrucción GO TO, ésta instrucción permitía romper el control del programa y transferirlo a cualquier parte de él pudiendo o no regresar a dicho punto de salto para seguir ejecutando instrucciones. Esto daba una gran facilidad de pasar de un lado a otro del programa pero también hacía que no tuviera una estructura lógica bien definida y ordenada de programación. Además el programa realizado no era fácil de modificar para correcciones futuras, ya que alguna modificación requería la completa revisión del programa para saber en que parte sería conveniente realizarla y no solo eso sino tratar de saber que tipo de lógica tuvo el autor del programa para el caso en que fuese un programador distinto.

El querer realizar algún cambio significaba afectar gran parte de la codificación para adecuarlo a la nueva lógica.

Debido a lo anterior, surge la necesidad de programar de una forma diferente, algo que pudiera permitir el realizar cambios fácilmente, propiciar una puesta a punto más rápida y que cuando se diera el caso, diferentes programadores pudieran interpretar los programas sin dificultad, sin que alguna modificación afecte a toda la codificación del mismo.

Una de las tendencias que se busca conseguir para la elaboración de un programa es la de desarrollar éstos en forma clara, simple y que puedan ser mantenidos y actualizados fácilmente.

Esto se consigue con el uso de la programación modular que da las bases para elaborar codificaciones más ordenadas, algunas de las ventajas que aporta frente a la programación clásica son :

- Los programas son más sencillos de escribir y depurar ya que se pueden hacer pruebas parciales con cada uno de sus módulos.

- El realizar algún cambio en un módulo es más sencillo y en general no tiene por qué afectar al resto de los módulos.

- Un programa se puede ampliar fácilmente con sólo diseñar los nuevos módulos necesarios.

- Un mismo módulo puede ser llamado desde varios puntos del programa, evitando la repetición de instrucciones ya escritas.

Aun con el surgimiento de la programación modular no se lograba, tener un mejor orden en la programación ya que aunque se tenían módulos de programación seguía existiendo el mismo problema que se presentaba con la programación convencional ya que este se presentaba ahora en cada uno de los módulos de la programación modular por que

se tuvo que continuar buscando nuevas técnicas para el desarrollo de programas por lo que vino a surgir la Programación Estructurada la cual tenia una serie de características que permitió tener un mejor orden de análisis y desarrollo de programas, así como también redujo la complejidad de las anteriores tipos de programación, ya que esta contaba con una serie de estructuras bien definidas con las cuales se podía desarrollar cualquier tipo de programa, sus ventajas sobre la programación convencional y modular son las siguientes:

- Tiene como herramienta el pseudocódigo para el análisis y desarrollo de programas complejos y muy grandes
- El mantenimiento de los Programas es más fácil
- Evita la utilización de la sentencia GO TO, ya que la utilización de esta se considera incoherente
- Obliga a analizar el problema de arriba a abajo (Top-Down) para una descomposición de elementos más simples
- Debido a las estructuras que utiliza es más fácil hallar errores y corregirlos
- No se depende de la existencia del autor del programa para poderlo modificar

Debido a las exigencias que presenta el mundo actual se ha hecho necesario el desarrollo de un nuevo tipo de programación que satisfaga las necesidades actuales y que sea de una alta calidad, fácil desarrollo, más realista , complejo pero a la vez sencillo de programar y de rápido mantenimiento por esa razón surge lo que ahora conocemos como

programación orientada a objetos. la cual tiene la siguientes ventajas sobre los demás tipos de programación.

- Se pueden construir sistemas muy grandes complejos y de alta calidad
- Esta expuesto a menos errores ya que las partes de que se compone ya fueron probadas
- Se pueden agregar y quitar módulos sin que afecte a todo el sistema
- El desarrollo de sistemas es más rápido y se realiza en menor tiempo
- La programación y mantenimiento es mas sencilla.

CAPÍTULO II

ANTECEDENTES
DE
BASES DE DATOS

Objetivo:

Investigar los conceptos de una base de datos, para conocer la estructura y las partes que la componen, describiendo cada una de ellas.

II. ANTECEDENTES DE BASES DE DATOS

Antes que surgieran las bases de datos electrónicas, las bases de datos se plasmaban en papel, las cuales se guardaban en carpetas y a su vez estas en armarios. Cuando se necesitaba información los ficheros debían extraerse de los armarios, debido a que no existía un buen control, los papeles corrían el riesgo de mezclarse, y por consiguiente esto ocasionaba que la búsqueda de información fuera más lenta e incluso solía haber una duplicidad de documentos, la actualización y consulta de información era muy lenta y complicada debido a que existían grandes volúmenes de información.

Debido al aumento de la creciente información se fueron necesitando medios más propios y adecuados para agilizar el manejo y control de información , es entonces cuando viene el surgimiento de los Sistemas de Gestión de Bases de Datos **(DBMS)**[1] , ya que el instrumento central de estos son las bases de datos.

En un principio la potencia de la gestión de las bases de datos era sólo usada por personal especializado en procesos de datos, en los grandes sistemas de ordenadores que generalmente se usaban en experimentos de investigaciones científicas, pero debido a que algunas personas se dieron cuenta de la potencia que podían ofrecer estos Sistemas de Gestión de Bases de datos empezaron a hacerlo más extensivo a todo el mundo con la introducción de uno de los primeros Sistemas de Gestión de bases de datos comerciales que Fue Dbase II.

En cualquier empresa de la actualidad se requiere de un sistema que permita almacenar datos en forma segura y eficiente y que además pueda accesar a ellos en cualquier instante, lo anterior ha llevado a considerar el gran valor que ocupa una base de datos y al desarrollo de conceptos y técnicas para la gestión eficiente de los datos.

[1] DBMS (Data Base Management System)

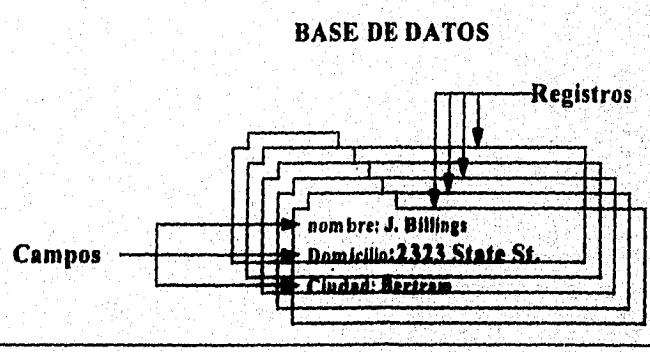
Para entender el significado de base de datos, es necesario explicar ciertos conceptos fundamentales que se relacionan con este término. Se habla de una necesidad de poder almacenar un volumen de información demasiado grande y poder accesar a esta en forma rápida y precisa sin mayores complicaciones. Teniendo en cuenta que dicha información debe encontrarse almacenada completamente y sin tener ninguna alteración, hasta el momento en que el usuario realice algún cambio en ella.

2.1 CONCEPTOS FUNDAMENTALES

Una base de datos es un conjunto de infonnación relacionada entre sí, almacenada en archivos. Se hace referencia al hardware de la computadora que se emplea para almacenarla y a los programas utilizados para manipularla. Con la definición anterior se toca una parte medular de una base de datos, la cual es el almacenamiento de información en archivos, un archivo es un conjunto de registros, mientras que un registro es un conjunto de campos, un campo es un conjunto de cadenas de información y una cadena es un conjunto de caracteres y un carácter es un conjunto de bits.

Ejemplos de Bases de datos:

- Un archivador de metal que contiene registros de los clientes.
- Un archivador de tarjetas con nombres y números de teléfono.
- Una agenda con una lista escrita a lápiz de un inventario de almacén.



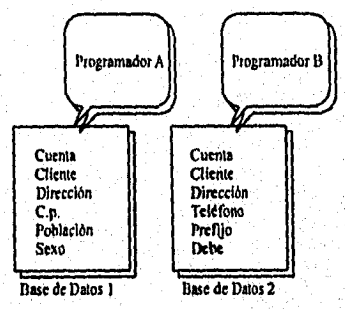
BASE DE DATOS EN FORMA DE TABLA DE DATOS

NOMBRE	DOMICILIO	CIUDAD	ESTADO	CP	NO TELEFONO	No CUENTE
J. Billings	2323 State St.	Bertram	CA	91113	234-8980	0004
R. Foster	RR 1 Box 62	Frio	CA	93360	845-3121	0001
L. Miller	P.O. Box 345	Dagget	CA	94567	484-996	0002
O. Roberts	2023 St.	[illegible]	[illegible]	[illegible]	[illegible]	0003

2.1.1 OBJETIVOS DE UNA BASE DE DATOS

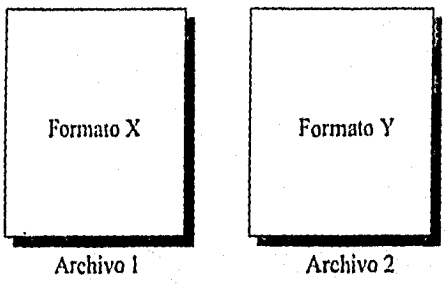
Uno de los objetivos de los sistemas de bases de datos es el de poder manipular la información que se encuentra almacenada, pero también el proporcionar y resolver ciertos aspectos de control. Algunos de estos aspectos son :

- ***Redundancia e Inconsistencia de los Datos:*** Muchas de las veces en las empresas se desarrollan gran cantidad de programas individuales realizados por distintos programadores, esto causa duplicidad de información , ya que cada programador incluye los campos necesarios en su base de datos sin saber que otra persona tiene los mismos campos pero en otra base de datos distinta. Esto causa el aumento de los costos de almacenamiento.

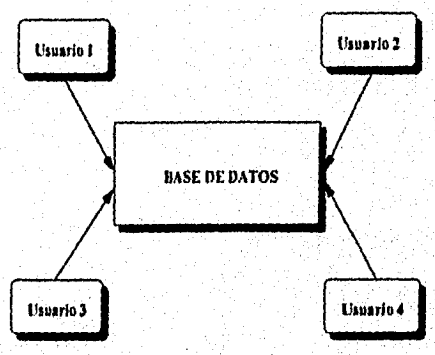


- ***Acceso a los Datos:*** Se debe tener en cuenta que se deben desarrollar sistemas de recuperación de datos para uso general.

- ***Aislamiento de los Datos:*** Este problema se da cuando los datos se encuentran repartidos en varios archivos y cada uno de ellos tiene diferentes formatos causando el no poder obtener los datos apropiados.



- ***Problemas de Acceso Simultáneo:*** En la actualidad se requiere de poder accesar a una base de datos pero no solamente una persona sino varias a la vez, esto causa una inconsistencia de la información ya que se puede dar el caso de que dos usuarios al mismo tiempo estén realizando una operación similar y que el sistema no actualice en el momento, alguna de las dos operaciones.



- ***Seguridad:*** Una parte importante de la base de datos es el proporcionar una integridad de los datos tanto en la información en sí, como en el acceso permitido de usuarios a cierta parte de la base de datos.

Lo anterior ha dado la pauta para desarrollar sistemas de gestión de bases de datos, los cuales permiten a los usuarios acceder y modificar los archivos de las bases de datos.

La organización de los datos en una base de datos debe representar la semántica de los mismos en forma correcta y eficiente y dicha organización se establece en los archivos, debemos recordar que cualquier tipo de almacenamiento de información se realizará en un archivo. Un archivo se define como un conjunto de registros semejantes almacenados en dispositivos de almacenamiento secundario como: unidades de disco, cintas magnéticas, etc... Dentro de cada archivo están contenidos los campos relacionados que contienen elementos datos elementales.

Al considerar una base de datos se debe tener en cuenta el tamaño de los archivos, es decir, tratar de justificar el por qué de la utilización de dicha base de datos, por lo general son grandes aunque esto depende del hardware y de las restricciones operativas aplicables en un ambiente dado. El término grande implica una cantidad de datos mayor que la que una sola persona puede manejar por sí misma, aunque la auxilie un sistema de computación, un ejemplo de base de datos grande la constituye un sistema integrado de datos de personal y de producción en una compañía manufacturera de aproximadamente 6 mil empleados, con más de 300 mil registros.

Una base de datos muy grande es una parte esencial de una empresa ya que será utilizada continuamente por muchos usuarios, al mismo tiempo requerirá de muchos dispositivos de almacenamiento. Una base de datos muy grande sería aquélla en la que el tiempo necesario para realizar una copia es mayor que el intervalo permitido entre las transacciones de actualización realizadas para mantener al día la base de datos. Algunos

otros ejemplos de bases de datos grandes serían los sistemas para una compañía telefónica, la administración de seguro social, etc...

Al leer o escribir archivos, los datos se transfieren entre los dispositivos de almacenamiento y de memoria del sistema de cómputo. Una base de datos se ocupa de los datos que permanecen al alcance del sistema. Los datos que se sacan, se vuelven accesibles para modificarlos y si vuelven a introducirse tendrán que considerarse como nueva entrada, ya que no se tiene seguridad de su consistencia. Al hablar de una base de datos se considerará que se dispone de un sistema operativo que proporciona capacidades adecuadas de entrada y salida.

Otro objetivo importante de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema debe esconder ciertos aspectos de cómo se almacenan y mantienen los datos. También debe ser lo suficientemente accesible y esto se logra cuando los datos se pueden extraer con rapidez.

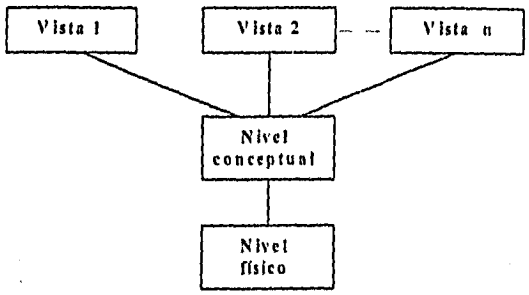
Muchos de los usuarios finales que utilizarán la base de datos a través de un sistema de gestión de bases de datos no tienen experiencia en computadoras, por lo cual se les esconde lo complejo del proceso a través de diversos niveles de abstracción. Existen 3 niveles de abstracción de datos, los cuales son :

1.- Nivel Físico: En este nivel se describe a detalle las estructuras de los datos y se indica como se almacenan realmente.

2.- Nivel Conceptual: Se describe qué datos son realmente almacenados y las relaciones que existen entre ellos. Generalmente este nivel lo usan los administradores de bases de datos, quienes deben decidir que información se va a guardar en la base de datos.

3.- Nivel de Visión: Este nivel describe solo una parte de la base de datos, a muchos de los usuarios no les interesará saber como se definen las estructuras de los datos,

únicamente como acceder a la información fácilmente. Dicho nivel es la interacción que un usuario tiene para con el sistema.



Niveles de Abstracción de Datos

Un ejemplo de lo anterior es lo siguiente :

Supongamos una empresa bancaria en la que se tiene un registro llamado cliente con tres campos, nombre, calle y ciudad. En el nivel físico, un registro de cliente se define como un bloque de palabras o bytes, en el nivel conceptual cada uno de estos registros se describe por medio de una definición de tipo, y se define la interrelación entre estos tipos de registro y en el nivel de visión el usuario visualiza solo una parte de la base de datos por ejemplo la información sobre las cuentas de los clientes. No pueden acceder a la información sobre los salarios de los empleados.

Algún tipo de definición de datos es :

```
type cliente = record
                    nombre : string;
                    calle : string;
                    ciudad : string;
               end;
```

2.1.2 RESUMEN DE LOS OBJETIVOS DE UNA BASE DE DATOS

- **Los Datos Podrán Utilizarse de Múltiples Maneras.**

  Diferentes usuarios, que perciben diferentemente los mismos datos, pueden emplearlos de distintas maneras.
- **Se Protegerá la Inversión Intelectual.**

  No será necesario rehacer los programas y las estructuras lógicas existentes (que representan muchos hombres-años de trabajo) cuando se modifique la base de datos.
- **Bajo Costo.**

  Bajos costo del almacenamiento y el uso de los datos y minimización del costo de los cambios.
- **Menor Aumento de Datos.**

  Las necesidades de las nuevas aplicaciones se satisfarán con los datos existentes más bien que creando nuevos archivos, evitándose así el excesivo aumento de datos que se advierte en las bibliotecas de cintas actuales.
- **Desempeño.**

  Los pedidos de datos se atenderán con la rapidez adecuada según el uso que de ellos habrá de hacerse.
- **Claridad.**

  Los usuarios sabrán qué datos se encuentran a su disposición y los comprenderán sin dificultad.
- **Facilidad de Uso.**

  Los usuarios tendrán fácil acceso a los datos. Las complejidades internas son ajenas al usuario, gracias al sistema de administración de la base.
- **Flexibilidad.**

  Los datos podrán ser utilizados o explorados de manera flexible, con diferentes caminos de acceso.
- **Rápida Atención de Interrogantes No Previstos.**

Los pedidos espontáneos de información se atenderán sin necesidad de escribir un programa de aplicación (lo que significa un cuello de botella por la pérdida de tiempo) sino utilizando un lenguaje de alto nivel para averiguación o generación de reportes.

- **Facilidad Para el Cambio.**

  La base de datos puede crecer y variar sin interferir con las maneras establecidas de usar los datos.

- **Precisión y Coherencia.**

  Se utilizarán controles de precisión. El sistema evitará las versiones múltiples de los mismos ítems de datos con diferentes estados de actualización.

- **Reserva.**

  Se evitará el acceso no autorizado a los datos. Los mismos datos podrán estar sujetos a diferentes restricciones de acceso para diferentes usuarios.

- **Protección Contra Pérdida o Daño.**

  Los datos estarán protegidos contra fallos y catástrofes, y contra delincuentes, vándalos, incompetentes y personas que intenten falsearlos.

- **Disponibilidad.**

  Los datos se hallarán inmediatamente disponibles para los usuarios casi todas las veces que los necesiten.

2.2 ESTRUCTURA DE UNA BASE DE DATOS

Una vez entendidos los conceptos anteriores, se llega a la descripción de la ***estructura de una base de datos***, ésta es la definición de los datos en el sistema, es decir, los vínculos con otros elementos dato, la forma de describir una base de datos es el ***modelo estructural*** y este a su vez se conoce como ***modelo de datos.***

Un ***modelo de datos*** es una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia. Dentro de estos tipos de modelos de datos se encuentran 3 grupos

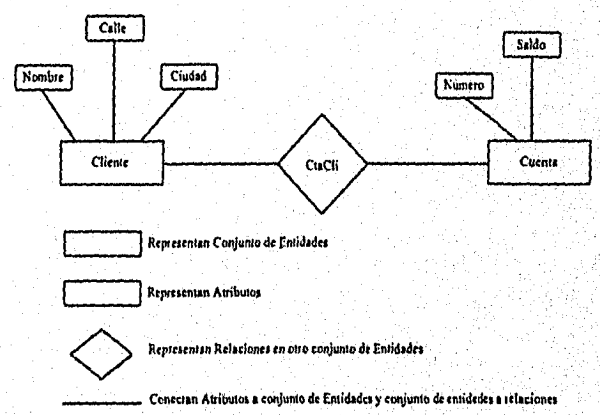
principales que son : ***modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos de datos.***

- **Los Modelos Lógicos Basados en Objetos**, en donde el elemento de referencia es el objeto, entendido como tal, a aquél que existe y puede distinguirse de otros.

Algunos ejemplos de este modelo son:

- Modelo Entidad-Relación
- Modelo Binario
- Modelo Semántico de Datos
- Modelo Infológico
- Modelo Funcional de Datos
- Modelo Orientado a Objetos

Modelo Entidad-Relación, esta basado en la captación o percepción del mundo real, el cual esta constituido por una colección de objetos básicos a los cuales se les llama ***entidades, y sus relaciones*** existentes entre ellos mismos. Una entidad es un objeto que se distingue de los demás objetos debido a que tiene un conjunto especifico de atributos y una relación es una asociación entre varias entidades, la siguiente figura muestra un ejemplo de este modelo.



Modelo Orientado a Objetos, de igual forma que el modelo Entidad-Relación, el modelo Orientado a Objetos se basa en una colección de objetos pero los cuales son manejados y vistos desde otro punto de vista. Los objetos contiene valores guardados en variables instancias dentro de ellos mismos, estos valores son objetos también, por consiguiente los objetos pueden contener objetos dentro de si mismos hasta un cierto nivel de anidamiento deseado.

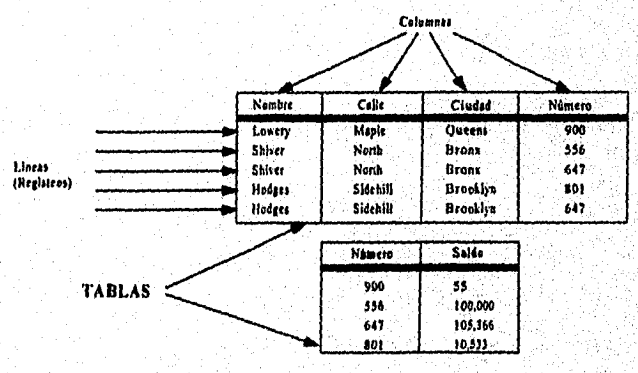
Un objeto tiene cierta parte de código que opera sobre el mismo, a esa parte de código se le llama ***método***, la forma en que se comunican los objetos es a través de mensajes, los cuales activan los métodos correspondientes del otro objeto al que se le está enviando el mensaje para que este realice una tarea determinada.

- Los Modelos Lógicos Basados en Registros, el elemento básico es una ocurrencia o conjunto de datos relacionados de algún modo.

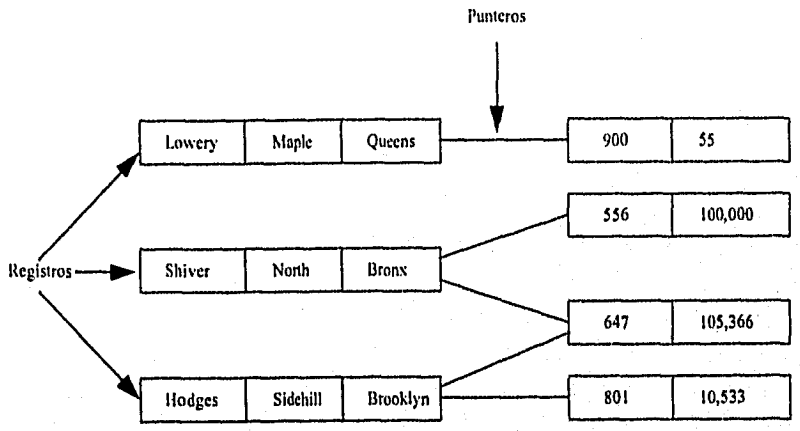
Algunos modelos de este tipo son:

- Modelo Relacional
- Modelo de Red
- Modelo Jerárquico

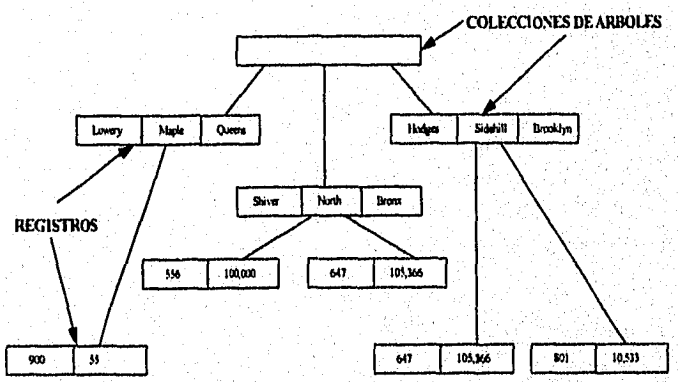
Modelo Relacional, este modelo usa ***tablas*** de dos dimensiones, que consisten de líneas y columnas para la representación de los datos y sus relaciones, cada una de estas tablas tiene un número de columnas con nombres únicos, abajo se muestra un ejemplo:



Modelo de Red, En este modelo los datos se representan por medio de colecciones de ***registros*** y sus relaciones entre los datos se pueden ver como ***punteros***, un ejemplo de este modelo es el siguiente:



Modelo Jerárquico, este modelo es muy parecido al modelo de red, debido a que los datos y las relaciones entre los datos se representan por medio registros y punteros. La diferencia es que en el modelo Jerárquico los registros están organizados como colecciones de ***árboles*** en vez de grafos arbitrarios como en el modelo de red, un ejemplo es el siguiente:



- **Modelos Físicos de Datos,** existen muy pocos modelos de este tipo, los dos más reconocidos son los siguientes:

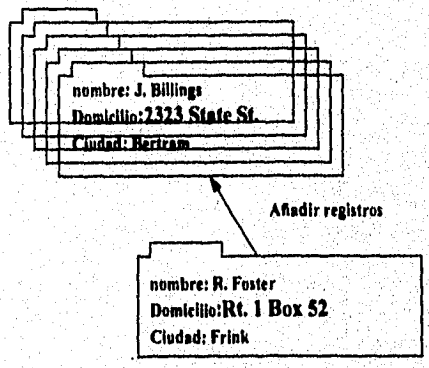
- Modelo Verificador
- Memoria de Elementos

Estos nos permiten describir datos en el nivel más bajo de la abstracción de datos, capturando aspectos de la implementación de la base de datos.

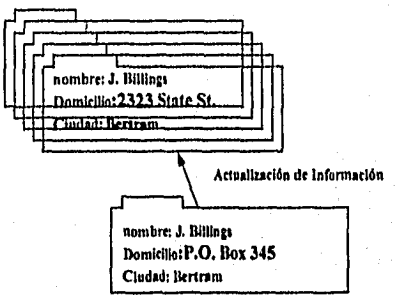
De dichos modelos de datos se hablará con más detalle en el siguiente capítulo.

Podemos mencionar que el aspecto estático de una base de datos, es el almacenamiento de datos como archivos, registros y campos. Ahora tocaremos el punto dinámico de una base de datos, es decir, las operaciones (por operación entendemos que es una sección de la base de datos que realiza un proceso determinado), dentro de éste tipo de operaciones se tienen :

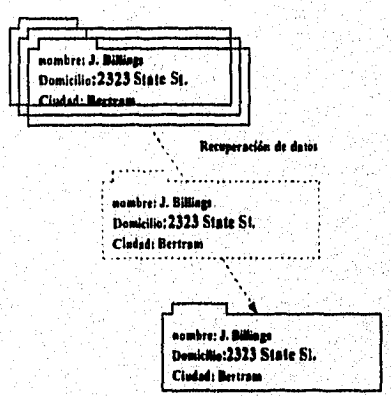
- ***Añadir Registros de Información:*** El añadir registros a una base de datos incluye la obtención de los datos su codificación y su captación.



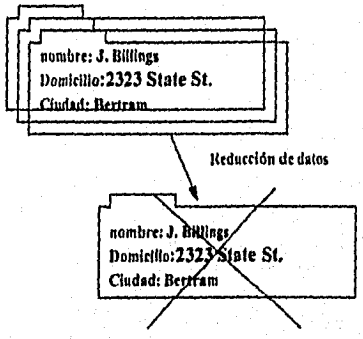
- ***Actualización de la Información:*** Esta es la parte de modificación de la base de datos, incluye colocar nuevos datos, modificar, cuando sea necesario valores de datos almacenados y eliminar datos no válidos u obsoletos.



- ***Recuperación de Datos:*** Dicha recuperación puede consistir en tomar un elemento específico para obtener un valor almacenado, o en la recolección de una serie de elementos relacionados para obtener datos referentes a alguna relación que se manifieste al unir los datos. Para traer un registro específico de datos se entrará a la base de datos mediante un argumento de búsqueda, el que se comparará con una llave en los registros de la base de datos.



- ***Reducción de Datos:*** Esto resulta necesario cuando el volumen de los datos importantes excede la capacidad de los solicitantes. En una base de datos estática este tipo de actividad tiende a ser dominante. En una base de datos dinámica la reducción de datos se utiliza principalmente para resúmenes periódicos y análisis de tendencia. Cuando la información deseada se encuentra distribuida en toda la base de datos, puede resultar necesario el acceso a la mayor parte de su contenido, para concentrar los datos. La reducción de datos exige mucho del desempeño de una base de datos. Debemos recordar que para poner en operación un sistema de bases de datos se invierten grandes cantidades de tiempo y dinero, por lo cual debemos hacer que nuestra base de datos sea flexible, segura y eficiente.



2.3 INSTANCIAS Y ESQUEMAS

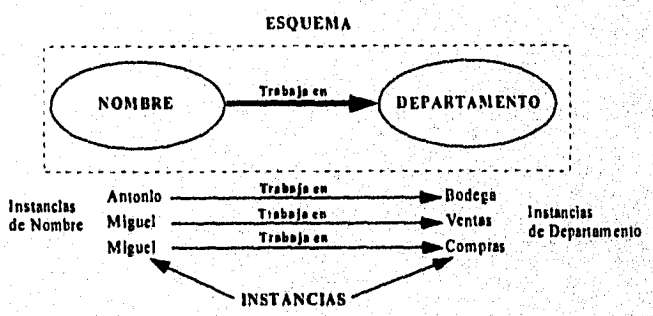
Se conoce con el nombre de instancia a la colección de información almacenada en la base de datos, en un determinado tiempo, mientras que al diseño global de la base de datos se le llama esquema de la base de datos. El término esquema se usa para referirse al diseño. El diseño consiste en una enumeración de tipos de entes u objetos con los cuales la base trabaja, las relaciones entre estos tipos de entes y las formas en las cuales los entes y las relaciones de un nivel de abstracción son expresadas en el siguiente nivel (más concreto). Dicho concepto de esquema corresponde a la idea de definición de tipo en el

lenguaje de programación . Una variable de algún tipo tiene un valor determinado en un instante de tiempo dado. Con esto podemos indicar que el concepto del valor de una variable en los lenguajes de programación corresponde al concepto de una instancia de un esquema de la base de datos.

Un sistema de bases de datos tiene varios esquemas, que se dividen de acuerdo a los niveles de abstracción tratados. En el nivel más bajo está el esquema físico; en el nivel intermedio, el esquema conceptual; en el nivel más alto, un subesquema.

2.3.1 DIFERENCIA ENTRE ESQUEMAS E INSTANCIAS

Ejemplo: Supóngase que se tiene una base de datos de empleados de una empresa. El esquema conceptual podría incluir los tipos de entes NOMBRE y DEPARTAMENTO así como la relación TRABAJA EN entre NOMBRE y DEPARTAMENTO, además de otras relaciones y otros tipos de entes. La base de datos conceptual podría incluir objetos como Antonio y Miguel del tipo NOMBRE, con la información de que " Antonio TRABAJA EN bodega", mientras que "Miguel TRABAJA EN ventas" y "Miguel TRABAJA EN compras", los objetos bodega, ventas y compras son instancias de DEPARTAMENTO.



Bajando al siguiente nivel de abstracción, el esquema para la base de datos física podría determinar que los entes de tipo NOMBRE y DEPARTAMENTO son cadenas de

10 caracteres y TRABAJA EN está representado por listas ligadas de DEPARTAMENTOS por cada empleado.

La base de datos física reside permanentemente en memoria secundaria (disco y/o cinta). Esta puede ser vista como una serie de niveles de abstracción, desde el nivel de registros y campos en un lenguaje de programación a través del nivel de registros lógicos, mantenido por el sistema operativo bajo el DBMS, hasta el nivel de bits y direcciones físicas del mecanismo de almacenamiento.

Por lo general los esquemas se cambian rara vez. La capacidad de cambiar o modificar una definición de un esquema en un nivel sin afectar la definición de un esquema en un nivel superior se llama independencia de datos, de la cual tenemos 2 subdivisiones :

- ***Independencia Física de Datos.*** Es la capacidad de modificar el esquema físico sin provocar que se vuelvan a escribir los programas de aplicación. Es decir, el hardware de almacenamiento y las técnicas físicas de almacenamiento podrán ser modificados sin obligar a la modificación de los programas de aplicación.

- ***Independencia Lógica de Datos.*** Es la capacidad de modificar el esquema conceptual sin provocar que se vuelvan a escribir los programas de aplicación. Es decir, podrán agregarse nuevos ítems de datos, o expandirse la estructura lógica general, sin que sea necesario reescribir los programas de aplicación existentes.

Tomando en cuenta estos dos tipos de independencia de datos, la independencia lógica es más difícil de lograr que la independencia física de datos, ya que los programas de aplicación generalmente dependen de la estructura lógica de los datos a los que acceden.

Se hace una analogía al decir que el concepto de independencia de los datos es similar en algunos aspectos al concepto de tipos abstractos de datos en lenguajes de programación modernos. Ambos ocultan detalles de implementación al usuario, lo cual permite que el usuario se concentre en la estructura general en vez de los detalles de implementación de bajo nivel.

2.4 LENGUAJE DE DEFINICIÓN DE DATOS (DDL)

Un DBMS proporciona un lenguaje de definición de datos para especificar un esquema conceptual por medio de un conjunto de definiciones las cuales se expresan mediante un lenguaje de definición de datos **(DDL)**[2] . El lenguaje de definición de datos es una notación para describir los tipos de entes y las relaciones entre éstos en términos de un modelo de datos particular. Este se usa cuando se está diseñando. El resultado de la compilación de sentencias de DDL es un conjunto de tablas las cuales se almacenan en un archivo especial llamado ***diccionario de datos*** (o directorio).

Un **Directorio de Datos** es un archivo que contiene ***metadatos***, es decir, (datos sobre datos). este archivo se consulta antes de leer o modificar los datos reales en el sistema de bases de datos.

La estructura de almacenamiento y los métodos de acceso usados por los sistemas de bases de datos se especifican por medio de un conjunto de definiciones en un tipo especial de DDL llamado lenguaje de almacenamiento y definición de datos. El resultado de la compilación de estas definiciones es un conjunto de instrucciones que especifican los detalles de implementación de los esquemas de bases de datos que normalmente se esconden a los usuarios.

[2] (DDL Data Definition Language)

2.5 LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

Un lenguaje de manipulación de datos (DML)[3] es un lenguaje que capacita a los usuarios a acceder o manipular datos según la organización por el modelo de datos adecuado. Existen básicamente 2 tipos:

- **Procedimentales**, los DML requieren que el usuario especifique qué datos se necesitan y cómo obtenerlos.
- **No Procedimentales**, los DML requieren que el usuario especifique qué datos se necesitan sin especificar cómo obtenerlos.

Podemos notar que los DML no procedimentales son más sencillos de aprender y usar que los procedimentales. Ya que el usuario no necesita especificar la forma de obtenerlos, sin embargo, estos lenguajes pueden generar código que no sea tan eficiente como el producido por lenguajes procedimentales. Este problema puede remediarse a través de técnicas de optimización.

Una **consulta** es un enunciado que solicita la recuperación de información. A la parte de un DML que implica recuperación de infomación se llama ***lenguaje de consultas***, generalmente este nombre se aplica como un sinónimo de ***lenguaje de manipulación de datos.***

Por manipulación de datos, se quiere decir:

- La Recuperación de Información.
- La Inserción de Infomación.
- La Supresión de Información.
- La Modificación de Datos.

[3] (DML Data Manipulation Language)

2.6 GESTIÓN DE BASES DE DATOS

Un sistema de gestión de bases de datos es una colección de archivos interrelacionados y un conjunto de programas que permiten a los usuarios acceder y modificar esos archivos. Generalmente conocidos por el nombre de Sistemas Administradores de Bases de Datos (DBMS). Algunas de las funciones de los DBMS son: Seguridad, integridad, sincronización, protección y recuperación de la información.

Interacción con el Gestor de Archivos. Los datos sin procesar se almacenan en el disco usando el sistema de archivos que normalmente es proporcionado por un sistema operativo convencional. El gestor de base de datos traduce las distintas sentencias DML a comandos del sistema de archivos de bajo nivel.

Implantación de la Integridad. Los valores de los datos que se almacenan en la base de datos deben satisfacer ciertos tipos de restricciones de consistencia. El gestor de la base de datos determina si las actualizaciones a la base de datos dan como resultado la violación de las restricciones, para el caso de ser así, tomar la acción apropiada.

Implantación de la Seguridad. No todos los usuarios de la base de datos necesitan tener acceso a todo el contenido. El gestor de la base de datos hace que se cumplan estos requisitos de seguridad.

Copia de Seguridad y Recuperación. Cualquier sistema informático, dispositivo mecánico o eléctrico, está sujeto a fallos. En cada uno de estos casos, se pierde la información referente a la base de datos. Es responsabilidad del gestor de la base de datos detectar tales fallos y restaurar la base de datos al estado que existía antes de ocurrir el fallo. Esto se lleva a cabo normalmente a través de la iniciación de varios procedimientos de copias de seguridad y recuperación.

Control de Concurrencia. Controlar la interacción entre los usuarios concurrentes es otra responsabilidad del gestor de la base de datos.

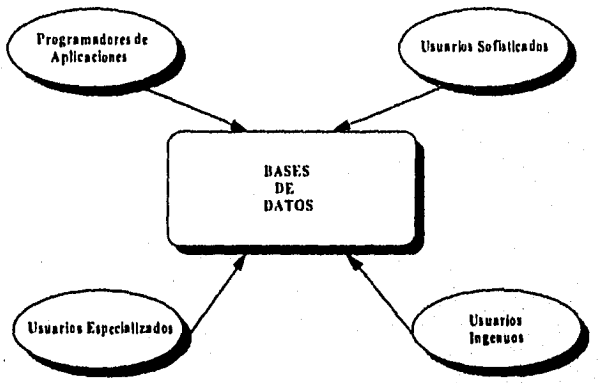
2.7 ADMINISTRADOR DE BASES DE DATOS

En toda base de datos se requiere de una persona que tenga el control central de los datos y de los programas que acceden a esos datos, dicha persona que tiene este control sobre el sistema se llama administrador de base de datos (DBA)[4]. Las funciones del administrador de base de datos incluyen.

- **Definición de Esquema.** El esquema original de la base de datos se crea por un conjunto de definiciones que son traducidas por el compilador de DDL a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.
- **Definición de la Estructura de Almacenamiento y del Método de Acceso.** Estas son un conjunto de definiciones de métodos de acceso y estructuras de almacenamiento adecuadas que son traducidas por el compilador del lenguaje de almacenamiento y definición de datos.
- **Modificación del Esquema y de la Organización Física.** Las modificaciones al esquema u organización física son poco comunes y se logran escribiendo un conjunto de definiciones que son usadas por el compilador de DDL o por el compilador del lenguaje de almacenamiento y definición de datos para generar modificaciones a las tablas internas apropiadas del sistema (ejemplo, el diccionario de datos).
- **Concesión de Autorización para el Acceso a Datos.** Dicha concesión de diferentes tipos de autorización permite al administrador de la base de datos regular que partes de ella van a poder ser accedidas por varios usuarios.
- **Especificación de las Restricciones de Integridad.** Esas restricciones de integridad se mantienen en una estructura especial del sistema que consulta el gestor de la base de datos cada vez que tiene lugar una actualización en el sistema.

[4] (DBA Data Base Administrator)

2.8 USUARIOS DE BASES DE DATOS



Hay cuatro tipos distintos de usuarios de sistemas de bases de datos, diferenciados por la forma en que esperan interaccionar con el sistema.

1.- Programadores de Aplicaciones. Los profesionales en computación interaccionan con el sistema por medio de llamadas en DML, las cuales están incorporadas en un programa escrito (Cobol, Pascal, C etc.). estos programas son los llamados programas de aplicación. Ya que la sintaxis de DML es diferente de la sintaxis del lenguaje principal, las llamadas en DML van normalmente precedidas por un carácter especial de forma que se pueda generar el código apropiado. Un preprocesador especial, llamado precompilador de DML, convierte las sentencias en DML a llamadas normales a procedimientos en el lenguaje principal. El Programa resultante se ejecuta por el compilador del lenguaje principal el cual genera el código de objeto apropiado.

Hay tipos especiales de lenguajes de programación que combinan estructuras de control de lenguajes como pascal con estructuras de control para el manejo de un objeto de la base de datos (por ejemplo relaciones). estos lenguajes llamados lenguajes de cuarta generación, por lo general incluyen características especiales para facilitar la generación

de formas y la presentación de datos en la pantalla. La mayoría de los sistemas de bases de datos comerciales incluyen un lenguaje de cuarta generación.

2.- Usuarios Sofisticados. Estos interactúan con el sistema sin escribir programas. En cambio escriben sus preguntas en un lenguaje de consulta de bases de datos. Cada consulta se somete a un procesador de consultas cuya función es tomar una sentencia en DML y descomponerla en instrucciones que entienda el gestor de base de datos.

3.- Usuarios Especializados. Algunos usuarios sofisticados escriben aplicaciones de bases de datos que no encajen en el marco tradicional de procesamiento de datos, algunas de estas aplicaciones son: Sistemas de diseño ayudados por computadora, sistemas expertos, sistemas que almacenan datos con tipos complejos de datos(Gráficas y datos de audio) y sistemas de modelación de entorno.

4.- Usuarios Ingenuos. Dichos usuarios no sofisticados interactúan con el sistema invocando alguno de los programas de aplicación permanentes que se han escrito anteriormente.

CAPÍTULO III

MODELOS
DE
BASES DE DATOS

Objetivo:

Investigar los modelos de bases de datos, para conocer sus características, con el fin de diferenciar éstos, con el modelo orientado a objetos, comparando cada uno de ellos.

III.- MODELOS DE BASES DE DATOS

3.1 MODELOS DE DATOS

Los modelos de bases de datos son una forma diferente de ver los datos ya que son un enfoque no tradicional de los datos, son un marco de referencia bajo el que describen las relaciones lógicas entre los datos que forman las bases de datos, pues no toman en cuenta ni hardware ni software ni los valores específicos de cada elemento de datos, se consideran solo los datos y no los procesos con los datos ya que los procesos cambian y los datos no, los modelos de bases de datos no tienen contenido las bases de datos si.

Existen tres grupos principales de modelos de bases de datos que son:

1) MODELOS LÓGICOS BASADOS EN REGISTROS

- Modelo Jerárquico
- Modelo de Red
- Modelo Relacional

2) MODELOS LÓGICOS BASADOS EN OBJETOS

- Modelo Entidad-Relación
- Modelo Binario
- Modelo Semántico de datos
- Modelo Infológico
- Modelo Funcional de datos
- Modelo Orientado a objetos

3) MODELOS FÍSICOS DE DATOS

- Modelo Verificador
- Memoria de Elementos

3.2 MODELOS LÓGICOS BASADOS EN REGISTROS

- *Modelo Jerárquico (1960)*

Los datos y las relaciones se presentan por medio de registros y ligas, los registros se organizan como conjuntos de árboles.

- *Modelo de Red*

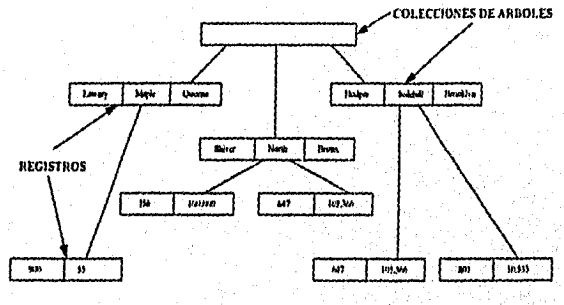
Los datos se representan mediante conjuntos de registros y las relaciones entre datos se representan con enlaces que son apuntadores.

- *Modelo Relacional (1970)*

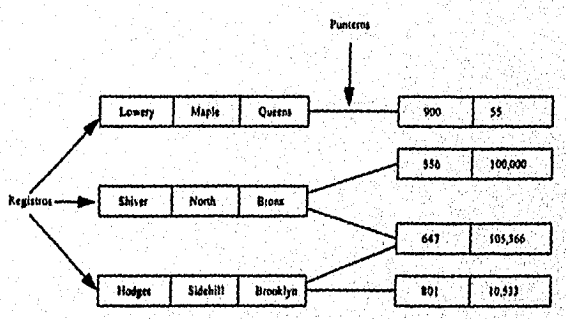
Los datos se representan por medio de tablas bidimensionales cada una de las cuales se compone de columnas que tienen nombres únicos.

3.2.1 REPRESENTACIÓN

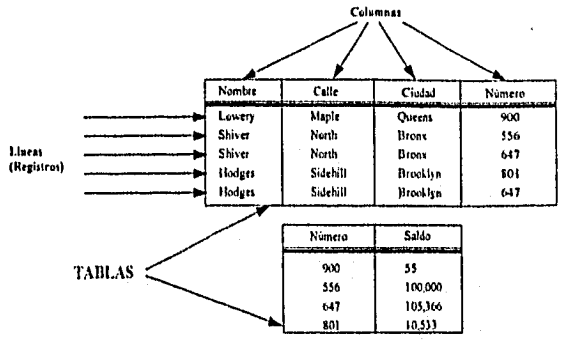
Modelo Jerárquico



Modelo de Red



Modelo Relacional



3.2.2 COMPONENTES

Los elementos básicos de los modelos de bases de datos son

- Entidad
- Atributos
- Relaciones

3.2.3 CARACTERÍSTICAS

JERÁRQUICO Y DE RED

- Fueron enfocados a estructuras físicas para almacenamiento optimo
- Poca importancia ala percepción de los datos por parte del usuario
- Existe redundancia de datos
- Carentes de fundamento
- No fueron modelos de datos bien definidos

RELACIONAL

- Nos permite crear modelos de datos más completos en base a relaciones (Tablas)
- Sistema de redundancia controlada
- Modelo fundamentado matemáticamente (Álgebra Relacional)

3.2.4 TERMINOLOGÍA

MODELO\ENTIDAD	ATRIBUTO	RELACIÓN
Jerárquico\Segmento	campos de segmento	Jerarquías
Redes\Registros	Campos de registros	sets
Relacional\Relación	Atributos de tablas	Relationship (Llaves de las tuplas)

3.3 MODELO JERÁRQUICO

Este modelo es muy inflexible ya que en las diferentes entidades de las que está compuesta una determinada situación, se organizan en niveles múltiples de acuerdo a una estricta relación Padre-Hijo.

3.3.1 TERMINOLOGÍA

Hablar del término Padre-Hijo significa que el modelo se organiza por niveles, es decir, van a existir en un nivel (Padre) ciertas entidades y debajo de estas en forma de árbol existirán otras entidades (Hijo). Un hijo únicamente puede tener un padre situado en el nivel inmediatamente superior al suyo.

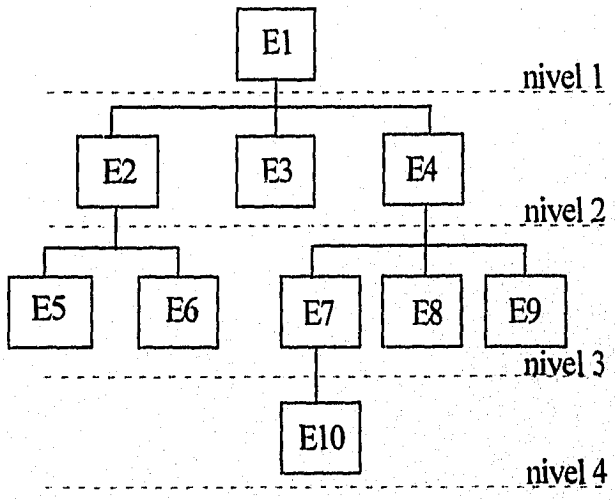
Las entidades, estructuras que agrupan todos aquéllos atributos o datos con características comunes, se denominan para este modelo Segmentos, mientras que los atributos reciben el nombre de Campos.

Los segmentos se organizan en niveles de manera que en un mismo nivel estén todos aquéllos segmentos que dependen de un segmento de nivel inmediatamente superior.

La relación Padre-Hijo implica que no puedan establecerse relaciones entre segmentos dentro de un mismo nivel.

La representación gráfica de un modelo jerárquico se realiza mediante la estructura de Árbol Invertido, en la que el nivel superior está ocupado por una única entidad, bajo la cual se distribuyen el resto de las entidades en niveles que se van ramificando. Los diferemes niveles quedan unidos por medio de las relaciones.

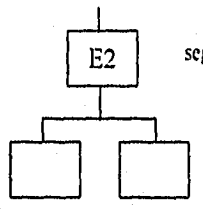
Al número de estos niveles se les denomina Altura del Árbol. Mientras que al número total de segmentos que constituye el árbol se denomina Peso del Árbol.



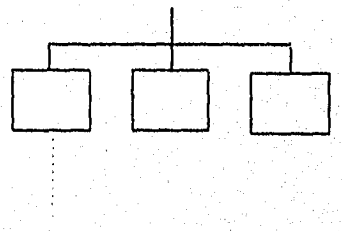
El árbol de la figura anterior tiene una Altura de 4 (No. de niveles) y el Peso del Árbol es de diez (No. total de segmentos que lo forman).

Los segmentos de acuerdo a su situación en el árbol y sus características, pueden denominarse como:

1) Segmento Padre: Es aquél que tiene descendientes, todos ellos localizados en el mismo nivel.

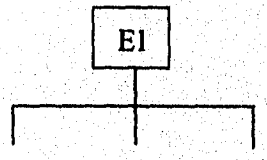


2) Segmento Hijo: Es aquél que depende de un segmento de nivel superior. Todos los hijos de un mismo padre están en el mismo nivel del árbol.

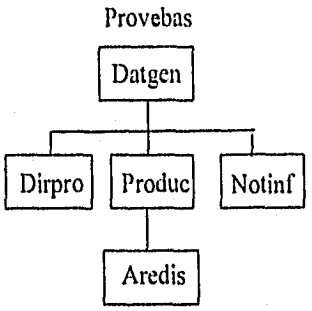


Como puede observarse, los segmentos podrán ser padre e hijo al mismo tiempo.

3) Segmento Raíz: La raíz es única y ocupa el nivel superior del árbol.



Ejemplo representativo de un árbol.....



Imaginemos una empresa nacional con sucursales en todo el país; esta empresa tiene centralizadas todas las compras de material de sus sucursales en la oficina central, para lo cual dispone de una base de datos jerárquica que le permite almacenar los datos de todos sus proveedores.

La base de datos se denomina Provebas (proveedores base de datos) y presenta cinco segmentos. El segmento raíz en el que se almacenan los datos que son comunes a todos los proveedores, como pueden ser : Nombre de la empresa, Director de la empresa, etc... Este segmento se denomina Datgen (datos generales). En el segundo nivel del árbol hay tres segmentos dependientes del segmento raíz. El primero de ellos contiene las direcciones de las sucursales de la empresa proveedora, indicando: Calle, Número, Ciudad, Tipo de direceión, etc...El nombre de este segmento es Dirpro (dirección del proveedor).

El segundo segmento contiene los datos de todos los produetos suministrados por cada una de las empresas proveedoras. Este segmento se denota como Produc (productos). El último segmento del segundo nivel es el que permite guardar las diferentes notas informativas que sobre un proveedor van remitiendo las sucursales a la oficina central; este segmento se denomina Notinf (notas informativas).

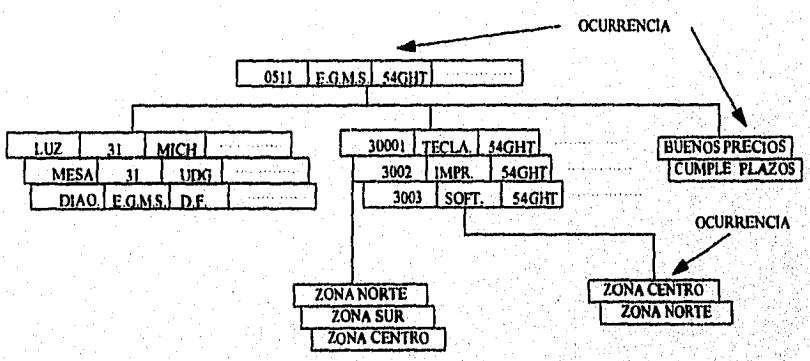
El tercer nivel del árbol está ocupado por un solo segmento que es el que permite almacenar las zonas de distribución de cada uno de los productos suministrados por los

diferentes proveedores; este segmento se reconoce como Aredis (área de distribución) y depende del segmento Produc.

Dentro de este modelo, al conjunto de valores particulares que toman todos los campos que componen un segmento en un momento determinado se le llama Ocurrencia de un segmento.

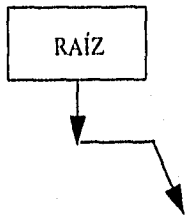
Un Registro de la base de datos es el conjunto formado por una ocurrencia del segmento raíz y todas las ocurrencias del resto de los segmentos de la base de datos que dependen jerárquicamente de dicha ocurrencia raíz.

Diagrama representativo de Ocurrencias..

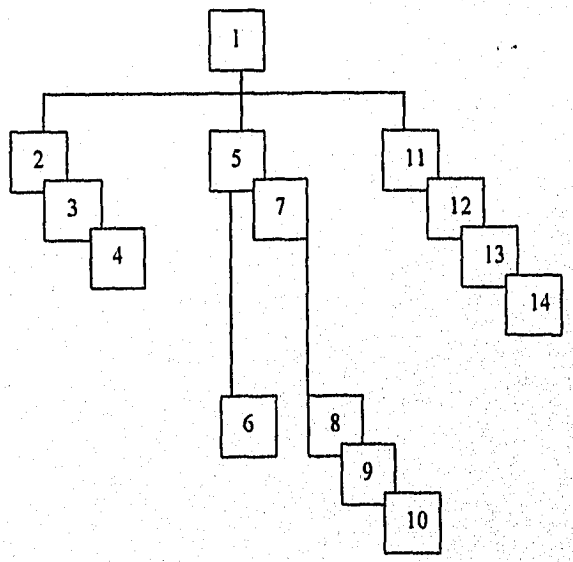


La relación Padre-Hijo en la que se apoyan las bases de datos jerárquicas, determina que el camino de acceso a los datos sea Único, a lo que se le conoce como Camino o Secuencia Jerárquica, comienza siempre en una ocurrencia del segmento raíz y recorre la base de datos de arriba a abajo y de izquierda a derecha. Gráficamente se puede representar este camino de la siguientes forma:

Diagrama de camino o secuencia jerárquica.



Un conjunto de registros de la base de datos PROVEBAS podría ser el que aparece a continuación:



Entre las restricciones propias de este modelo se pueden resaltar:

A) Cada árbol debe tener un único segmento raíz.

B) No puede definirse más de una relación entre dos segmentos dentro de un árbol.

C) No se permiten las relaciones reflexivas de un segmento consigo mismo.

D) No se permiten las relaciones N:M.

E) No se permite que exista un hijo con más de un padre.

F) para cualquier acceso a la información almacenada, es obligatorio el acceso por la raíz del árbol, excepto en el caso de utilizar un índice secundario.

G) El árbol debe recorrerse siempre de acuerdo a un orden prefijado: el camino jerárquico.

H) la estructura del árbol, una vez creada, no se puede modificar.

3.3.2 DISEÑO DE ÁRBOLES

Cabe señalar que el diseño de árboles parte del modelo Entidad-Relación que determina una determinada situación. Los pasos a seguir al aplicar este modelo son:

a) Determinar las entidades que pueden extraerse de dicha situación, agrupando bajo un nombre genérico todos aquéllos datos con características comunes.

B) Definir, de entre todas las relaciones existentes entre las entidades, aquéllas que se vayan a implantar en la futura base de datos jerárquica. Para definir estas relaciones hay que tener en cuenta las restricciones fijadas por la realidad del problema.

La representación en un esquema de las entidades y las relaciones existentes entre ellas da lugar a un modelo (nivel conceptual) que se puede emplear como punto de partida para determinar el tipo de base de datos a emplear.

C) Un tercer paso del diseño sería aplicar las normas y restricciones del modelo jerárquico al esquema anterior, para así obtener el árbol o árboles necesarios para almacenar la información de la situación problema inicial.

La definición de las entidades y de las relaciones en una base de datos jerárquica se realiza con el L.D.D. del sistema gestor (Lenguaje de Definición de los Datos) siguiendo el

orden que determina el camino jerárquico; esto implica que, en primer lugar, habría que definir la entidad o segmento raíz y a continuación y de arriba a abajo y de izquierda a derecha, el resto de los segmentos que componen la base de datos. Todo segmento que no sea el segmento raíz incluirá en su definición quién es su padre jerárquico.

3.4 MODELO DE RED

Una base de datos en el modelo de red se forma de una colección de registros(Mediante los cuales son representados los datos), los cuales están asociados entre si por medio de enlaces, los cuales son una asociación entre dos registros exclusivamente.
Un registro es parecido a una entidad en el modelo Entidad-Relación, así como también cada registro esta compuesto de campos (Atributos en otros modelos), y estos están organizados en forma de grafo arbitrarios.

3.4.1 DIAGRAMA DE ESTRUCTURA DE DATOS

Los diagramas de estructura de datos son esquemas mediante los cuales podemos representar el diseño de una base de datos de red, sus componentes básicos son:

- Cajas, representan tipos de registros
- Líneas, representan enlaces

Ejemplo de diagramas de estructura de datos y sus tipos de relaciones.

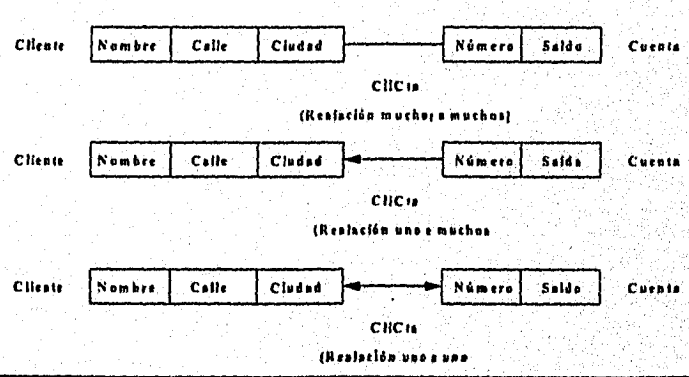
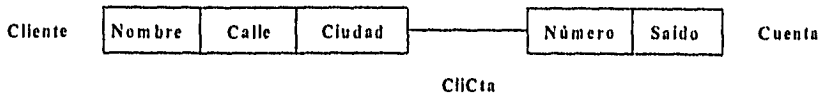
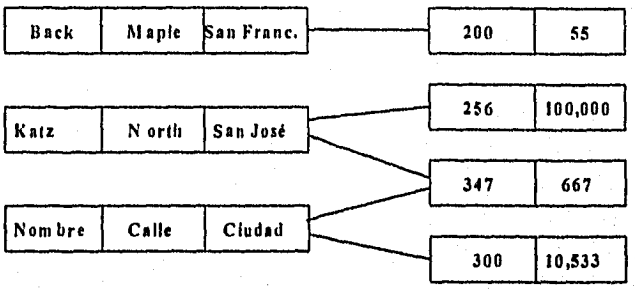


Diagrama de Estructura

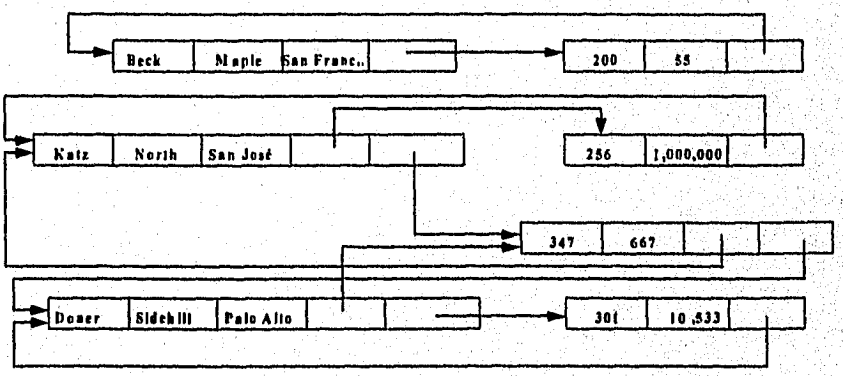


Base de Datos del Diagrama de Estructura



Una base de datos de red consta de registros y enlaces, los enlaces se forman agregando campos de punteros a los registros que se asocian por medio de enlaces, por cada enlace de un registro a otro, cada registro debe tener un puntero.

Ejemplo de implementación de una base de datos de red



Debido a que el enlace CliCta entre cliente y cuenta es muchos a muchos, cada registro puede estar asociado a un número ilimitado de registros, por lo que no es posible limitar el número de campos de puntero de un registro, por lo que aunque un registro sea de longitud fija, el registro real implementado físicamente es u registro de longitud variable, es por esta razón que los arquitectos de modelo CODASYL DBTG restringieron los tipos de enlaces en este modelo para simplificar la implementación en el modelo de red.

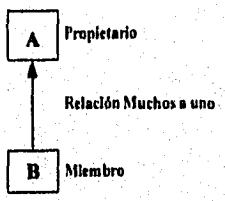
3.4.2 EL MODELO CODASYL DBTG

A finales de los sesenta aparecieron varios sistemas comerciales de bases de datos basados en el modelo de red, el Grupo de trabajo sobre Bases de Datos saca la primera especificación estándar de una base de datos, a la cual le llamaron informe CODASYL 1971.

Este modelo toma una consideraciones para simplificar su implementación, las cuales son:

- Los enlaces muchos a muchos se prohiben
- Los enlaces uno a uno se representan utilizando un enlace muchos a uno

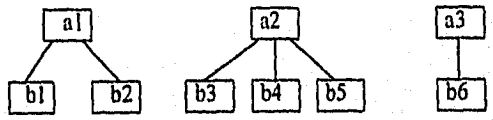
Debido a que sólo se puede utilizar enlaces muchos a uno en este modelo, un diagrama de estructura de datos que conste de dos tipos de registros enlazados entre si se representa de la siguiente manera:



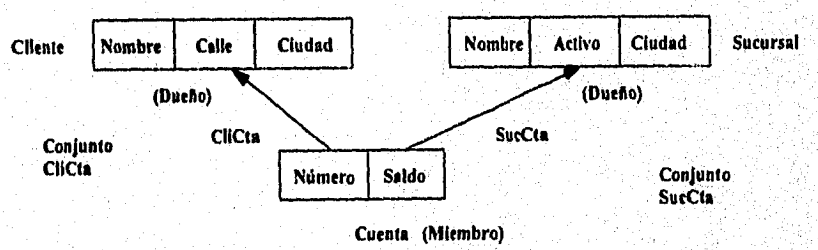
Conjunto DBTG

A esta estructura en el modelo DBTG se le llama conjunto DBTG, el nombre que recibe este conjunto es generalmente el nombre del enlace que conecta a los dos registros. Al tipo de registro A se le llama dueño(o padre en el modelo jerárquico)del conjunto y al tipo de registro B se le llama miembro (o hijo en el modelo jerárquico) del conjunto.
Cada conjunto DBTG puede tener cualquier número de ocurrencias del conjunto, o sea instancias reales de registros enlazados.

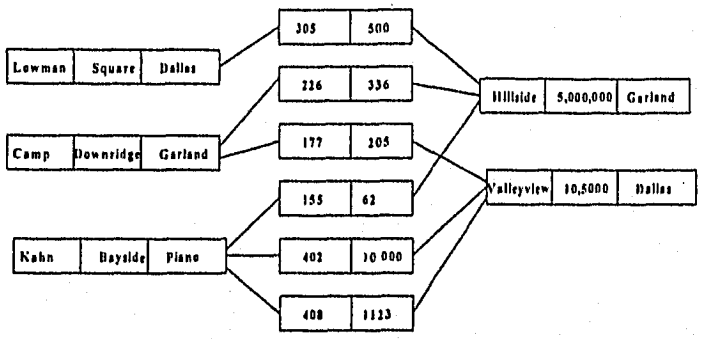
Ejemplo de ocurrencias del conjunto DBTG anterior



Ejemplo de un modelo DBTG



Ocurrencias del modelo DBTG



El lenguaje de manipulación de datos del modelo DBTG esta compuesto de ordenes que se incorporan en un lenguaje principal (Cobol, PL/1, Fortran, RPG etc.), estas ordenes acceden y manipulan variables de la base de datos como variables que se declaran localmente.

Algunas de las ordenes usadas en DBTG son las siguientes:

- Find
- Get
- Store
- Erase
- Modify
- Connect
- Disconnect
- Reconnect

Las cuales tienen funciones en este modelo de localizar, copiar, actualizar, modificar, insertar, sacar registros.

Los sistemas comerciales en el modelo DBTG que surgieron en ese entonces fueron:

- Total
- IDMS

3.5 MODELO RELACIONAL

Este modelo es uno de los más utilizados actualmente por la gran facilidad y flexibilidad para representar un problema de la vida real en una base de datos. Surge después de la aparición de los modelos de bases de datos jerárquico y de red.

3.5.1 TERMINOLOGÍA

Para entender dicho modelo, debemos conocer el significado de algunos conceptos que se manejarán en este capítulo. Uno de ellos es el llamado Entidad que se refiere a la especificación de un objeto concreto o abstracto que será representado en un sistema de base de datos. por ejemplo: el objeto fruta, coche, empleado, etc...

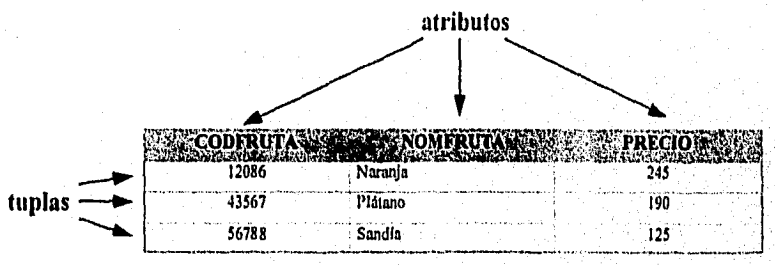
Existen los siguientes tipos de entidad:

- *Entidad Fuerte o Propia:* Es aquélla cuyas propiedades que la identifican sólo hacen referencia a la propia entidad. Por ejemplo: alumno, matrícula, almacén, etc...

- *Entidad Débil o Regular:* Es aquélla que sólo tiene sentido gracias a las propiedades que identifican a otras entidades (fuertes o a su vez débiles). Por ejemplo: rfc que está definida por las propiedades de la existencia de un empleado.

A partir de la existencia de una entidad existe un concepto llamado Atributo el cual es conocido en el medio tradicional de archivos como campo, este es un conjunto de cadenas de caracteres. Ejemplo: matricula, dirección del almacén, nombre de la fruta, etc.... Cada atributo se identifica por un nombre y este puede encontrarse en varias entidades.

Diagrama representativo de: atributos en varias entidades.



Cuando se habla del termino relacional se está mencionando implícitamente a la Estructura Relacional de los Datos que no es más que la forma en que se estructuran los datos en este modelo y se realiza a partir del concepto de Tablas.

3.5.2 TABLAS

Una tabla consiste de columnas y filas lo que es lo mismo atributos y tuplas respectivamente, con estos nombres se conocen en el modelo relacional. Para que una tabla forme parte de una estructura relacional, esta debe cumplir con las siguientes condiciones:

- Debe tener un sólo tipo de fila, cuyo formato queda definido por el diseño de la tabla o relación. Todas las filas tienen las mismas columnas.
- Cada fila debe ser única y no pueden existir filas duplicadas.
- Cada columna debe ser única y no pueden existir columnas duplicadas.

- Cada columna debe estar identificada con un nombre específico.
- El valor de una columna para una fila debe ser único. Por lo tanto, no pueden existir múltiples valores en una posición de una columna.
- Los valores de una columna deben pertenecer al dominio que representa, y es posible que un mismo dominio se utilice para definir los valores de varias columnas.

Una tabla que cumple estas condiciones, se denomina tabla relacional o relación. El concepto de relación se utiliza para indicar que la tabla mantiene una asociación con otras tablas. El concepto relacional se utiliza para expresar que los atributos que contiene no relacionan objetos fuertes, y por tanto representa por sí misma un objeto o entidad propia.

Al cumplir estas condiciones, la tabla tiene asociadas las siguientes propiedades:

- Las filas pueden estar en cualquier orden.
- A una fila se le hace referencia mediante todos los valores que la forman.
- Las columnas pueden estar en cualquier orden.
- Se hace referencia a una columna mediante el nombre que la identifica.

Otro concepto conocido en las tablas es el de Grado de una tabla relacional, el cual es el número de atributos que la forman, y Cardinalidad de una tabla relacional que es el número de tuplas que contiene.

Ejemplo de una tabla, indicando la entidad, atributos, tuplas, grado y cardinalidad.

DATGRAL

MATRICULA	NOMBRE	DIRECCION	TELEFONO
GOHG700427	NORMA	CEDROS #34	3374654
NBFH560803	HILARIO	CAMPANA #2	6544444
HGFL671095	LAURA	CAMPOS #2	6554343

La tabla anterior llamada DATGRAL (Datos generales), contiene los campos : matrícula, nombre, dirección y teléfono, es una tabla de grado=4 (No. de atributos), y una cardinalidad=3 (No. de tuplas que contiene).

3.5.3 DOMINIOS

El dominio es el conjunto de todos los posibles valores para una o más columnas de una tabla relacional, se distinguen dos tipos de dominios:

- *Dominios Generales o Continuos:* Son aquéllos que contienen todos los posibles valores entre un máximo y un mínimo predefinido. Por ejemplo:
  - Clave: Todos los números enteros y positivos de cuatro dígitos.
  - Peso del material: Todos los números reales y positivos.

- *Dominios Restringidos o Discretos:* son aquéllos que contienen ciertos valores específicos entre un máximo y un mínimo predefinido. Por ejemplo:
  - Estado Civil: Compuesto por soltero, casado, viudo, divorciado.
  - País: Uno de los países asiáticos.

Hablando de dominios podemos mencionar el universo total de los datos, es decir, el conjunto completo de todas las tablas relaciónales almacenadas en la base de datos a lo que se conoce como Modelo de Datos. mientras que un Esquema es el conjunto de declaraciones que describen el modelo.

Submodelo es el conjunto de relaciones a las que puede acceder un usuario determinado, y subesquema las declaraciones correspondientes al submodelo.

3.5.4 CLAVES

Para poder accesar a una tupla concreta dentro de una tabla se utiliza el concepto de Clave, la cual es un atributo o conjunto de atributos cuyos valores distinguen univocamente una tupla específica de una tabla.

Debido a que en una tabla no pueden existir filas duplicadas, esto implica, que siempre tiene que existir al menos una clave, que en un caso extremo estará formada por todos los atributos.

Para buscar la clave de una tabla relacional, se deberá hacer a través de todos los posibles valores (dominio) de los atributos. De tal forma que tomando algunos valores determinados se pueda identificar una única tupla de la tabla.

Ejemplo ilustrativo de selección de una clave para una tabla relacional.
Por ejemplo, sean las tablas:

DATGRAL

MATRICULA	NOMBRE	DIRECCION	TELEFONO
GOHG700427	NORMA	CEDROS #34	3374654
NBFH560803	HILARIO	CAMPANA #2	6544444
HGFL671095	LAURA	CAMPOS #2	6554343

CALIF

MATRICULA	ESPAÑOL	MATEMATICAS	FISICA
GOHG700427	8	10	6
NBFH560803	3	8	5
HGFL671095	10	10	10

En las tablas anteriores podemos notar que el atributo que tomaremos como clave es matrícula, ya que cualquier valor del dominio de dicho atributo identifica a una única tupla. Por lo tanto no puede aparecer un mismo valor de la matrícula en dos tuplas diferentes.

Es posible que en una tabla, más de una columna (o combinación de ellas) puedan servir de clave. Cuando se presente este caso, a estas claves se les denomina Claves Candidatas, ya que dentro de todas ellas se elegirá una sola que será la que identifique la tupla, a esta clave se le denomina Clave Principal o Primaria y a las demás Claves Alternativas o Secundarias.

Cuando se analice la clave principal, tendremos en cuenta sobre su dominio las siguientes consideraciones:

- Sus valores siempre deben ser conocidos (diferentes de nulos).
- La memoria que ocupen debe ser mínima.
- Su codificación debe ser sencilla.
- El contenido de sus valores no debe variar.
- Que se utilice en otras tablas para crear una interrelación (mediante claves ajenas).

Se denomina Clave Ajena a aquél atributo o conjunto de atributos que en la tabla donde se encuentran no son clave, y sus valores se corresponden con la clave principal de la misma u otra tabla.

Las claves ajenas se deben tomar muy en cuenta, ya que la integridad de la base de datos se mantiene o se elimina en una gran parte mediante las transacciones que se realizan sobre dichas claves.

3.5.5 INTERRELACIONES

Una interrelación es una asociación entre tablas mediante atributos que tienen el mismo dominio. Generalmente estos atributos hacen referencia a los mismos conceptos y la

interpelación se establece entre la clave ajena de una tabla (tabla hija), y la clave principal de la otra tabla (tabla padre). Por ejemplo: imaginemos una tabla "personal" (tabla padre) y la tabla "coches personal" (tabla hija). Esta interrelación se caracteriza por :

- Interrelación: "propiedad-coche"
- Tabla padre: "personal"
- Tabla hija: "coches-personal"
- Clave ajena: "rfc"

Todas las interrelaciones se deben identificar por un nombre, en nuestro caso "propiedad-coche".

3.5.6 VISTAS

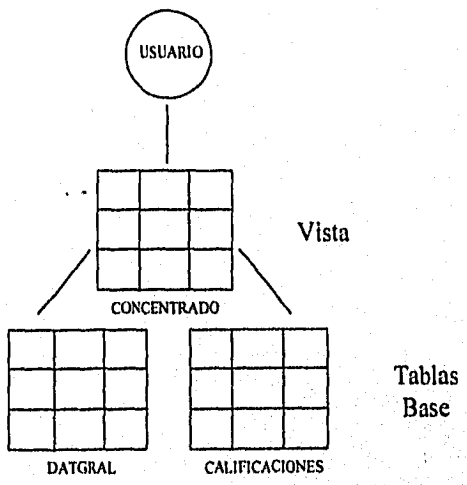
Debemos distinguir y aclarar dos conceptos, uno es el de una tabla base que es aquélla que se define, se crea y sus datos se encuentran almacenados físicamente en memoria auxiliar y otro tipo de tabla son las tablas ficticias o también conocidas como tablas vistas que son aquéllas que se forman a partir de una o más tablas base. Estas obtienen vistas parciales de datos para que los usuarios sólo accedan a determinada información.

Estas tablas vistas tienen las siguientes características:

- Sus columnas se obtienen a partir de múltiples tablas base e incluso pueden estar calculadas a partir de valores de las tablas base.
- Pueden estar definidas a partir de otras vistas.
- Sus datos se obtienen como resultado de realizar operaciones de recuperación (lectura) de datos.
- Se puede almacenar su definición (esquema o estructura) para una utilización posterior.

La vista es una tabla virtual que no existe en realidad como una tabla base en memoria auxiliar (disco duro, cinta, etc...); sólo se almacena, si se desea. Este tipo de tablas es una forma de ver determinados datos de tablas base. Por lo tanto, tal y como se manipulen las tablas base con modificaciones, inserciones y borrados, así quedarán afectadas las vistas que se tengan sobre ellas. Cabe señalar que los datos que se obtienen de una vista no son datos duplicados.

Diagrama de Tablas Vista, tablas ficticias.



Una tabla vista que vaya a admitir las operaciones de actualización (modificación, inserción y borrado) debe estar limitada por las siguientes restricciones:

- Debe estar derivada de una sola tabla base.
- Cada fila o tupla distinta de la vista se debe corresponder con una única tupla de la tabla base.
- Cada columna o atributo distinto de la vista se debe corresponder con un único atributo de la tabla base (no puede tener columnas calculadas).

Las principales ventajas que ofrecen las vistas son:

- La visión de los datos está simplificada para el usuario.
- Los mismos datos pueden verse de diferente manera desde distintos usuarios.
- Las vistas no quedan afectadas tras:
  - Aumento de otras tablas.
  - Aumento de la estructura con nuevos atributos.
  - reestructuración de la posición de los atributos de una tabla.
- Se aumenta la seguridad para aquélla información que no se desea mostrar.

3.6 MODELO ENTIDAD-RELACIÓN

Los modelos Jerárquico y de Red no son entendibles por los usuarios debido a que fueron orientados a las computadoras (Estructuras físicas para el almacenamiento óptimo), es por esta razón que viene el surgimiento del modelo relacional, pero este también es un modelo no óptimo ya que no esta orientado a la comprensión por parte del usuario, ya que requiere un alto grado de abstracción para imaginar la información en tablas normalizadas, es entonces cuando surge el modelo Entidad-Relación el cual fue propuesto por Peter Chen en 1976 en su escrito:
" The Entity-Relationship Model Toward a Unified View of Data" (Marzo-1976).

Este modelo es una técnica gráfica para representar la estructura lógica de una base de datos, incluye símbolos para representar entidades, atributos y relaciones, por lo que reduce ambigüedades de textos narrativos simplificando explicaciones, debido a las anteriores características es el modelo de datos más usado como herramienta de diseño lógico de bases de datos ya que aquí hay un análisis inmiscuido.

Este modelo visualiza el mundo real como un conjunto de objetos básicos llamados ***entidades y relaciones*** existentes entre estos objetos, así también nos facilita la tarea del

diseño de bases de datos, ya que con este modelo obtenemos un esquema de la estructura lógica global de la base de datos.

3.6.1 ENTIDAD Y CONJUNTO DE ENTIDADES

Una *entidad* es un objeto que existe en el mundo real y que es distinguible de los demás objetos en el universo, este puede ser concreto, como una persona, un libro o también puede ser abstracto, como un día festivo o un concepto. Así al conjunto de entidades del mismo tipo se le llama ***Conjunto de Entidades.*** Por ejemplo al conjunto de todas las personas que tienen una cuenta en un banco, puede definirse como el conjunto de entidades cliente, análogamente el conjunto de entidades cuenta podría representar el conjunto de todas las cuentas del banco.

Una entidad se representa por un conjunto de ***atributos*** y para cada atributo hay un conjunto de valores permitidos, llamados ***dominio.***

Ejemplo de atributos y dominio de una entidad cliente

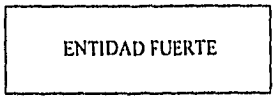
ATRIBUTOS	DOMINIO
Nombre-Cliente	Conjunto de todas las cadenas de texto de una cierta longitud
Seguridad-Social	Conjunto de todos los enteros positivos
Calle	Conjunto de todas las cadenas de texto de una cierta longitud
Ciudad-Cliente	Conjunto de todas las cadenas de texto de una cierta longitud

Cada entidad se describe por medio de pares (Atributo, Dominio). El concepto de un conjunto de entidades corresponde a la noción de definición de tipo en los lenguajes de programación, mientras que una variable corresponde al concepto de una entidad en el modelo E-R.

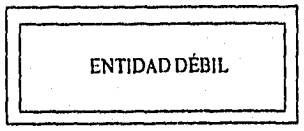
Las entidades de este modelo son de dos tipos:

- Entidades Fuertes
- Entidades Débiles

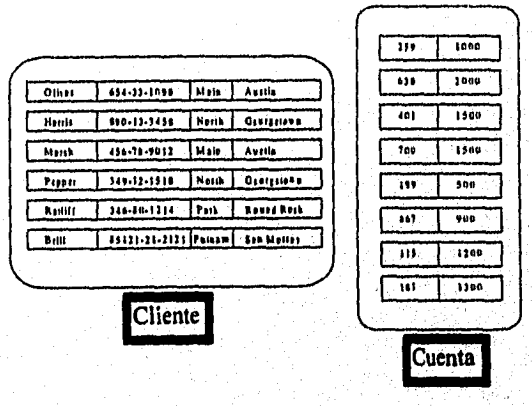
Las entidades fuertes son aquellas que tienen identificadores propios y únicos, o sea que tienen existencia propia y no dependen de otra entidad.



Las entidades débiles derivan su existencia de los atributos que son identificadores de una o más entidades fuertes.



Ejemplo de conjunto de entidades cliente y cuenta



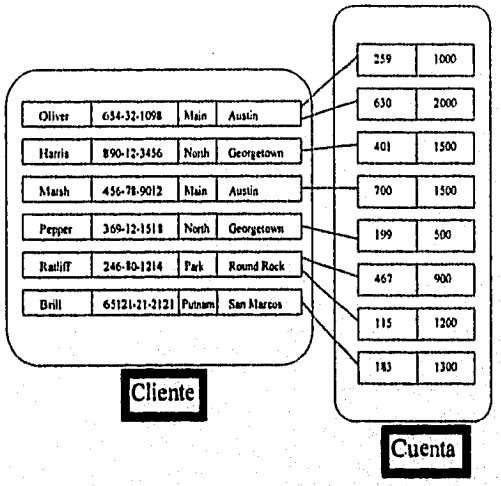
3.6.2 RELACIONES Y CONJUNTO DE RELACIONES

Se entiende por ***relación*** a la asociación entre varias entidades, así al conjunto de relaciones del mismo tipo se le llama ***conjunto de relaciones***.

Las relaciones en este modelo pueden ser entre dos entidades (Binaria) o también se permiten relaciones de una entidad consigo misma (unitarias).

Las relaciones tienen semántica, ya que indican la manera en que se asocian las entidades, cosa que en el modelo relacional no se podía ver con solo tablas.

Ejemplo de conjunto de relaciones que implican a los conjuntos de entidades cliente y cuenta



La mayoría de los conjuntos de relaciones en un sistema de bases de datos son conjunto de relaciones binarias (Implica a dos conjuntos de entidades), aunque ocasionalmente hay conjuntos de relaciones que implican más de dos conjuntos de entidades.

La función que juega una entidad en una relación se llama ***papel,*** estos generalmente son implícitos y no se suelen especificar, pero son útiles cuando el significado de una relación necesita ser clarificada.

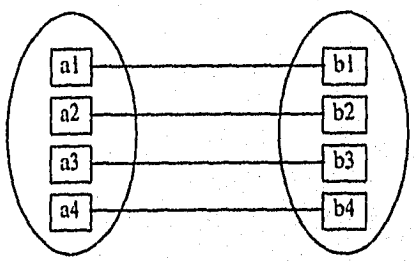
3.6.3 RESTRICCIONES DE ASIGNACIÓN (MAPPING)

En el modelo de base de datos E-R, la planificación del contenido de una base de datos debe cumplir ciertas restricciones a las que debe ajustarse como son:

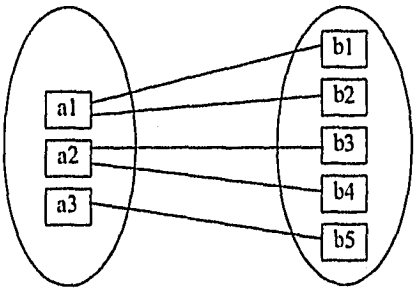
- Cardinalidad de asignación, la cual determina el número de entidades con las que puede asociarse otra entidad por medio de un conjunto de relaciones, este tipo de cardinalidad de asignación son eficientes cuando se describen conjuntos binarios de relaciones, aunque también contribuyen a la descripción de otro tipo de conjuntos de relaciones.

Cardinalidad de asignación de un conjunto de relaciones R, entre los conjuntos de entidades A y B, son las siguientes:

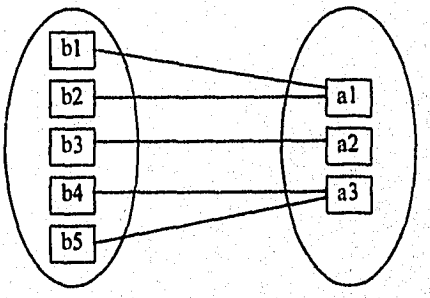
Una a Una: Una entidad A esta relacionada a lo máximo con una entidad en B, y una entidad en B esta relacionada a lo máximo con una entidad en A.



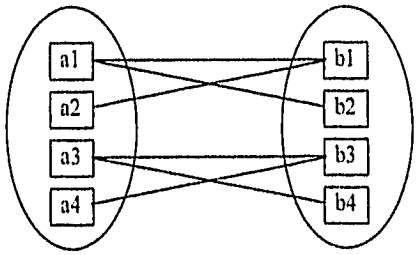
Una a Muchas: Una entidad en A esta relacionada con un número cualquiera de entidades en B. Una entidad en B sólo puede estar relacionada a lo máximo con una entidad en A.



Muchas a Una: Una entidad A esta relacionada a lo máximo con una entidad en B, Una entidad en B, esta relacionada con un número cualquiera de entidades en A.



Muchas a Muchas: Una entidad en A esta relacionada con un número cualquiera de entidades en B, y una entidad en B esta relacionada con un número cualquiera de entidades en A



Para las anteriores cardinalidades de asignación, la más adecuada para un conjunto de relaciones depende del mundo real que se este modelando. Un ejemplo de una relación de una a muchas es: Cuando en un banco cualquiera una cuenta, puede pertenecer únicamente a un cliente, y un cliente puede tener varias cuentas (De cliente a Cuentas).

- Dependencias de existencia: Esto es si la existencia de la entidad X depende de la entidad Y, entonces si se suprime Y, también se suprime X, por lo que se dice que la entidad Y es una entidad dominante y X una entidad subordinada.

La dificultad de definir entidades y relaciones se debe a lo que para un analista es una entidad, para otro pude ser una relación, esta confusión es común en el caso de las entidades que son eventos, como una venta, ya que una venta puede ser una entidad o una relación entre cliente y producto.

3.6.4 CONSIDERACIONES PARA ESTE MODELO DE DATOS

- Algo que no implique al menos dos elementos de datos para describirlo, probablemente no sea una entidad, si no un atributo.
- Las entidades son usualmente objetos o eventos
  ejemplo:
- clientes, proveedores, partes, productos, gente etc.
- Los objetos permanecen durante el tiempo
- Los eventos ocurren en un tiempo especificado

- Un evento puede representar la asociación entre dos o más objetos
- Un objeto puede tener dos o más eventos asociados con el

ejemplo:

El inicio y termino de un proyecto

La contratación y despido de un empleado

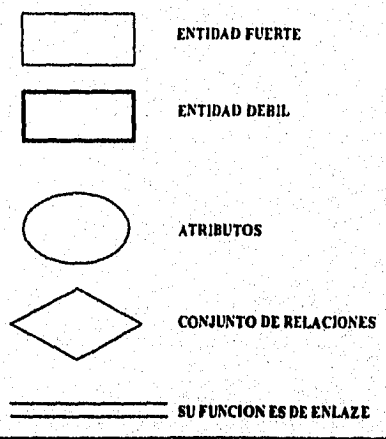
3.6.5 DIAGRAMA ENTIDAD RELACIÓN

Para obtener este diagrama se requiere de un análisis de la realidad, esta se concentra en entidades de datos, y muestran procesos y entidades de interés.

Su objetivo es mostrar las entidades que participan o interesan del mundo real, para ser expresadas por medio de datos, y como se relacionan dichas entidades.

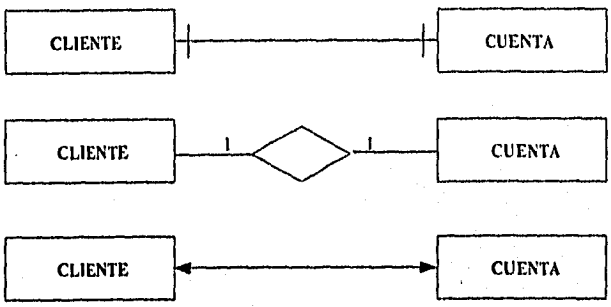
3.6.6 COMPONENTES DE UN DIAGRAMA ENTIDAD-RELACIÓN

Rectángulos con línea simple	Representan conjuntos de entidades fuertes
Rectángulos con línea doble	Representan conjunto de entidades débiles
Elipses	Representan atributos
Rombos	Representan conjunto de relaciones
Líneas	Enlazan atributos a conjunto de entidades y conjunto de entidades a conjunto de relaciones

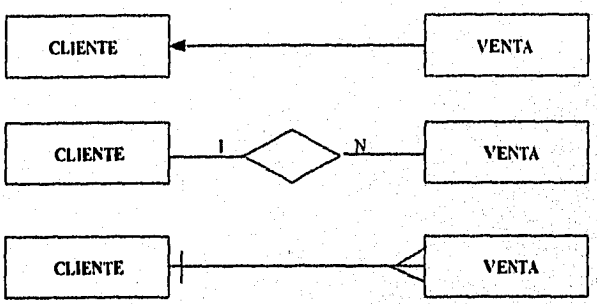


3.6.7 REPRESENTACIÓN DE LAS RELACIONES EN UN DIAGRAMA ENTIDAD-RELACIÓN

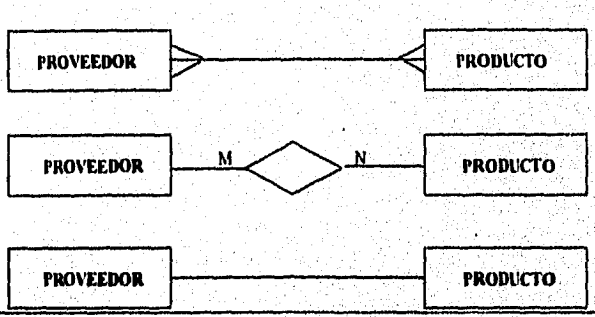
Una a Una



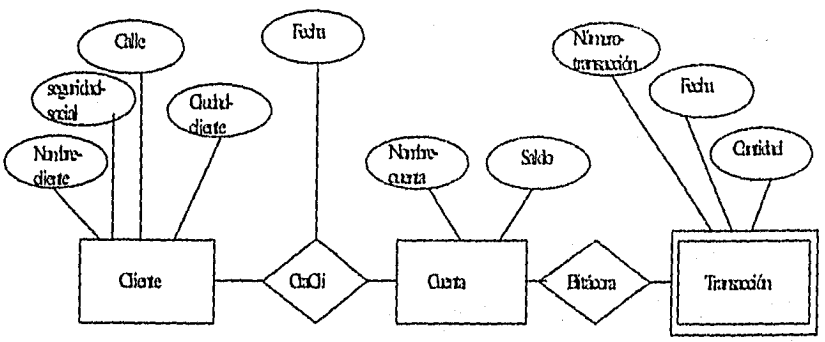
Una a Muchas



Muchas a Muchas



EJEMPLO DE UN DIAGRAMA ENTIDAD-RELACIÓN



3.6.8 CONSTRUCCIÓN DE UN DIAGRAMA ENTIDAD-RELACIÓN

Para la construcción de un diagrama Entidad-Relación se siguen los siguientes pasos:

1) Identificar y listar las entidades de interés acerca de las cuales queremos almacenar datos.
   Se construye un bloque para cada entidad identificada y se distribuyen espaciadamente.
   Los nombres de las entidades se ponen dentro del bloque y deben ser en singular.
2) Se identifican las asociaciones de cada entidad con respecto a las demás, y se dibuja la relación entre ellas.
3) En caso de existir relación entre dos entidades es necesario verificar la conectividad o sea la manera en que se asocian.
   La asociación se hace analizando para cada elemento de una entidad, cuantos elementos de la otra entidad están relacionados y en sentido contrario.
   Los tipos de relaciones de acuerdo a su conectividad pueden ser:

   Uno a Uno
   Uno a muchas

Muchas a Muchas

4) Se analiza Opcionalidad, algunas veces, dos entidades "pueden" estar relacionadas pero no con todas sus ocurrencias, esto es una relación opcional. Es importante definir las relaciones opcionales cuando se busca cumplir con la integridad referencial.

   Ejemplo:

   Empleado <------> Proyecto

3.6.9 REDUCCIÓN DE LOS DIAGRAMAS ENTIDAD-RELACIÓN A TABLAS

Ya obtenido el diagrama E-R este puede ser expresado por medio de tablas, para cada conjunto de entidades y conjunto de relaciones, existe una tabla única a la que se le asigna el nombre del conjunto de entidades o del conjunto de relaciones.

Ejemplo de reducción del diagrama Entidad-Relación a tablas del diagrama E-R anterior

Número-cuenta	Saldo
259	1000
630	2000
401	1500
700	1500
199	500
467	900
115	1200
183	1300

Número-cuenta	Número-transacción	Fecha	Cantidad
259	5	11 de mayo 1990	+50
630	11	17 de mayo 1990	+70
401	22	23 de mayo 1990	-300
700	69	28 de mayo 1990	-500
199	103	3 de junio 1990	+900
467	6	7 de junio 1990	-44
115	53	13 de junio 1990	+120
183	104	17 de junio 1990	-200

Nombre-cliente	Seguridad-social	Calle	Ciudad-cliente
Oliver	654-32-1098	Main	Harrison
Harris	890-12-3456	North	Rye
Marsh	456-78-9012	Main	Harrison
Pepper	369-12-1518	North	Rye
Ratliff	246-80-1214	Park	Pittsfield
Brill	121-21-2121	Putnam	Stamford
Evers	135-79-1357	Nassau	Princeton

Seguridad-social	Número-cuenta	Fecha
654-32-1098	259	11 de mayo 1990
890-12-3456	630	17 de mayo 1990
456-78-9012	401	23 de mayo 1990
369-12-1518	700	28 de mayo 1990
246-80-1214	199	3 de junio 1990
121-21-2121	467	7 de junio 1990
135-79-1357	115	13 de junio 1990

Seguridad-social	Número-cuenta	Nombre-sucursal
654-32-1098	259	Downtown
890-12-3456	630	Redwood
456-78-9012	401	Perryridge
369-12-1518	700	Downtown
246-80-1214	199	Mianus
121-21-2121	467	Round Hill
135-79-1357	115	Pownal

CAPÍTULO IV

MODELO ORIENTADO

A

OBJETOS

Objetivo:

Investigar los conceptos relacionados con la orientación a objetos, para conocer sus características e identificar el modelo OO, describiendo dicho modelo.

IV.- MODELO ORIENTADO A OBJETOS

4.1 ANTECEDENTES DE LA ORIENTACIÓN A OBJETOS

La orientación a objetos tiene su origen en Noruega a finales de los años 60 con un lenguaje llamado ***Simula67*** el cual fue desarrollado por Krsiten Nygaard y Ole-Johan Dahl, simula67 introdujo los conceptos de ***clases, corrutinas y subclases*** que son muy parecidas a los lenguajes orientados a objetos de nuestros tiempos, también demostró el poder de modelación de un lenguaje de programación basado en clases, así como la idea de que ***los datos y operaciones debían almacenarse juntos.***

Posteriormente al surgimiento de simula67, a mitad de los años 70, los científicos del Centro de Investigaciones Palo alto de Xerox (Xerox Parkc) desarrollaron el lenguaje ***Smalltalk***, que fue el primer lenguaje orientado a objetos, más puro, consistente y completo, pero debido a las necesidades de plataformas especializadas de calculo que necesitaba smalltalk hizo poco atractivo económicamente el empleo de este lenguaje a los desarrolladores de software comercial y empresarial, por tal motivo vino el fracaso de apreciar la temprana aparición del paradigma de la orientación a objetos.

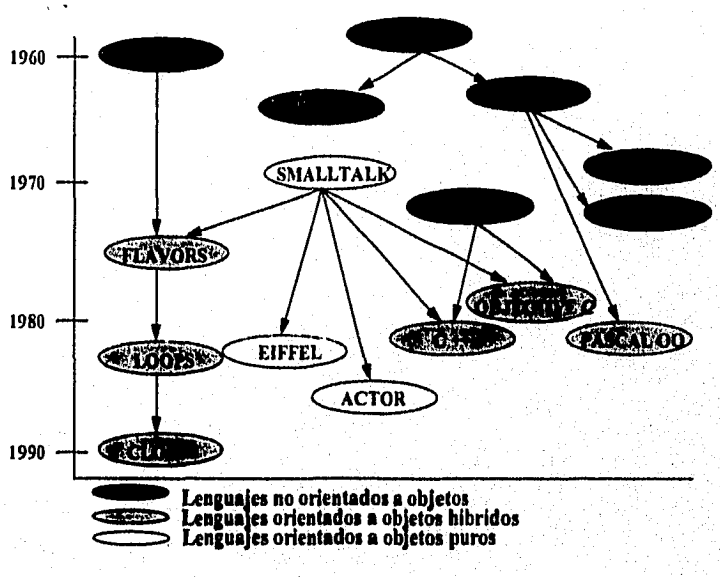
Debido a las crecientes necesidades de software que se requiere para satisfacer las necesidades actuales se hizo indispensable volver a retomar el paradigma de la orientación a objetos, que apareció con aquellos lenguajes como simula67 y smalltalk.

En los años 80 **C** era el lenguaje de desarrollo más popular en todos los entornos informáticos y a principios de la década de los años 80, Bjame Stroustrup de los laboratorios de Bell de AT&T amplio el lenguaje **C** para crear **C++** que era un lenguaje que soportaba la programación orientada a objetos y que los programadores de C les era más fácil aprender el paradigma orientado a objetos sin tener que invertir mucho tiempo en volver a aprender por completo un nuevo lenguaje orientado a objetos.

Así también los obstáculos que se presentaban en lo que respectaba a precio/rendimiento que evitaban el uso de la tecnología orientada a objetos cayeron debido a los grandes avances tecnológicos que se habían dado en el Hardware y con la introducción de los entorno integrados orientados a objetos como la Macintosh en 1984 y la máquina

NeXT en 1988 fueron los hechos más sobresalientes en la aparición de la orientación a objetos en la corriente principal de la informática.

Los lenguajes orientados a objetos cuando aparecieron no se les dió la importancia y trascendencia que les correspondía, debido a esto surgieron nuevos tipos de programación que no eran orientados a objetos sino procedimentales, pero debido a las necesidades de software que eran cada vez más exigentes se volvió a retomar el paradigma orientado a objetos, y con la extensión de C para crear C++ aparecieron los ***lenguajes orientados a objetos híbridos*** que son una extensión de los lenguajes procedimentales para que incorporaran la tecnología orientada a objetos, debido al éxito que estaba presentando este paradigma, en la siguiente figura se muestra como aparecieron los lenguajes orientados a objetos puros, los lenguajes orientados a objetos híbridos y los lenguajes no orientados a objetos.

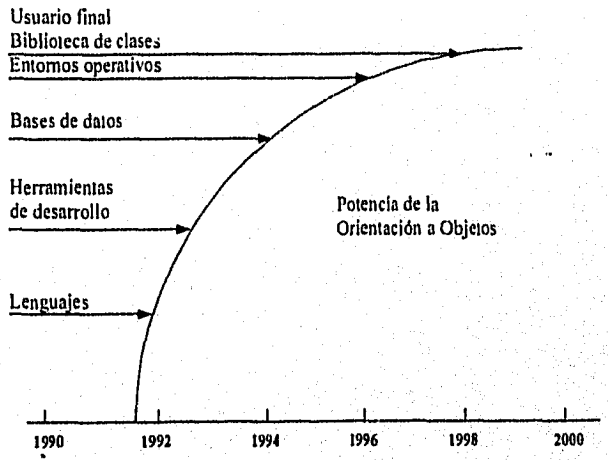


Aunque el paradigma de la orientación a objetos estuvo en sombras por un gran periodo de años, hoy en día muchos componentes de software como son:

- Lenguajes
- Interfaces de usuario
- Bases de datos
- Sistemas operativos

Se están encaminando hacia la orientación a objetos. Ejemplos del éxito de la orientación a objetos son las ampliaciones de los lenguajes como pascal, cobol etc., para que puedan soportar el manejo y uso de la orientación a objetos.

La siguiente figura muestra una secuencia para la integración total hacia la orientación a objetos:

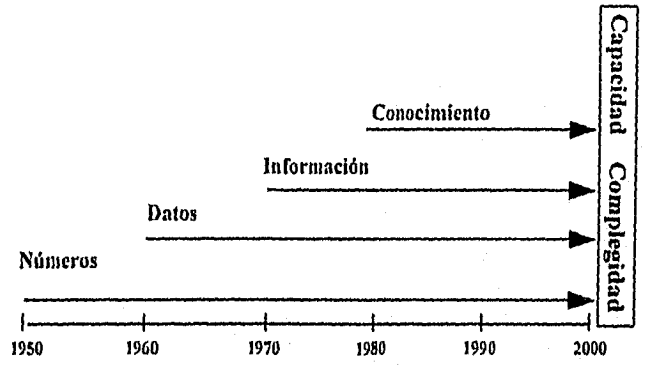


Las fuerzas que impulsan y guían la orientación a objetos son las siguientes:

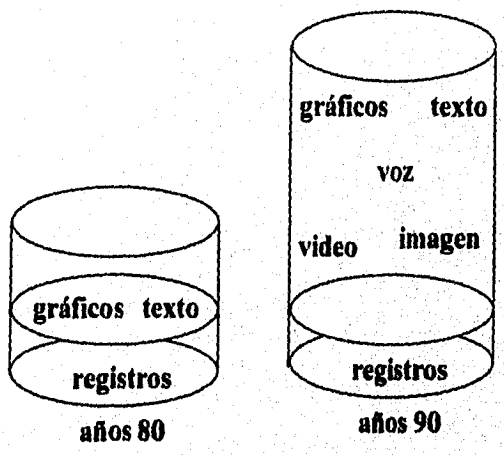
- Las herramientas de desarrollo
- Los entornos de sistemas y aplicaciones que permitan la información multimedia
- La computación del usuario final
- El procesamiento distribuido

Los sistemas han crecido y continúan creciendo en complejidad y no solo en lo que respecta a la funcionalidad si no que también están creciendo los diferentes tipos de datos

que manejan ya que los sistemas futuros procesaran seguramente imagen, voz y vídeo además de texto y números, como muestra la siguiente figura, la cual muestra la creciente complejidad del software.



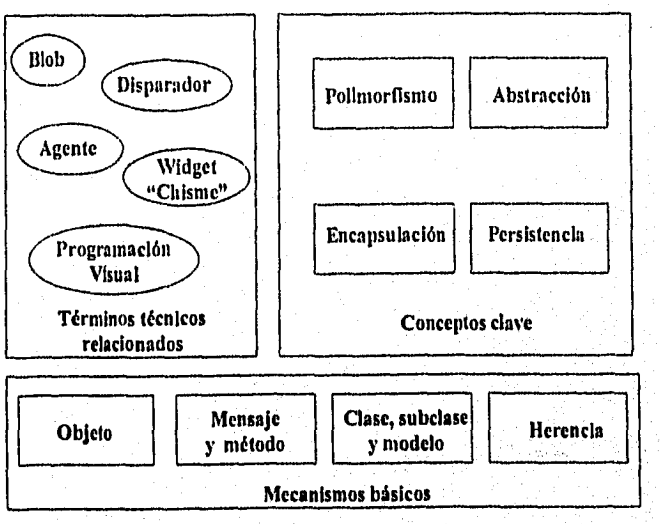
La siguiente figura muestra como eran los componentes que se requerían para los años 80 y 90.



Uno de los puntos que permitirá aumentar la rapidez con que se desarrollan sistemas son las cuidadosas librerías de objetos que serán desarrolladas por programadores especializados y las cuales apoyarán a los programadores menos sofisticados que deseen ensamblar aplicaciones rápidamente a partir de secciones prefabricadas.

4.2 CONCEPTOS BÁSICOS DEL MODELO ORIENTADO A OBJETOS

El modelo orientado a objetos se basa en ver al mundo como un conjunto de entidades y objetos los cuales están relacionados entre si, y que se comunican unos con otros por medio de mensajes, este modelo hace énfasis en los objetos y no en los procesos, la siguiente figura muestra los pilares básicos de la orientación a objetos.



4.2.1 OBJETO

Los conocimientos que hemos adquirido a través del tiempo nos permiten sentir y razonar respecto a las cosas que existen en el mundo. A estas cosas se les llama ***objetos***, los cuales pueden ser reales o abstractos, lo siguiente son ejemplos de objetos:

- Una factura
- Una organización
- Una figura en un programa de dibujo
- Una pantalla con la que interactúa el usuario
- Un campo o nodo de la pantalla de una herramienta CASE
- Un mecanismo en un dispositivo de robótica
- Todo un plano de ingeniería
- Un componente de un plano de ingeniería
- Un texto y fotografía utilizados en una plana de un periódico
- Un avión
- El vuelo de un avión
- Una reservación aérea
- Un icono en la pantalla al que un usuario puede apuntar y "abrir"
- Un proceso para llenar un pedido
- El proceso para escribir esta línea

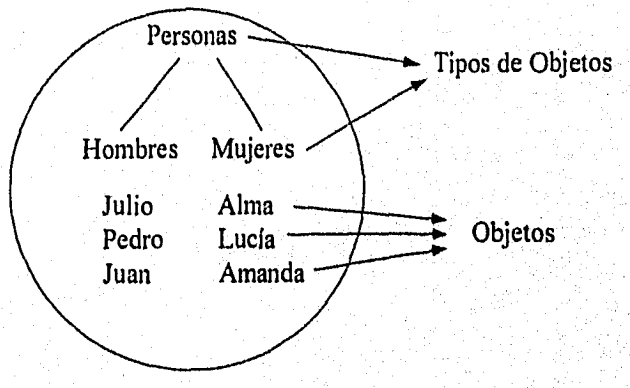
Los objetos son entidades que contienen en su interior datos y las instrucciones que operan sobre esos datos. Los objetos pueden ser de dos tipos en una aplicación, los cuales son:

- **Objeto pasivo: Es aquél que solamente actúa cuando se le hace una petición, es decir cuando el botón se pulsa con el mouse en algún programa de aplicación.**

- Objeto activo: Es aquél que va siguiendo los sucesos que ocurren en una aplicación y actúa de forma autónoma, denominado a veces como ***agente***, generalmente alertan a los usuarios respecto a una operación que se encuentra mal o hacer una operación después de otra. Este tipo de objeto se asimila mucho a un virus de computadora ya que este actúa de forma autónoma pero en vez de destruir ayuda a que los problemas se corrijan.

4.2.2 TIPOS DE OBJETO

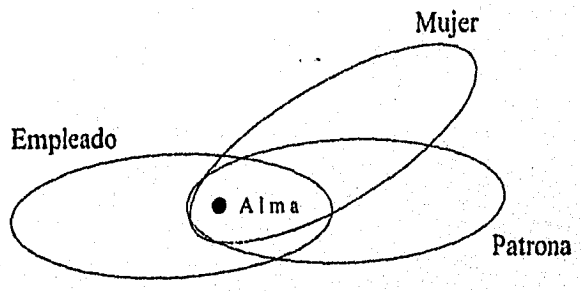
Los conocimientos y conceptos que tenemos sobre las cosas nos permiten distinguir características entre un objeto y otro, a esta distinción se le conoce como tipo de objeto. Los tipos de objetos son categorías de objetos, es decir, un grupo generalizado de determinados objetos. En la siguiente figura se muestra lo anterior:



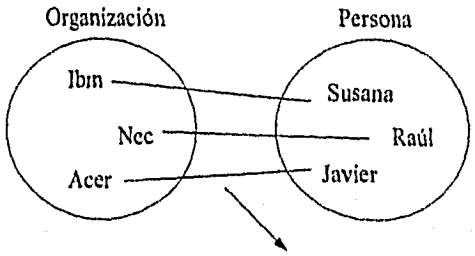
Otros ejemplos de tipos de objetos pueden ser:

Intangible	Relación	Eventos	Concretos
Tiempo	Matrimonio	Venta	Persona
Calidad	Sociedad	Compra	Animal
Compañía	Propiedad	Falla del sistema	Automóvil

Un objeto también se puede categorizar en más de una forma: Por ejemplo, el objeto Alma del ejemplo anterior, puede ser parte de un tipo de objeto llamado patrona, empleado, mujer, dueña, etc... Por que su jardinero la considera su patrona, su jefe la ve como un empleado, y la sociedad como una mujer.



Existen asociaciones entre objetos de un determinado tipo de objeto con los objetos de otro tipo, esto se muestra en la siguiente figura:



Un tipo de objeto puede ser supertipo de otro o tener subtipos, sub-subtipos, sub-sub-subtipos, etc. Por ejemplo hombre es un subtipo de persona y persona es supertipo de hombre.

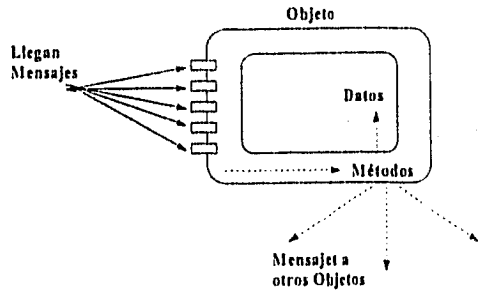
4.2.3 INSTANCIAS

Una instancia es el valor de un objeto, es decir los datos almacenados en ese objeto. por ejemplo, del tipo de objeto empleados, algunas instancias serían Gerardo González Hernández, Norma Alamilla López, etc...

4.2.4 MÉTODO

Los métodos son las operaciones que determinan la forma en que se controlan los datos de los objetos y nos representan los diferentes tipos de comportamiento de estos. Cada uno de los métodos es una parte de código de todo un sistema.

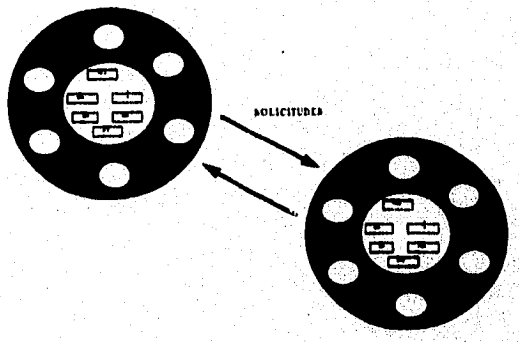
Los métodos se encuentran dentro del objeto y determinan como va a actuar el objeto cuando reciba un mensaje ya que al ser recibido un mensaje se ejecuta un determinado método el cual solamente puede modificar los datos que se encuentran dentro del objeto o puede ser que el método envíe mensajes hacia otros objetos, ya que los métodos determinan el comportamiento que van a tener los objetos, como se muestra en la siguiente figura. .



Anatomía y funcionamiento de un Objeto

4.2.5 MENSAJES

Son solicitudes para que uno o varios objetos lleven a cabo una tarea o función especifica, estas hacen que se ejecute una operación determinada, y la operación determina que métodos se van a ejecutar. La solicitud contiene el nombre del objeto, nombre de la operación y algunas veces un grupo de parámetros.



4.2.6 CLASES Y SUBCLASES

Las clases son una implantación de un tipo de objeto en software. Especifica una estructura de datos y los métodos operativos permisibles que se aplican a cada uno de los

objetos de esta clase, las clases son una descripción de un conjunto de objetos casi idénticos debido a que muchos objetos diferentes actúan de forma muy similar.

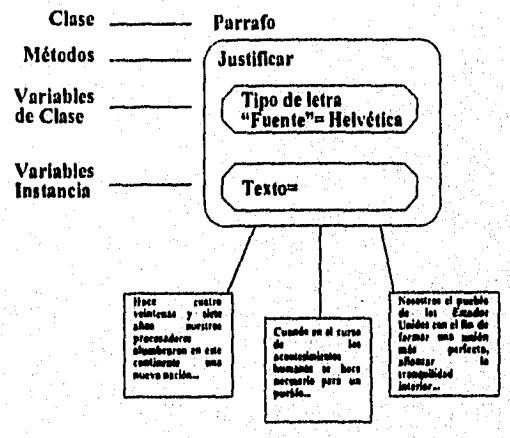
Las clases están formadas de:

- Métodos
- Datos: Los cuales pueden ser de dos tipos:
  - Variables de clase: Tienen valores almacenados en una clase
  - Variables Modelo o de Instancia: Tiene valores asociados únicamente con cada instancia u objeto creado a partir de una clase.

Estas dos características resumen las características comunes de un conjunto de objetos.

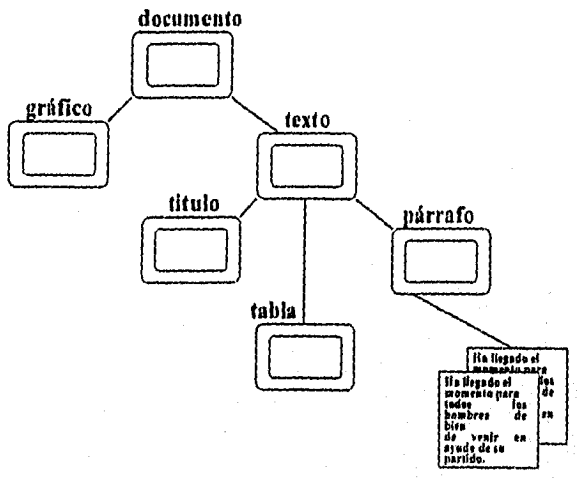
La definición de una clase nos permite crear código reutilizable para evitarnos de esta manera volverlo y volverlo a escribir una y otra vez.

Un objeto es un modelo o instancia de una clase, y este es creado cuando recibe un mensaje solicitando creación por la clase padre por lo que este toma los métodos y datos de su clase padre.



Ejemplo de una Clase párrafo con tres instancias

Una clase puede también resumir elementos comunes para un conjunto de subclases como se muestra en las subclases de la clase documento en la siguiente figura.



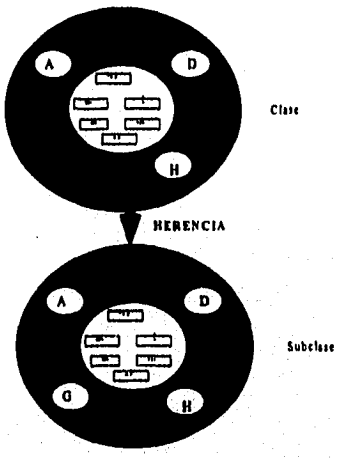
4.3.7 HERENCIA

Al igual como nosotros heredamos de nuestros padres rasgos y características, nuestros hijos heredarán ciertos rasgos y características de nosotros, así también existe en el software este concepto.

Una subclase hereda propiedades de una clase; una sub-subclase hereda propiedades de la subclase, los tipos de propiedades que se pueden heredar en software son:

- La estructura de datos
- Los métodos

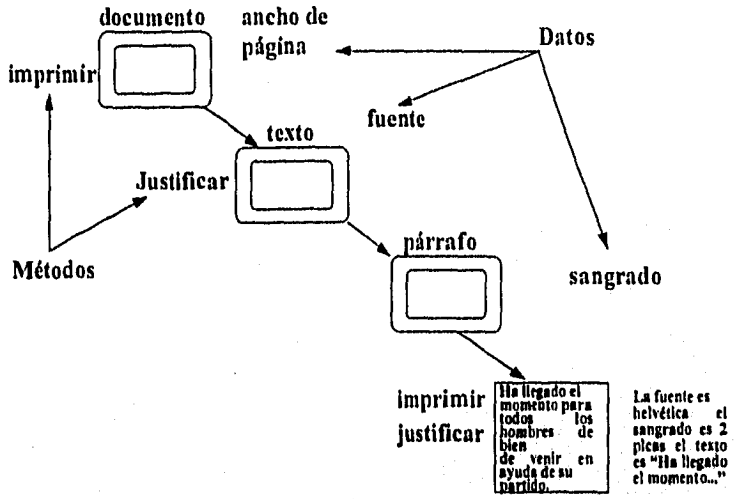
Las subclases y sub-subclases pueden heredar las anteriores propiedades, pero también tienen sus métodos e incluso sus propios tipos de datos.



Una clase puede tener sus propios métodos y estructuras de datos, así como también heredarlos de una superclase.

La herencia es el medio por el cual podemos compartir automáticamente métodos y datos entre las clases, subclases y objetos, así como también la herencia nos permite crear clases programando solamente las diferencias con la clase padre.

Por ejemplo cuando se declara a un párrafo como una subclase de texto todos los métodos y variables modelo asociados con texto son heredados automáticamente por párrafo. Si la clase texto tiene métodos que no son adecuados para la subclase párrafo, entonces estos se pueden obviar y escribir nuevos métodos y almacenarlos como parte de la clase párrafo.



Ejemplo: un modelo hereda de sus clases padre

En la programación orientada a objetos se usan dos tipos de herencia los cuales se mencionan a continuación:

- Herencia Simple. Una clase puede heredar datos y métodos de una clase simple así como añadir o sustraer comportamiento por si misma.
- Herencia Múltiple. Es la posibilidad de una subclase de adquirir los datos y métodos de más de una clase.

4.3.8 POLIMORFISMO

Este fenómeno se origina cuando un mismo mensaje provoca acciones diferentes al ser recibido por diferentes objetos, ya que con el polimorfismo se puede enviar un mensaje genérico y dejar los detalles de realización a los objetos receptores, por ejemplo el mensaje

imprimir al ser recibido por una figura o un diagrama ejecutará diferentes métodos de impresión.

4.3.9 ABSTRACCIÓN

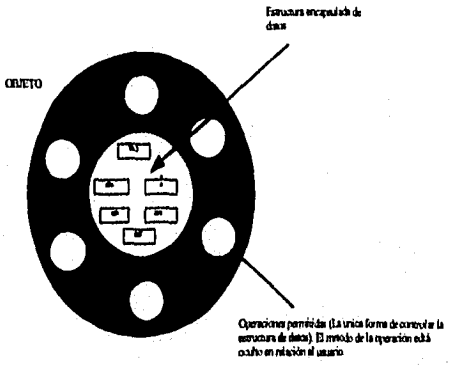
La orientación a objetos recomienda que se piense en las aplicaciones en términos abstractos, esto se hace con el fin de que al momento de diseñar un sistema se tome muchos aspectos importantes como son:

- Factores de comportamiento común
- Situar los objetos en superclases abstractas
- Las bibliotecas de clases
- La herencia

Ya que todo esto facilita el trabajo de programación, debido a que se encuentra más código reutilizable.

4.3.10 ENCAPSULADO

Se llama así al empaquetamiento conjunto de los datos y métodos. Cada objeto esconde sus datos de los demás objetos y solo permite el acceso a sus datos mediante sus propios métodos, esto permite ocultar al usuario los detalles de la implantación del objeto, ya que solamente los usuarios pueden ver que el objeto responde a ciertas solicitudes pero desconocen la forma interna en que esta construido el objeto. El encapsulado evita la corrupción de los datos de los objetos.



Cada objeto encapsula una estructura de datos y métodos. Las estructuras de datos sólo pueden ser utilizadas por dichos métodos.

4.3.11 PERSISTENCIA

Se refiere al tiempo en el cual se asigna espacio y mantiene accesible en memoria de la computadora un objeto, los objetos almacenados permanentemente se denominan persistentes.

4.3.12 CARACTERÍSTICAS DE LAS TÉCNICAS ORIENTADAS A OBJETOS

- Cambian nuestra forma de pensar sobre los sistemas
- Los sistemas suelen construirse a partir de objetos ya existentes
- La complejidad de los objetos sigue en aumento, ya que unos objetos se construyen a partir de otros
- Creación de sistemas con un funcionamiento correcto y más fácil

- Las técnicas OO (orientadas a objetos) se ajustan de manera natural a la tecnología CASE.

4.3.13 BENEFICIOS DE LA TECNOLOGÍA ORIENTADA A OBJETOS

- Reutilización de clases
- Estabilidad. Esta surge cuando una clase se utiliza muy frecuentemente
- El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel
- Se construyen clases cada vez más complejas
- Confiabilidad
- Nuevos mercados para el software
- Un diseño más rápido
- Diseño de mayor calidad
- Programación más sencilla
- Mantenimiento más sencillo
- Invención
- Esmero durante la construcción
- Modelos más realistas
- Mejor comunicación entre los profesionales de los sistemas de información y los empresarios
- Imágenes, vídeo y expresión
- **Independencia del diseño**
- **Interacción. El software de varios proveedores puede funcionar como conjunto**
- **Computación cliente-despachador**
- **Computación de distribución masiva**
- **Computación paralela**
- **Mayor nivel de automatización de las bases de datos**

- Eficacia de la máquina
- Mejores herramientas CASE
- Bibliotecas de clases para las industrias y empresas

4.3 BASES DE DATOS ORIENTADAS A OBJETOS

4.3.1 INTRODUCCIÓN

En sus principios las Bases de datos fueron pensadas para soportar aplicaciones que procesaran grandes volúmenes de información uniformemente estructurados con una pequeña estructura interna o sin ella y no pensaron que ese tipo de bases de datos en el futuro no iban a cumplir con las necesidades que fueran a surgir en los años venideros. Debido a que las nuevas aplicaciones de nuestros tiempos requieren soporte para estructuras de datos extensibles y complejas, acceso a datos complejos, alto rendimiento, así como también debido al resurgimiento de la programación orientada a objetos y a su tremendo impacto que ha causado actualmente. Fue por esta razón que vino el surgimiento de las Bases de Datos Orientadas a Objetos, ya que estas capturan la estructura de los datos.

Las aplicaciones técnicas y de ingeniería fueron las primeras en requerir bases de datos que manipularan tipos de datos complejos y capturaran las estructuras de los datos. Llevadas por estas diversas necesidades de aplicación, a finales de los años 80, comenzaron a parecer Bases de Datos Orientadas a Objetos, muy pocas comerciales y muchas otras basadas en investigación. La mayoría de estas iniciales bases de datos Orientadas a Objetos eran utilizadas en aplicaciones basadas en el diseño en los campos científicos y de ingeniería.

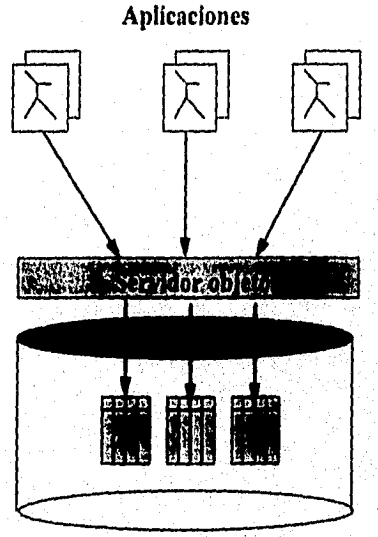
Debido a que en sus principios las Bases de Datos Orientadas a Objetos tenían características que limitaban su empleo (complejidad de uso) y a que en ese momento el SQL era el estándar del área relacional, tuvieron muy poco éxito. Pero a pesar de eso las Bases de Datos Orientadas a Objetos siguieron luchando y para lograr introducir en el

mercado las bases de datos orientadas a objetos extendieron las bases de datos relacionales para que aceptaran la tecnología orientada a objetos, logrando así la cualidad *híbrida* de las bases de datos relaciónales existentes para acomodar así buena parte de la funcionalidad de las Bases de Datos Orientada a Objetos.

Las extensiones de objetos para el modelo relacional centran la optimización para las aplicaciones comerciales en el rendimiento, la presentación física de los objetos para tratar con un entorno heterogéneo, y la ampliación del modelo SQL para incluir en este el paradigma Orientado a Objetos. Estos esfuerzos han sido evidentes en el trabajo actual de los vendedores establecidos de bases de datos relacionales.

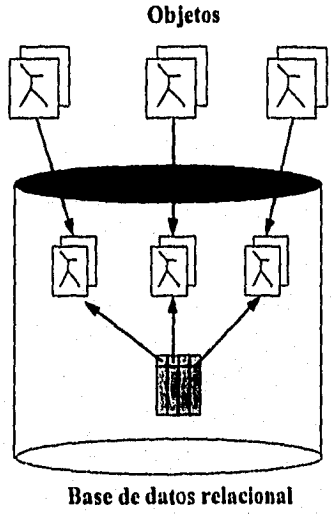
Estas extensiones del modelo de base de datos relacional para soportar objetos han tomado uno de los dos modelos siguientes:

- Permitir que los mismos objetos actúen como tablas relacionales



Ejemplo de base de datos relacional con objetos reducidos en tablas

- Permitir que la tabla relacional contenga punteros hacia los objetos.



Ejemplo de base de datos relacional con tablas apuntando a objetos

En las siguientes figuras se muestra la evolución de la tecnología de las bases de datos (figura A) y el volumen, complejidad y variedades de bases de datos (figura B).

Figura A

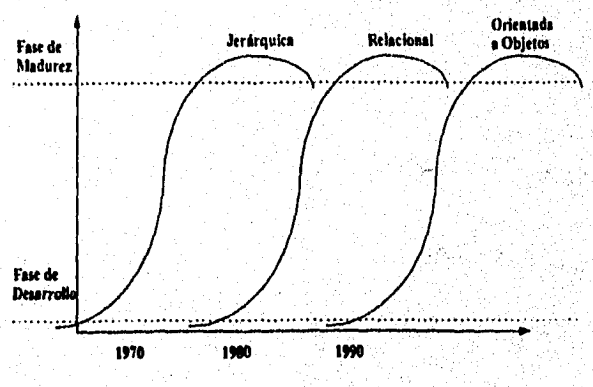
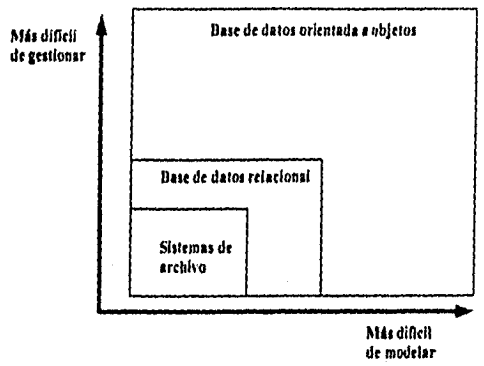


Figura B



Las extensiones Orientadas a Objetos de las bases de datos relacionales son la dirección que están tomando las compañías de bases de datos relacionales para introducir poco a poco las bases de datos orientadas a objetos puras en los años venideros.

Si las bases de datos relacionales se siguen ampliando para soportar aplicaciones más complejas es probable que las Bases de Datos Orientadas a Objetos no reemplacen a corto plazo a los sistemas relacionales.

4.3.2 FUNCIONALIDAD DE LAS BASES DE DATOS ORIENTADAS A OBJETOS

Las Bases de Datos Orientadas a Objetos ofrecen la misma funcionalidad que los lenguajes orientados a objetos, estas permiten:

- La encapsulación dentro de objetos de los datos y métodos que actúan sobre ellos
- Permiten la declaración de relaciones jerárquicas entre los objetos a través del uso de la herencia.
- Activan métodos mediante mensajes a los objetos

- Ofrecen la persistencia y participación cosa que los lenguajes orientados a objetos carecen. Ya que aquí estamos hablando de una base de datos la cual almacena por largo tiempo datos.

Las Bases de datos Orientadas a Objetos proporcionan almacenamiento central para los datos, mecanismos para compartir dichos datos entre los usuarios y formas de asegurar la integridad y seguridad de los datos. La tecnología de Bases de Datos ha estado luchando por encontrar formas de continuar proporcionando ventajas de almacenamiento de datos complejos y diversos.

Al igual que en la Programación Orientada a Objetos, las Bases de Datos Orientadas a Objetos intentan reducir la complejidad aumentando la capacidad de los fundamentos básicos del Software. Las Bases de Datos Orientadas a Objetos no almacenan datos en forma aisladamente si no que siguen el paradigma de la Programación Orientada a Objetos de vincular los datos junto con el comportamiento asociado dentro de los objetos. Las capacidades funcionales de las Bases de Datos Orientadas a Objetos incluyen soporte en construcciones ricas en el modelo de datos, soporte directo de inferencia o deducción y la posibilidad de almacenar tipos de datos complejos y diversos, como por ejemplo:

- Imágenes
- Audio o voz
- Vídeo

Una base de datos orientada a objetos es una base de datos inteligente, almacena datos y métodos, está diseñada para ser eficaz, desde el punto de vista físico, para almacenar objetos complejos. Evita el acceso a los datos si esto no se realiza mediante los métodos almacenados en ella.

4.3.3 ELEMENTOS BÁSICOS DE LAS BASES DE DATOS ORIENTADAS A OBJETOS

La tecnología de Bases de datos Orientadas a Objetos depende de varias características importantes que van más allá de las que ofrecen los lenguajes orientados a

objetos debido a que son bases de datos y no lenguajes orientados a objetos, estas características son:

- ***Persistencia***
- ***Integridad***
- ***Comparición***
- ***Consulta***

Persistencia: Creando objetos que sobreviven al proceso que los creó.

Integridad: Garantizando que los cambios en las bases de datos no destruyen su consistencia.

Comparición: Permitiendo procesos múltiples para coordinar el acceso simultáneo a los objetos de la base de datos.

Consulta: Utiliza expresiones lógicas para definir un subconjunto de la base de datos para su acceso.

Las bases de datos orientadas a objetos tienen características de las bases de datos normales junto con las características que tiene el paradigma orientado a objetos, como se muestra en el siguiente cuadro.

4.3.4 CARACTERÍSTICAS DE LAS BASES DE DATOS ORIENTADAS A OBJETOS

Objetos	**Persistencia**
Métodos	**Integridad**
Herencia	**Compartición**
	Consulta

Los lenguajes orientados a objetos y las Bases de Datos Orientadas a Objetos son complementos naturales uno del otro. Los lenguajes enfatizan el procesamiento, la estructuración compleja y los datos locales, las bases de datos se centran en un método más

declarativo, datos compartidos fuera del dominio de las aplicaciones y soporte para grandes cantidades de datos.

Uno de los objetivos, tanto de los lenguajes orientados a objetos como para las Bases de Datos Orientadas a Objetos, es crear una fusión limpia entre ellos, conservando no obstante, sus ventajas individuales.

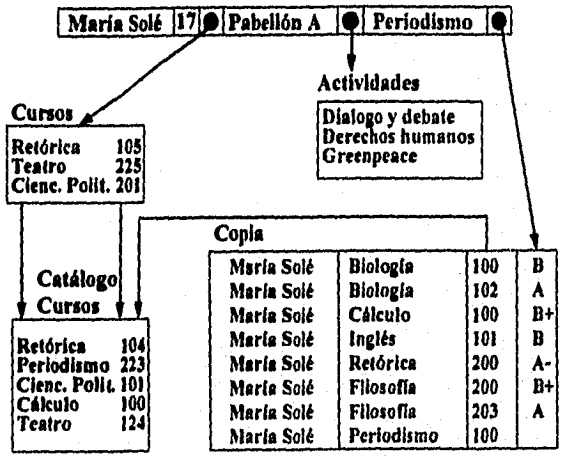
Las Bases de datos Orientadas a Objetos difieren del funcionamiento de las bases de datos relaciónales en los siguientes puntos:

- Las bases de datos relaciónales se basan en derivar una estructura virtual en la ejecución basada en los valores de los conjuntos de datos almacenados en tablas. Las Bases de Datos Orientadas a Objetos contienen objetos predefinidos que no necesitan derivarse en el momento de la ejecución.
- En una base de datos relacional, las vistas se construyen seleccionando datos de múltiples tablas y cargándolos en una única tabla. En una Base de Datos Orientada a Objetos una vista se obtiene pasando punteros de objeto en objeto.
- Una Base de datos Orientada a Objetos juega un papel muy activo, mientras que una base de datos relacional es pasiva.
- Mientras la base de datos relacional ofrece principalmente la capacidad de añadir o borrar registros, las Bases de Datos orientadas a Objetos ofrecen la capacidad de incorporar métodos dentro de los objetos, permitiendo así a la base de datos incorporar muchas de las operaciones que deban dejarse a la aplicación con una base de datos relacional.

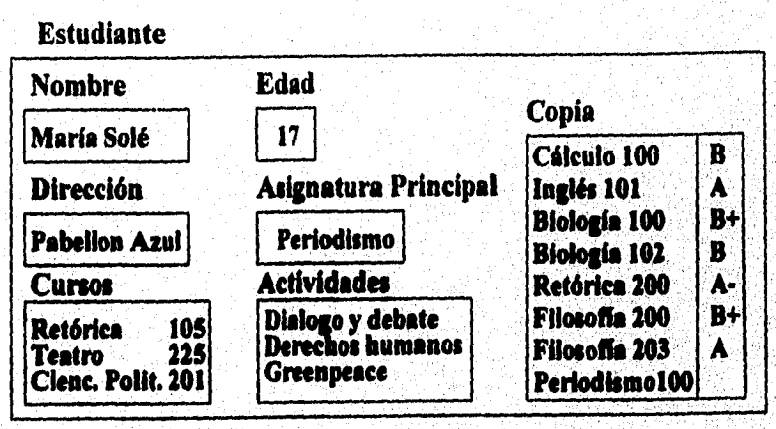
4.3.5 OBJETOS

Los objetos son elementos que una Base de Datos Orientada a Objetos almacena, modifica y recupera por orden del programa de la aplicación. estos objetos pueden ser tan

simples o tan complejos, además de que tienen cabida los tipos múltiples de datos incluyendo gráficos, sonido y vídeo. La ventaja mas importante de las Bases de Datos Orientadas a Objetos de la bases de datos tradicionales es que pueden representar como objeto cualquier entidad del mundo real.



Ejemplo de datos académicos almacenados en tablas relaciónales



Ejemplo de datos académicos almacenados como un conjunto de objetos relacionados

Los objetos pueden contener otros objetos en cualquier nivel de anidamiento. Debido a esta razón existe una tremenda flexibilidad al crear nuevos tipos de objetos. Los objetos hacen que sea mas fácil la escritura de las aplicaciones, por que todos los datos de una entidad determinada están localizados en un lugar. Además de los datos, los objetos pueden almacenar relaciones, representadas internamente como enlaces a otros objetos y pueden almacenar el comportamiento, representado, internamente como métodos.

Los objetos pueden hacer también que las aplicaciones se ejecuten más rápido. Como los datos de una entidad están relacionados lógicamente, la Base de datos Orientada a Objetos tiene los medios para optimizar su localización física.

El acceso concurrente multiusuario, el procesamiento de transacciones, el control de versión, la seguridad y los esquemas de recuperación, son más fáciles de realizar debido a que puede ponerse un único bloqueo en todos los datos relevantes a nivel objeto, en vez de requerir del programa de la aplicación que intente establecer bloqueos múltiples a través de los datos relacionados, aunque dispersos, en una base de datos relacional.

4.3.6 MÉTODOS

Cada objeto en una Base de Datos Orientada a Objetos puede tener encapsulados en su interior un numero de métodos para actuar sobre sus datos. Estos métodos son activados cuando el programa de la aplicación envía mensaje al objeto.

Aunque el software de aplicación puede realizar la comprobación de la integridad de los datos, es más fácil y menos propenso a error dejar que la base de datos se encargue de ello.

4.4.7 HERENCIA

La herencia proporciona un medio de reducir el esfuerzo necesario para desarrollar y mantener una base de datos. La herencia permite que se añadan nuevas características y tipos de datos a una base de datos exigiendo solamente unos cambios muy concretos.
Cada objeto de una Base de datos Orientada a Objetos es un miembro de una clase, que determina la estructura del objeto y define que operaciones puede realizarse con él. Las nuevas clases se crean a partir de las clases existentes por medio de la técnica de subclasificación y obviando los métodos heredados. Esto da como resultado operaciones y tipos de datos complejos arbitrariamente, que pueden ser tratados entonces como si fueran tipos incorporados.

La herencia no solamente ayuda en este proceso de definir el tipo y las relaciones de los datos a almacenar en una base de datos, si no que también reduce el esfuerzo necesario para acomodar los inevitables cambios en la estructura de la base de datos.

El mecanismo de la herencia soporta funciones de bases de datos como bloqueo, autorización y consulta.

4.4.8 BIBLIOTECA DE CLASES

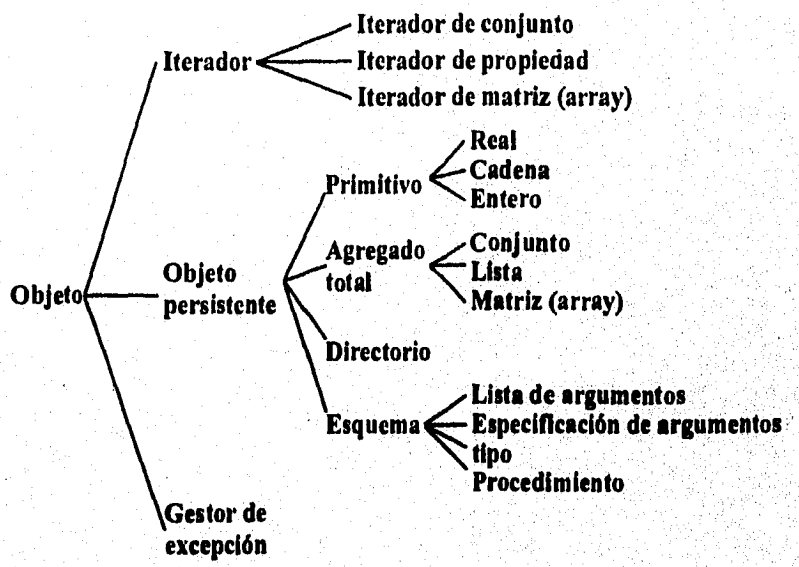
Las bibliotecas de clases permiten la reutilización para que así los programadores no tengan que volver a crear las estructuras más comunes que son:

- **Matrices**
- **Diccionarios**
- **Tablas**
- **Fechas**
- **Métodos (como recorrido lineal o dispersión {hashing})**

Los programadores de Bases de Datos Orientadas a Objetos perfeccionan las bibliotecas de clases para satisfacer las necesidades de la aplicación.

Las bibliotecas de clases para las Bases de Datos Orientadas a Objetos difieren de las bibliotecas de clases para los lenguajes orientados a objetos en el tipo de clases que definen, ambos incluyen clases para los tipos básicos (enteros, reales, cadenas, conjuntos, matrices etc.) pero las Bases de Datos Orientadas a Objetos incluyen clases que tienen más de funcionalidad propias de una bases de datos como son:

- Clases para objetos persistentes
- Clases para excepciones
- Clases para directorios
- Clases para bloqueos
- Clases para esquemas



Ejemplo de una biblioteca de clases típica para una bases de datos orientada a objetos

4.4.9 PERSISTENCIA

Si un objeto después de terminado el programa de aplicación sigue existiendo entonces ese objeto posee persistencia, los objetos creados en los lenguajes orientados a objetos existen durante la ejecución de la aplicación y al termino de esta se destruyen, cosa que en los objetos de las Bases de datos Orientadas a Objetos no sucede ya que estos sobreviven múltiples sesiones.

La persistencia de un objeto esta muy ligada al concepto de identidad de objeto, la cual requiere que todo objeto tenga algo que sea único e invariable con independencia de la forma en que el objeto cambie, debido a esto para eliminar un objeto que posee persistencia se sigue un mecanismo diferente, el cual consiste en borrar todas las referencias al objeto y sólo se elimina el objeto hasta cuando ya no hay ninguna referencia a este objeto, entonces el espacio que ocupaba es recuperado por la aplicación.

4.4.10 CONSULTA

La diferencia principal entre los sistemas de Bases de datos Orientadas a Objetos y las bases de datos relacionales se encuentra en la cantidad de información que se mueve entre la aplicación y el sistema de gestión de bases de datos, la forma en que se lleva acabo una consulta en una base de datos relacional es la siguiente:

La aplicación envía una consulta a la base de datos relacional, la cual devuelve un número de valores, estos valores son almacenados normalmente como parte de la estructura de datos de la aplicación donde se manipulan y se vuelven a entregar a la base de datos para su almacenamiento.

4.4.11 VENTAJAS DE LAS BASES DE DATOS ORIENTADAS A OBJETOS

- Posibilitan soporte de aplicaciones complejas que no son soportadas perfectamente por el resto de los modelos.
- El rendimiento, concepto siempre importante en la realización del sistema, puede mejorarse significativamente utilizando un modelo Orientado a Objetos en lugar de un modelo relacional.
- Las Bases de Datos Orientadas a Objetos pueden almacenar no sólo componentes complejos de aplicación si no también estructuras de datos mayores.
- Simplifica el control de la concurrencia con una Base de Datos Orientada a Objetos
- La integridad, una exigencia clave de las bases de datos, puede soportarse mejor.
- La capacidad de almacenar los objetos permite también que los objetos sean compartidos en un entorno distribuido.
- Una Base de Datos Orientada a Objetos puede permitir solamente que los objetos utilizados activamente sean cargados en memoria.
- Las Bases de Datos Orientadas a Objetos que disponen de grandes objetos no sufren una degradación en rendimiento por que los objetos no necesitan ser descompuestos y vueltos a ensamblar por las aplicaciones.

CAPÍTULO V

ESQUEMAS
DE
BASES DE DATOS

Objetivo:

Investigar una técnica de análisis y diseño orientado a objetos, para conocer su metodología de desarrollo de sistemas con orientación a objetos, estudiando cada una de las partes que la componen.

V.- ESQUEMAS DE BASES DE DATOS

5.1 ANTECEDENTES

Como hemos visto anteriormente, para desarrollar cualquier sistema o programa de computación, se requiere de realizar un análisis, un diseño y una codificación. Ahora que conocemos los conceptos fundamentales de la orientación a objetos, podemos crear un modelo del problema a resolver utilizando un análisis y diseño respectivo.

Con el análisis orientado a objetos, la forma de modelar la realidad difiere del análisis convencional. Se modela en términos de tipos de objetos y lo que le ocurre a éstos. El analista orientado a objetos ve el mundo como objetos (con estructuras de datos y métodos) y eventos que activan operaciones, las cuales modifican el estado de los objetos. Las operaciones aparecen como objetos que hacen solicitudes a otros objetos. El analista crea diagramas de la estructura de los objetos y de los eventos que los modifican. El modelo del diseñador es similar al modelo del analista, pero se toma con el detalle suficiente como para crear el código. El análisis y diseño orientado a objetos intenta lograr la reutilización masiva de las clases de objetos.

Generalmente, la forma en que construimos sistemas es la siguiente: El analista crea un modelo del área de interés. El modelo se convierte en un diseño y después en un código. El modelo debe representar cómo perciben los usuarios finales el área o lo que los usuarios desean que realice el sistema. Poco a poco, el modelo debe tener una forma comprensible para los usuarios y ayudarlos a ser creativos con respecto a sus necesidades. El técnico utiliza el modelo y crea un diseño a partir del cual se pueda escribir o generar un código.

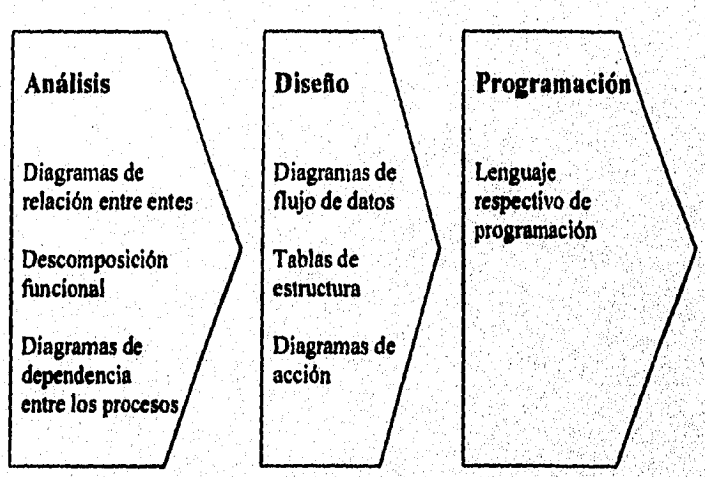
En las metodologías tradicionales para la generación de sistemas, los modelos conceptuales utilizados para el análisis difieren de los que se emplean para el diseño. La programación es también un tercer punto de vista. Los analistas utilizan los modelos de relación de entes, la descomposición funcional y las matrices. Los diseñadores utilizan

diagramas de flujo de datos, tablas de estructura y diagramas de acción. Los programadores utilizan las construcciones en su lenguaje de programación respectivo.

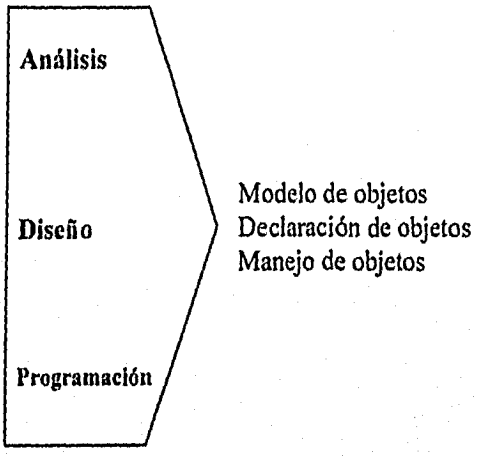
En las técnicas orientadas a objetos, todos utilizan el mismo modelo conceptual: Analistas, diseñadores, programadores y, de forma particularmente importante , los usuarios finales. Todos piensan en los tipos de objetos, los objetos y su comportamiento. La transición del análisis al diseño es tan natural, que a veces es difícil saber donde finaliza el análisis y donde comienza el diseño.

Cuando se utiliza la forma tradicional para realizar una aplicación, del paso del análisis al diseño, se pierde información y aparecen las incomprensiones. Dicha transición consume tiempo y a menudo reduce la calidad del producto final. Lo anterior se ilustra con el siguiente dibujo:

Metodología Tradicional
(Distintos Modelos Conceptuales)



Metodología Orientada a Objetos
(Un solo Modelo Conceptual)



El utilizar un modelo conceptual único junto con una herramienta I-CASE para ese modelo tiene como ventajas:

- Una mayor productividad
- Menos errores
- Una mejor comunicación entre los usuarios, analistas, diseñadores y técnicos
- Resultados de mejor calidad
- Más flexibilidad
- Mayor inventiva

En el análisis orientado a objetos, se construyen dos tipos de modelos, los cuales se relacionan uno con otro estrechamente: Un modelo de los tipos de objetos y sus estructuras; y un modelo de lo que ocurre a los objetos. Los modelos se representan mediante diagramas llamados esquemas. Los esquemas de objetos muestran las estructuras del objeto y los esquemas de eventos muestran lo que ocurre a los objetos.

El primer modelo se refiere a los tipos de objetos, clases, relaciones entre los objetos y herencia; se le conoce como Análisis de la Estructura de Objetos (AEO) y Diseño de la Estructura de Objetos (DEO). Y el segundo modelo se refiere al comportamiento de los objetos y lo que les ocurre al paso del tiempo; este se conoce como Análisis del Comportamiento de Objetos (ACO) y Diseño del Comportamiento de Objetos (DCO).

A continuación se definen algunos de los aspectos que tratan cada una de las partes anteriores:

Análisis de la Estructura de Objetos (AEO)	***Análisis del Comportamiento de Objetos (ACO)***
Se ocupa de los tipos de objetos y sus asociaciones Tipos de objetos y asociaciones Diagramas de generalización Diagramas de relación entre los objetos Diagramas de componentes	Se ocupa de lo que le sucede a los objetos al paso del tiempo Diagramas de flujo de datos Esquemas de eventos Diagramas de funcionamiento que muestran las funciones y la secuencia en que deben ocurrir Estados de objetos y cambios en dichos estados Reglas de activación que ligan la causa y el efecto
Diseño de la Estructura de Objetos (DEO)	***Diseño del comportamiento de Objetos (DCO)***
Se ocupa de las clases, métodos y herencia	**Se ocupa del diseño de métodos**
Clases **Superclases y subclases** **Herencia** **Estructuras de datos** **Diseño de una base de datos**	**Métodos y funciones** **Lógica de procedimiento** **Código no por procedimiento** **Entrada para los generadores de códigos** **Diseño de la pantalla y del diálogo** **Fabricación de prototipos**

Como podemos notar en el cuadro anterior, el análisis de la estructura de objetos se ocupa de definir las categorías de objetos y la forma en que se asocian. Nos preguntamos: ¿Qué tipos de objetos hay?, ¿Cuáles son sus relaciones y funciones?, ¿Qué subtipos y supertipos son útiles?, ¿Hay algún tipo de objeto compuesto por otros objetos?.

Cuando el análisis pasa a la etapa del diseño de la estructura de objetos, se identifican las clases, es decir, la implantación de tipos de objetos. Se definen las superclases, subclases, rutas de herencia y los métodos a utilizar y se lleva a cabo el diseño en detalle de las estructuras de datos.

El análisis del comportamiento de objetos se ocupa de modelar lo que ocurre a los objetos al paso del tiempo. Nos preguntamos: ¿En qué situación pueden estar las clases de objetos?, ¿Qué tipo de eventos cambian estos estados?, ¿Qué sucesión de eventos ocurre?, ¿Qué funciones resultan de estos eventos y cómo se activan?.

Una vez terminado lo anterior sigue la etapa de diseño, la cual se ocupa del diseño detallado de métodos, ya sea con técnicas por procedimiento o de no por procedimiento. Se crea la entrada para los generadores de códigos. Se diseña la pantalla y se generan los diálogos. Por último, se construyen y desarrollan los prototipos.

5.2 ANÁLISIS DE OBJETOS

Debemos tomar en cuenta que al análisis de objetos comprende dos partes, como ya se ha explicado anteriormente, una de ellas es el análisis de la estructura de objetos (AEO) y la otra el análisis del comportamiento de objetos (ACO).

5.2.1 ANÁLISIS DE LA ESTRUCTURA DE OBJETOS

En esta etapa se presentan los modelos que se crean en el análisis de la estructura de objetos, los cuales son una guía para la definición de clases y sus estructuras de datos en el diseño de la estructura y comportamiento del objeto. El AEO se ocupa de identificar:

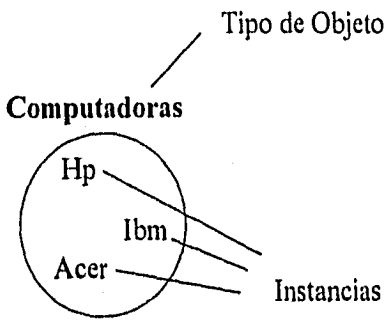
a) ¿ Qué son los tipos de objetos y cómo se asocian? La identificación de los objetos y sus asociaciones se representan mediante esquemas de objetos. Esta es la información que guía al diseñador en la definición de clases y estructuras de datos.

b) ¿ Cómo se organizan los tipos de objetos en supertipos y subtipos? Aquí se hace uso de las jerarquías de generalización, es decir, realización de diagramas de organización, los cuales indicarán al diseñador las direcciones de herencia.

c) ¿ Cuál es la composición de los objetos complejos? Se elaboran diagramas de jerarquías compuestas. Dicha composición guía al diseñador en la definición de mecanismos que controlen adecuadamente a los objetos dentro de otros objetos.

5.2.1.1 TIPOS DE OBJETOS

Una de las partes importantes del análisis de la estructura de objetos es la de identificar los tipos de objetos en un sistema, más que identificar los objetos individuales. Recordemos que el tipo de objeto es una categorización de objetos, es decir, una generalización. Mientras que un objeto individual es una instancia de un tipo de objeto.



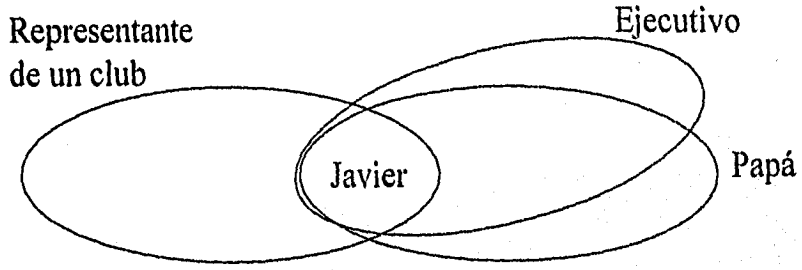
Debemos tener en cuenta que los tipos de objetos son importantes, ya que crean los bloques conceptuales de construcción para el diseño de sistemas. Estos bloques de construcción guiarán al diseñador en la definición de las clases y estructuras de datos.

Algunos ejemplos de tipos de objetos son:

Concreto	Intangible	Juicio
computadora	tiempo	salario alto
automóvil	calidad	buen ejemplo
persona	compañía	trabajo productivo
Relación	Eventos	Otros
matrimonio	compra	señal
sociedad	venta	imagen
propiedad	falla	club

Los tipos de objetos son un índice para los procesos de un sistema, ya que existen varias actividades que están ligadas al tipo de objeto y que cambian su estado. Por ejemplo el tipo de objeto reservación aérea tiene actividades como reservación aérea solicitada, reservación aérea cancelada, etc... Al pensar en tipos de objetos, debemos hacerlo de acuerdo a la comprensión de nuestro mundo, es decir, a la vida real.

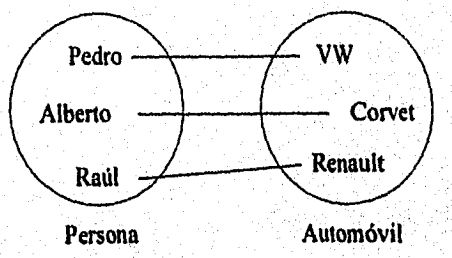
Un objeto se puede categorizar en más de una forma, o sea, que puede ser parte de varios tipos de objetos o asociaciones, como lo indica la figura:



Lo anterior, significa que el objeto Javier es un ejecutivo de una empresa, pero también es papá de Carlitos o es representante de un club de alguna organización.

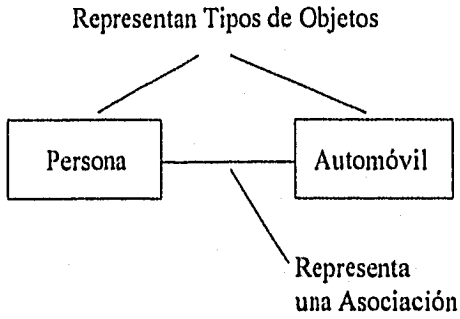
5.2.1.2 ASOCIACIÓN DE OBJETOS

Una vez identificados los tipos de objetos procedemos a establecer sus asociaciones y relaciones con otros tipos de objetos, para ello nos auxiliamos utilizando diagramas de relación de objetos. en la figura se ejemplifica una asociación simple de dos tipos de objetos:

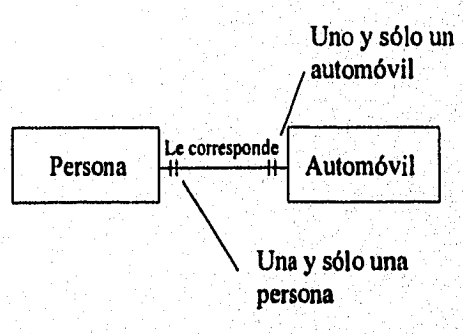


Como podemos observar, a la persona Pedro le corresponde el tipo de automóvil VW, a Raúl el tipo de automóvil Renault y a la persona Alberto le corresponde el tipo de

automóvil Corvet, con el fin de esquematizar aún más este diagrama de relación se utilizará la siguiente notación:



Se utilizará un rectángulo con esquinas rectangulares cuando se quiera representar un tipo de objeto y una línea para representar una asociación de un tipo de objeto con otro. En el diagrama anterior no se indica el significado de la asociación, ni tampoco la cantidad de objetos con los que un objeto dado puede y debe asociarse. En el análisis de la estructura de objetos, es útil nombrar de alguna forma a las asociaciones e indicar la cantidad de objetos de un tipo dado que se deben asociar con los objetos de otro tipo, puesto que esto le da significado y aumenta la comprensión de la asociación. Por ejemplo:

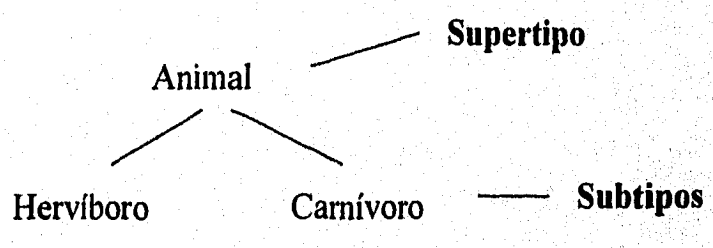


En el diagrama anterior se incluyó un significado de la relación entre el tipo de objeto persona y automóvil la cual es: "Le corresponde". Y se indica también la cantidad de "una y

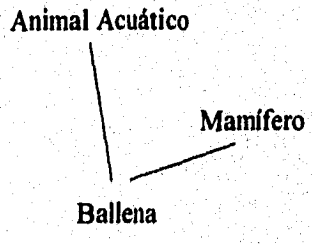
solo una persona", así como, "uno y sólo un automóvil". Con esto se explica de mejor forma el diagrama, el cual lo podrá entender cualquier persona, ya sea un usuario, un analista, un diseñador o un técnico.

5.2.1.3 JERARQUÍAS DE GENERALIZACIÓN

Se utilizarán los diagramas de generalización, para representar gráficamente los supertipos y subtipos del análisis de la estructura de un objeto. De esta forma se organizan los tipos de objetos de una forma general a los más específico, es decir, la generalización es el resultado de distinguir un tipo de objeto como más general, o incluso, que es más que otro. Todo lo que se aplique a un tipo de objeto también se aplica a sus subtipos. Cada instancia de un tipo de objeto es también una instancia de sus supertipos. Ejemplo:



De la figura, el tipo de objeto Animal es un supertipo de Herbívoro y Carnívoro y éstos son subtipos de Animal. Notamos que en este diagrama se aplica el clásico concepto de estructura de árbol, sin embargo, no siempre un subtipo contiene sólo un supertipo, a veces se da que contenga más de un supertipo, como lo muestra la siguiente figura:



En este caso, Ballena consta de dos supertipos los cuales son: Animal acuático y Mamífero. Con esto concluimos que no siempre se tendrán diagramas de jerarquías de generalización de tipo árbol sino también de tipo no árbol.

Cabe destacar que, un diagrama de jerarquía de generalización también indica la dirección de herencia de características que pasarán de un tipo de objeto a otro, es decir, de un supertipo a un subtipo, teniendo así el paso de parámetros que se aplican a sub-subtipos, etc....

Esto significa que si el supertipo Animal Acuático consta de características como: Ser viviente en agua, contiene escamas, vive de peces, etc... y Mamífero consta de características como: no nace de huevo, es grande, etc.... Todas estas características pasarán a ser heredadas al subtipo Ballena, el cual podrá tener incluso sus propias características las cuales se agregarán a las anteriores.

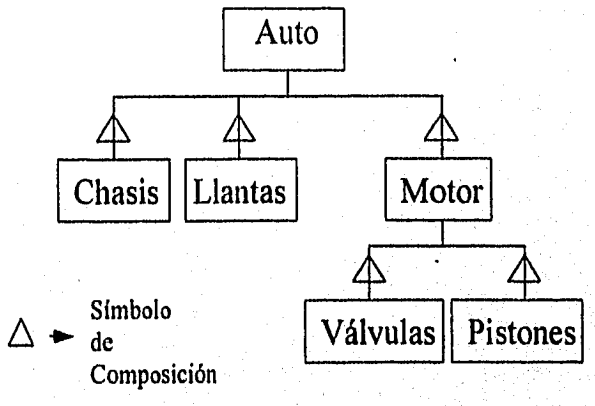
5.2.1.4 JERARQUÍAS COMPUESTAS

Un diagrama de jerarquías compuestas es aquél que indica que un tipo de objeto está formado por otros objetos. Muchas de las veces los tipos de objetos no son tan simples, es decir, se consideran complejos. Por ejemplo, el tipo de objeto Auto se compone de chasis, llantas, motor, etc... Y el motor a su vez está compuesto de válvulas, pistones, monoblock, etc... Además cada uno de ellos se compone de otras partes, según el detalle que se quiera tener. Cuando suceda lo anterior, se tendrá que utilizar diagramas de jerarquías compuestas con el fin de comprender cada una de las partes de un tipo de objeto complejo.

En el análisis OO, la composición de un objeto nos ayuda a describir el hecho de que los dibujos están formados por determinada configuración de símbolos, los trabajos lo están por tareas específicas, las organizaciones por otras organizaciones, y así sucesivamente. En un sistema de tecnología avanzada, el analista describirá la forma en que un pedido no sólo

puede constar de elementos en cada renglón , sino contener también instrucciones verbales del cliente o un diagrama hecho a mano. Los pedidos de este tipo se llaman objetos complejos. Cada pedido se puede controlar como un objeto único que conste de otros, los cuales, a su vez, se pueden controlar de forma independiente, en caso necesario.

Diagrama de Jerarquías Compuestas



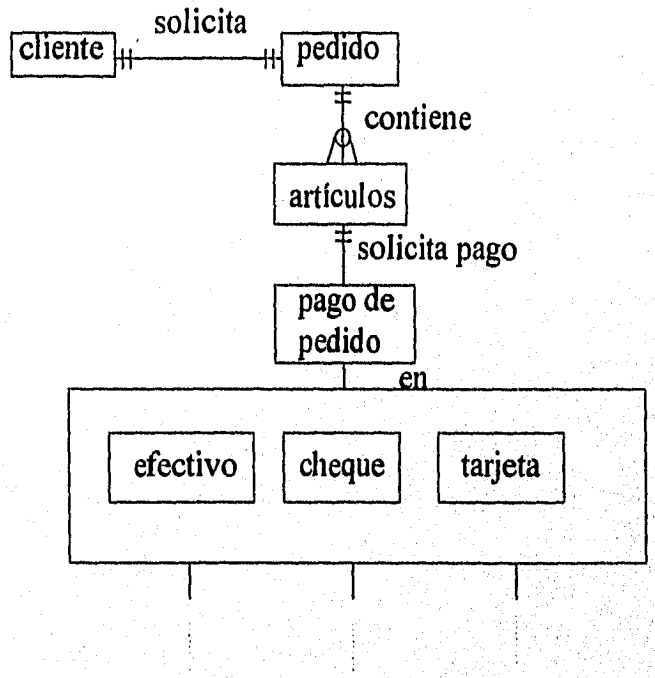
En este diagrama, podemos notar como el tipo de objeto Auto se descompone en Chasis, llantas y motor, de la misma forma Motor se descompone en Válvulas y Pistones. Se observa el uso de un triángulo sin relleno alguno, el cual es el símbolo utilizado para denotar que un objeto es parte de un tipo de objeto.

5.2.1.5 ESQUEMAS DE OBJETOS

Una vez que se han elaborado los diagramas correspondientes a los tipos de objetos, sus asociaciones y relaciones, identificación de supertipos y subtipos, jerarquías de generalización y jerarquías compuestas, se procede a realizar un diagrama que integre todos los demás, con el fin de visualizar en forma rápida todas las operaciones en un análisis de estructura de objetos y este recibe el nombre de Esquema de Objetos.

Debemos mencionar, que el realizar diagramas independientes de un análisis de la estructura de objetos es una forma fácil y sencilla de comprender un modelo, sin embargo para usuarios más sofisticados resulta útil presentarlo todo en el mismo diagrama.

El siguiente es un ejemplo de un diagrama de esquemas de objetos:



5.2.2 ANÁLISIS DEL COMPORTAMIENTO DE OBJETOS

En esta parte se realizan esquemas de eventos, la secuencia en que ocurren y cómo dichos eventos cambian el estado de los objetos. Por lo tanto un esquema de eventos se debe expresar en términos de un esquema de objetos, puesto que los eventos cambian el estado de determinados tipos de objetos.

En este análisis se debe identificar lo siguiente:

a) ¿ En qué estados puede estar un objeto ?, Un objeto puede tener varios conjuntos de estados.

b) ¿ Qué transiciones de estado se pueden dar ?, Estas se determinan en el diagrama de transición de estado.

c) ¿ Qué eventos ocurren ?, (un evento es el cambio de estado de un objeto).

d) ¿ Qué operaciones se llevan a cabo ?, Se traza un esquema de eventos, para mostrar la secuencia de operaciones y eventos.

e) ¿ Qué interacciones ocurren entre objetos ?, Se traza un diagrama para mostrar los mensajes transferidos entre las clases.

f) ¿ Cuáles son las reglas de activación que se utilizan para reaccionar ante el evento ?

g) ¿ Cómo se representan las operaciones en los métodos ?, Se utilizan los diagramas, enunciados declarativos u otros medios para determinar los métodos con detalles suficientes como para generar código.

5.2.2.1 ESTADOS DE UN OBJETO

Un objeto puede existir en varios estados, es decir, puede formar parte de varios tipos de objetos, ejemplo: El objeto Pedido puede ser una instancia de alguno de los siguientes tipos de objeto:

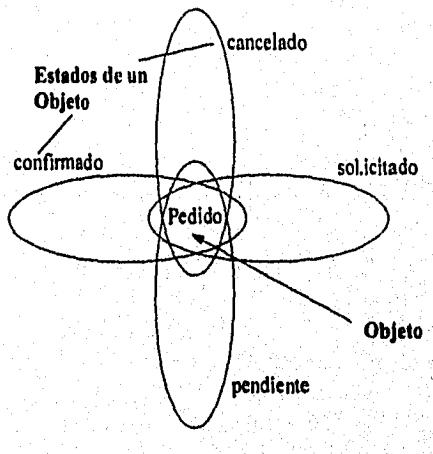
- **Pedido solicitado**
- **Pedido confirmado**
- **Pedido cancelado**

- Pedido pendiente

Los tipos de objetos anteriores, serían los distintos estados que podría adoptar en un momento dado el tipo de objeto Pedido, además sabemos que dicho objeto puede tomar una gran variedad de perspectivas, es decir, el mismo objeto Pedido, podría fomar parte de otros tipos de objeto como:

- Pedido pagado
- Pedido con pago de depósito
- Pedido no liquidado

Podemos definir el estado de un objeto como una colección de los tipos de objeto que se aplican a él , sin embargo, cuando se implanta en un lenguaje de programación OO, el estado se registra en los datos almacenados con relación al objeto. Se determina mediante las clases y valores de los campos de los datos asociados con el objeto. De esta forma un programador OO define el estado de un objeto como: La colección de asociaciones que tiene un objeto.



En el lenguaje OO, una solicitud se envía y provoca la activación de los métodos. Los métodos cambian el estado del objeto. El estado se registra en los datos del objeto.

5.2.2.2 EVENTOS

En el análisis orientado a objetos el mundo se describe en términos de los objetos y sus estados, así como de los eventos que modifican esos estados. Un evento es un cambio en el estado de un objeto. Algunos ejemplos de eventos son los siguientes:

- Se casó tu hermana
- Nos visitó la abuela
- Reservación aérea confirmada
- Saldo negativo
- Préstamo

En el ambiente de contaduría, se dan cambios de estado como: Aparecen en la cuenta de un empleado saldos negativos o en un balance general se detecta que se gastó una cantidad mayor en un producto que en otro. Esto significa que la mayoría de estos cambios no los percibimos o tomamos en cuenta, pero cuando se quiera saber de ellos y reaccionar de alguna forma, debemos notar que ocurren. Así , los eventos sirven como indicadores de los instantes en que ocurren los cambios de estado de un objeto. Por ejemplo:

Supongamos, que Pedro tiene guardados en una cuenta bancaria $ 1,000.00 y según la restricción del banco el puede agregar o sacar dinero cada mes de acuerdo a sus necesidades. Si Pedro al primer mes agrega a su cuenta otros $ 1,000.00, entonces él tendría un total de $ 2,000.00. Para que el banco supiera que Pedro a introducido tal cantidad, necesita que "algo" le indique que Pedro ha almacenado dinero, este "algo" sería un evento llamado "agregar a cuenta de", si el banco no tuviera este indicador de cambio de estado del objeto cuenta, éste no sabría del aumento.

Sin los eventos, nada cambiaría. Se podrían construir y generalizar bases de datos sin preocuparnos por actualizarlas. Sin embargo en la mayoría de las aplicaciones, sí debe

cambiar el contenido de las bases de datos. Debido a que deseamos saber de esos cambios y reaccionar en forma adecuada ante ellos, deberemos entender y modelar los eventos.

Así como identificamos tipos de objetos, también debemos identificar los tipos de eventos, esto con el fin de detectar solamente los cambios de estado que nos interesen y no cada evento que ocurra en una organización. Los tipos de eventos indican los cambios sencillos en el estado de un objeto. Básicamente, los tipos de eventos describen las siguientes formas de cambios de estado:

Un objeto se crea. Por ejemplo se crea un pedido.

Un objeto se termina. Un contrato se termina o un producto se destruye.

Un objeto se clasifica como una instancia de un tipo de objeto. Por ejemplo, una novia se convierte en esposa, una empresa se convierte en cliente, un empleado se convierte en gerente.

Un objeto se desclasifica como una instancia de un tipo de objeto. Por ejemplo, una firma deja de ser cliente; un producto sale del catálogo de ventas.

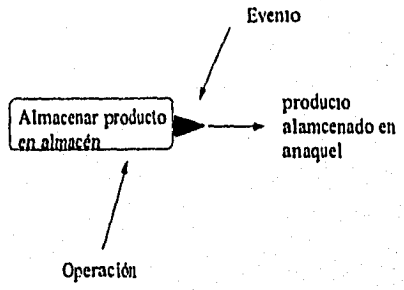
Un objeto cambia de clasificación. Por ejemplo, un abogado pasa de ayudante a socio; una cuenta cambia de normal a atrasada.

El atributo de un objeto se cambia.

Un evento puede asociar un objeto con otro. Por ejemplo, el objeto Persona y el objeto Cuenta de un banco se asocian con un tipo de evento "Cuenta de la persona".

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

Una operación hace que un evento ocurra. En un diagrama (esquema) de eventos, una operación se representa con un rectángulo de esquinas redondeadas y el evento con un triángulo lleno ya que este indica los puntos en el tiempo en que se da el cambio de estado de un objeto. Generalmente aparecen unidos a la caja de operación, la figura muestra lo anterior:

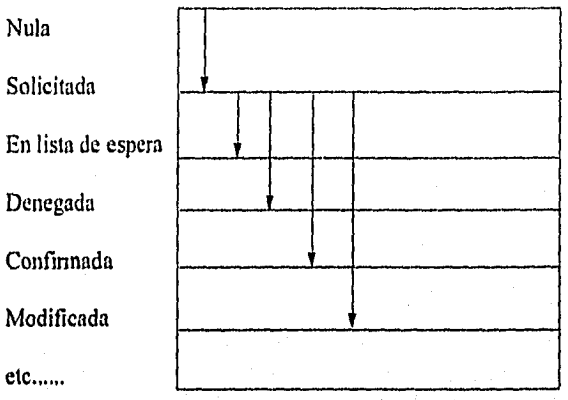


Según el área que se modele, puede ocurrir más de un evento al terminar una operación, y cada uno de éstos puede activar operaciones independientes.

5.2.2.3 CICLO VITAL DE UN OBJETO

El ciclo vital de un objeto es la sucesión de eventos que pueden ocurrirle a un objeto y la modificación de su estado que causa cada uno de ellos. En el análisis orientado a objetos, se dibuja un diagrama que muestre el ciclo vital de un objeto. A este diagrama se le conoce con el nombre de diagrama de rejas o diagrama de transición de estados. Dicho diagrama además de mostrar los estados posibles de los objetos, también muestra los cambios de estado permisibles.

Tipo de Objeto: Reservación Aérea



Las líneas horizontales representan los estados posibles estados del objeto Reservación Aérea. Las líneas verticales muestran las transiciones entre los estados.

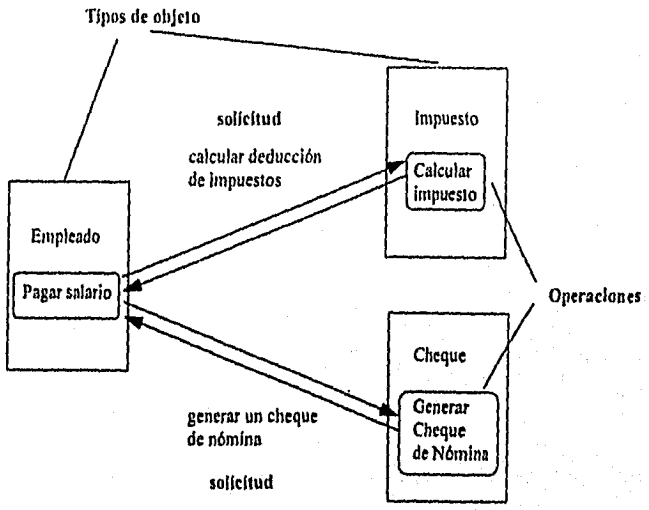
Debemos considerar que dicho objeto puede tener varias perspectivas y que otros tipos de objetos pueden determinar el estado del objeto. Por ejemplo, la figura anterior no indica el estado de pago de una reservación, para lo cual se necesitaría realizar otro diagrama que muestre los cambios de estado y una vez hecho, hacer que los dos diagramas creados interactúen entre ellos.

Cuando en un diagrama de transición de estados, aparezca el signo + al lado de un estado, dicho símbolo significa que el estado se puede descomponer en subestados.

5.2.2.4 INTERACCIONES ENTRE TIPOS DE OBJETOS

Un diagrama de transición de estados es útil para expresar el ciclo vital de un objeto particular. Sin embargo, la mayoría de los procesos requieren la interacción de varios objetos. Para representar dicha interacción se utilizarán los llamados diagramas de red, el

cual muestra como los distintos tipos de objeto cambian de estado y pueden solicitar a otros objetos que cambien su estado en el proceso. Para completar este tipo de diagramas se puede incluir en ellos a las operaciones de cada tipo de objeto. Ejemplo:



Cuando los diagramas se expresan así, es fácil ver la forma de implantarlos como estructuras de un programa OO: los tipos de objetos se implantan como clases y las operaciones se convierten en operaciones del programa OO. Si, durante el análisis, el usuario puede expresar los requisitos de procesamiento de esta forma, se facilita el trabajo del diseñador de lenguajes de programación orientada a objetos (LPOO).

5.2.2.5 ESQUEMAS DE EVENTOS

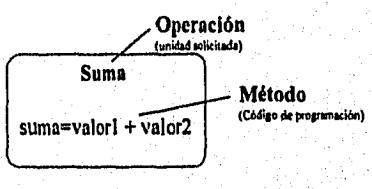
Un esquema de eventos representa los cambios en el ciclo vital de un objeto particular, el cual activa cambios en el ciclo vital de otro objeto. Los esquemas de eventos son adecuados para la descripción de procesos en términos de eventos, de reglas de activación, de condiciones y de operaciones.

Para adentrarnos en la realización de esquemas de eventos debemos conocer algunos conceptos como: ¿Qué es una operación?, fuentes externas de eventos, reglas de activación, condiciones de control, esquemas jerárquicos de eventos, aislamiento de la causa y el efecto, así como los diagramas de flujo de objetos.

A continuación se explicará brevemente cada uno de estos conceptos:

5.2.2.6 OPERACIONES

Una operación es una unidad de procesamiento que puede ser solicitada. Su procedimiento se implanta mediante un método. El método es la especificación de cómo llevar a cabo la operación (es el guión de la operación). A nivel programa, el método es el código que implanta la operación.

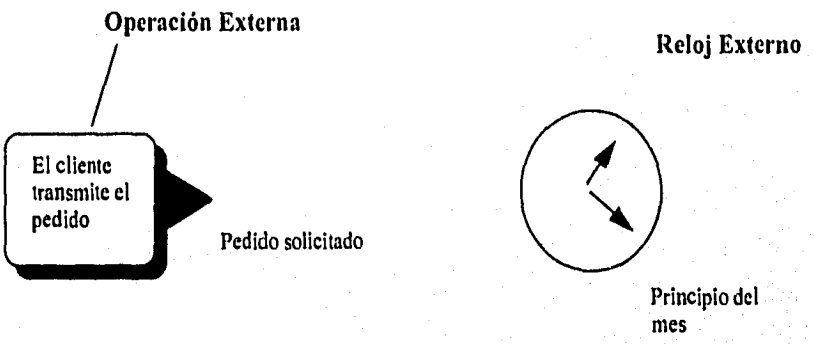


una operación puede o no cambiar el estado de un objeto. Si lo cambiara, ocurriría un evento. Recordemos que una operación se representa mediante cuadros con esquinas redondeadas y los tipos de eventos se representan mediante triángulos sólidos negros conectados a la caja.

5.2.2.7 FUENTES EXTERNAS DE EVENTOS

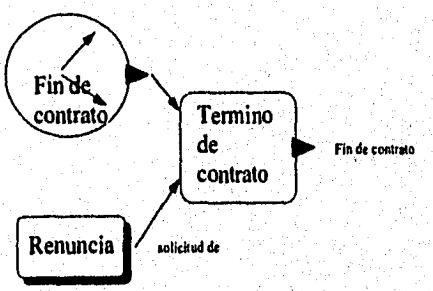
Existen 2 tipos de fuentes externas de eventos que se manejan en los diagramas del análisis del comportamiento de objetos, los cuales son: Tipo de evento externo y tipo de evento de reloj externo. El primero es un evento que se produce por "algo" exterior al sistema, por ejemplo cuando un cliente transmite un pedido.

Para representar dicho evento, se utilizará una caja sombreada con esquinas redondeadas. El tipo de evento de reloj externo, indica que un proceso externo emitirá señales de reloj con cierta frecuencia determinada con anterioridad: como cada segundo, al final de cada día, el principio de cada mes o cada determinada fecha. Estos relojes externos se representan como carátulas de reloj. Ejemplo:



5.2.2.8 REGLAS DE ACTIVACIÓN

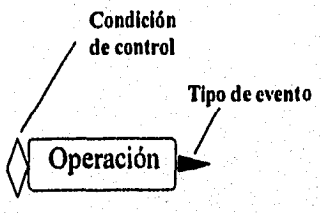
Se les llama reglas de activación a los eventos que hacen que una operación se active, es decir, que podemos tener uno o más eventos detrás de una operación y que cuando alguno de éstos se active la operación funcionará, mientras no. Ejemplo:



En este ejemplo se puede observar que la operación Termino de contrato tiene dos reglas de activación que son Fin de contrato y Renuncia, al momento que alguna de las dos se active, hará que la operación Termino de contrato funcione. Así, las reglas de activación definen la relación entre la causa y el efecto. Siempre que ocurra un evento de cierto tipo, la regla de activación invoca a una operación ya definida.

5.2.2.9 CONDICIONES DE CONTROL

Una condición de control en un diagrama de esquema de eventos, es una condición que deben cumplir las reglas de activación para que una operación pueda ejecutarse, esto es, que antes de invocar a la operación, se verifica su condición de control. Una condición de control se representa por medio de un rombo colocado antes de la operación. Esto significa que las condiciones de control garantizan que un conjunto de eventos esté completo antes de proceder con una operación.

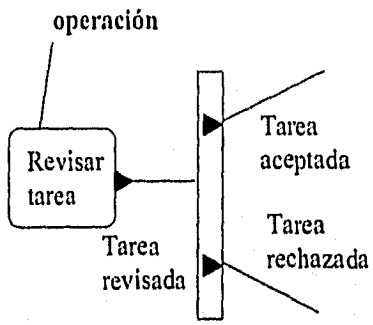


5.2.2.10 SUBTIPOS Y SUPERTIPOS DE EVENTOS

De la misma forma que existen esquemas en donde se representan los supertipos y subtipos de objetos, también los hay para representar los subtipos y supertipos pero de eventos. Esto es, que un evento se puede dividir en dos o más eventos. Al crear este tipo de diagramas, se emplea un término llamado tipos ajenos que son los subtipos de un tipo de evento y cada uno de ellos realizará una tarea específica. Un ejemplo, sería el siguiente:

Supongamos que tenemos el evento Revisar tarea, pero este evento una vez realizado debe indicar si la tarea fue aceptada o fue rechazada. pues bien, el evento tarea aceptada y

tarea rechazada son subtipos del evento Revisar tarea y éste es supertipo de los dos anteriores. Ejemplo:



5.2.2.11 AISLAMIENTO DE LA CAUSA Y EL EVENTO

Cuando se activa una operación, se realiza por medio de reglas de activación, sin embargo, la operación en sí no sabe que evento la activó ni por qué, esto permitirá que dicha operación se pueda volver a utilizar. Este es uno de los objetivos principales del uso de técnicas orientadas a objetos, es decir, se aísla la causa y el efecto que producirá una operación, por lo que solamente se activa la operación para producir un cambio de estado en un objeto dado.

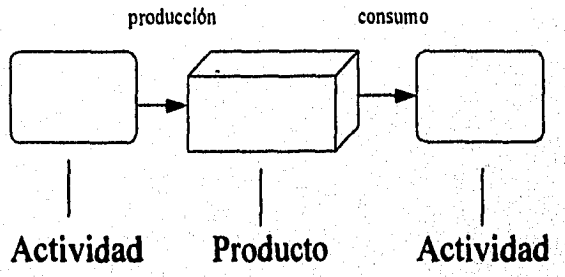
Con los conocimientos anteriores, podemos desarrollar esquemas de eventos, es decir, un guión de procesamiento que cambia los estados de los objetos. Estos se deben expresar en términos de esquemas de objetos, puesto que los eventos se expresan al cambiar el estado en determinados tipos de objetos.

5.2.2.12 DIAGRAMAS DE FLUJO DE OBJETOS

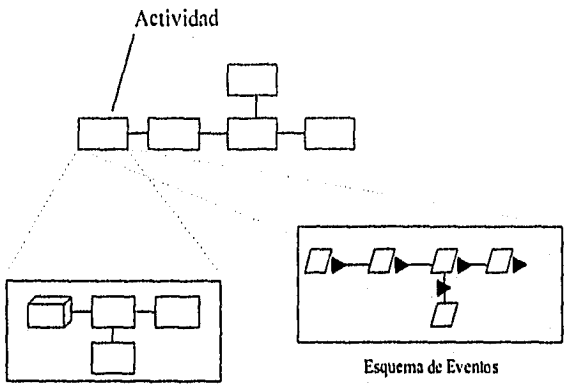
Un esquema de eventos es adecuado para la descripción de procesos en términos de eventos, de reglas de activación, de condiciones y de operaciones. Sin embargo, para

procesos grandes y complejos no es muy recomendable su utilización. En este tipo de casos se podrá utilizar un diagrama de flujo de objetos (DFO), los cuales son parecidos a los diagramas de flujo de datos (DFD) , los cuales muestran las actividades que interactúan con otras.

Generalmente en un DFD se representa la transferencia de datos, pero en un DFO no se limita a la transferencia de datos, sino que debe representar cualquier cosa que se transfiera de una actividad a otra, ya sean pedidos, partes, artículos, hardware, software, etc... Es decir, que el DFO indica los objetos que se producen y las actividades que los producen e intercambian.



Podemos observar, que de una actividad (operación) se genera un producto y que este producto se consume a su vez por otra actividad. Una actividad se puede expresar en términos de un DFO, un esquema de eventos, o ambos, como lo muestra la figura:



Debemos recordar, que para expresar un proceso de forma más detallada y poder generar un código, lo adecuado es un esquema de eventos. Por lo tanto, un diagrama de flujo de objetos es útil para representar las estructuras básicas de control y el flujo del procesamiento, cuando no se entienda totalmente la dinámica de los eventos y las claves de activación.

En resumen, el análisis del comportamiento de objetos puede utilizar los siguientes tipos de diagramación:

- Esquema de eventos.
- Diagrama de dependencias entre procesos.
- Diagrama de flujo de objetos.
- Diagrama de transición de estados.

5.3 DISEÑO DE OBJETOS

El diseño orientado a objetos, consta de dos partes: El diseño de la estructura de objetos (DEO) y el diseño del comportamiento de objetos (DCO). El estudio de ambos

conceptos se realizará en forma conjunta debido a la estrecha relación que guardan cada uno de ellos.

La información que debemos identificar en el diseño de la estructura y comportamiento de objetos, es la siguiente:

a) ¿ Qué clases se implantarán ?, Los tipos de objetos en el AEO serán la guía en esta decisión.

b) ¿ Qué estructura de datos utilizará cada clase ?, Se puede hacer un diagrama para representar la estructura de datos.

c) ¿ Qué operaciones ofrecerá cada clase y cuáles serán sus métodos ?, Se enumeran las operaciones y se especifican sus métodos en determinado momento.

d) ¿ Cómo se implantará la herencia de clases y cómo afectará ésta las especificaciones de los datos y las operaciones ?

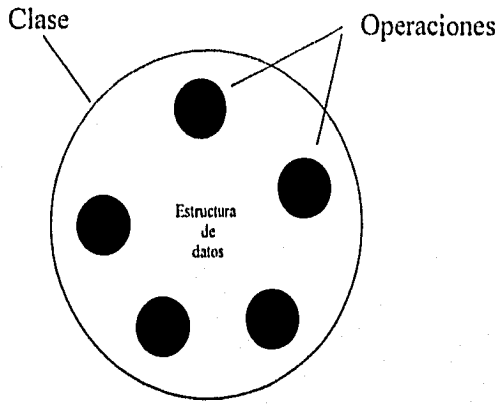
e) ¿ Cuáles son las variantes ?, Se identifican las probables variantes de las clases. ("Igual que, excepto..." se aplica a la mayoría de los componentes reutilizables).

En este capítulo se busca lograr la implantación de los tipos de objetos, sus operaciones y sus métodos. Para lograr esto, debemos conocer algunos conceptos como clases, métodos, etc....

5.3.1 CLASE

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

significa que dentro de una clase se encuentran las estructuras de datos y las operaciones que van a permitir modificar dicha estructura. Ejemplo:

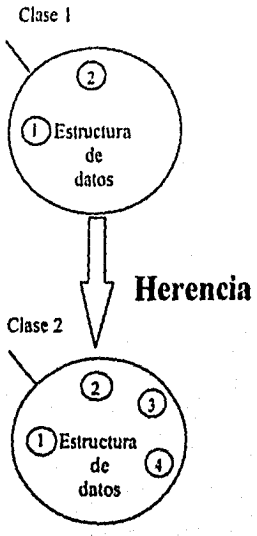


Los datos del objeto se almacenan dentro de él y se tiene acceso a ellos, y se les modifica sólo mediante las operaciones permisibles. A esta restricción al acceso se le llama encapsulado. El encapsulado protege los datos del uso libre de un usuario. El acceso o la actualización directa de los datos de un objeto por parte del usuario violaría el encapsulado.

En el concepto anterior se habló de operaciones y métodos, nos debemos preguntar, ¿qué diferencia existe entre una operación y un método?. Pues bien , la operación es una unidad de procesamiento que puede ser solicitada, es decir, es un procedimiento que está disponible, mientras que el método es el código de programación de dicha operación.

Un método se encuentra dentro de una clase, no dentro de cada objeto, ya que esto llevaría a duplicar información. Cada uno de estos métodos u operaciones, se heredan a otras clases, de aquí nace el concepto de herencia. Aunque debemos tomar en cuenta que los métodos de una clase controlan solamente a los objetos de esa clase. No pueden tener acceso directo a las estructuras de datos de un objeto en una clase distinta. Para utilizar las estructuras de datos en una clase diferente, deben enviar una solicitud a ese objeto.

5.3.2 HERENCIA

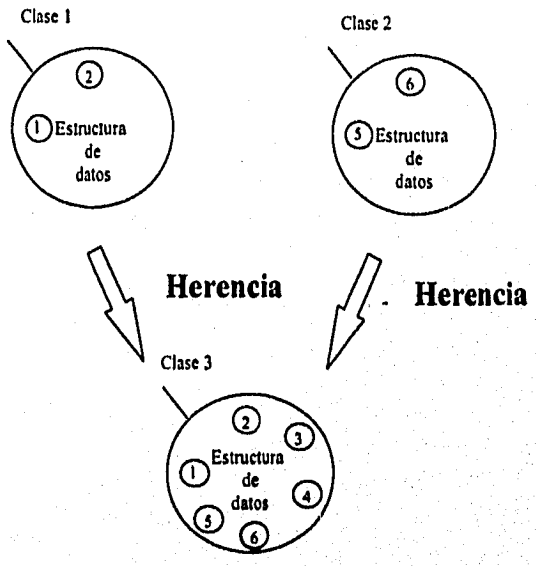


En la figura podemos observar, que la Clase 1 contiene 2 métodos, los cuales al existir una subclase (Clase 2), estos se heredarán y se podrá disponer de ellos a parte de los métodos propios de la subclase 2.

El concepto de herencia en realidad es una implantación de la generalización, y como hemos estudiado anteriormente, significa que las propiedades de un tipo se aplican a sus subtipos. La herencia de clase (conocida como herencia), hace que la estructura de datos y operaciones sean disponibles para su reutilización por parte de sus subclases. la herencia de las operaciones de una superclase permite que las clases compartan el código (en lugar de volverlo a definir). La herencia de estructura de datos permite la reutilización de la estructura.

A partir del concepto anterior, se encuentra el concepto de herencia simple la cual es que una clase pueda heredar la estructura de datos y operaciones de una superclase. Y también el concepto de herencia múltiple que se da cuando una clase puede heredar la estructura de datos y operaciones de más de una superclase.

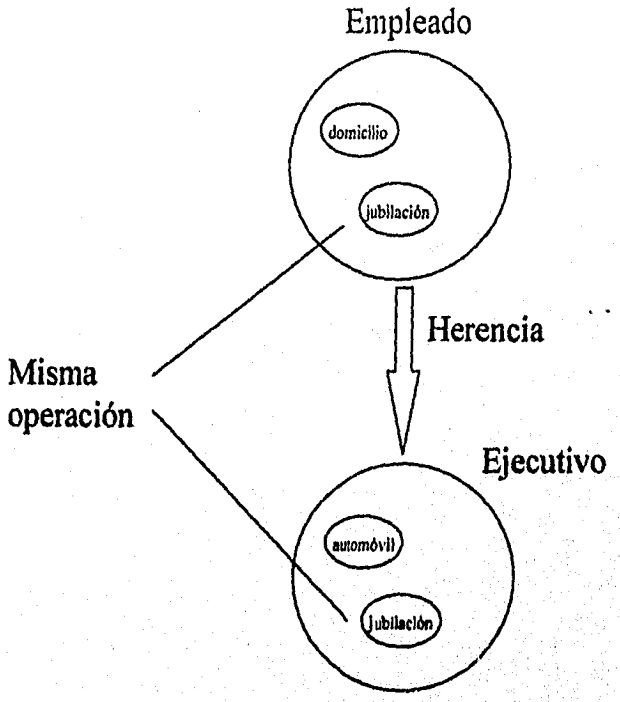
Herencia Múltiple



5.3.3 POLIMORFISMO

El término de polimorfismo se aplica cuando se utiliza una operación que está definida en dos clases, pero que en cada una de ellas aplica métodos diferentes. Recordemos que uno de los objetivos de las técnicas OO es utilizar otra vez el código. Sin embargo, algunas de las operaciones requieren adaptación para resolver necesidades particulares. La figura muestra lo anterior:

En el diagrama, se representa una clase llamada Empleado y una subclase llamada Ejecutivo, nosotros sabemos por los conceptos aprendidos anteriormente, que las operaciones domicilio y jubilación pasan por herencia a formar parte de la subclase Ejecutivo, sin embargo, la operación jubilación para cada una de las clases ejecutará métodos diferentes, ya que no es lo mismo jubilar a un empleado general que a un



empleado ejecutivo, a modo de ejemplo demostrativo, a continuación se muestra el método supuesto que ejecutaría cada una de las operaciones de las dos clases:

Clase: Empleado	**Clase: Ejecutivo**
Operación: Jubilación	**Operación: Jubilación**
Método:	**Método:**
jubilar= años trabajados * 3000	**jubilar= años trabajados * 6000**

Las técnicas orientadas a objetos deben permitir la adaptación de las clases. Cualquier persona debe poder tomar una clase de un depósito y adaptarla a sus necesidades. no se quiere que el implantador cambie el código de una clase, del mismo modo que no se quiere que nosotros cambiemos los circuitos de una videograbadora, al contrario que se reutilicen para formar clases más complejas. Por lo tanto, hay que diseñar la clase de modo que se pueda adaptar con facilidad. La frase "igual que, excepto..." describe la mayor parte de la reutilización.

El analista o el diseñador que genere una clase debe preguntarse: ¿ Cómo se utilizará esta clase en el futuro ? Debe crear la clase de forma que se pueda adaptar con facilidad a las necesidades futuras. en un ambiente OO bien administrado, todo se construye a partir de clases ya existentes o se crean nuevas clases que serán utilizadas de nuevo en el futuro. Todo se relaciona con el reuso en el pasado o en el futuro. Al desarrollar aplicaciones a partir de clases de objetos y adaptar con facilidad los objetos de forma " igual que, excepto..." se alcanzará el más alto nivel de automatización.

CAPÍTULO VI

APLICACIÓN DE BASES DE DATOS CON PROGRAMACIÓN OO A UN PROBLEMA REAL

Objetivo:

Aplicar una técnica de análisis y diseño orientado a objetos, para desarrollar un sistema con programación orientada a objetos, desarrollando y aplicando la metodología de dicha técnica a un problema real.

VI. APLICACIÓN DE BASES DE DATOS CON PROGRAMACIÓN ORIENTADA A OBJETOS A UN PROBLEMA REAL

6.1 PLANEACIÓN DEL SISTEMA

6.1.1 ANTECEDENTES

El Centro Regional de Estudios y Carreras Especializadas (C.R.E.C.E), es una escuela particular que ofrece carreras técnicas de Administración, Contabilidad y Computación. Dicho centro de estudios lleva un calendario de clases semestral para la impartición de clases.

Existe un período de inscripción, con el cual se capta semestralmente determinado número de alumnos, los cuales estudiarán una carrera y asistirán a clases de acuerdo al horario elegido, siempre y cuando este lo ofrezca la escuela. Para el caso en que una persona ya es alumno de la escuela y desea continuar sus estudios, se llevará a cabo un proceso de reinscripción, mientras que para alumnos de nuevo ingreso el proceso es de inscripción al ciclo escolar.

Para que una persona pueda ser considerada alumno de la escuela, debe cumplir ciertos requisitos previos a su inscripción (entregar documentos oficiales de escolaridad y datos personales, así como efectuar un primer pago por la membresía que se le asigna, el cual le da derecho a asistir 1 semana de clases) y obtener una carta de aceptación por las autoridades del Centro de Estudios. Una vez que se es alumno, se deben registrar y almacenar sus datos. Procedimiento que se lleva a cabo de la siguiente forma:

El alumno llena un formato en el cual se le pide que anote sus datos personales (nombre, dirección, teléfono, etc...), así como la carrera y horario que desea cursar (éstos se le dan a conocer antes de realizar cualquier trámite por medio de una pequeña demostración de las instalaciones de la escuela, para que este decida si desea estudiar ahí).

Para llevar un control de la información general del alumno, se genera un expediente (folder) el cual se identifica por el número de membresía asignado, dicho folder contiene la documentación original del alumno, el formato de llenado inicial y una copia de la carta aceptación, firmada por el alumno y su tutor.

Con lo que respecta, al pago de colegiaturas del alumno, se tiene un formato elaborado en Excel, el cual contiene las siguientes columnas: Número de Inscripción, Año y Período del Semestre, Número de Membresía (clave) asignado al alumno, Nombre, Carrera, Semestre, Horario, 27 columnas correspondientes a los pagos de colegiatura que deberá cubrir el alumno al término del curso y una última de observaciones. En caso de que el

alumno no pagara a tiempo cada semana, entonces se le adiciona un recargo por cada semana de atraso.

Los reportes que se generan, en base a esta información son: el cálculo diario de colegiaturas pagadas por los alumnos, un reporte semanal de número de alumnos que deben más de una semana, reporte semanal de cambios efectuados (carrera, horario, etc...).
Como podemos observar, lo anterior es funcional para cuando se tienen menos de 100 alumnos, pero se torna problemático llevar el control tanto de información del alumno como de los pagos realizados por este, cuando se tienen más alumnos, ya que esto implica revisar y registrar diariamente la información de cada uno de ellos, para saber si va atrasado en algún pago o realizó algún cambio. Esto significa que cualquier reporte que se quiera generar, tardará horas o días el realizarlo.

Debido a la necesidad de contar con información precisa, eficiente y rápida sobre el control de información y registro de pagos de los alumnos, se requiere llevar a cabo la automatización de dichos procesos a través del desarrollo de un Sistema, el cual permitirá un mejor control de dicha operación.

6.2 ANÁLISIS Y DISEÑO DE LA APLICACIÓN

El objetivo de desarrollar un sistema de computación para el Centro Regional de Estudios y Carreras Especializadas (C.R.E.C.E), es el de poder llevar el control preciso, eficiente y rápido de la información de los alumnos, así como también, el de controlar los pagos de colegiatura que tienen que realizar cada semana. Para ello, se utilizará una técnica de análisis y diseño OO, que divide en cuatro partes fundamentales, los procesos a realizar. Dichas partes son: Análisis de la Estructura y Comportamiento de Objetos, y Diseño de la Estructura y Comportamiento de Objetos.

6.2.1 ANÁLISIS DE LA ESTRUCTURA DE OBJETOS (AEO)

Debemos recordar que esta parte se ocupa de los tipos de objetos y sus asociaciones. Para lo cual identificaremos: Objetos, tipos de objetos, asociaciones, supertipos, subtipos. Lo anterior se representará por medio de diagramas de jerarquías de generalización, de jerarquías compuestas y finalmente se elaborará un Esquema de Objetos.

6.2.1.1 TIPOS DE OBJETOS

Tipos de Objetos

Alumno	Credencial	Colegiatura
Pago	Membresía	Otros
Inscripción	Horario	Carta de Aceptación
Adeudo	Carrera	
Recibo	Datos Grales.	

El objeto Alumno, generaliza a todas aquéllas personas que realizaron su inscripción o reinscripción y que por lo tanto para la institución son considerados alumnos (debemos recordar que un tipo de objeto, es una categorización de un grupo de objetos).

Pago será el tipo de objeto que identifica cualquier operación de cobro al alumno, ya sea por colegiaturas o el pago de algún servicio extra como: expedición de una credencial, constancia, pago de algún extraordinario, etc... Este tipo de objeto se relaciona con Adeudo, ya que este último es el monto o cantidad de dinero que el alumno debe pagar a la escuela por algún servicio de los mencionados anteriormente.

El tipo de Objeto Inscripción, en realidad es un proceso administrativo, el cual no tiene ningún efecto sobre el sistema que se va ha realizar, sin embargo, es importante mencionarlo, ya que, indica como una persona externa se considera alumno al cumplir ciertos requisitos, este objeto está compuesto de 3 más, que son: Pago de Membresía, Carta de Aceptación y Credencial.

En el momento que se captura la información del alumno, se le debe asignar un número de membresía, el que servirá como clave para realizar cualquier operación dentro de la escuela. Por lo cual se considera un objeto llamado Membresía, lo mismo sucede con Horario y Carrera.

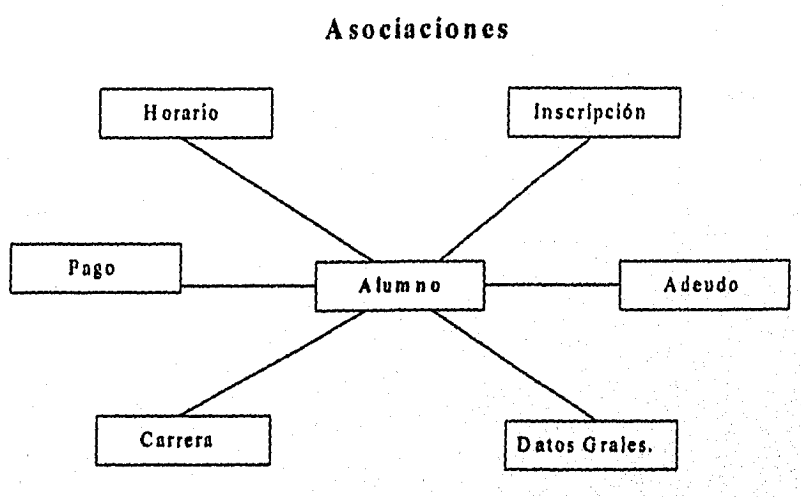
Una parte importante del sistema es la de registrar los pagos realizados por el alumno durante el transcurso del semestre. Este pago o adeudo se indica con el objeto Colegiatura, que se diferencia del objeto Otros, ya que este último considera el pago o adeudo de alguna constancia, credencial, etc...

Recibo es el tipo de objeto que indica la generación de un documento que avala el pago realizado de una colegiatura o de otros.

El tipo de objeto Datos Generales, es un objeto compuesto por la información general del alumno (nombre, dirección, teléfono, etc...).

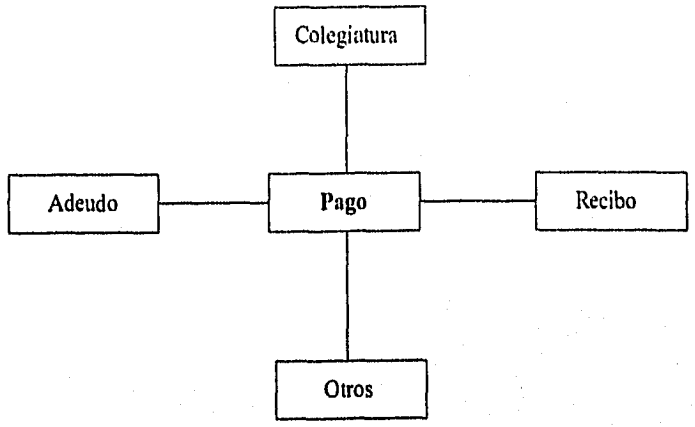
6.2.1.2 ASOCIACIONES Y RELACIONES DE TIPOS DE OBJETOS

Las asociaciones del objeto Alumno para con otros objetos se representan en la siguiente figura:

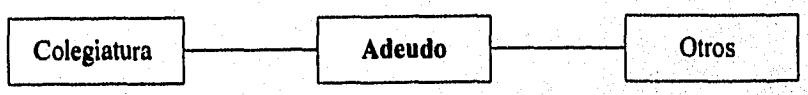


Es importante indicar al sistema como se asocian los objetos internamente, con lo que respecta al Alumno (objeto), sabemos que se asocia con un objeto Inscripción (trámite administrativo), a través del cual se obtienen Datos Generales, al igual que el nombre de la Carrera a cursar y el Semestre, también se indica un Horario seleccionado y el Estado que guarda un alumno dentro de la institución (activo, inactivo, baja temporal, etc...), de la misma forma el Pago y Adeudo.

El hablar de un Pago, significa el cubrir un Adeudo de alguna Colegiatura u Otros y emitir un Recibo, estas asociaciones se indican a continuación:



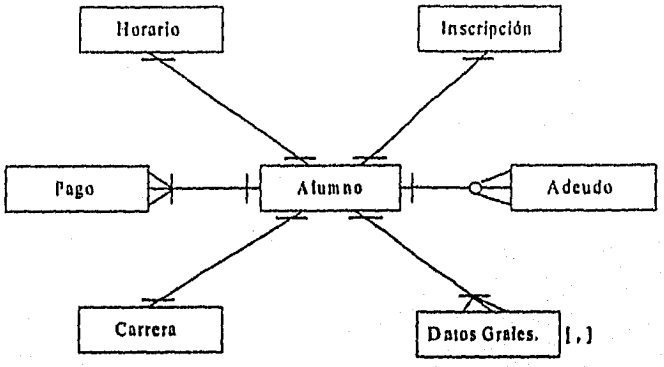
Un Adeudo, solamente puede ser de Colegiaturas u Otros:



Podemos observar que existen tres Tipos de Objetos, que relacionan a todos los demás, éstos son: Alumno, Pago y Adeudo.

Con lo que respecta a las relaciones de cada objeto para con los demás, se tiene:

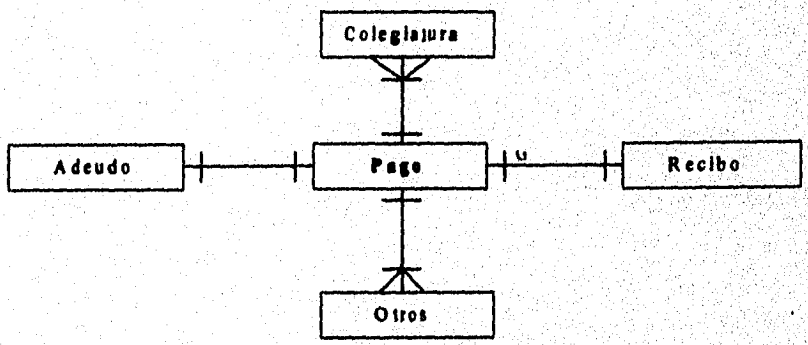
Relaciones



En este diagrama se observan algunos símbolos de cardinalidad, los cuales pueden consultarse en el apéndice A de esta tesis. Es importante recordar que el hablar de un Alumno, implica la generalización de instancias, es decir, que Alumno (tipo de objeto) contiene Instancias (valores), por ejemplo: Alberto, José, Martha, etc.... Se indica el tipo de relación arriba de la línea de asociación y en sus extremos la cardinalidad de un objeto a otro.

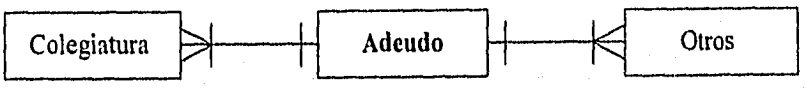
Podemos darnos cuenta, como el tipo de objeto Alumno, se relaciona con sus datos generales, pagos que realice o adeudos que tenga para con la institución.

En el caso del tipo de objeto Pago, sus relaciones son:



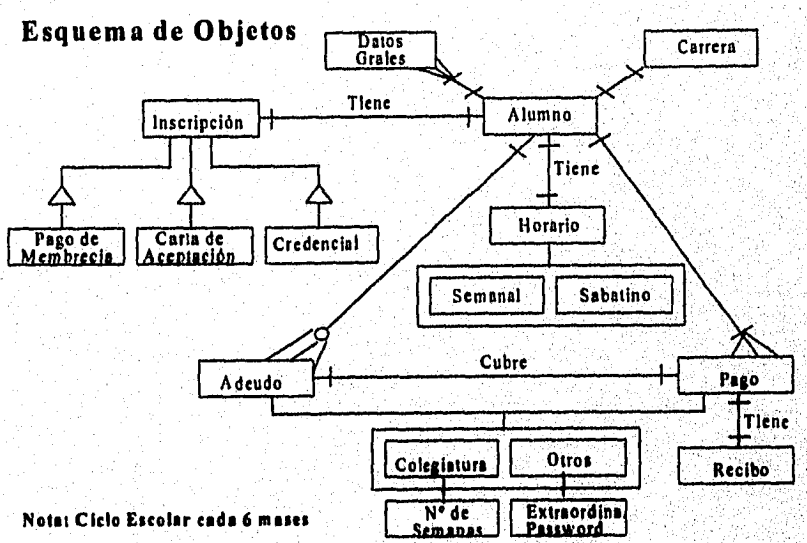
El pago que se realiza será, o de Colegiatura o de Otros, adeudando cierta cantidad y generándose un recibo por la cantidad pagada.

Adeudo tiene las siguientes relaciones:



6.2.1.3 ESQUEMA DE OBJETOS

En base a los diagramas obtenidos anteriormente, se genera otro llamado Esquema de Objetos, el cual está compuesto de los diagramas de asociaciones, relaciones, supertipos y subtipos de objetos. Para el caso en que no se tiene experiencia en el análisis OO, resulta muy útil visualizar dicho diagrama:



En este diagrama se puede observar, las jerarquías de generalización, es decir, la jerarquía de los objetos, identificando los Supertipos, para este caso, Alumno, Adeudo y Pago, así como los subtipos: Datos Generales, Carrera, Semestre, Horario, Colegiatura, Otros y Recibo. También podemos observar algunos objetos que están compuestos por otros (jerarquías de composición), como el objeto Inscripción que se compone por: Pago de Membresía, Carta de Aceptación y Credencial.

6.2.2 ANÁLISIS DEL COMPORTAMIENTO DE OBJETOS (ACO)

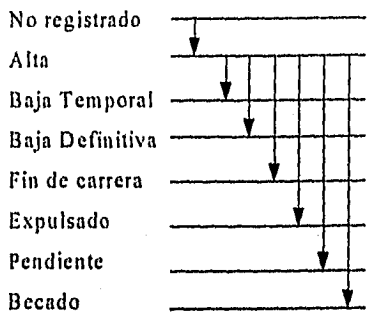
En esta parte se estudia, lo que le sucede a los objetos al paso del tiempo. Se identifican los Estados de un Objeto, Transiciones de Estado (como pasa de un estado a otro un objeto), eventos, tipos de eventos y se realiza un diagrama final, llamado Esquema de Eventos.

6.2.2.1 ESTADO Y TRANSICIÓN DE OBJETOS

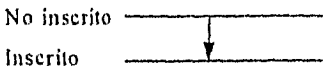
Para mostrar a detalle, los diferentes estados por los que pasa un objeto al paso del tiempo, se utilizan los diagramas de rejas (diagramas que representan la transición de los estados de un objeto). A continuación se muestran los diagramas de rejas de cada uno de los tipos de objetos:

Diagramas de Estado y Transición de Estados

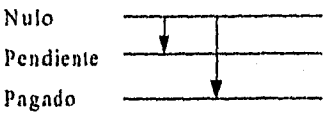
ALUMNO



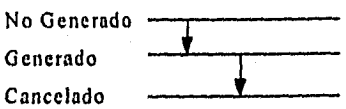
INSCRIPCION



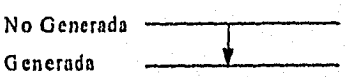
ADEUDO



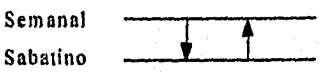
RECIBO



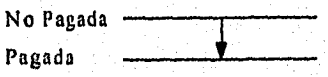
CREDENCIAL



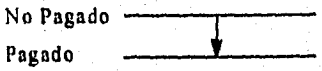
HORARIO



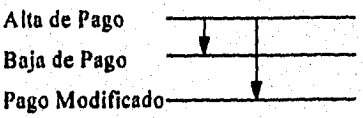
COLEGIATURA



OTROS



PAGO

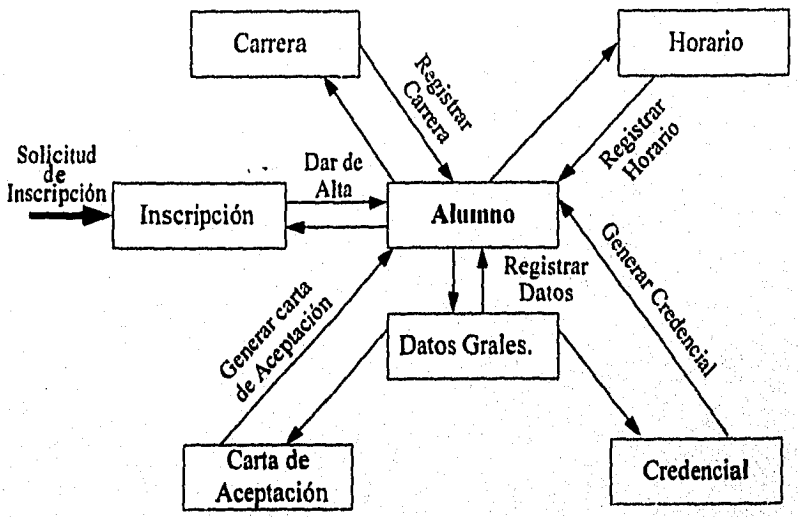


CARTA DE ACEPTACION

6.2.2.2 IDENTIFICACIÓN DE EVENTOS

Para identificar los eventos que ocurren en nuestro sistema, es necesario, primero, identificar los diferentes mensajes que se enviarán de un objeto a otro para comunicarse, a continuación se muestran cada uno de estos mensajes ilustrados en los diagramas siguientes:

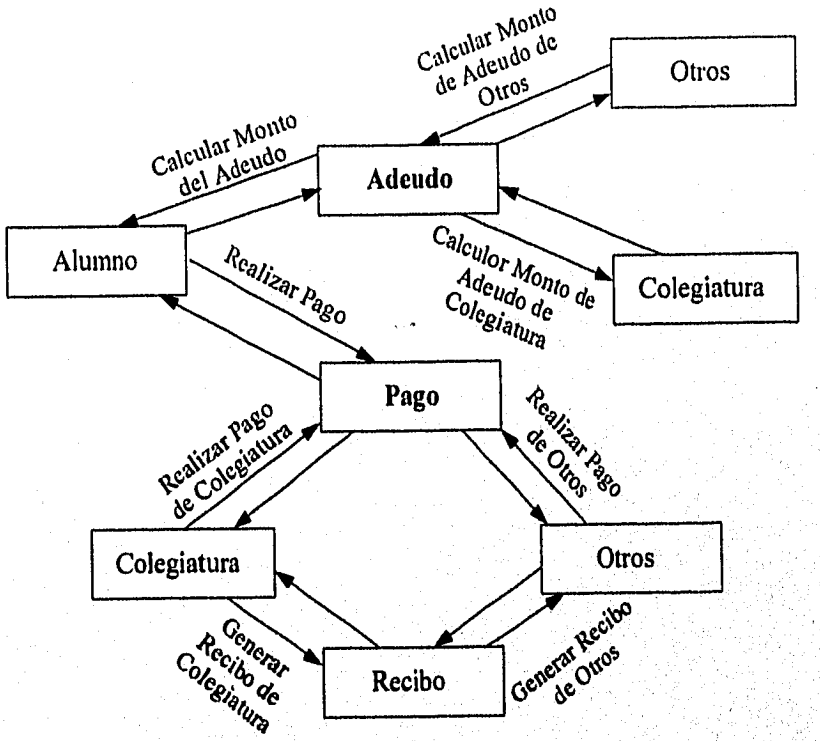
Diagramas de Mensajes



En el diagrama anterior, se muestra la solicitud que hace el objeto Inscripción, para dar de alta a un alumno y este a su vez solicita el registro de los Datos Generales, para después generar una Carta de Aceptación y una Credencial.

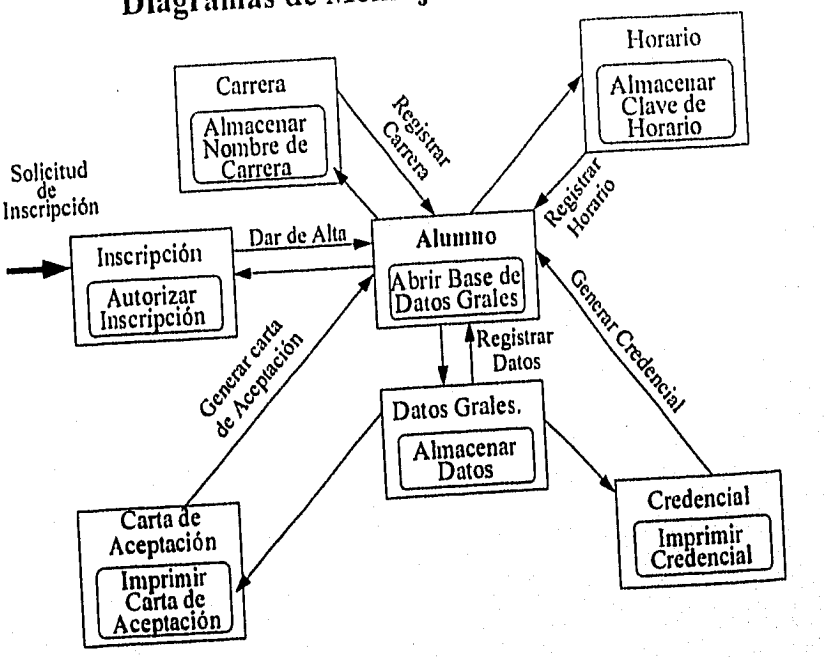
Para el caso de un Adeudo del alumno, debemos indicarle al sistema por medio de un mensaje, que el Alumno solicita el cálculo del monto del Adeudo, lo que Adeudo solicitará a Colegiatura que se calculen las colegiaturas adeudas o en su caso el Adeudo de Otros. Si es el caso de Pago de un alumno, entonces, se enviará un mensaje de Alumno a Pago, solicitándole realizar un pago, este a su vez, mandará otro mensaje, de Pago a

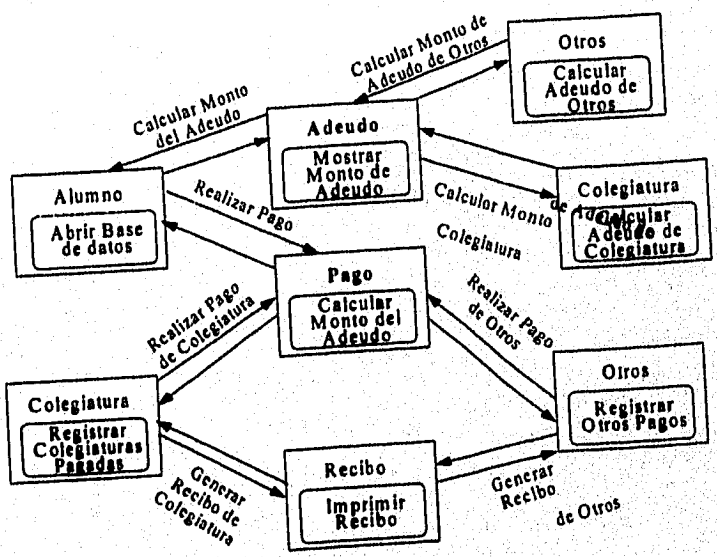
Colegiatura u Otros, solicitando realizar pago de colegiatura o realizar Pago de Otros. Después, ya sea, Colegiatura u Otros, según corresponda, enviará un mensaje a Recibo, solicitándole, generar un recibo de pago de colegiatura o un pago de Otros. Lo anterior se muestra a continuación:



Ahora, se deben indicar las operaciones, que harán posible que los mensajes solicitados entre objeto se realicen. Para ello, utilizaremos los diagramas anteriores, adicionando en cada uno de ellos, las operaciones correspondientes:

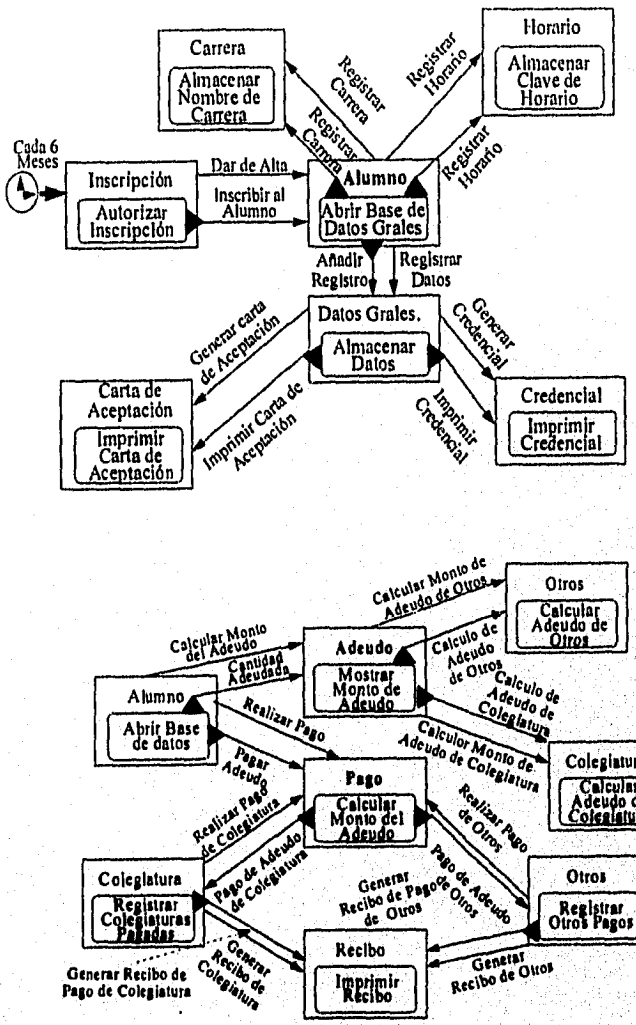
Diagramas de Mensajes con Operaciones





Por último, se elabora un diagrama llamado Esquema de Eventos, que contiene, las operaciones, los eventos y mensajes a realizar entre los objetos. De esta forma queda completamente claro, cuales será la secuencia a seguir, para cuando se solicite alguna información, ya sea, Información del Alumno (datos generales) o el pago de colegiatura o pago de otros de un alumno.

Diagramas de Mensajes con Operaciones y Eventos



El diagrama anterior, servirá de base para realizar, el diseño de la estructura de datos y comportamiento de objetos, que se describirá en el siguiente capítulo.

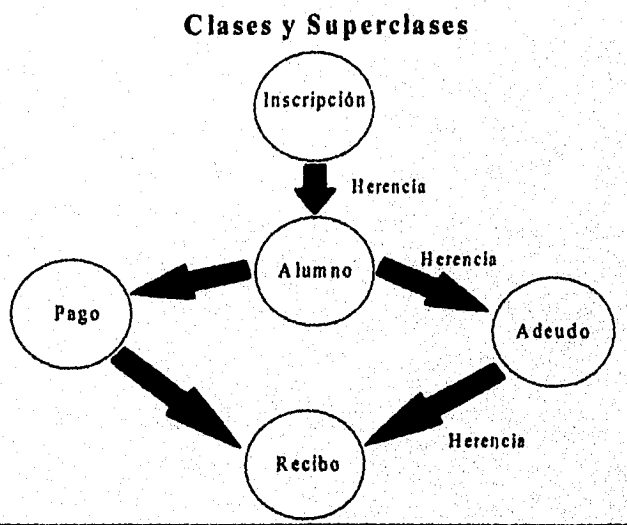
6.2.3 DISEÑO DE LA ESTRUCTURA DE OBJETOS (DEO)

El realizar el diseño de la aplicación, significa, tomar en cuenta el análisis realizado, el cual nos proporciona un amplio panorama del problema a resolver, y determinar si se necesitan componentes adicionales para mejorar el dominio del problema. Para el caso de la estructura de objetos, se deberá identificar las Superclases y subclases, Herencia y Estructuras de los datos (asociaciones y relaciones de las tablas), así como el diseño de las bases de datos.

6.2.3.1 IDENTIFICACIÓN DE CLASES - HERENCIA

Se identifican, las siguientes Superclases: Alumno, Pago y Adeudo. La clase Recibo, es una subclase de Pago y Adeudo. Las clases Pago y Adeudo, son subclases de Alumno, mientras Alumno es una subclase de Inscripción. Lo anterior, se muestra en el siguiente diagrama:

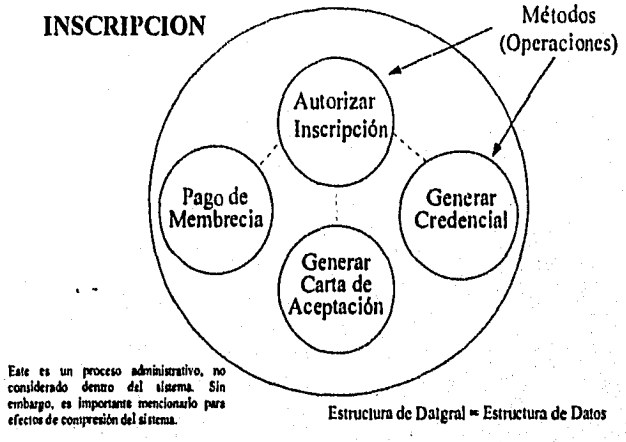
En el diagrama, se observa la herencia entre las clases.



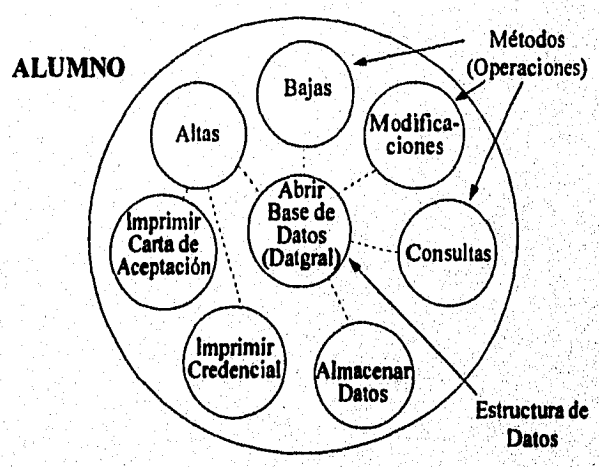
6.2.3.2 MÉTODOS Y ESTRUCTURAS DE DATOS DE LAS CLASES

A continuación se presentan diagramas, que especifican los métodos de cada una de las clases observadas en el diagrama anterior:

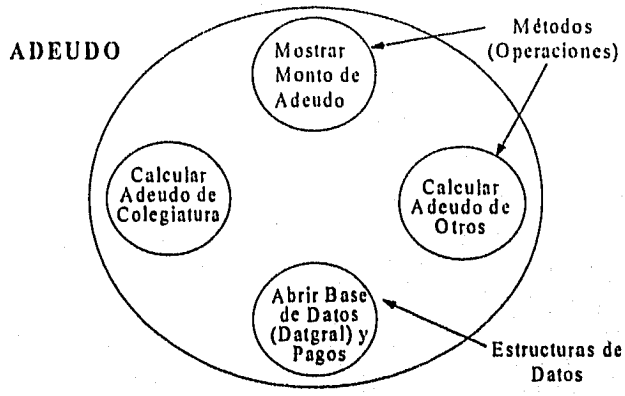
Métodos y Estructuras de Datos de la Clase Inscripción



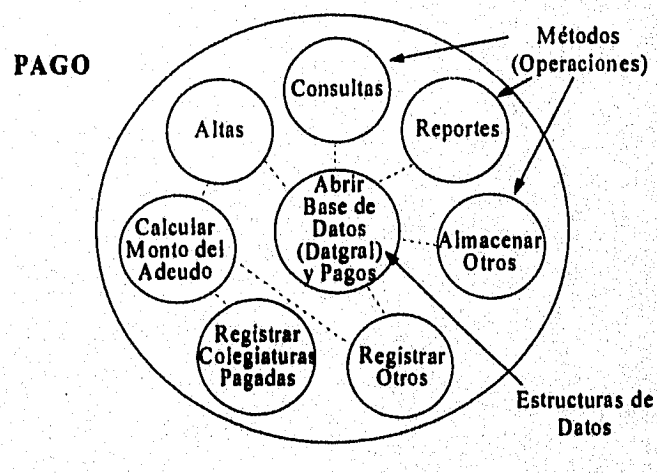
Métodos y Estructuras de Datos de la Clase Alumno



Métodos y Estructuras de Datos de la Clase Adeudo



Métodos y Estructuras de Datos de la Clase Pago



6.2.4 DISEÑO DEL COMPORTAMIENTO DE OBJETOS (DCO)

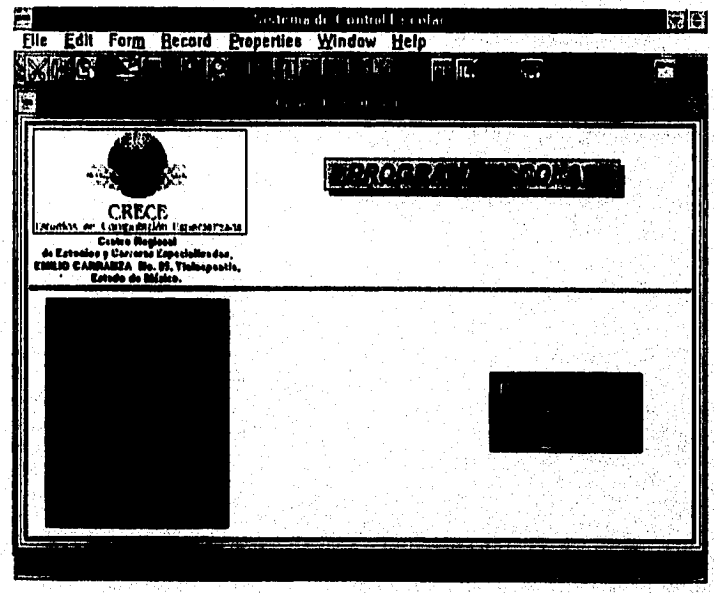
Esta última parte de la técnica aplicada para el desarrollo de la aplicación, se encarga del diseño de métodos. Se identificarán métodos, lógica de procedimientos, diseño de las interfases que interactúan con el usuario y la fabricación de prototipos.

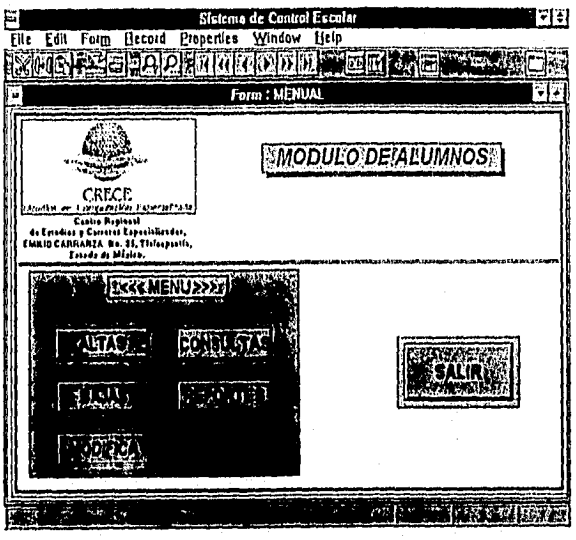
6.2.4.1 MÉTODOS

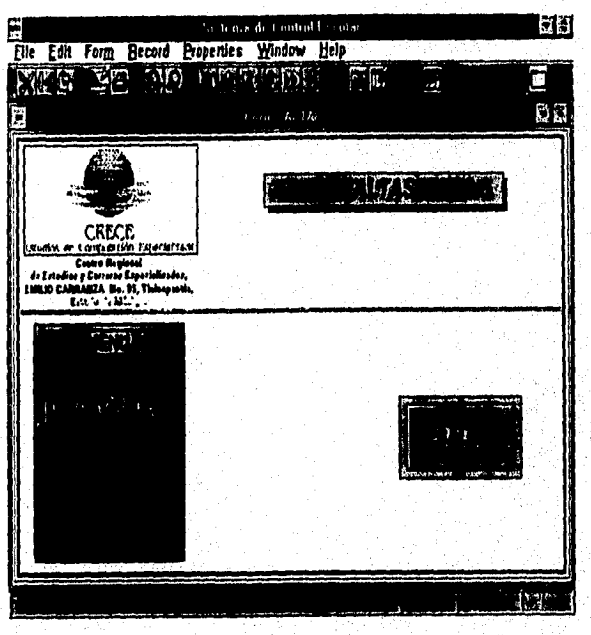
En esta parte, se debe indicar la codificación de los métodos y funciones del sistema.

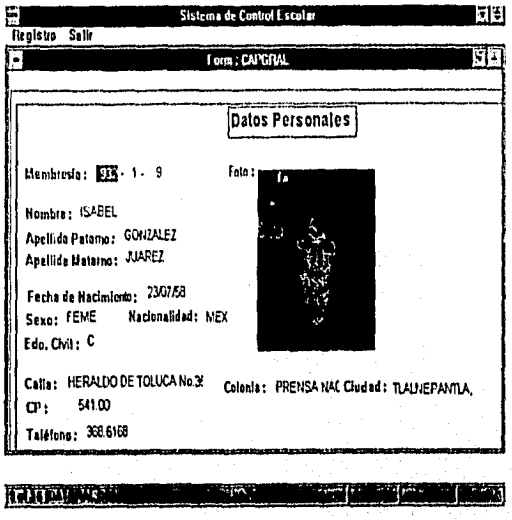
6.2.4.2 DISEÑO DE INTERFASES DEL SISTEMA

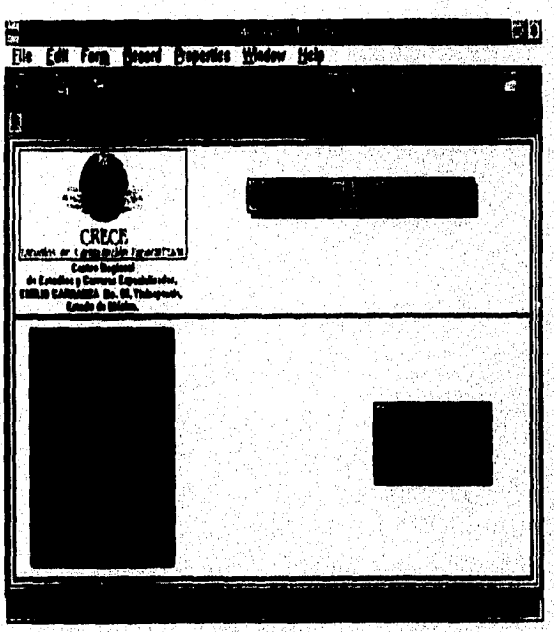
Las pantallas que se manejarán en el sistema, serán las siguientes:

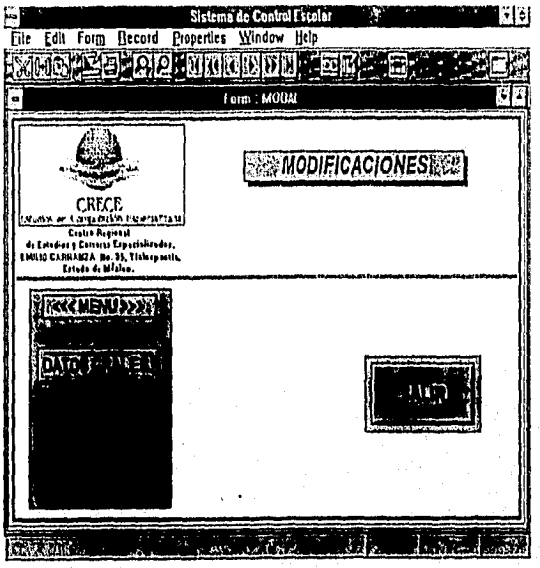


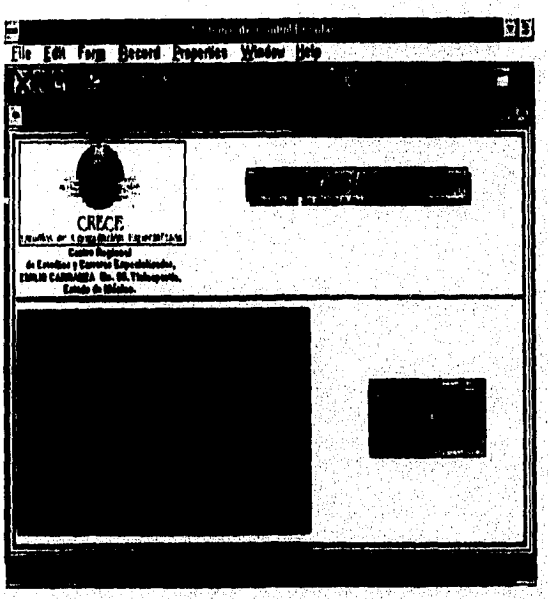

Sistema de Control Escolar
File Edit Form Record Properties Window Help
Form : MENUAL
CRECE
Centro Regional
de Estudios y Carreras Especializadas,
EMILIO CARRANZA No. 31, Tlalnepantla,
Estado de México.
MODULO DE ALUMNOS
<<< MENU >>>
ALTAS
CONSULTAS
SALIR

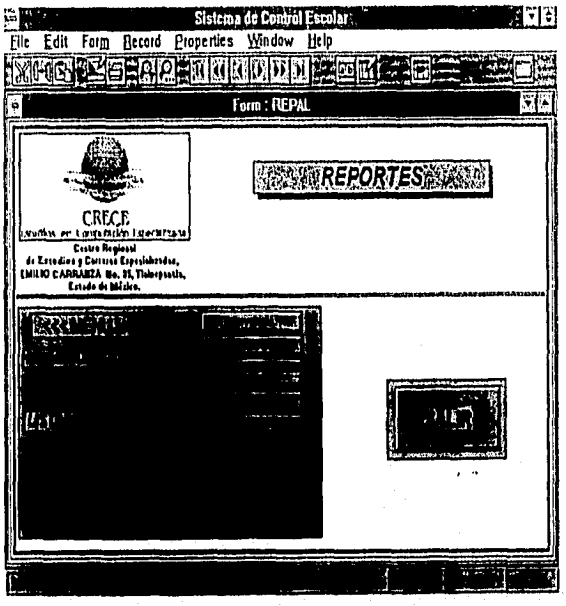

File Edit Form Record Properties Window Help
CRECE
Centro Regional
de Estudios y Carreras Especializadas,
EMILIO CARRANZA No. 31, Tlalnepantla,

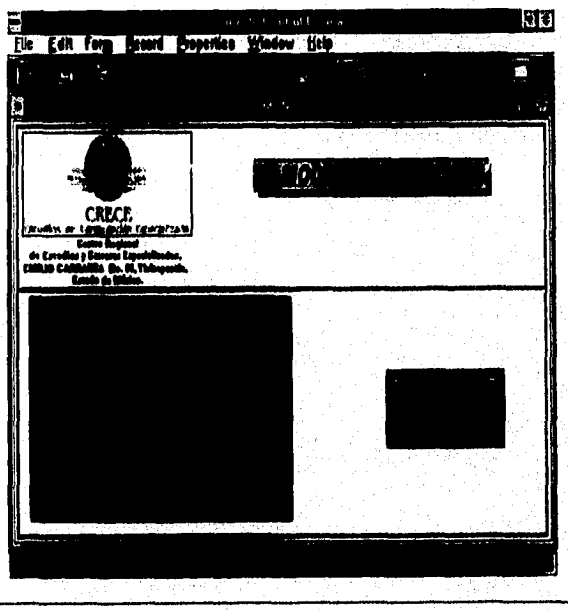

Sistema de Control Escolar
Registro Salir
Form : CAPGRAL
Datos Personales
Membresía: 93 - 1 - 9
Foto:
Nombre: ISABEL
Apellido Paterno: GONZALEZ
Apellido Materno: JUAREZ
Fecha de Nacimiento: 23/07/58
Sexo: FEME
Nacionalidad: MEX
Edo. Civil: C
Calle: HERALDO DE TOLUCA No.3
Colonia: PRENSA NAC Ciudad: TLALNEPANTLA,
CP: 541.00
Teléfono: 368.6168

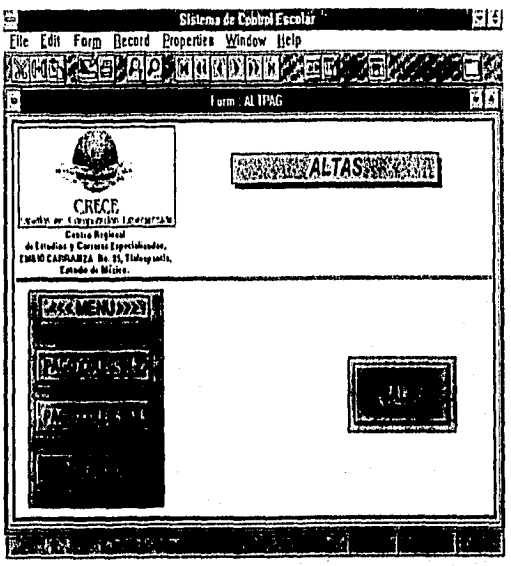

File Edit Form Record Properties Window Help
CRECE

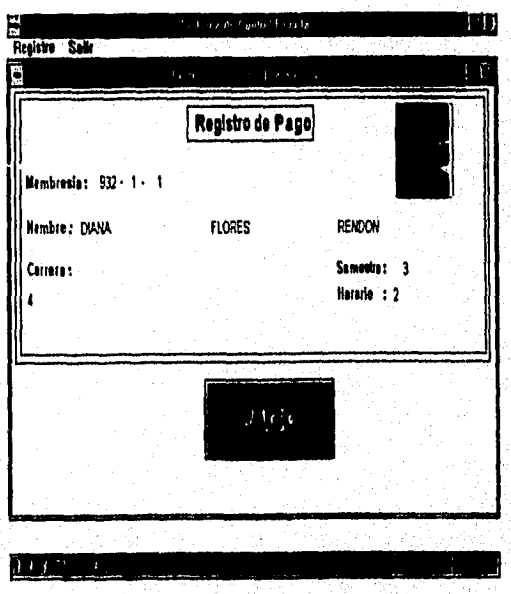

Sistema de Control Escolar
File Edit Form Record Properties Window Help
Form : MODAL
CRECE
MODIFICACIONES
<<<MENU>>>

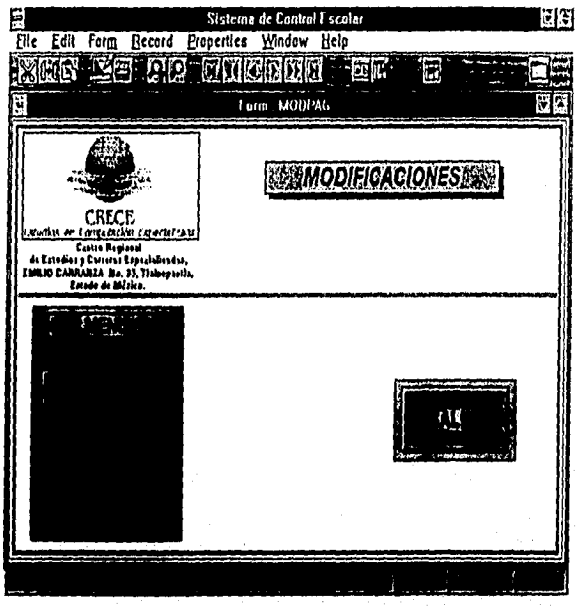

File Edit Form Record Properties Window Help
CRECE

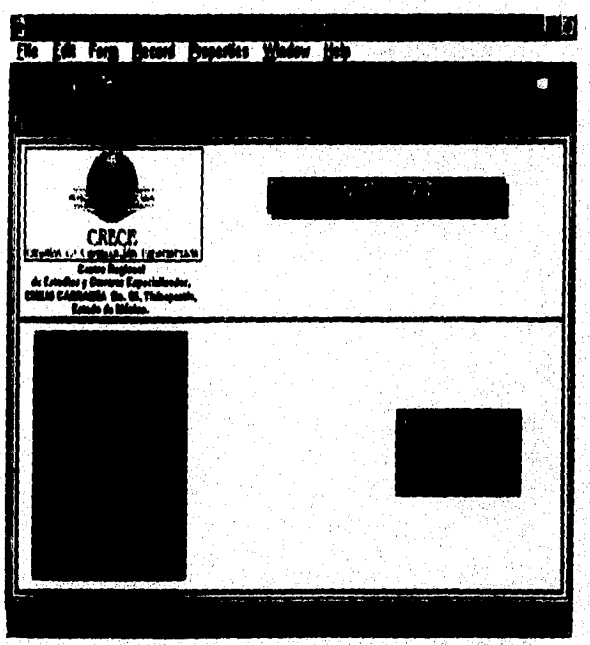

Sistema de Control Escolar
File Edit Form Record Properties Window Help
Form : REPAL
REPORTES
CRECE

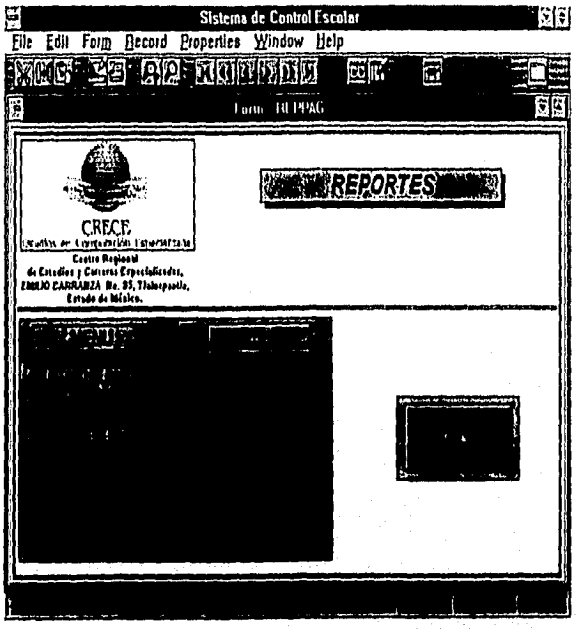

File Edit
CRECE

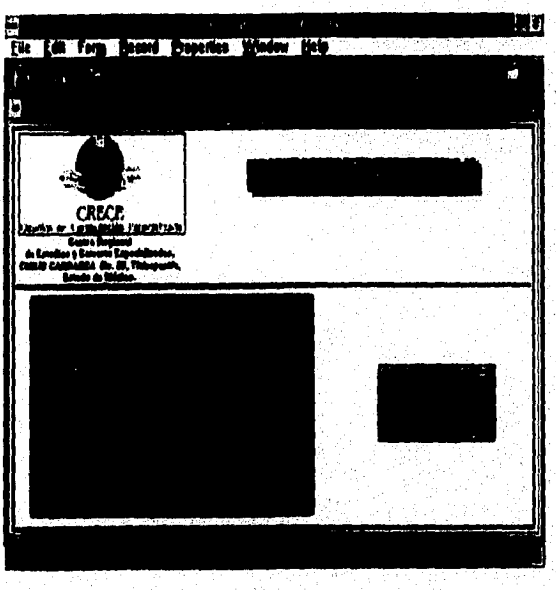

Sistema de Control Escolar
File Edit Form Record Properties Window Help
Form : ALTPAG
ALTAS
CRECE
Centro Regional
de Estudios y Cursos Especializados,
EMILIO CARRANZA No. 15, Tlalnepantla,
Estado de México.
<<<MENU>>>


Registro Salir
Registro de Pago
Membresia: 932 - 1 - 1
Nombre: DIANA
FLORES
RENDON
Carrera:
4
Semestre: 3
Horario : 2


Sistema de Control Escolar
File Edit Form Record Properties Window Help
Form MODIFAL
CRECE
MODIFICACIONES


File Edit Form Record Properties Window Help
CRECE


Sistema de Control Escolar
File Edit Form Record Properties Window Help
REPORTES
CRECE


CRECE

6.2.4.3 PROTOTIPO

CONCLUSIONES

En base a los conocimientos adquiridos a lo largo de nuestra tesis, hemos aprendido a diferenciar un lenguaje orientado a objetos de un lenguaje orientado a eventos. Programas como Access y Visual Basic corresponden a este último grupo, siendo que C++ y SmallTalk son lenguajes orientados a objetos.

Para distinguir un lenguaje de programación orientado a objetos (LPOO) de un lenguaje de programación orientado a eventos (LPOE), se debe tomar en cuenta lo siguiente:

Un LPOO debe contener una estructura de programación para definir una clase, esto permitirá la existencia del concepto de Herencia.

En un LPOO, existen operaciones que proporcionan una protección y evitan el uso inadecuado de los objetos, es decir, se aplica el concepto de encapsulado de los datos.

Un LPOO debe poder reconocer las solicitudes de operaciones y la selección del método adecuado.

Un LPOE contiene herramientas visuales (en pantalla) como botones, gráficos, figuras, etc..., que permiten mejorar la apariencia de la aplicación a desarrollar, sin embargo, en sus instrucciones de programación no cuenta con las características mencionadas anteriormente para un LPOO, más bien se caracteriza, por poseer funciones implementadas dispuestas para ser ejecutadas como parte de la programación, las cuales cambiarán el estado de un objeto, a este cambio de estado es a lo que se le llama evento o suceso. Siendo un ejemplo de evento, la activación de un botón en pantalla, el click dado a un botón de la pantalla, etc...

Cabe mencionar, que al principio del estudio de este tema de tesis, creíamos que todo lenguaje que utilizará elementos gráficos en pantalla como iconos, botones, campos foto, etc... era un lenguaje orientado a objetos, ya que los mismos autores de los manuales de dichas aplicaciones manejaban dichos elementos con el nombre de objetos, pero en el transcurso del desarrollo de nuestra tesis hemos podido llegar a diferenciar claramente entre un LPOO de un LPOE.

Debido a que el desarrollo de la aplicación se realizó en PARADOX y observando su lenguaje de programación , el cual no cumple con las características de un LPOO, llegamos a la conclusión de que PARADOX es un software manejador de bases de datos híbridas (es decir, estas bases presentan características de bases de datos relacionales y de bases de datos orientadas a objetos), que contiene un lenguaje de programación (ObjectPal) orientado a eventos.

Para lograr el análisis y diseño de la aplicación, se utilizó una metodología que descompone en cuatro partes fundamentales la descripción de este proceso, siendo estas, el análisis de la estructura y comportamiento de objetos, así como, el diseño de la estructura y comportamiento de objetos. Esta metodología no es la única, de hecho en los libros consultados se maneja una secuencia libre de elaboración del análisis y diseño de la aplicación, no importando si algún paso se realiza antes que el otro, sin embargo, para fines prácticos, consideramos que la secuencia de la metodología aplicada, permitía una mejor comprensión, facilitando el entendimiento del problema.

Lamentamos, el hecho de no contar con alguna herramienta CASE, para poder corroborar de forma práctica todos los conceptos aprendidos sobre programación orientada a objetos, así como el no haber podido visualizar y elaborar diagramas en forma práctica con dicha herramienta.

Por último, queremos mencionar, que con el estudio de este tipo de temas nuevos y pioneros en nuestro país, se destaca una gran falta de conocimiento, incertidumbre y malos entendidos entre la gente del argot computacional acerca del tema de orientación a objetos, debido en gran parte por la información inequívoca proporcionada por los publicistas de dichos productos.

GLOSARIO

Programa: Es un conjunto de líneas de instrucciones que nos permiten indicarle a la computadora lo que va a hacer cuando estas se ejecuten.

Lenguaje de programación: Es donde podemos escribir nuestros programas, para que posteriormente cuando se ejecuten sean traducidos a lenguaje máquina mediante los compiladores y ensambladores que posee el lenguaje de programación.

Constante: Es un valor fijo que no cambia

Variable: Es un valor cambiante, es decir, en un cierto momento tiene un valor y en otro uno distinto.

Operador: Es el símbolo que nos permite realizar operaciones matemáticas o lógicas como sumas, restas, comparaciones, etc...

Expresiones: Es el conjunto de constantes, variables y operadores que definen algún cálculo matemático o lógico y en éstos se pueden usar paréntesis para poder distinguir alguna operación respecto a otra.

Sentencias de Asignación: Son aquéllas expresiones a las cuales se les asigna una variable.

Sentencias de Entrada y Salida: Permite comunicarnos con el exterior (Usuarios, periféricos y la memoria principal de la computadora).

Sentencias de Control: Permiten efectuar una ruptura de secuencia de instrucciones en el programa.

Rutinas: Módulos que componen el programa y que son llamados constantemente.

Procedimientos: Módulos que componen el programa y que son llamados constantemente pero que no regresan ningún valor cuando de les llama.

Funciones: Módulos que componen el programa y que son llamados constantemente pero que regresan algún valor cuando de les llama.

Diagramas de Flujo: Los cuales contienen una serie de símbolos universales para representar cierto tipo de operaciones en un programa.

Módulo: Se define como un conjunto formado por una o varias instrucciones.

Sentencia de asignación: Es el operador mediante el cual hacemos una asignación ya sea de una variable a otra o de un campo a otro, o alguna otra operación en la cual se va a almacenar un valor.

Ordinogramas estructurados: Es un diagrama de flujo de proceso.

Algoritmo: Es un conjunto de instrucciones en un orden determinado para resolver algún problema especifico.

Pseudocódigo: Es la forma alternativa a los diagramas de flujo con que se puede analizar y resolver problemas complejos y grandes.

Estructuras de datos: Son las diferentes formas que se utilizan para poder manipular información en una forma especifica.

CASE: Computer Aided Software Engineering. Ingeniería de Software Asistida por Computadora.

Programación visual: Es aquél tipo de programación que hace que la programación sea en forma gráfica y visible para el usuario, por ejemplo en paquetes de dibujos los objetos pueden dibujarse y modificarse, manipulando directamente el objeto en pantalla y sin tener que cambiar los datos en forma manual para poder modificarlo.

Generadores de código: Programas que permiten generar el código a partir de definir los diagramas necesarios para crear una aplicación.

lenguajes no por procedimientos: Lenguajes que generan automáticamente el código del programa , con tan solo la definición del problema que hace el usuario, evitando de esta forma que el usuario programe el código para resolver el problema.

Bibliotecas de clases: Es una colección de clases genéricas que pueden adaptarse para una aplicación especifica, estas maximizan la reutilización.

Codificación: Son las instrucciones o palabras reservadas de un lenguaje de programación para escribir un programa.

Mensaje: Es una solicitud que se envía a un objeto para que realice una operación determinada o entregue un valor.

Base de datos: Es un conjunto de información relacionada entre sí, almacenada en archivos.

Base de datos estática: Es aquél tipo de base de datos en la cual el tipo de datos que se grabo en ella no se vuelven a modificar.

Base de datos dinámica: Son las bases de datos que constantemente se están actualizando y modificando.

DBMS : Sistemas de Gestión de Bases de Datos.

Archivo: Es un conjunto de registros.

Registro: Es un conjunto de campos.

Campo: Es un conjunto de cadenas de información.

Cadena: Es un conjunto de caracteres.

Integridad de los datos: Es la protección de los datos ante un borrado accidental o una modificación incontrolada.

Dispositivos de almacenamiento: Son los periféricos donde podemos almacenar nuestra información (Discos duros, diskettes, cintas etc.).

Sistema operativo: Es el programa que nos permite comunicarnos con la computadora, además de que controla y planifica la ejecución de muchos programas.

Administradores de bases de datos: Son quienes deben decidir que información se va a guardar en la base de datos.

Bytes: Es una unidad de almacenamiento que equivale a 8 bits, o sea a un caracter.

bits: Es la unidad mínima conque se construyen caracteres para que los pueda leer la computadora, ya que un valor de un bit puede ser 0 o 1 que son precisamente los valores que lee la máquina internamente ya que los circuitos que la componen son digitales.

Lenguaje de alto nivel: Son los lenguajes opuestos a los lenguajes máquina y ensamblador ya que son más ricos en estructuras y más fáciles de programar.

Estructura de una base de datos: Esta es la definición de los datos en el sistema, es decir, los vínculos con otros elementos dato, la forma de describir una base de datos es el modelo estructural y este a su vez se conoce como modelo de datos.

Entidad: Es un objeto que se distingue de los demás objetos debido a que tiene un conjunto especifico de atributos y una relación es una asociación entre varias entidades.

Punteros: Son estructuras de programación que se utilizan en los lenguajes de programación.

Instancia: A la colección de información almacenada en la base de datos, en un determinado tiempo.

Esquema: Se usa para referirse al diseño.

Listas ligadas: Son estructuras de programación que se utilizan en los lenguajes de alto nivel.

Memoria secundaria: Se le llama así a los periféricos donde podemos almacenar información (disco y/o cinta).

DDL: Lenguaje de definición de datos.

Metadatos: Es definir datos sobre datos.

DML: Lenguaje de manipulación de datos.

Consulta: Es un enunciado que solicita la recuperación de información a la parte de un DML que implica recuperación de información se llama lenguaje de consultas, generalmente este nombre se aplica como un sinónimo de lenguaje de manipulación de datos.

DBA: Administrador de base de datos.

Compilador: Es un traductor de los lenguajes de programación a lenguaje máquina (O o 1) para que estos puedan ser leídos y entendidos por la computadora.

Modelos de bases de datos: Son una forma diferente de ver los datos ya que son un enfoque no tradicional de los datos, son un marco de referencia bajo el que describen las relaciones lógicas entre los datos que forman las bases de datos.

Atributos: Es una propiedad o característica de un objeto.

Relaciones: Es la cardinalidad que existe de un objeto a otro.

Segmento: Son las entidades, estructuras que agrupan todos aquéllos atributos o datos con características comunes.

Altura de un Árbol: Es el número de niveles que tiene un árbol.

Peso de un Árbol: Es el número total de segmentos que constituye un árbol.

Diagramas de estructura de datos: Son esquemas mediante los cuales podemos representar el diseño de una base de datos de red.

CODASYL: Se le llamó así a la primera especificación estandar de una base de datos, que apareció a finales de los sesenta.

Entidad Fuerte o Propia: Es aquélla cuyas propiedades que la identifican sólo hacen referencia a la propia entidad. Por ejemplo: alumno, matrícula, almacén, etc...

Entidad Débil o Regular: Es aquélla que sólo tiene sentido gracias a las propiedades que identifican a otras entidades (fuertes o a su vez débiles). Por ejemplo: rfe que está definida por las propiedades de la existencia de un empleado.

Tabla: Consiste de columnas y filas lo que es lo mismo atributos y tuplas respectivamente, con estos nombres se conocen en el modelo relacional.

Relación: Se utiliza para indicar que la tabla mantiene una asociación con otras tablas.

Grado de una tabla relacional: El cual es el número de atributos que la forman, y Cardinalidad de una tabla relacional que es el número de tuplas que contiene.

Dominio: Es el conjunto de todos los posibles valores para una o más columnas de una tabla relacional, se distinguen dos tipos de dominios.

Dominios Generales o Continuos: Son aquéllos que contienen todos los posibles valores entre un máximo y un mínimo predefinido.

Dominios Restringidos o Discretos: Son aquéllos que contienen ciertos valores específicos entre un máximo y un mínimo predefinido.

Submodelo: Es el conjunto de relaciones a las que puede acceder un usuario determinado, y subesquema las declaraciones correspondientes al submodelo.

Clave: La cual es un atributo o conjunto de atributos cuyos valores distinguen unívocamente una tupla específica de una tabla.

Clave Candidata: Es cuando en una tabla, más de una columna (o combinación de ellas) puedan servir de clave.

Clave Primaria: Se le llama así a la clave candidata que se eligió para que identifique a la tabla.

Claves alternativas o secundarias: Son todas las claves que no son llaves primarias.

Clave Ajena: Aquél atributo o conjunto de atributos que en la tabla donde se encuentran no son clave, y sus valores se corresponden con la clave principal de la misma u otra tabla.

Interrelación: Es una asociación entre tablas mediante atributos que tienen el mismo dominio. Generalmente estos atributos hacen referencia a los mismos conceptos y la interpelación se establece entre la clave ajena de una tabla (tabla hija), y la clave principal de la otra tabla (tabla padre).

Vista: La vista es una tabla virtual que no existe en realidad como una tabla base en memoria auxiliar (disco duro, cinta, etc...); sólo se almacena, si se desea. Este tipo de tablas es una forma de ver determinados datos de tablas base.

Entidades fuertes : Son aquéllas que tienen identificadores propios y únicos, o sea que tienen existencia propia y no dependen de otra entidad.

Entidades débiles: Derivan su existencia de los atributos que son identificadores de una o más entidades fuertes.

Relación: A la asociación entre varias entidades, así al conjunto de relaciones del mismo tipo se le llama conjunto de relaciones.

Relaciones binarias: son aquellas que Implica a dos conjuntos de entidades.

Cardinalidad de asignación: La cual determina el número de entidades con las que puede asociarse otra entidad por medio de un conjunto de relaciones.

Lenguajes orientados a objetos híbridos: Es una combinación de los lenguajes procedimentales con parte de los lenguajes orientados a objetos.

CLOS: Common Lisp Object System, sistema de objetos comunes de Lisp.

EIFEEL: Es un lenguaje orientado a objetos derivado de Smalltalk.

ACTOR: Es un lenguaje de programación orientado a objetos similar a pascal.

Blob: Binary Large Object, objeto binario de gran tamaño.

Agente: Es un objeto activo que sigue los sucesos que ocurren en una aplicación y actúa de forma autónoma para alertar a los usuarios respecto a operaciones que están mal, o que operación sigue después de otra.

Disparador: Es un procedimiento que se ejecuta automáticamente siempre que surge una condición predeterminada.

Widget "Chisme": Es un elemento de ventana predefinido que se utiliza en X-Windows.

El procesamiento distribuido: Es una interconexión de computadoras en una red, en la que cada computadoras gestionan su propia carga de trabajo.

Objeto: Los objetos son entidades que contienen en su interior datos y las instrucciones que operan sobre esos datos.

Objeto pasivo: Es aquél que solamente actúa cuando se le hace una petición, es decir cuando el botón se pulsa con el mouse en algún programa de aplicación.

Herencia Simple: Una clase puede heredar datos y métodos de una clase simple así como añadir o sustraer comportamiento por si misma.

Herencia Múltiple: Es la posibilidad de una subclase de adquirir los datos y métodos de más de una clase.

Bases de Datos Híbridas: Son bases de datos que incluyen las características relacionales con parte de las características de las bases de datos orientadas a objetos.

Bases de Datos Orientadas a objetos: Una base de datos orientada a objetos es una base de datos inteligente, almacena datos y métodos, está diseñada para ser eficaz, desde el punto de vista físico, para almacenar objetos complejos. Evita el acceso a los datos si esto no se realiza mediante los métodos almacenados en ella.

Acceso concurrente multiusuario: Es cuando varios usuarios accesan a la misma fuente de información al mismo tiempo.

AEO: Análisis de la Estructura de Objetos.

DEO: Diseño de la Estructura de Objetos.

ACO: Análisis del Comportamiento de Objetos.

DCO: Diseño del Comportamiento de Objetos.

OO: Orientado a Objetos.

Diagrama de rejas o de transición de estados: Diagrama que muestre el ciclo vital de un objeto, dicho diagrama además de mostrar los estados posibles de los objetos, también muestra los cambios de estado permisibles.

Diagramas de Red: Diagrama que se utiliza para representar los procesos que requieren la interacción de varios objetos, así también muestra como los distintos tipos de objeto cambian de estado y pueden solicitar a otros objetos que cambien su estado en el proceso. Para completar este tipo de diagramas se puede incluir en ellos a las operaciones de cada tipo de objeto.

LPOO: Lenguajes de programación orientada a objetos.

DFO: Diagrama de flujo de objetos.

DFD: Diagramas de flujo de datos.

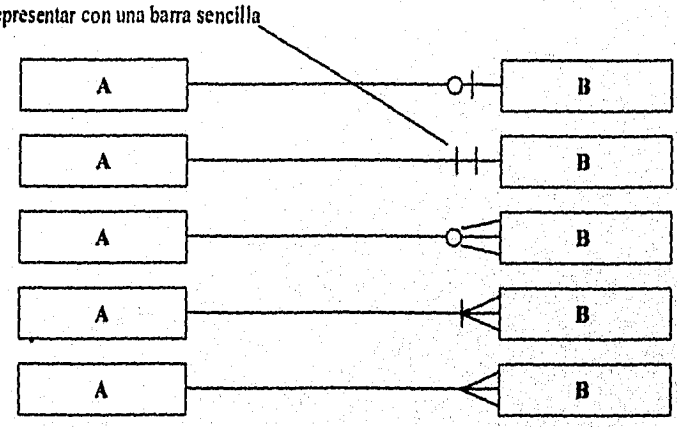
Apéndice A

Diagramas Convencionales

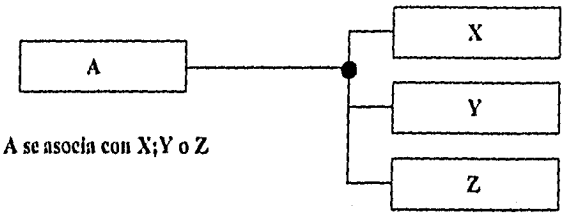
DATOS
Los cuadros con esquinas rectangulares representan datos (tipos de entes, subtipos de entes, registros, conjuntos de datos

INTERSECCION DE DATOS
Un registro de intersección se utiliza para resolver relaciones muchos a muchos.

CARDINALIDAD
Esto se suele representar con una barra sencilla



EXCLUSIVIDAD MUTUA
Sólo se considera una y sólo una rama:



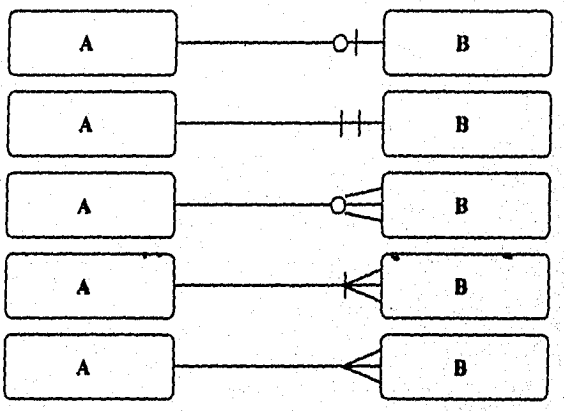
A se asocia con X;Y o Z

ACTIVIDADES
Los cuadros con esquinas redondeadas representan actividades (funciones, procesos, procedimientos, módulos del programa

FLUJO O SECUENCIA

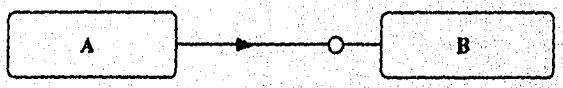
Esto se representa a menudo como una barra sencilla

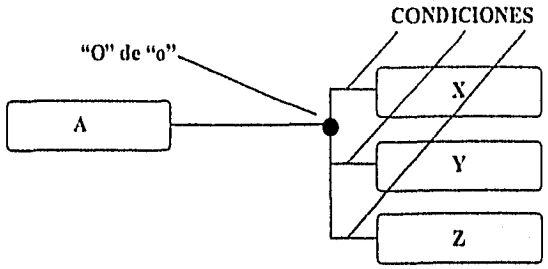
CARDINALIDAD
A se asocia con los siguientes de B:



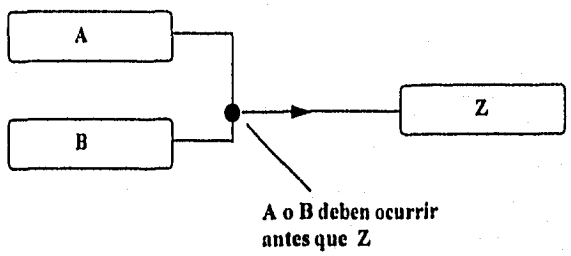
Mínimo	Máximo
0	1
1	1
0	Más de 0
1	Más de 1
Más de 1	Más de 1

Nota: La barra "1" se omite por lo general en los diagramas de actividad.





Las condiciones se asocian con enlaces en los que la cardinalidad puede ser cero y enlaces que se excluyen mutuamente.



A o B deben ocurrir antes que Z

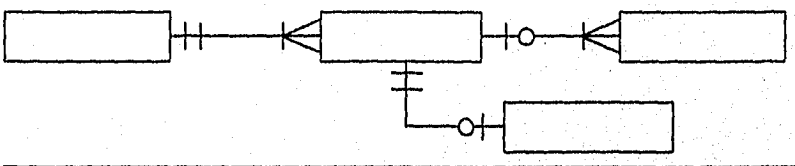
Apéndice B

Diagramas Orientados a Objetos

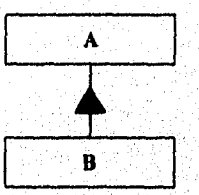
TIPOS DE OBJETOS O CLASES
(y Subtipos)

TIPOS DE OBJETOS O CLASES
EXTERNOS

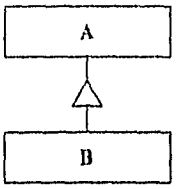
RELACIONES DE OBJETOS O CLASES



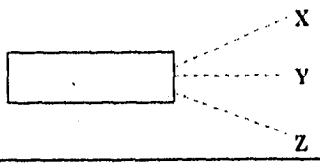
GENERALIZACION
(y Herencia)
B es un subtipo de A



COMPOSICION
(Un tipo particular de relación)
B es un subtipo de A

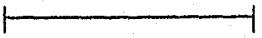


INSTANCIAS
X,Y y Z son objetos de tipo A

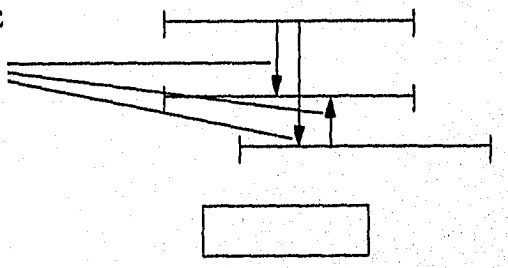


COMUNICACION ENTRE LAS CLASES

ESTADOS
(y Subestados)



TRANSICIONES ENTRE LOS ESTADOS
(Transiciones de estado

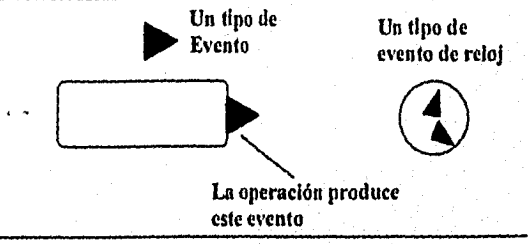


ACTIVIDADES
(Operaciones, procesos, procedimientos, métodos)

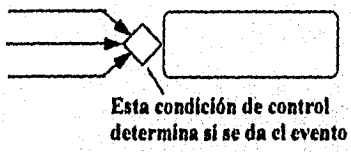
ACTIVIDADES EXTERNAS
(Externas al sistema que se diagrama

SECUENCIAS DE ACTIVIDADES

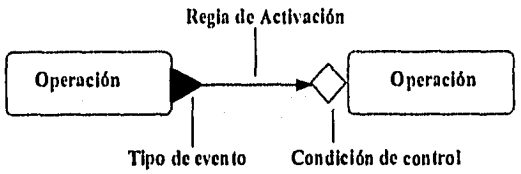
TIPOS DE EVENTOS



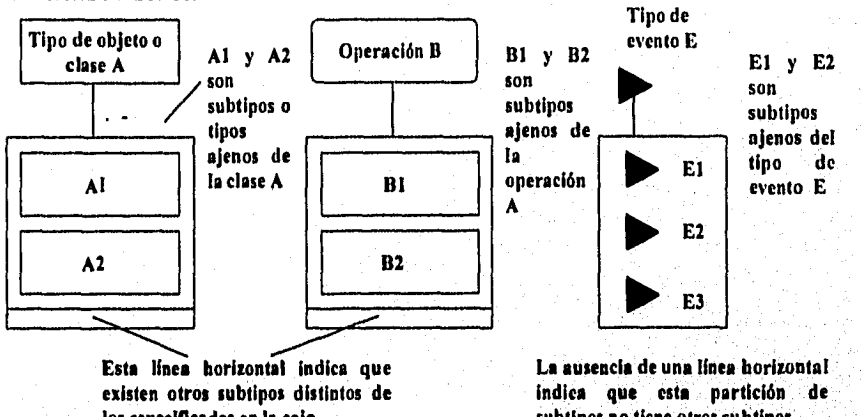
CONDICIONES DE CONTROL



REGLAS DE ACTIVACION



CAJAS DE SUBTIPO



BIBLIOGRAFÍA

-Begoña Albizuri Romero Miren

Estructura de Datos e Introducción a Bases de Datos

Ed. Limusa, Primera Edición 1989, No. Págs. 197.

-Flores Iván, Traducción: Sr. Edmundo Said

Arquitectura de Bases de Datos

Ed. El Ateneo, 1986, No. Págs. 366.

-F. Korth Henry, Silberschatz Abraham

Fundamentos de Bases de Datos

Ed. Mc. Graw Hill, 2a. Edición, No. Págs. 739.

-Lucas Gómez Angel

Diseño y Gestión de Sistemas de Bases de Datos

Ed. Paraninfo, 1993, No. Págs. 472.

-L. Winblad Ann, D. Edwards Samuel, R. King David

Software Orientado a Objetos

Ed. Addison-Wesley Iberoamericana, S.A., 1993, No. Págs. 314.

-Martin James, J. Odell James

Análisis y Diseño Orientado a Objetos

Ed. Prentice Hall, Impreso en México 1992, No. Págs. 546.

-Martin James

Organización de las Bases de Datos

Ed. Prentice Hall, impreso en México 1992, No. Págs. 543

- Norton, Peter

Toda la PC

Ed. Prentice Hall, impreso en México 1994, No. Págs. 609

- Paradox 5.0 para Windows

Sánchez Navarro José Daniel, Flores Gutiérrez Mariano

Ed. Mc. Graw Hill, Impreso en México, 1995, No. Págs. 321.

- Wiederhold, Gio

Diseño de Bases de Datos

Ed. Mc. Graw Hill, 2a. Edición, No. Págs. 921