

Tema 1

MODELO CONCEPTUAL DE DATOS.
ENTIDADES, ATRIBUTOS Y RELACIONES.
REGLAS DE MODELIZACIÓN.
DIAGRAMAS DE FLUJO DE DATOS.
REGLAS DE CONSTRUCCIÓN.
DESCOMPOSICIÓN EN NIVELES.
FLUJOGRAMAS.

© Centro de Estudios Adams
www.adams.es

ISBN: 978-84-9084-994-1



La reproducción total o parcial de esta obra por cualquier medio, existente o de próxima invención, sin permiso por escrito de los propietarios del © será perseguido de acuerdo con la legislación vigente

Anagrama «LUCHA CONTRA LA PIRATERÍA»
propiedad de Unión Internacional de Escritores.

Guion-resumen

- | | |
|--|---|
| <ul style="list-style-type: none">1. Modelo conceptual de datos2. Entidades, atributos y relaciones<ul style="list-style-type: none">2.1. Concepto de entidad2.2. Concepto de relación2.3. Concepto de atributo3. Análisis entidad/relación (reglas de modelización)4. Diagramas de flujo de datos: reglas de construcción<ul style="list-style-type: none">4.1. Organigramas4.2. Ordinogramas4.3. Pseudocódigo4.4. Paso de pseudocódigo a diagrama de flujo y viceversa | <ul style="list-style-type: none">5. Descomposición en niveles. Flujogramas<ul style="list-style-type: none">5.1. Diagramas de Flujo de Datos (DFD)5.2. Modelos de datos5.3. Diagramas de datos (DED)6. Conclusiones |
|--|---|



1. Modelo conceptual de datos

Un modelo de datos es una representación gráfica orientada a la obtención de las estructuras de datos de una forma metódica y a la vez sencilla. El modelo se suele representar con el modelo entidad/relación de Chen. Este modelo percibe el mundo real como una serie de objetos que se relaciona entre sí y pretende representarlos gráficamente mediante un mecanismo de abstracción basado en símbolos, reglas y métodos.

El diseño conseguido no es el nexo de unión entre el mundo del usuario (nivel externo) y el mundo del ordenador (nivel interno), solo es una representación de las propiedades lógicas de los datos y por tanto, dicha información no es accesible directamente por el Sistema de Gestión de Bases de Datos (SGBD). Es un método de representación abstracta del mundo real y, por lo tanto, no es directamente traducible a un SGBD, sino que necesita una traducción al modelo relacional de dicho SGBD.

En esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este esquema se le denomina **esquema conceptual**. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos de la empresa: encuentran entidades, atributos y relaciones. El objetivo es comprender:

- La perspectiva que cada usuario tiene de los datos.
- La naturaleza de los datos, independientemente de su representación física.
- El uso de los datos a través de las áreas de aplicación.

El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que esta maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema. La más popular es la notación del modelo entidad-relación, que se describirá en el capítulo dedicado al diseño conceptual.

El esquema conceptual se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario. El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el SGBD que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física. Durante todo el proceso de desarrollo del esquema conceptual este se prueba y se valida con los requisitos de los usuarios. El esquema conceptual es una fuente de información para el diseño lógico de la base de datos.

2. Entidades, atributos y relaciones

El modelo entidad/relación tiene sus estructura propias que son conocidas con el nombre de diagramas entidad/relación. Los elementos que componen dicho modelo son las entidades, los atributos y las relaciones.

2.1. Concepto de entidad

Una entidad es cualquier tipo de objeto (físico) o concepto del cual se pueda extraer información. Las entidades se representan mediante rectángulos.



Por ejemplo un bote de bebida es un objeto del que emana gran cantidad de información: la marca, el sabor, la capacidad, los ingredientes, la caducidad, etc. A este paso se le denomina **abstracción**; después, toda esta información hay que organizarla; ¿cómo?, en tablas, por ejemplo. Una entidad solo aparece una vez en el modelo conceptual. Se denomina **ocurrencia de entidad** a la implementación concreta de una entidad.

2.2. Concepto de relación

Es la asociación entre dos o más entidades. Se representan gráficamente entre rombos con el nombre dentro. Al número de participantes se le llama grado de la relación (binaria, ternaria, etc).

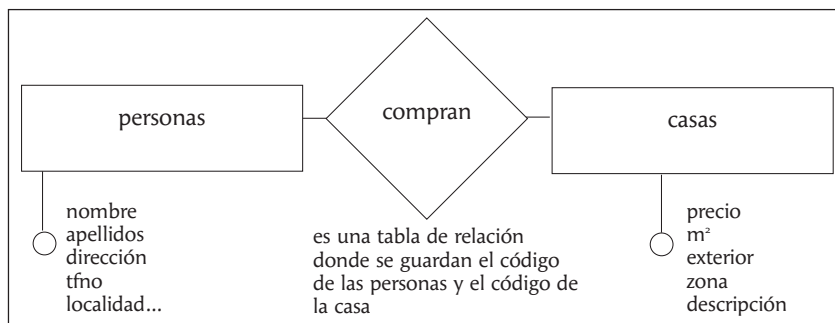
2.3. Concepto de atributo

Es una característica de interés de la entidad, por ejemplo la marca, la capacidad o el sabor.

Cada atributo tiene un conjunto de valores asociados que se les denomina **dominio** y se encarga de definir todos los valores posibles que puede tomar un atributo. Los atributos pueden ser simples o compuestos. Se dice que un atributo es simple cuando este solo tiene un componente, por lo cual no se puede dividir en partes más pequeñas. El atributo es compuesto cuando este consta de componentes que a su vez se pueden dividir en partes más pequeñas. Se representa con una elipse.

En función de sus características respecto a la entidad se distinguen dos tipos de atributos:

- **Atributo descriptor.** Caracteriza una ocurrencia pero no la distingue del resto. Se representa por una elipse o círculo sin relleno.
- **La clave primaria o identificador.** Conjunto de atributos pertenecientes a la misma entidad y que hacen único el acceso a cada ocurrencia de la entidad. Se representa mediante un círculo relleno. Es posible pensar en la existencia de varias claves sobre la misma entidad. Al conjunto de todas ellas se las denomina claves candidatas. Solo una de ellas conformará la clave primaria. Por el hecho de estar formada por un solo atributo o varios se llamará clave simple o múltiple (compuesta).



3. Análisis entidad/relación (reglas de modelización)

Es el modelo más utilizado para el diseño conceptual de bases de datos. El análisis entidad-relación (E/R) abstrae las tablas en forma de objetos y enlaza dichos objetos mediante punteros de relación de la misma forma que enlazaríamos objetos si estuvieran en la memoria de nuestro PC.

Veamos un ejemplo: nos plantean un problema que consiste en crear una base de datos para llevar el control de una gestoría de inmuebles. Necesitaremos como mínimo dos entidades, una donde quede plasmada la información de las personas (clientes) y otra donde quede plasmada la información de las viviendas (casas). En un principio son dos entidades diferentes pero si se pretende acceder a la información para visualizar las casas vendidas y sus compradores, no queda más remedio que relacionarlas. Una acción que vincula a las personas con las viviendas es, por ejemplo, que estas son compradas por parte de las anteriores. El análisis entidad/relación quedaría así:

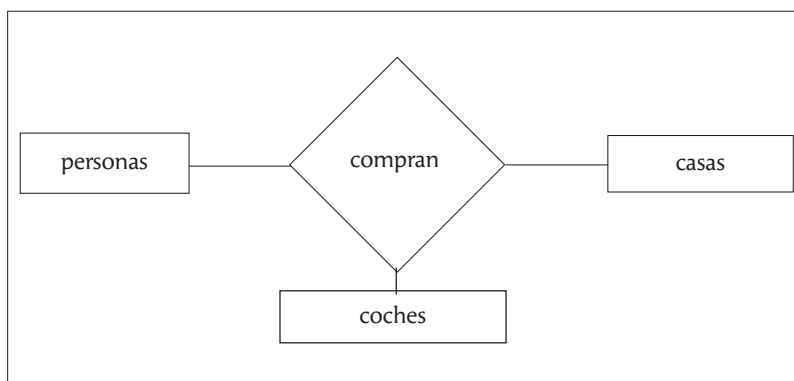
Ahora lo abstraemos a sus respectivas tablas:

personas				casas			
idpersona	nombre	direcc	Tfno	idpersona	decripción	metros	Zona
1	Juan	C/Pez nº 20	917885522	1	Chalet	250	La Moraleja
2	María	C/Córdoba nº 60	913225577	2	Piso	90	Pinar Chamartín
3	Pedro	Avda Jucal nº 12	917184051	3	Caserón	300	El clavín

Idpersona	Idcasa
1	1
2	1
3	2
3	3

tabla de referencia (ventas)

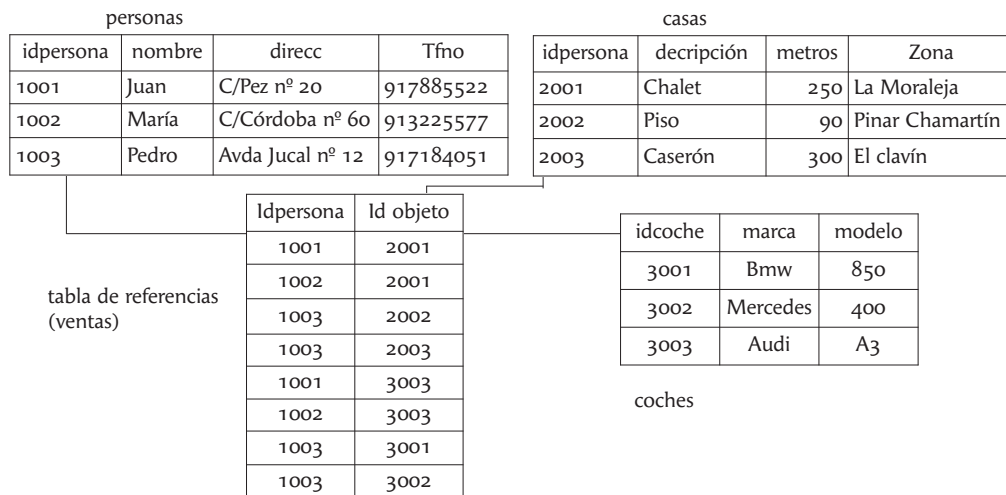
Juan y María comparten chalé. Pedro ha comprado el caserón y el piso. Ahora vamos a añadir otra entidad.



A) Identificador único

Los identificadores de casa y persona están duplicados. Esto puede generar problemas cuando queramos expresar relaciones con la tabla de coches. Para solucionar esto necesitamos dos cosas:

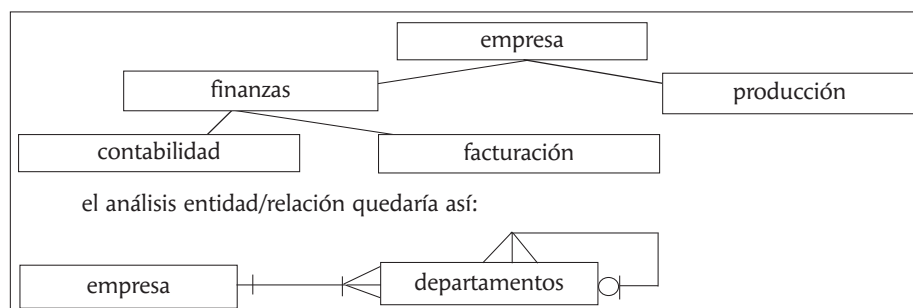
1. Identificador único: nunca debe haber dos identificadores iguales para objetos (instancias) del mismo o diferente tipo.
2. Que en el identificador vaya implícita la información sobre el tipo de objeto: los identificadores de personas empiezan por 1000, los de casas por 2000 y los de coches por 3000.



De este modo podemos tener tablas (tipos) que almacenan diferentes objetos (instancias) de diferente tipo.

B) Relaciones recursivas

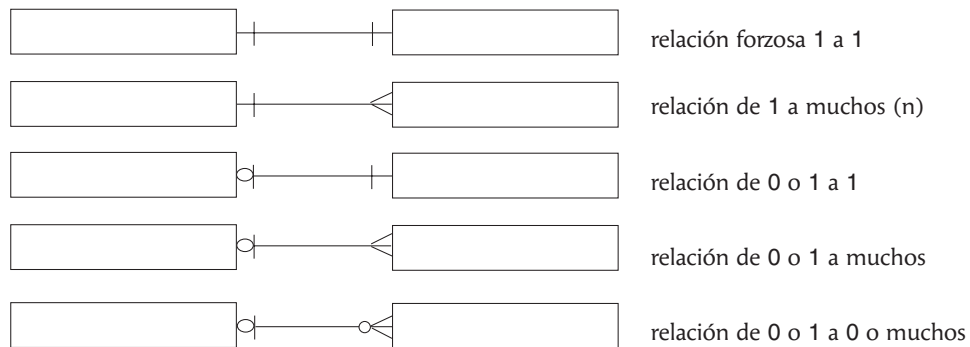
La relación recursiva es la que relaciona objetos consigo mismo. Un claro ejemplo es el de una empresa que tiene varios departamentos y estos, a su vez, tienen subdepartamentos.



En este caso no hay una tabla de relación, pues una empresa puede tener varios departamentos, pero un departamento no tiene varias empresas.

C) Simbología de cardinalidad

La cardinalidad es la proporción de relaciones entre objetos.



Una empresa puede tener varios departamentos (1 a n).

Un departamento puede tener o no varios subdepartamentos (1 o 0 a n) y así con el resto de las combinaciones.

No todos los analistas y programadores utilizan la misma simbología, aunque esta es la más utilizada.

El modelo entidad/relación ha dado origen al **modelo relacional**, que es el de mayor aceptación en la actualidad.

D) Metodología de diseño conceptual

Como hemos comentado anteriormente, es el primer paso para crear la base de datos. Este esquema se construye mediante la información que nos da el usuario, examinando programas, formularios e informes ya existentes que se utilizan en la empresa y siguiendo el flujo de información con la que trabajan esos usuarios. A estas visiones de la información se las llama vistas. Puede haber varias vistas recogidas de varios usuarios, por ejemplo de distintos departamentos, por lo cual se suelen crear varios esquemas conceptuales, llamados locales. Cada uno de estos esquemas se compone de entidades, relaciones, atributos, dominios de atributos e identificadores.

Los pasos que se deben seguir para la creación de estos esquemas son:

1. **Identificar las entidades:** se buscan objetos existentes, como los empleados, clientes, proveedores, etc.
2. **Identificar las relaciones:** una vez identificadas las entidades se debe buscar la forma de relacionarlas y qué tipo de cardinalidad surge.



3. **Identificar los atributos y asociarlos a entidades y relaciones:** se buscan nombres para plasmar la información, por ejemplo: idempleado, nombre, apellidos, o mejor, apellido1 y apellido2, dirección, etc. La mejor técnica para encontrar los atributos es hacerse la pregunta “si necesito visualizar... ¿qué atributo necesito?”. Así, por ejemplo, si queremos un listado de las empleadas, necesitaremos un atributo, por ejemplo, mujer. Si necesito un listado de los empleados más veteranos, necesitaremos un atributo fecha_incorporación, etc. De cada atributo se debe ir anotando el nombre que se le asignará, el tamaño, el tipo de dato que albergará, si se permiten valores nulos, etc.
4. **Determinar los dominios de los atributos:** valores que puede tomar ese atributo, su tamaño, formato y tipos de operaciones que se pueden realizar sobre ellos.
5. **Determinar los identificadores:** cada entidad por lo menos ha de tener un identificador o clave. A las entidades que no tienen identificador se las denomina débiles.
6. **Determinar las jerarquías de generalización** (si las hay): en este paso se deben analizar las entidades existentes, pues pueden dar origen a otras entidades o subentidades.
7. **Dibujar el diagrama entidad-relación:** cuando tengamos identificados los conceptos anteriores se procede a dibujar la estructura de la base de datos mediante un diagrama.
8. **Revisar el esquema conceptual local con el usuario:** presentarle el esquema al usuario (o cliente) para revisarlo y comentar los posibles cambios.

4. Diagramas de flujo de datos: reglas de construcción

Para el diseño de algoritmos se utilizan técnicas de representación. Una de estas técnicas son los denominados **diagramas de flujo**, que se definen como la representación gráfica que, mediante el uso de símbolos estándar conectados o unidos mediante líneas de flujo, muestran la secuencia lógica de las operaciones o acciones que debe realizar un ordenador, así como la corriente o flujo de datos en la resolución de un programa.

Los diseños deben de ser **normalizados** para facilitar el intercambio de documentación entre el personal informático (analistas y programadores). Para ello existen normas en las que basarse, dictadas por distintas organizaciones, como la ISO (*International Standard Organization*), ANSI (*American National Standard Institute*), etc.

Los diagramas de flujo se pueden clasificar en dos grandes grupos:

- a) Organigramas.
- b) Ordinogramas.

Una de las principales diferencias entre ambos es que pertenecen a distintas fases o etapas de la resolución de un programa. Mientras que los organigramas corresponden a la fase de análisis, los ordinogramas corresponden a la fase de diseño.



4.1. Organigramas

También denominados diagramas de flujo de sistemas o **diagramas de flujo de configuración**. Son representaciones gráficas del flujo de datos e información entre los periféricos o soportes físicos (de entrada/salida) que maneja un programa.

Todo organigrama debe reflejar:

- Las distintas áreas o programas en los que se divide la solución del problema, así como el nombre de cada uno de ellos.
- Las entradas y salidas de cada área, indicando los soportes que serán utilizados para el almacenamiento, tanto de los datos pendientes de elaborar o procesar, como de los resultados obtenidos.
- El flujo de los datos.

Todo ello debe proporcionar:


- Una visión global de la solución del problema.
- Una fácil realización de futuras correcciones.
- Un control de todas las posibles soluciones.

Los organigramas deben respetar las siguientes reglas de representación:











- En la parte central del diseño debe figurar el símbolo de proceso.
- En la parte superior del diseño, y siempre por encima del símbolo de proceso, deben figurar los soportes de entrada.
- En la parte inferior del diseño, y siempre por debajo del símbolo de proceso, deben figurar los soportes de salida.
- A izquierda y derecha del diseño y, por tanto, a ambos lados del símbolo de proceso, figurarán los soportes que son tanto de entrada como de salida.

La simbología que se utiliza es la siguiente:






A) Símbolos de soporte de información

SÍMBOLO	DENOMINACIÓN	TIPO DE DISPOSITIVO
	Teclado	Entrada
	Soporte magnético	Entrada






SÍMBOLO	DENOMINACIÓN	TIPO DE DISPOSITIVO
	Pantalla/CRT	Salida
	Impresora	Salida
	Tarjeta perforada	Entrada/Salida
	Cinta de papel	Entrada/Salida
	Disco magnético	Entrada/Salida
	Disco magnético	Entrada/Salida
	Cinta magnética	Entrada/Salida
	Cinta encapsulada	Entrada/Salida
	Disco flexible	Entrada/Salida
	Tambor magnético	Entrada/Salida

B) Símbolos de proceso

SÍMBOLO	FUNCIÓN
	Proceso u operación.
	Clasificación u ordenación de datos en un fichero.
	Fusión o mezcla de dos o más ficheros en uno solo.
	Partición o extracción de datos de un fichero.
	Manipulación de uno o varios ficheros (<i>intercalación</i>).

C) Líneas de flujo de datos

SÍMBOLO	FUNCIÓN
	Dirección del proceso o flujo de datos.
	Líneas de teleproceso (<i>transmisión de datos</i>).
	Línea conectora. Permite la unión entre unidades o elementos de información.

4.2. Ordinogramas

También denominados **diagramas de flujo de programas**. Son representaciones gráficas que muestran la secuencia lógica y detallada de las operaciones que se van a realizar en la ejecución de un programa.

Se puede decir que los diseños resultantes, por estética, deben guardar cierto equilibrio y simetría, facilitando así, en la medida en la que sea posible, su entendimiento y comprensión, procurando limitar al máximo el uso de comentarios aclaratorios.

El diseño de todo ordinograma debe reflejar:

- Un principio o inicio que marca el comienzo de ejecución del programa y que viene determinado por la palabra "INICIO".
- La secuencia de operaciones, lo más detallada posible y siguiendo siempre el orden en el que se deberán ejecutar (arriba-abajo e izquierda-derecha).
- Un fin que marca la finalización de ejecución del programa y que viene determinado por la palabra "FIN".

Las reglas que hay que seguir para la confección de un ordinograma son las siguientes:


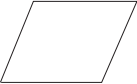


- Todos los símbolos utilizados en el diseño deben estar conectados por medio de líneas de conexión o líneas de flujo de datos.
- Queda terminantemente prohibido el cruce de líneas de conexión, pues ello indica que el ordinograma no está correctamente diseñado.
- A un símbolo de proceso pueden llegarle varias líneas de conexión o flujo, pero de él solo puede salir una.
- A un símbolo de decisión pueden llegarle varias líneas de conexión o flujo de datos, pero de él solo puede salir una línea de cada una de las dos posibilidades existentes (verdadero o falso).



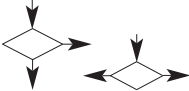
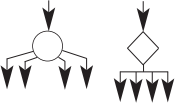

- e) A un símbolo de inicio de proceso no llega ninguna línea de conexión o flujo y de él solo puede partir una línea de conexión.
- f) A un símbolo de final de proceso o ejecución de programa pueden llegar muchas líneas de conexión pero de él no puede partir ninguna.

La simbología utilizada es la siguiente:

A) Símbolos de operación o proceso

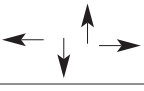

SÍMBOLO	FUNCIÓN
	Terminal (marca el inicio, final o una parada necesaria realizada en la ejecución del programa).
	Operación de E/S en general (utilizada para mostrar la introducción de datos desde un periférico a la memoria del ordenador y la salida de resultados desde la memoria del ordenador a un periférico).
	Proceso u operación en general (utilizado para mostrar cualquier tipo de operación durante el proceso de elaboración de los datos depositados en la memoria).
	Subprograma o subrutina (utilizado para realizar una llamada a un subprograma o proceso, es decir, un módulo independiente cuyo objetivo es realizar una tarea y devolver el control de ejecución del programa al módulo principal).

B) Símbolos de decisión



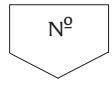
SÍMBOLO	FUNCIÓN
	Decisión de dos salidas (indica operaciones lógicas o comparativas seleccionando en función del resultado entre dos caminos alternativos que se pueden seguir).
	Decisión múltiple con "n" salidas (indica el camino que se puede seguir entre varias posibilidades según el resultado de la operación lógica o comparación establecida).
	Bucle definido, empleado para modificar una instrucción del bloque de instrucciones que a su vez producen una alteración o modificación en el comportamiento del programa.




C) Líneas de flujo

SÍMBOLO	FUNCIÓN
	Flechas indicadoras de la dirección del flujo de datos.
	Línea conectora, también llamada línea de flujo de datos (permite la conexión entre los diferentes símbolos utilizados en el diseño).

D) Símbolos de conexión

SÍMBOLO	FUNCIÓN
	Conector (este símbolo es utilizado para el reagrupamiento de líneas de flujo).
	Conector de líneas de flujo en la misma página (utilizado para enlazar dos partes cualesquiera del diseño a través de un conector de salida y un conector de entrada).
	Conector de líneas de flujo en distintas páginas (utilizado para enlazar dos partes cualesquiera del diseño a través de un conector de salida y un conector de entrada).

E) Símbolos de comentarios

SÍMBOLO	FUNCIÓN
	Permite escribir comentarios a lo largo del diseño realizado.

4.3. Pseudocódigo

Para evitar el exceso de espacio de los métodos anteriores y hacer una primera visión del desarrollo de un programa, nació el pseudocódigo, que consiste en una **técnica para expresar mediante lenguaje natural (sujeta a unas determinadas normas) el lógico desarrollo de un programa, es decir, su flujo de control.**

Cuenta con la ventaja de poder ser desarrollado con mayor facilidad y en poco tiempo, y luego sirve de soporte a la programación real al ser utilizado como base para la codificación del algoritmo en el lenguaje de programación que más nos interese.



El pseudocódigo ha de ser considerado como una herramienta para el diseño de programas y no como una notación para la descripción de los mismos. Gracias a su flexibilidad permite obtener la solución de un problema mediante aproximaciones sucesivas, es decir, mediante el denominado diseño descendente.

Todo pseudocódigo debe permitir la descripción de:

- Las instrucciones de Entrada/Salida.
- Las instrucciones de proceso.
- Las sentencias de control de flujo de ejecución.
- Acciones compuestas que hay que refinar posteriormente.
- Cualquier proceso relacionado con los datos:
 - Describir datos.
 - Definir tipos de datos.
 - Definir y usar constantes y variables.
 - Archivos.
 - Objetos.

4.3.1. Acciones simples

También denominadas instrucciones primitivas, son aquellas que el procesador ejecuta de forma inmediata:

- Asignación: `variable_valor`.
- Entrada: leer variable.
- Salida: escribir expresión.

4.3.2. Sentencias de control

También denominadas sentencias estructuradas, son aquellas que controlan el flujo de ejecución de otras instrucciones.

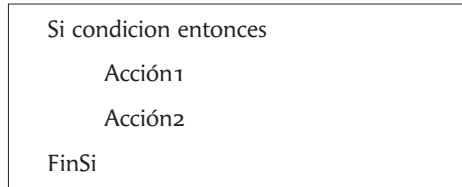
- **Secuencial.** Se ejecutan en el mismo orden en el que aparecen escritas.

```
Leer nota
suma=suma+nota
media=suma/5
escribir media
```

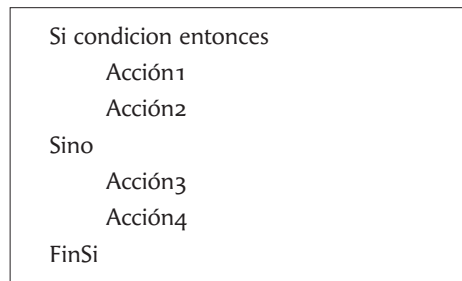


- **Alternativa**

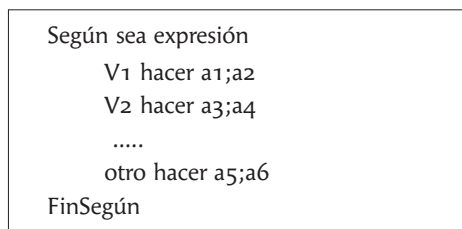
- Simple



- Doble

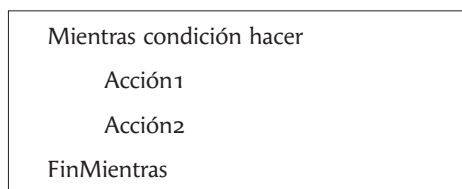


- Múltiple

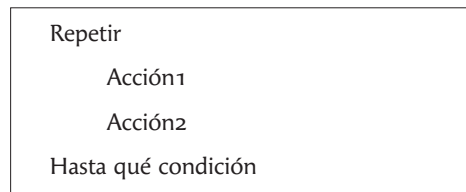


- **Repetitiva**

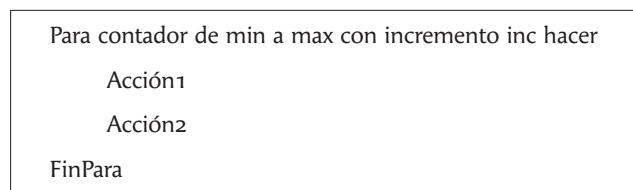
- Mientras



— Repetir



— Para



4.3.3. Acciones compuestas

Una acción compuesta es una acción que ha de ser realizada dentro del algoritmo, pero que aún no está resuelta en términos de acciones simples y sentencias de control. Se denominan subprogramas.

4.3.4 Comentarios

Los comentarios son líneas aclarativas cuyo fin es el de aclarar la comprensión del programa. Estas líneas son ignoradas a la hora de ejecutar el programa. Se escribirán en cualquier línea a partir de un símbolo que los identifique, como por ejemplo **.

** Comentario aclarativo de cualquier aspecto del programa
--

4.3.5. Datos del programa

La finalidad primordial de un programa es la de realizar cálculos con una serie de datos. La información, ya sea de entrada o generada como salida, debe guardarse en algún lugar. Es necesario saber previamente cuál será su cantidad, sus nombres y sus tipos, con lo que podré limitar el conjunto de operaciones que se podrán realizar con ellos a lo largo del programa. Este conjunto de datos se denomina **entorno**.



Entorno:
 I es numérica entera
 Euros es numérica real
 Apellidos es alfanumérica

4.3.6. Programa

El programa es la solución final de un problema. Consiste en la unión del entorno y del algoritmo precedido cada uno de una etiqueta.

Programa Nombre_Programa
 Entorno:
 ** Descripción de los datos
 ...
 Algoritmo:
 ** Descripción de las acciones
 ...
 FinPrograma

Idéntico para los programas

SubPrograma Nombre_Programa
 Entorno:
 ** Descripción de los datos
 ...
 Algoritmo:
 ** Descripción de las acciones
 ...
 FinSubPrograma



4.4. Paso de pseudocódigo a diagrama de flujo y viceversa

- PSEUDOCÓDIGO

ORDINOGRAMA

Variable=Expresión

Variable_Expresión

Leer Variable

Leer variable

Escribir Expresión

Escribir expresión

Acción 1

Acción1

Acción 2

Acción2

Acción 3

Acción3

Si condición entonces

Acción 1

Acción 2

FinSi

Condición

Acciones



Si condición entonces

Acción 1

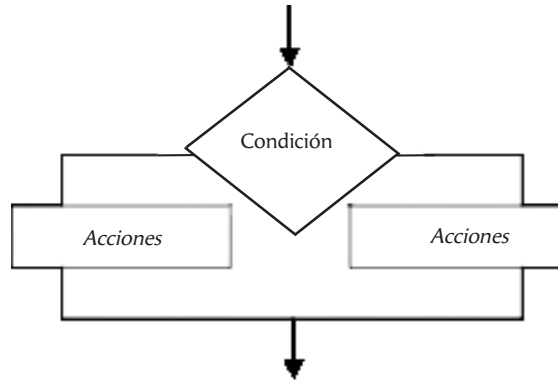
Acción 2

Sino

Acción 3

Acción 4

Finsi



Según sea expresión

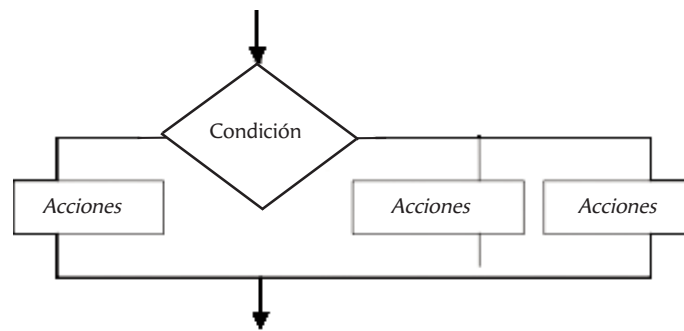
V1 hacer a1;a2

V1 hacer a3;a4

...

otro hacer a5;a6

Fin según

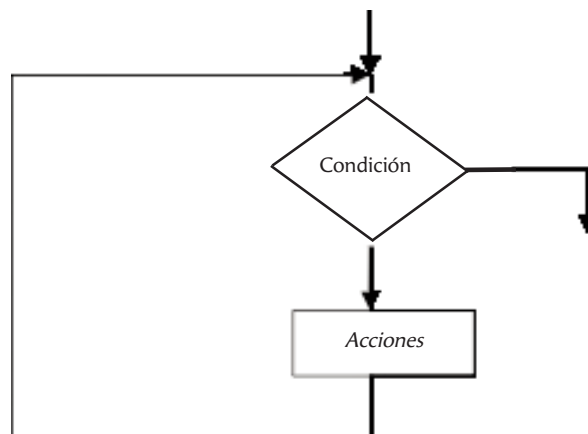


Mientras condición hacer

Acción1

Acción2

FinMientras

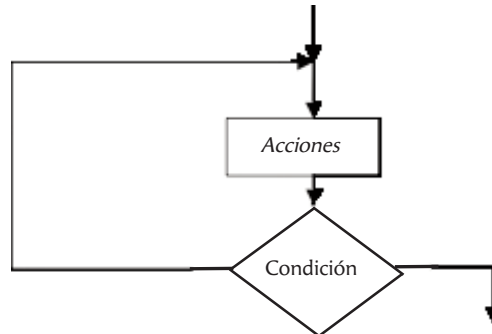


Repetir

Acción1

Acción2

Hasta que condición

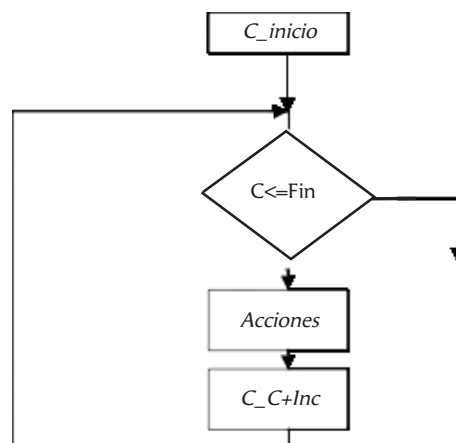


Para C de inicio a fin con
incremento inc hacer

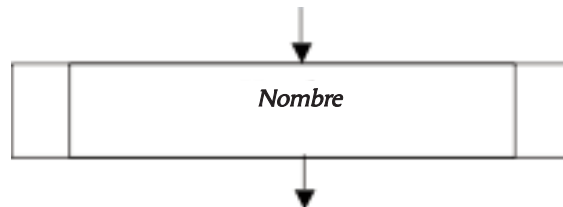
Acción1

Acción2

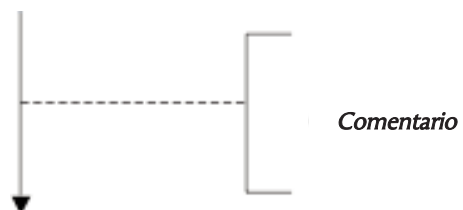
FinPara

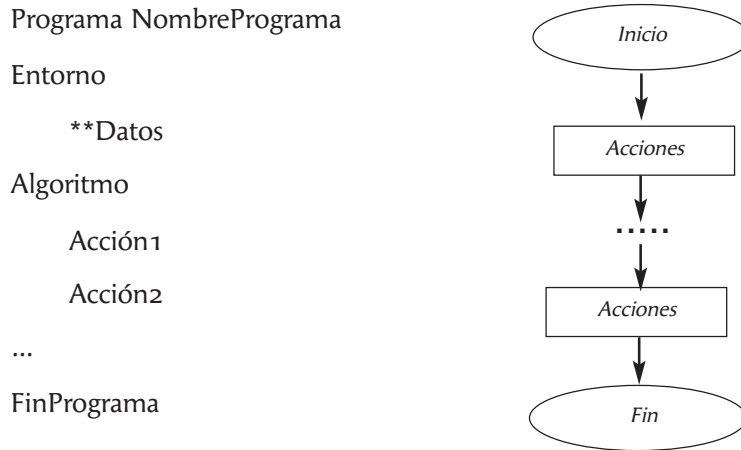


Acción compuesta



**Comentario





5. Descomposición en niveles. Flujogramas

5.1. Diagramas de Flujo de Datos (DFD)

Los Diagramas de Flujo de Datos (DFD) **se utilizan fundamentalmente en la fase de análisis para la elaboración del modelo lógico de procesos**. En un DFD se representa, definiendo los límites del sistema en estudio, el movimiento o flujo de la información a través del sistema, los procesos que transforman dicha información y los almacenamientos intermedios que son necesarios; todo ello desde un punto de vista puramente lógico, sin tener en cuenta ni representar ninguna restricción o aspecto físico ni ningún tipo de secuencia u orden de ejecución.

Los DFD se utilizan para representar los sucesivos niveles de descomposición realizados durante el análisis del sistema, comenzando con una descripción muy general del sistema denominada de nivel 0 (DFD-0) a partir de la cual se producen sucesivos diagramas con niveles de detalle cada vez mayor (DFD-1, DFD-2, etc). Así, en un DFD-0, también conocido como diagrama de contexto, figura un único proceso que representa al sistema completo y las entidades externas que interactúan con él; en el DFD-1 se representan los subsistemas; en los DFD-2 (uno por subsistema) las funciones de cada subsistema, en el DFD-3 las subfunciones asociadas a los eventos del sistema; y, finalmente, en el DFD-4, último nivel al que normalmente llegará la descomposición, los procesos necesarios para el tratamiento de cada subfunción.

Los componentes que aparecen en un DFD son:

- Las **entidades externas** (simbolizadas mediante rectángulos o elipses e identificadas por su nombre) que representan otros sistemas, organizaciones o personas externas al sistema pero que interactúan con él recibiendo o aportando información. Las entidades externas proporcionan la conexión del sistema con el mundo exterior.
- Los **procesos** (rectángulos que encierran la identificación del proceso) que representan las actividades que suponen transformación o



manipulación de datos. Un proceso no puede ser ni fuente ni sumidero de datos, por lo que un proceso siempre tendrá al menos un flujo de datos de entrada y al menos uno de salida.

- Los **almacenes de datos** (dos líneas paralelas con una identificación) que representan depósitos de información dentro del sistema, ya sean permanentes por afectar a información que el sistema debe guardar, o transitorios, por ser información que se utiliza en un proceso diferente del que la produce con el que no está sincronizado (recuérdese que no se tienen en cuenta restricciones físicas). Siempre deberá existir un proceso entre una entidad externa y un almacén de datos. Los almacenes de datos sirven de enlace del DFD con el modelo lógico de datos: cada almacén principal de un DFD representa un conjunto completo de entidades del modelo de datos (una o varias entidades), y cada entidad del modelo de datos pertenece a un único almacén principal de un DFD; esto facilitará las validaciones cruzadas entre los dos diagramas.
- Los **flujos de datos** (flechas con la identificación de la información que transportan), que representan la comunicación entre procesos, almacenes y entidades externas. Los flujos de datos portan información, no son activadores de procesos.

5.2. Modelos de datos

Los modelos de datos se usan en diferentes fases del desarrollo. Nos referiremos aquí al modelo conceptual de datos, descripción de alto nivel utilizada en la fase de planificación de sistemas, para la que se recomienda utilizar la técnica del modelo entidad relación, y al modelo lógico de datos elaborado en la fase de análisis de sistemas, para la que se recomienda la técnica del diagrama de estructura de datos (DED). En ambos casos se trata de representaciones del nivel lógico de los datos, válidas para describir las necesidades de información mediante estructuras no redundantes, sin inconsistencias, seguras e íntegras, y desprovistas de todo tipo de condicionantes, como pudieran ser los impuestos por los procesos que deba sufrir, o de tipo físico, como almacenamientos, etc.

Será en la fase de diseño del sistema cuando el modelo lógico de datos servirá de base para especificar las estructuras de datos físicas (esquema interno) que deben implantarse para el almacenamiento de datos, normalmente sobre un SGBDR (Sistema Gestor de Bases de Datos Relacional) y las visiones específicas de los datos (esquema externo) precisadas por los diferentes componentes o programas del sistema.

Las dos técnicas son muy similares en cuanto a los elementos que las constituyen, por lo que su descripción se centrará en la del modelo entidad-relación, señalándose seguidamente las diferencias para el diagrama de estructura de datos.

En un **modelo entidad-relación (MER)** aparecen los elementos siguientes:

- Las **entidades**, simbolizadas con cajas rectangulares e identificadas por un nombre, que representan objetos o conceptos del mundo acerca de los cuales el sistema precisa manejar información. Dicha



información se modeliza como atributos de dichas entidades. Por ejemplo, un sistema de gestión de pedidos representará la entidad CLIENTE con, entre otros atributos, el nombre del cliente, la dirección, teléfono, etc.; la entidad PEDIDO, con atributos número de pedido, estado del mismo; la entidad PRODUCTO, serie, precio, etc.

- Las **relaciones entre entidades**, simbolizadas por rombos sobre las líneas que enlazan las entidades relacionadas e identificadas por un nombre, que representan las interrelaciones existentes entre entidades. Las relaciones se caracterizan por su cardinalidad (1:1, 1:N o N:M), según sean una o varias las ocurrencias de las entidades de cada lado que participan en las instancias de la relación (así un pedido puede consistir de varios [M] productos y cualquier producto puede figurar en pedidos diferentes [N]). Las relaciones pueden ser obligatorias (cuando exigen al menos una ocurrencia de cada una de las entidades que participan), opcionales (cuando la ocurrencia de la entidad opcional no es necesaria) o exclusiva (cuando la ocurrencia de una de las relaciones de la entidad implica que no tiene lugar la ocurrencia de otras posibles relaciones con otras entidades).

5.3. Diagramas de datos (DED)

Las diferencias fundamentales en el diagrama de estructura de datos son:

- En el DED solo se admiten relaciones de cardinalidad 1:N. Las relaciones 1:1 dan lugar a una única entidad. Las relaciones N:M se representan definiendo una entidad adicional sin correspondencia con un objeto real que sirve de enlace con las dos entidades originales mediante relaciones (1:N) y (1:M) respectivamente.
- Las relaciones son de tipo binario, esto es, solo entre dos entidades, mientras que en el modelo entidad relación pueden existir relaciones entre más de dos entidades, razón por la cual su riqueza descriptiva es mayor y se prefiere este modelo para representar modelos del nivel conceptual.

6. Conclusiones

En línea con lo expresado en los apartados anteriores, y con objeto de dotar a las diferentes Unidades de Tecnología de la Información de la Administración de un entorno que facilite la construcción de sus sistemas de información siguiendo prácticas metodológicas, el Consejo Superior de Informática y para el impulso de la Administración Electrónica, en el desarrollo de su línea estratégica de mejora de la calidad y productividad en el desarrollo de software, promovió la elaboración de una metodología para el desarrollo de sistemas de información para su uso en proyectos informáticos de las Administraciones Públicas. Fruto de ello es la metodología MÉTRICA, actualmente en su versión 3, conocida como MÉTRICA 3.

La metodología esta constituida por fases. Cada una de estas fases, a su vez, se estructura en módulos de contenido homogéneo para los que se des-



criben las actividades y tareas a realizar, así como los productos a obtener y una recomendación sobre la posible o posibles técnicas a utilizar en cada punto. La identificación de los productos a obtener en cada momento facilita la introducción de hitos en el proyecto de desarrollo, elemento imprescindible para la planificación y el seguimiento y control de la ejecución del proyecto. Por otro lado, estos productos permiten enlazar con las actividades de garantía de calidad previstas en el Plan General de Garantía de Calidad que se describen más adelante en este tema.

Aún siendo altamente formal en su planteamiento, MÉTRICA 3 es una metodología que pretende tener un carácter flexible en su adaptación a una amplia variedad de proyectos de desarrollo de sistemas de información, debiéndose en cada caso, en función de las características específicas de cada proyecto, adoptar el modelo de ciclo de vida que es más apropiado para efectuar el desarrollo y, a partir de esta elección, decidir qué actividades han de efectuarse, qué productos obtenerse, etc, con qué énfasis y en qué secuencia. La utilización de herramientas CASE que soporten, de una manera completa, las técnicas propuestas en la metodología, permitirá optimizar considerablemente el esfuerzo de desarrollo.

En la siguiente web puedes consultar:
http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3#.V-tytf3_qrR.

