UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ciencias

Bases de Datos y su Aplicación en el Diseño de un Sistema Escolar Utilizando dBase III

T E S I S

Que para Obtener el Título de:

ACTUARIO

PRESENTA

María del Pilar Otaola Esquivel

MEXICO, D. F. 1 9 8 7

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

La humanidad a lo largo de la historia, se ha visto necesitada de información que le sirva de ayuda para una mejor realización de sus diferentes actividades.

Ya en el año 3500 a. C. aproximadamente, los mercaderes babilónicos conservaban información en registros impresos en tablas de arcilla.

El hombre fue ingeniando las formas de recopilar y procesar datos que le iban siendo de utilidad. Pero fue hasta el siglo XV, con la invención de la imprenta, cuando la humanidad se vio con la posibilidad de registrar, almacenar, recobrar, informar y transmitir datos de un modo más efectivo.

El procesamiento de datos e información logra dar un paso más en el siglo XX con los Medios de Comunicación Masiva. Y en este mismo siglo, se llega a lograr otro gran avance al aparecer la computadora digital.

En los últimos años ha sido impresionante observar cómo crecen en volumen e importancia las cantidades de datos que utilizan las computadoras, siendo también cada vez mayor la complejidad de su organización. Por lo que se ha visto la necesidad y utilidad de crear BASES de DATOS cuya función primordial sea dar apoyo al sistema de información y lograr un mejor manejo de los datos.

Estas BASES de DATOS se consideran ya como un recurso de vital importancia para el mejor funcionamiento de las empresas, así como indispensables para la toma de decisiones en todo tipo de organización humana.

Por esta razón, con este trabajo que presento, he querido profundizar en el tema de las BASES de DATOS para que, aprovechando los avances de la tecnología actual, pueda ofrecer un mejor servicio en el campo de la computación aplicada al ámbito escolar.

Además de una información general sobre las BASES de DATOS, presento una metodología a seguir propuesta por James Martin, para lograr un mejor MODELO de DATOS recomendable para cualquier BASE de DATOS con la que se trabaje.

En el último capítulo, propongo un sistema de calificaciones utilizando dBASE III. Dada la capacidad de este manejador de datos, quiero presentarlo como subsistema de un posible sistema escolar de mayor amplitud, en el que muchos otros subsistemas puedan agregarse.

1. CONCEPTOS GENERALES DE LAS BASES DE DATOS

1.1 ANTECEDENTES

En 1959 se organizó CODASYL (Conference on Data System Languages) como respuesta a la necesidad de crear un lenguaje común orientado a los negocios. Tal lenguaje fue llamado con el nombre de COBOL (Common Business Oriented Language).

COBOL permite definir en el mismo programa los datos y procesos necesarios para realizar la aplicación deseada. Sin embargo, la implementación de aplicaciones más complejos, acentuó la necesidad de crear métodos nuevos de almacenamiento y recuperación de datos mediante mecanismos de acceso directo o aleatorio, ya que antes sólo se contaba con cintas de acceso secuencial.

El DBTG (Data Base Task Group), organismo perteneciente a CODASYL, dedicó varios años al estudio para dar solución a esta nueva dificultad que se presentaba.

Con la aparición de dispositivos de acceso aleatorio se facilitó el manejo de los datos con estructuras diferentes a la secuencial, permitiendo ésto manejar relaciones entre archivos a los que se les llamaría Bancos o Bases de Datos. Y para 1971 se tenía ya un lenguaje para la descripción y manejo de los datos almacenados dentro de una Base de datos, diseñado de tal forma que resultaba independiente de cualquier lenguaje de programación. Sin embargo, las especificaciones del lenguaje para el manejo de datos estaban diseñadas para ser usadas como una extensión de los lenguajes de programación ya existentes.

Actualmente, el software se ha desarrollado tanto que ya hay

también múltiples lenguajes-manejadores de Bases de datos que ofrecen un "lenguaje propio" y totalmente independiente con el que se pueden realizar variadas aplicaciones sin necesidad de utilizar algún otro lenguaje de programación.

1.2. DESCRIPCION DE UNA BASE DE DATOS

Después de lo visto en el punto anterior, se puede decir, entonces, que una Base de datos es una colección de datos interrelacionados y almacenados en conjunto para obtener así una mayor integración de los mismos. Esto, a su vez, minimiza la redundancia y disminuye también los errores e inconsistencias en los datos.

Su finalidad es que los mismos datos puedan ser aprovechados para tantas aplicaciones como sean posibles.

Al crear una Base de datos se busca que los datos sean almacenados de modo que resulten independientes de los programas que los utilizan. Se emplean métodos bien determinados para incluir datos nuevos y para modificar o extraer los datos almacenados sin necesidad de hacer ajustes importantes a los programas ya existentes y la actualización de archivos se realiza simultáneamente.

Una Base de datos permite responder con mayor rapidez a las necesidades del usuario y la información que puede proporcionar es más amplia porque permite al usuario formular preguntas imprevistas.

Las Bases de datos constan de una estructura física que es la forma en que los datos se encuentran físicamente almacenados y una estructura lógica, que describe las conexiones o la forma

lógica con que los datos se presentan para su utilización. En las siguientes secciones se describen ambas estructuras con detalle.

1.3. ELEMENTOS DE UNA BASE DE DATOS

1.3.1. Estructura Física.

Como se decía anteriormente, se refiere a la forma como los datos se registran en el medio de almacenamiento.

Las palabras que describen las subdivisiones físicas de los datos son las siguientes:

* Registro físico: Es la unidad básica de datos que se escribe por medio de una única orden de entrada/salida dada a la computadora. Un registro físico generalmente comprende varios registros lógicos.

* Bloque: Se le llama así al grupo de datos que contiene un registro físico.

* Pista: Es el área que contiene datos que pueden ser leídos o escritos por una cabeza de lectura y escritura individual.

* Cilindro: Se refiere a un grupo de pistas al que se puede acceder sin necesidad de cambiar la posición del mecanismo de acceso. Los cilindros sólo existen en paquetes de discos.

Actualmente, se cuenta con dispositivos que almacenan los datos sobre superficies magnetizadas, de manera que se tiene

acceso a ellos y se manipulan mediante computadoras electrónicas.

Un medio de almacenamiento es el Disco Magnético que brinda capacidades de almacenamiento muy grandes con moderadas velocidades de operación.

Una memoria de disco magnético está constituida por discos rotantes cubiertos con un material magnético, con un espacio entre cada disco.

La información se graba sobre la superficie de los discos rotantes por medio de cabezas magnéticas que son posicionadas sobre el disco.

Cada banda de información alrededor de un disco determinado forma una pista.

Sobre una de las caras de un disco típico pueden existir de cien a varios miles de estas pistas de información.

Una innovación en el almacenamiento en discos, desarrollada originalmente en IBM, usa un disco flexible, de una base plástica en lugar del disco rígido convencional de base metálica.

En el disco flexible la información se lee o escribe a través de una abertura. Las conveniencias de su uso y el bajo costo han ampliado el uso de memorias de disco flexible en muchas aplicaciones.

En la mayoría de las unidades de disco flexible, el conjunto de cabezas de lectura y escritura está en contacto físico con el material de grabación.

La Cinta Magnética no es recomendable como medio de almacenamiento principal debido a su tiempo de acceso tan grande y a su acceso exclusivamente secuencial.

La información en las cintas magnéticas suele grabarse en bloques de registros. La superficie de la cinta, por lo general, está en contacto con las cabezas de lectura/escritura y casi siempre se almacena un caracter por fila a lo largo de la cinta.

La forma como los datos se almacenan físicamente es a menudo totalmente distinta de su forma lógica.

1.3.2. Estructura Lógica.

Se refiere a la forma con que los datos se presentan al programador de aplicaciones o al usuario.

Incluye los siguiente componentes:

* Palabra: Es el grupo de bits más pequeño con dirección propia.

* Campo: Es el grupo de datos nominado más pequeño. Tiene atributos que definen su clase y longitud.

* Entidades: Es una colección de campos dentro de un registro al que se nomina como un todo.

* Registro: Es una colección nominada de campos o entidades afines considerados como una unidad.

* Archivo: Es la colecciòn nominada de todas las ocurrencias de un tipo de registro dado. Sabiendo que una ocurrencia existe cuando un valor para cada campo de un tipo de registro existe.

* Diccionario de datos: Es una descripciòn completa de los campos en una Base de datos y de las relaciones entre ellos.

* Clave o Llave: Es un identificador singular de un registro. Puede ser un campo o un grupo de campos.

* Indice: Es una tabla de nùmeros de registros. Estos nùmeros llamados "apuntadores" estàn dispuestos para ayudar a encontrar ràpidamente un registro particular mediante una llave. El indice permite la recuperaciòn de registros en orden de secuencia y permite la inserciòn de nuevos registros.

1.3.3. Relaciòn lògica entre los datos almacenados.

La complejidad de una Base de Datos se tiene no tanto en el almacenamiento de los datos sino al tener que indicar las relaciones que existen entre los diversos datos que almacena. Pero establecer dichas relaciones resulta esencial para completar la estructura de los datos que cumpla con todos los requerimientos de una Base de datos.

Por principio, conviene definir los siguientes tèrminos:

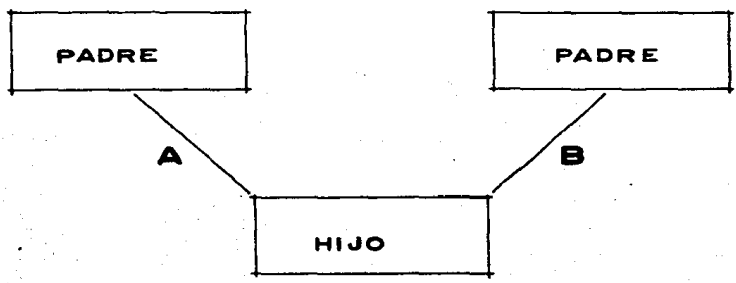
Registro Padre o Propietario: es aquèl del cual se desprende uno o màs registros hijos.

Registro Hijo o Miembro: es aquèl que se desprende de

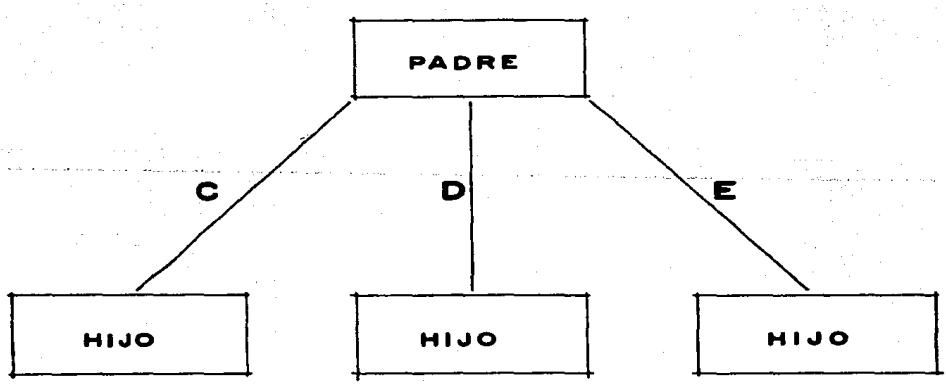
un registro padre.

Existen cuatro reglas básicas para formar conjuntos de relaciones entre registros:

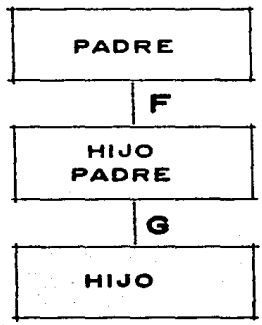
1. Cualquier registro puede participar como hijo en uno o más conjuntos de relaciones.



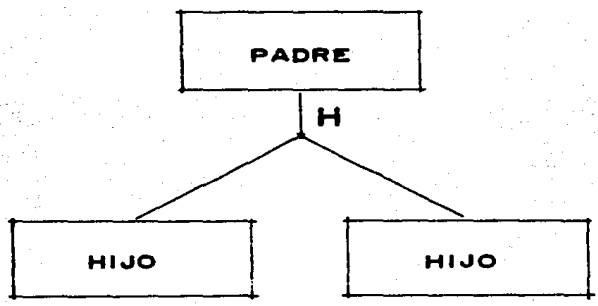
2. Cualquier tipo de registro puede ser padre de uno o más conjuntos.



3. Cualquier registro puede ser hijo en cualquier número de conjuntos y a la vez ser padre de uno o más conjuntos.



4. Un conjunto puede tener sólo un registro como padre, pero uno o más registros como hijos.



1.4. ESQUEMAS Y SUBESQUEMAS.

Definir los conceptos de un esquema y subesquema puede completar la descripción de los elementos de una Base de datos.

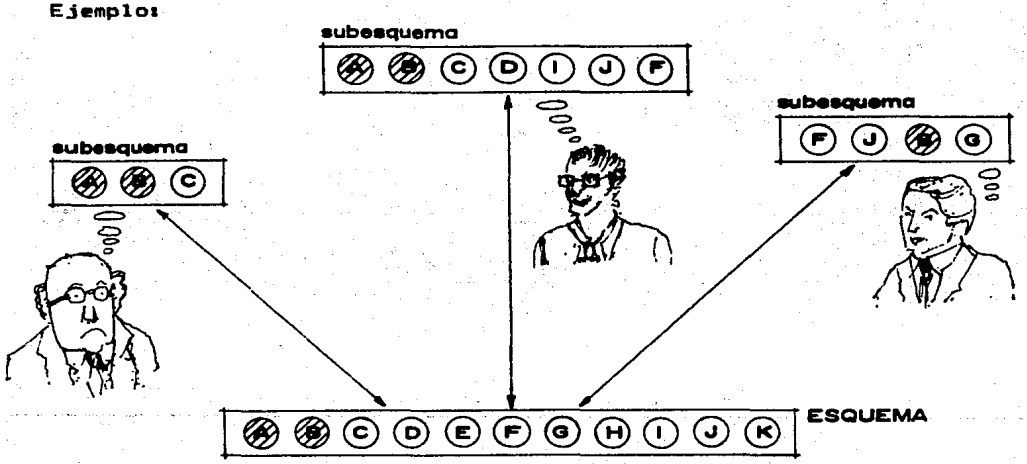
Un esquema describe la organización del conjunto de todos los datos existentes en una Base de datos. Es decir, es el diagrama de todos los tipos de datos que se utilizan, que sirve como mar-

co para inscribir los valores de los campos. Proporciona los nombres de las entidades y sus atributos y especifica las relaciones que existen entre dichas entidades.

Un subesquema es un subconjunto lôgico del esquema que sôlo nombra los campos, registros y relaciones que son de utilidad para determinado programa de aplicaciòn.

En cualquier Base de datos hay sôlo un esquema pero puede haber muchos subesquemas.

Ejemplo:



1.5. CLASIFICACION DE ARCHIVOS SEGUN SU ORGANIZACION

En una Base de datos los archivos deben organizarse de acuerdo a la manera en que van a ser utilizados.

1.5.1. Archivos Secuenciales.

Son los que tienen los registros de datos organizados en secuencia (siguiendo una clave). Si se quiere tener acceso a cualquier registro del archivo, es necesario leer primero todos los que le preceden. Y para insertar un registro hay que crear un nuevo archivo secuencial.

Estos archivos resultan muy eficientes cuando su consulta es alta.

1.5.2. Archivos indexados.

Son aquellos para los que se establece una tabla de índices con los que se compara la clave correspondiente al registro buscado.

El índice seleccionado localiza y lee el registro en el archivo usando la dirección real o una dirección próxima de dicho registro.

Si el archivo es secuencial, no es necesario que el índice incluya todos los registros, sino, más bien, referencias a bloques de registros, los que, una vez localizados, son explorados secuencialmente hasta encontrar el registro deseado.

Esto reduce el tamaño del índice permitiendo un mejor aprovechamiento del almacenamiento.

La organización secuencial de los archivos, junto con el índice, constituye una forma de direccionamiento de archivos muy común.

1.5.3. Archivos de acceso directo.

En estos archivos se almacenan los registros de datos y se tiene acceso a ellos convirtiendo una llave de un registro en la dirección real, por medio de una función donde :

f(llave) --> dirección de almacenamiento hacia donde se dirige por sí mismo el mecanismo de acceso.

Con este procedimiento se tiene un acceso más rápido a un registro determinado.

Cuando los archivos tienen gran volatilidad, es decir, un gran número de adiciones, supresiones y cambios efectuados en un período determinado, resulta ventajosa la organización directa. Pero si el archivo tiene una elevada proporción de consultas solamente, el procesamiento es mucho más costoso que con los archivos en secuencia.

1.6. CLASIFICACION DE LAS BASES DE DATOS

Las Bases de datos se pueden clasificar de acuerdo al tipo de estructura al que se ajusten las relaciones entre sus datos. De aquí que una Base de datos, en términos generales, pueda ser jerárquica, tipo red o relacional.

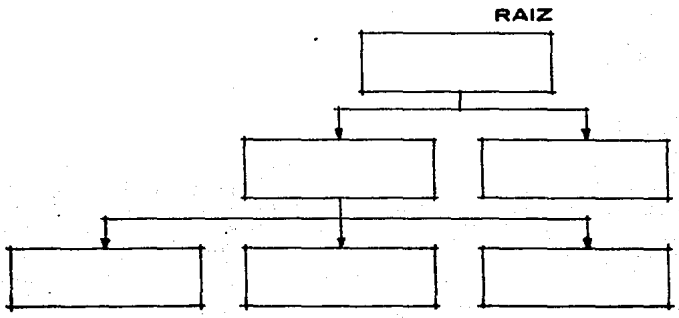
1.6.1. Base de datos jerárquica.

En una Base de datos jerárquica los datos son representados mediante estructuras de árboles. Estas estructuras son no lineales, jerárquicas y de varios niveles.

El nivel más alto de la jerarquía está constituido por

un solo registro denominado raíz.

Ej. estructura Árbol



Con excepción de la raíz, todos los demás registros están vinculados a un registro padre. Esta vinculación queda establecida por cierta asociación que existe entre ellos.

Es condición necesaria que ningún registro tenga más de un padre, aunque sí pueden tener cualquier número de hijos.

En una Base de datos jerárquica las conexiones entre archivos se establecen desde el principio y no dependen de los datos en sí.

C.J. Date en su libro "An Introduccion to Database Systems" (1) aclara estos conceptos con un ejemplo basado en una Base de datos concerniente a proveedores, productos y pedidos. Cada uno de estos registros contiene los siguientes campos:

---

(1) C.J. Date, An Introduction to Database Systems, U.S.A.,Addison-Wesley Publishing Company, 1977, p.52 .

reg. de proveedor

# PROV.	NOMBRE PROV.	EDAD	CIUDAD

reg. de productos

# PROD.	NOMBRE PROD.	COLOR	PESO	CIUDAD

reg. de pedidos

# PROV.	# PROD.	CANTIDAD

Con estos registros se puede ejemplificar una Base de datos jerárquica cuya estructura queda de la siguiente manera:

1	NUECES	CAFE	12	D.F.

REGISTRO DE UN PRODUCTO

2	VICTOR	31	IRAPUATO	300
1	JUAN	23	PUEBLA	300

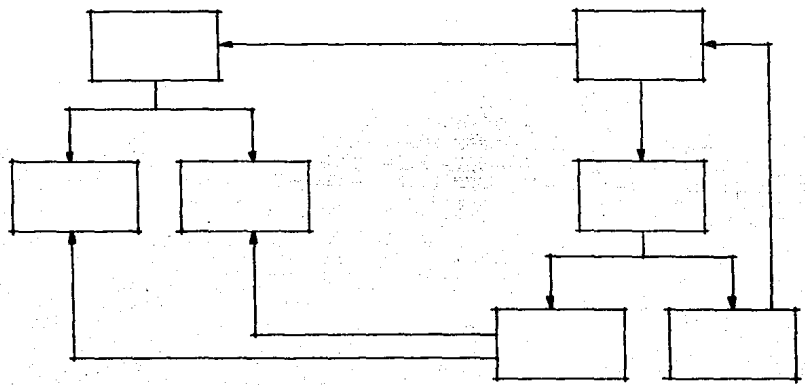
REGISTROS DE PROVEEDORES

1.6.2. Base de datos red.

Esta Base de datos está organizada mediante una estructura red que permite que el acceso pueda iniciarse con cualquier registro de un archivo y buscarle en cualquier dirección y por toda la jerarquía. Es decir, lo que permite la estructura de

redes es que un registro pueda ser hijo de màs de un conjunto de registros y ser a la vez padre de uno o màs registros.

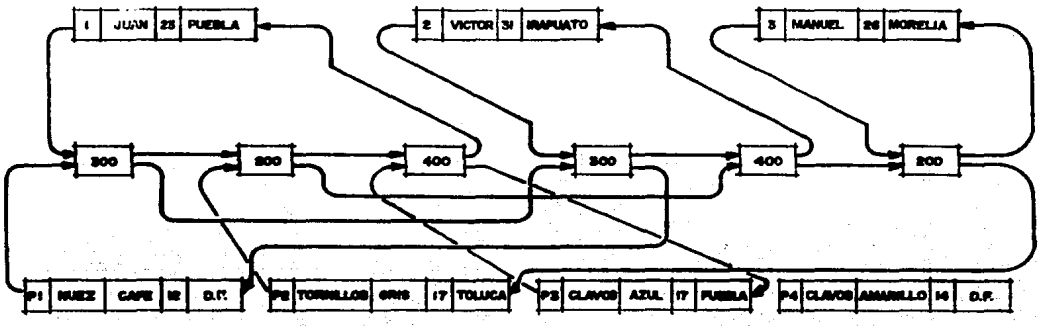
Ej. estructura red



Las asociaciones existentes estàn establecidas por medio de conectores o ligas que vinculan los registros.

La estructura interna de una Base de datos red es màs compleja que en el caso de una Base de Datos jeràrquica.

De nuevo un ejemplo de la Base de datos referente a proveedores y pedidos puede clarificar. C. J. Date muestra este ejemplo introduciendo un conector que contienen la cantidad pedida. Dicho conector representa la asociaciòn entre un proveedor y un producto. La correspondencia entre un proveedor y el conector asociado es de uno a varios, lo que muestra que las jerarquías pueden fàcilmente ser representadas en un sistema red y viceversa.



1.6.3. Base de datos relacional.

La Base de datos relacional, como su nombre lo indica, está construida por medio de relaciones.

Los datos en una Base de datos relacional están representados de una manera uniforme y sencilla, particularmente en forma de tablas normalizadas. Dichas tablas cumplen con las siguientes características:

* cada tabla representa una relación o archivo cuyos registros se encuentran en cierto orden
* cada archivo contiene sólo un tipo de registro
* cada registro tiene un número fijo de campos y un único campo identificador.

Las asociaciones entre archivos son dadas entonces, mediante registros relacionados por medio de un campo común.

A cada columna de la relación se le denomina "dominio".

Una Base de datos relacional permite mucha flexibilidad en la formulación de interrogantes de forma no prevista y hace posible ampliar la Base de datos sin necesidad de modificar la estructura lógica existente. Esto es una gran ventaja sobre las Bases de datos basadas en estructuras de árbol o red, ya que éstas limitan muchas veces el cambio que el crecimiento de una Base de datos exige.

Antes se mencionó que las estructuras red pueden también representarse como estructuras árbol admitiendo cierta redundancia. De igual modo, cualquier estructura red o árbol, puede representarse en la forma tabular requerida para tener una Base de datos relacional y viceversa.

Utilizando el mismo ejemplo de J. Date los datos quedan organizados en las tablas siguientes:

# PROV.	NOMBRE PROVEEDORES	EDAD	CIUDAD
1	JUAN	23	PUEBLA
2	VICTOR	31	IRAPUATO
3	MANUEL	26	MORELIA

# PROD.	NOMBRE PRODUCTOS	COLOR	PESO	CIUDAD
1	NUEZ	CAFE	12	D.F.
2	TORNILLOS	GRIS	17	TOLUCA
3	CLAVOS	AZUL	17	PUEBLA
4	CLAVOS	AMARILLO	14	D.F.

# PROV.	# PROD.	CANTIDAD
1	1	300
1	1	200
1	3	400
2	1	300
2	2	400
3	2	200

2. BASES DE DATOS EN LAS MICROCOMPUTADORAS.

El trabajo de Bases de datos para microcomputadoras ha seguido un desarrollo más informal en comparación con lo que se han trabajado las Bases de datos para otro tipo de computadoras, pero ha resultado también muy útil.

Se utilizará la terminología DBMS (Sistemas de manejo de Bases de datos, también llamados Sistemas de administración de Bases de datos) para referirse a los paquetes de programas y documentación que permiten instalar y utilizar Bases de datos.

Es importante considerar que las Bases de datos para microcomputadoras no llegan a cumplir con todas las características de las Bases de datos ideales.

En el campo de las microcomputadoras, tomando en cuenta la clasificación general mencionada en el capítulo anterior, los DBMS se clasifican en tres categorías:

* Sistemas de manejo de archivos o de información (FMS)
* Sistemas de manejo de Bases de datos de relación (RDBMS)
* Sistemas de manejo de Bases de datos de tipo jerárquico/red (NDBMS)

2.1. CLASIFICACION DE LAS BASES DE DATOS EN LAS MICROCOMPUTADORAS

2.1.1. Sistemas de manejo de archivos o de información.

Esencialmente, un FMS automatiza la construcción de programas para computadoras con fines mercantiles concentrándose en la definición de archivo, programación de entrada de datos,

clasificiaciòn y creaciòn de informes.

Un FMS tradicional permite la creaciòn de sistemas de archivo ùnico y a menudo proporciona un ìndice para el acceso directo. El acceso suele ser a travès de menùs. Esto ofrece como ventaja que resulta muy sencillo de utilizar por usuarios que no tengan muchos conocimientos. Se utiliza para aplicaciones muy elementales.

El archivo se define comunicando a la computadora el nombre y la longitud de cada campo y si el campo es numèrico o alfabètico. Una vez definidos los campos se pueden introducir los datos que se deseen. Si se quiere buscar un dato en particular, se introduce el nùmero de identificaciòn y el programa "explora" el archivo completo desde el comienzo del mismo o utiliza el ìndice para encontrar de forma directa el dato buscado. Una vez encontrado el registro, se puede borrar o modificar.

Un FMS permite tambièn crear e imprimir informes simplemente utilizando menùs. Se especifican los campos que se imprimirèn, incluyendo los encabezados de las columnas y los tìtulos de los informes. El FMS almacena estas especificaciones en disco para usos futuros.

Los programas de FMS, publicados màs recientemente, pueden manejar archivos mùltiples. Por ejemplo, se puede procesar un archivo de facturas y luego buscar el nombre del cliente en otro archivo basado en el nùmero del cliente.

En un FMS convencional, el diccionario de datos, los formatos de informes y el control de menùs son todos almacenados en archivos de discos especiales.

Como ejemplos de FMS están: TIM y FMS-80.

2.1.2. Sistemas de manejo de Bases de datos de relación.

Un RDBMS es más fácil de aprender que otras Bases de datos y permite construir sistemas complejos por pasos. El nombre proviene del concepto matemático relación que, como se veía en el capitulo anterior, en realidad se trata de una tabla. Esta tabla se almacena en la computadora como un archivo en donde los registros son las filas horizontales y los campos son las columnas verticales.

La capacidad propia de un RDBMS es que puede utilizar varios archivos a la vez. Para aprovechar al máximo su potencia se deben entender algunos conceptos abstractos de modelos de datos.

Otra ventaja de los RDBMS es que no tiene que prever todas sus necesidades cuando establece sus archivos.

Dos operaciones importantes de los RDBMS son la "proyección y la "unión". La proyección crea una nueva relación seleccionando determinadas columnas de una relación existente y la unión combina dos relaciones separadas.

Se ejemplifica de la siguiente manera:

Dada la relación:

NOMBRE	DIRECCION	CIUDAD	EDAD
Gonzalo	Buenavista # 2	Jalapa	23
Pablo	Av. Morelos # 144	Morelia	19
Santiago	Calle 5 # 23	Toluca	27

La PROYECCION puede crear la siguiente relación:

NOMBRE	EDAD
Gonzalo	23
Pablo	19
Santiago	27

Si por otro lado tuviéramos también la relación:

NOMBRE	ESCOLARIDAD	OCUPACION	EDO. CIVIL
Gonzalo	Actuario	Analista	Soltero
Pablo	Preparatoria	Operador	Soltero
Santiago	Dr. en Ciencias	Supervisor	Casado

La UNION puede combinar las relaciones anteriores de la siguiente manera:

NOMBRE	EDAD	OCUPACION
Gonzalo	23	Analista
Pablo	19	Operador
Santiago	27	Supervisor

Para trabajar adecuadamente como una relación, una Base de datos tiene que estar en forma normal, esto significa que se debe organizar de acuerdo a ciertas reglas matemáticas. (1)

Ejemplos de sistemas de manejo de Bases de datos relacionales son: CONDOR II, dBASE II y dBASE III.

---

(1) Ver capítulo 3, pág. 36

2.1.3. Sistemas de manejo de Bases de datos tipo jerárquico/red.

De todos los sistemas de manejo de Bases de datos de microcomputadoras los NDBMS son la copia más fácil de los sistemas de Bases de datos comerciales que se suelen utilizar en las grandes computadoras.

Así como la unidad básica en los RDBMS es la "relación", los NDBMS tienen el "conjunto" como unidad básica. Cada conjunto queda establecido por el registro padre, los registros hijos y las relaciones entre ellos, estableciendo una estructura árbol o red según el caso.

En los RDBMS se especifica la organización de los datos con un diccionario de datos, con una entrada por campo y con todos los registros en la Base de datos en el supuesto de que son los mismos; en los NDBMS, hay que aprender un lenguaje de descripción de datos (DDL) que describe la estructura de los datos. Además, en contraste con los RDBMS, en los NDBMS las relaciones entre registros quedan establecidas desde el principio por lo que se tiene la necesidad de asegurar que la estructura fundamental de los datos no va a cambiar.

Mediante los NDBMS no hay manera de recuperar directamente los datos interactivos, por lo que es necesario escribir algunos programas para poder hacerlo.

La planificación y establecimiento de un NDBMS requiere mucho análisis y diseño. Además, se debe ser capaz de escribir programas en un lenguaje establecido. El NDBMS no resulta tan fácil de usar pero tiene una potencia que permite crear aplicaciones muy completas.

Ejemplos de Bases de datos de este tipo son: MDBS III y DBVISTA.

2.2. CARACTERISTICAS DE LOS SISTEMAS DE MANEJO DE BASES DE DATOS

A continuación se presentan las características específicas de los sistemas de manejo de Bases de datos. Algunas de estas características son consideradas elementos necesarios, aunque otras están consideradas como opciones útiles para aplicaciones particulares. Conocer esta información resulta útil para la evaluación de los sistemas.

* Diccionario de datos.

El diccionario de datos es una parte esencial en las Bases de datos. Indica casi todo acerca de los datos. Los FMS y los RDBMS utilizan un diccionario para describir un archivo individual, pero los NDBMS lo utilizan para describir la Base de datos completa .

Cada campo tiene una entrada de diccionario con la información mínima siguiente:

- el nombre del campo o número
- el tipo de datos (alfabético, numérico, etc.)
- la longitud en octetos o dígitos

* Medios de consulta.

En un DBMS debe existir la manera de ver y actualizar los datos. Generalmente, los FMS utilizan menús; los RDBMS y los NDBMS utilizan órdenes especiales para tener acceso a la Base de datos.

* Generador de informes.

En un departamento de procesamiento de datos resultan necesarios los informes que muestren los datos en un cierto orden, con totales y subtotales y con los títulos correspondientes.

Los DBMS deben ofrecer la posibilidad de generar informes.

* Compatibilidad de archivos con otros programas.

Los DBMS deben tener la capacidad, en un momento dado, de leer datos de otros programas en BASIC y COBOL y, además, deben ser capaces de escribir datos para utilizarlos con estos programas.

* Capacidad de reestructuración.

Los DBMS deben dar la oportunidad de hacer ciertos cambios en la estructura de la Base de datos una vez establecida.

* Manipulación efectiva de errores.

Un buen DBMS debe proporcionar mensajes de error útiles y claros y permitir corregir el problema.

* Buena documentación y apoyo del software.

Los DBMS deben ofrecer un manual que contenga los pasos y conocimientos necesarios para saber utilizar el sistema y deben tener también una sección de referencia organizada por orden o función.

Es deseable que los DBMS tengan también las siguientes características:

* capacidad para manejar archivos múltiples
* que permita edición de pantalla completa
* capacidad de generar formatos de presentación visual en pantalla
* mecanismos de seguridad
* capacidad de multiusuarios

3. METODOLOGIA PARA EL DISENO DE UN MODELO DE DATOS

Antes de querer utilizar algún DBMS, conviene crear un modelo de datos que de algún modo nos muestre la realidad del ambiente de datos con el que vamos a trabajar.

Este diseño lógico de la Base de datos es un factor muy importante ya que un buen modelo de datos determinará en gran parte el buen funcionamiento de la Base de datos creada.

En este capítulo se presentará una metodología para el diseño de modelos de datos propuesta por James Martin en su libro "Managing the Data Base Environment". (1)

Se eligió esta metodología porque su accesibilidad facilita la creación e interpretación del modelo de datos.

Será al final del capítulo cuando se señalen los pasos a seguir para la creación de un modelo de datos. Primeramente, se explicarán ciertos conceptos y términos que son parte esencial dentro de esta metodología.

3.1. CARTAS DE BURBUJA

Las cartas de burbuja descritas a continuación, explican las ideas básicas de las estructuras de datos de una manera muy sencilla. Ofrecen un camino de representación de los datos y de las asociaciones entre ellos.

Cada TIPO de campo será manejado como una burbuja.

---

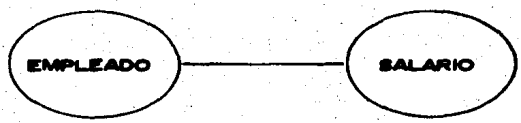
(1) James Martin, Managing the Data-Base Environment, New Jersey, Prentice-hall Inc., 1983, pp. 171 ss.

Ejemplo:

El cuidado que hay que tener es que probablemente, los datos han sido tratados descuidadamente en el pasado y quizá el mismo campo ha sido definido de manera diferente en lugares distintos, por lo que las definiciones de los campos deben ser establecidas y documentadas. Para este proceso pueden ayudar mucho los usuarios.

Un campo en sí mismo puede no ser interesante. El interés está cuando es asociado con otro campo.

Ejemplo:

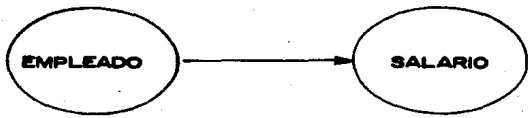


Estas asociaciones van a quedar representadas por 2 tipos de flechas:

- flechas con una sola punta
- flechas con doble punta

Si una flecha con una sola punta va del campo A al campo B significa que en cada momento, cada valor de A tiene uno y sólo un valor de B asociado con él; esto implica que si se conoce el valor de A, se puede encontrar el valor de B.

Ejemplo:



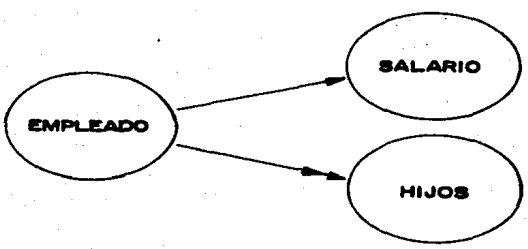
La notación "flecha con una sola punta" es consistente con la notación de lógica matemática en la cual A --> B significa que A identifica a B.

Una flecha con doble punta de A a B significa que un valor de A tiene $\varnothing$, uno o muchos valores de B asociados con él.

Ejemplo:

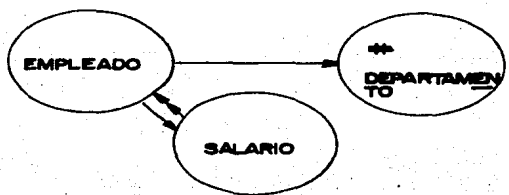
Mientras que un empleado puede tener sólo un salario en un tiempo dado; por otro lado, dicho empleado puede tener $\varnothing$, uno o más hijos.

Sintetizando las dos cartas de burbuja previas resulta:



Las dos cartas sintetizadas se refieren a dos diferentes vistas de usuarios, uno interesado en el salario y otro en los hijos de los empleados. Con ésto tenemos creada una simple estructura de datos que incorpora dos vistas de usuarios.

Si en el ejemplo anterior se agrega el campo "# departamento", se puede establecer la asociación contraria de "# departamento" para "empleado" en el caso de que un usuario quiera conocer què empleado trabaja en un departamento dado.



Entre N campos hay entonces N(N-1) posibles asociaciones.

Una Base de datos grande puede tener miles de asociaciones entre campos por lo que, en estos casos, conviene màs colocar dichos campos en grupos y referir las asociaciones entre grupos.

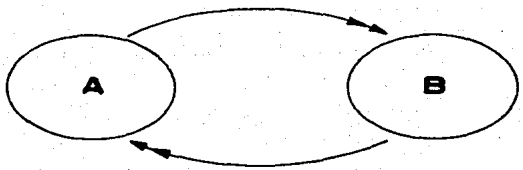
Una flecha con una sola punta entre burbujas serà llamada una ASOCIACION 1. Una flecha con doble punta se refiere a una ASOCIACION M.

Existen entonces 3 tipos de relación entre burbujas:

* relación 1 a 1 (1 : 1)

* relación 1 a MUCHOS (1 : M)

* relación MUCHOS a MUCHOS (M : M)



Dado este método de la carta de burbuja, se pueden precisar 3 definiciones importantes:

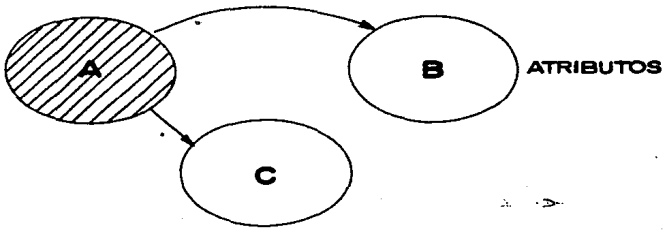
* llave primaria
* llave secundaria
* atributos

Una LLAVE PRIMARIA es una burbuja de la cual sale una o más flechas con una sola punta.

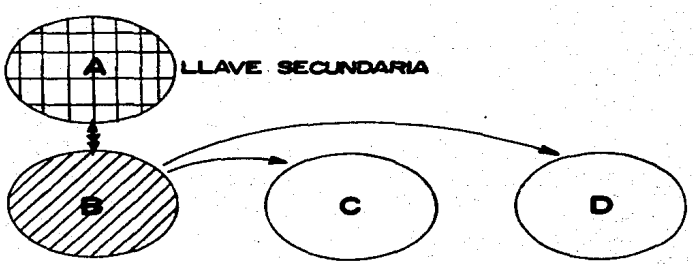
Sólo una llave primaria puede identificar muchos campos.

Ejemplo:

Los campos que no son llaves primarias son referidos como atributos no primarios. Es decir, un ATRIBUTO es una burbuja de la cual no salen flechas con una sola punta.

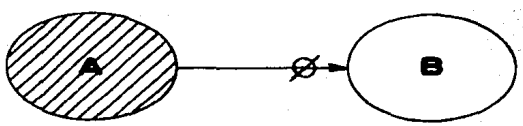


Una LLAVE SECUNDARIA es una burbuja de la cual parten una o más flechas de doble punta. Es decir, un valor de una llave secundaria es asociado con $\varnothing$, 1 o muchos valores de otro campo. No identifica de manera única a ningún campo.



Algunas veces la relación que liga una burbuja con otra puede ser opcional.

Es decir, para un valor de A puede haber o no un valor de B, lo denotamos de la siguiente manera:

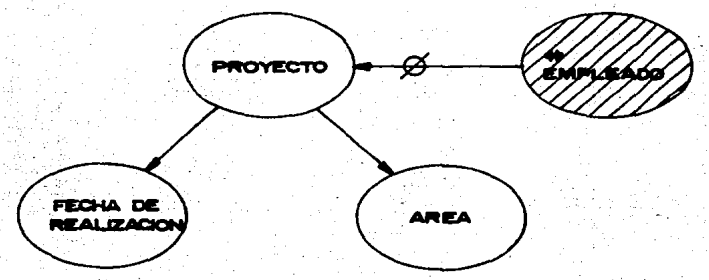


No habrà necesidad de poner un ∅ en una flecha con doble punta porque ésta incluye ya la posibilidad de que sea ∅ , 1 ò màs valores de campos los asociados con el atributo colocado al principio de la flecha.

Cuando la punta de una "liga opcional" vaya a una llave primaria la situaciòn seria diferente. Porque ésto significa que un registro entero puede o no existir.

Ejemplo:

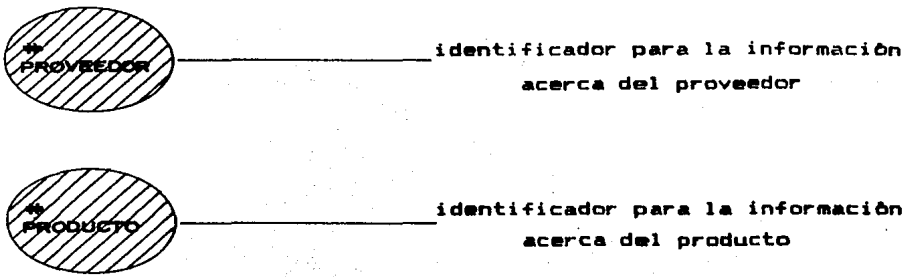
Un empleado puede o no ser asignado para un proyecto



En la carta de burbuja que resulta de combinar diferentes vistas de usuarios, las burbujas deben ser agrupadas por llaves primarias. Cada llave primaria serà el identificador ùnico de un grupo de campos.

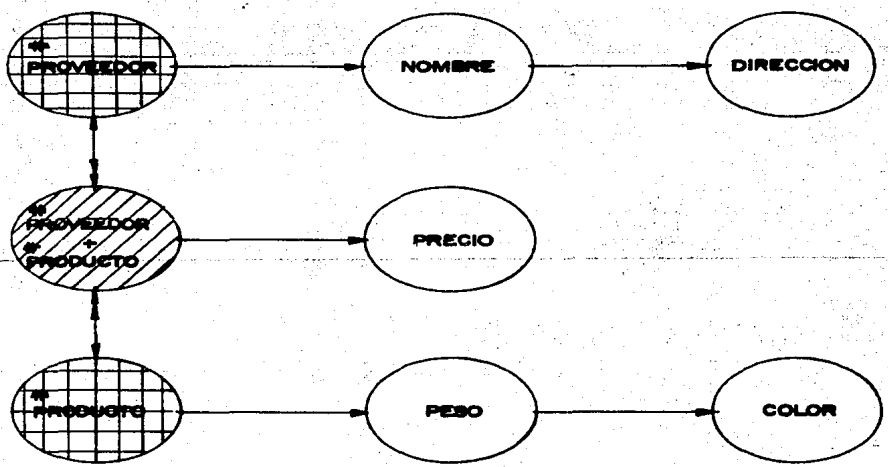
Hay atributos que no son identificados por un solo campo, necesitan que su llave primaria resulte de la combinaciòn de dos o màs campos. A estas llaves se les llama LLAVES CONCATENADAS.

Ejemplo:



Ninguna de estas dos llaves es suficiente para identificar el precio ya que el precio depende del proveedor y del producto. Entonces hay que crear una nueva llave juntando "# proveedor" y "# producto".

Ejemplo:



3.2. NORMALIZACION DE DATOS

NORMALIZAR es un procedimiento a través del cual se examinan datos o grupos de datos en conjunto y se aproximan a una forma que es capaz de ajustarse a futuros cambios, minimizando el impacto de los cambios en las aplicaciones.

Es importante tener presente que la normalización describe la representación lógica de los datos, no la física.

En la vida real, los datos existen como grupos de campos. Estas agrupaciones no están en forma normalizada. Los datos que no están normalizados pueden ocasionar serios problemas en el futuro.

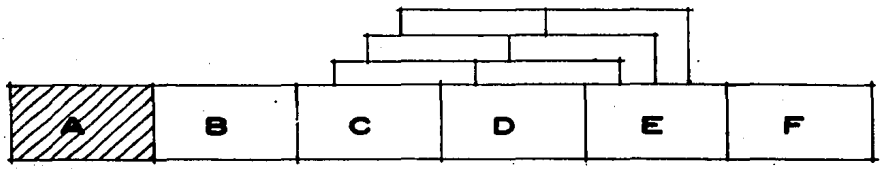
Al normalizar, resulta una estructura de datos más clara y sencilla, la cual es necesaria para muchos de los subsecuentes pasos en la ingeniería de información. La estructura resultante es también más estable y a la vez, más "abierta" a los posibles cambios.

$ Primera forma normal.

Se refiere a una colección de datos organizados dentro de registros en la que no se encuentra repetición de grupos de campos dentro de un mismo registro. Es decir, son archivos planos, matrices bidimensionales de datos.

Ejemplo:

Sean A, B, C, D, E, F campos o distintos grupos de campos de un registro R representados de la siguiente manera:



Es decir, para cada ocurrencia de A existe repetición en los campos o grupos de campos C y D.

Al aplicar la primera forma normal se convierte este registro en 2 archivos planos:

* Segunda forma normal.

Un registro R está en segunda forma normal si está ya en primera forma normal y cada campo no primario de R depende funcionalmente de manera plena de cada llave candidata de R.

Para comprender mejor la definición anterior, se definen a continuación los siguientes términos: dependencia funcional, dependencia funcional plena y llaves candidatas.

Dependencia funcional.-

El campo B de un registro R es funcionalmente dependiente de un campo A de R si, para cada instancia de tiempo, cada valor en A no tiene más que un valor en B asociado con él en el registro R.

Por ejemplo, en un registro de empleado, el campo "salario" es funcionalmente dependiente del "# de empleado", porque para cada número de empleado hay un determinado salario. En cambio, "# de empleado" no es funcionalmente dependiente del campo "salario" porque más de un empleado puede tener el mismo salario.

Dependencia funcional plena.-

Un campo o una colección de campos B de un registro R puede tener dependencia funcional plena sobre otra colección de campos A del registro R, si B es funcionalmente dependiente de toda la colección de campos A pero no de cualquier subconjunto de A.

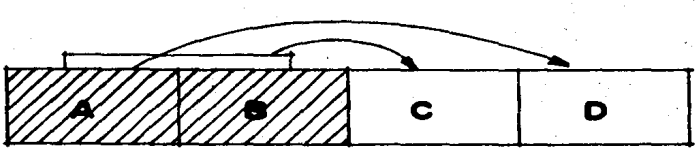
Llave candidata.-

Una llave candidata de un registro normalizado es aquella que cumple con las siguientes propiedades:

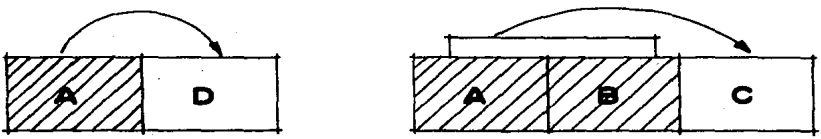
1. Para cada ocurrencia de un registro, la llave identifica de manera única a dicho registro.
2. No tiene redundancias.

Ejemplo:

Se tiene el registro R con los campos o grupos de campos A, B, C y D relacionados como sigue:



La segunda forma normal exige que se separe el registro anterior de la siguiente forma:



en donde los campos no primarios D y C dependen funcionalmente de las llaves candidatas A y AB respectivamente.

Hay necesidad de separar el registro porque D no tiene una dependencia funcional plena sobre la llave AB sino que sólo depende de A.

* Tercera forma normal.

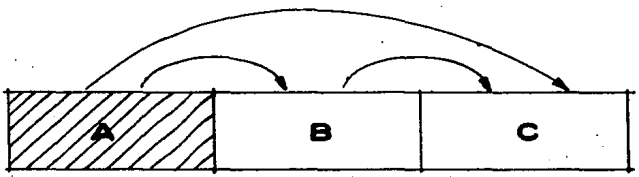
Un registro que está en segunda forma normal puede tener otro tipo de anomalía. Puede tener un campo que en sí no es una llave pero identifica a otros campos. A ésto se le llama DEPENDENCIA TRANBITIVA. Las dependencias transitivas pueden causar problemas. La tercera forma normal quita las dependencias transitivas.

Entonces, formalmente se define que un registro R está en tercera forma normal si está ya en segunda forma normal y ningún campo no primario de R depende transitivamente de la llave primaria de R.

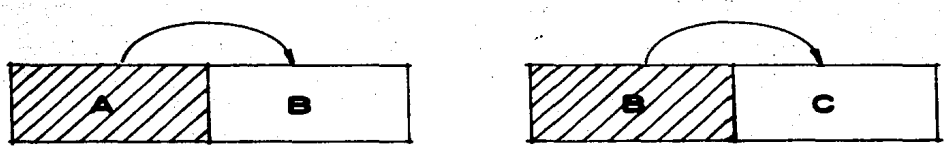
Ejemplo:

Sean A, B y C 3 campos o distintas coleccio-

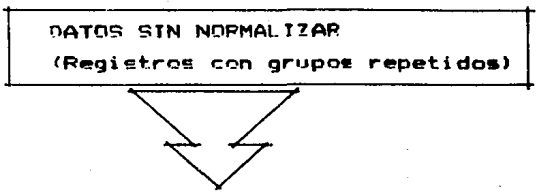
nes de campos de un registro R. Si C es funcionalmente dependiente de B y B es funcionalmente dependiente de A, entonces C es funcionalmente dependiente de A.



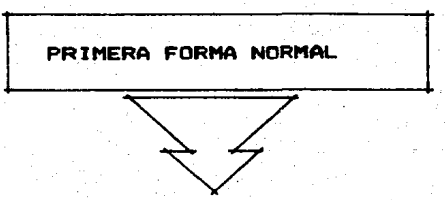
La conversión a la tercera forma normal quita la dependencia transitiva dividiendo el registro en dos:



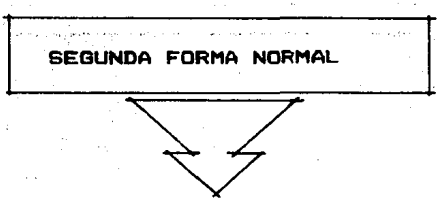
El siguiente esquema muestra el proceso de conversión a la tercera forma normal:



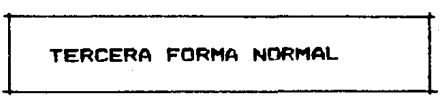
Descomponer todos los registros no planos en registros bidimensionales



Para registros cuyas llaves tengan màs de un campo, asegurarse que todos los otros datos sean dependientes de toda la llave. Dividir los registros si es necesario, para asegurar êsto.



Eliminar todas las dependencias transitivas



3.3. SINTESIS CANONICA

El proceso de síntesis canónica crea el modelo lógico de los datos que, posteriormente, podrá ser manejado por el software.

A continuación, se describe el procedimiento a través del cual se puede obtener un modelo canónico. Los registros en esta estructura resultante se encuentran en tercera forma normal.

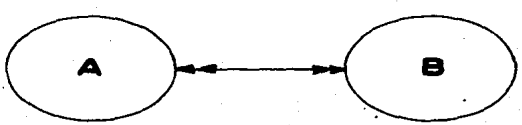
1. Se toma la primera vista de datos de algún usuario y se maneja en forma de una carta de burbuja, hasta asegurarnos que la respresentación de la vista del usuario está en tercera forma normal.

2. Tomar la siguiente vista del usuario, representarla o incluirla dentro de la gráfica existente. Eliminar los sinónimos u homónimos, si los hay.

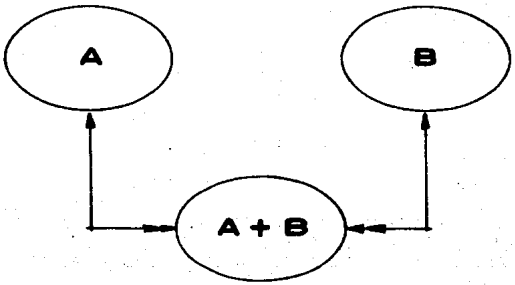
3. En la gráfica resultante distinguir entre los atributos y las llaves primarias. Señalar de algún modo estas llaves.

4. Para cada asociación entre llaves, agregar la asociación inversa si no está ya en la gráfica. Si resulta una relación M:M entre llaves, determinar si la asociación inversa tiene verdadera posibilidad de ser usada. Si es posible que se use en el futuro, entonces reemplazarla introduciendo una llave concatenada extra incorporando los campos llave que estaban ligados.

Ejemplo:



quedaría:



5. Examinar las asociaciones e identificar las que sean redundantes y eliminarlas.

6. Repetir los 4 pasos anteriores hasta que todas las vistas de los usuarios sean incluidas en la gráfica.

7. Identificar las llaves raíces. Una LLAVE RAIZ es una llave primaria de la cual salen flechas de doble punta hacia otras llaves.

Para mayor claridad, conviene arreglar las gráficas con las llaves raíces en la cima.

8. Observar si la gráfica contiene cualquier atributo aislado. Un ATRIBUTO AISLADO es aquél que no está identificado por ninguna llave primaria. Es decir, es una burbuja que no tiene flechas con una sola punta ni entrando ni saliendo de ella.

Si se tienen atributos aislados, se les puede tratar por cualquiera de los siguientes caminos:

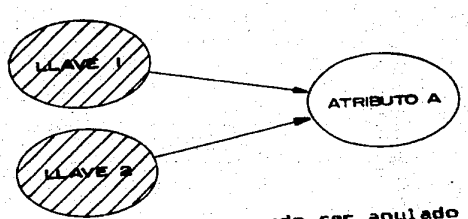
a) Podrá ser implementado como un atributo repetido en un registro de longitud variable.

b) Podrá ser tratado como una llave solitaria o como un registro con un solo campo.

c) Podría ser el resultado de un error en la interpretación de los datos, en este caso, el error deberá corregirse.

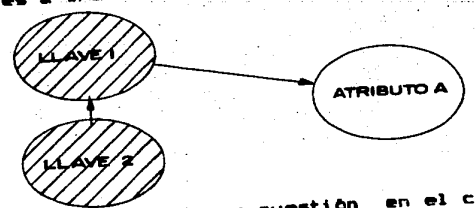
9. Ajustar la gráfica para evitar atributos intersectados. Un ATRIBUTO INTERSECTADO es aquél al que llegan más de una flecha con una sola punta.

Ejemplo:



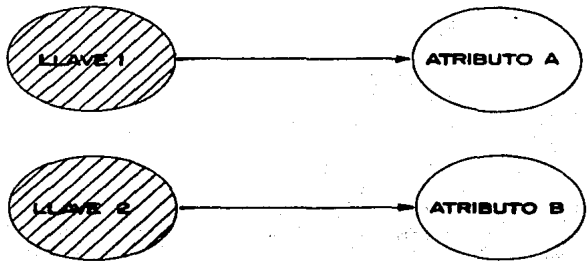
Un atributo intersectado puede ser anulado por cualquiera de los 3 siguientes caminos:

a) Todas excepto una liga pueden ser reemplazadas con ligas equivalentes a través de una llave existente.

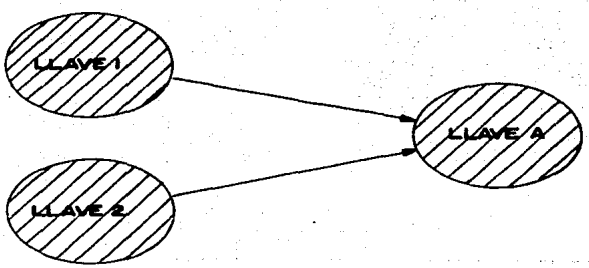


b) Duplicar el atributo en cuestión en el caso de que no exista la posibilidad de que este atributo se utilice como llave

primaria en el futuro.



c) Tratarlo como una llave solitaria (registro con un solo campo).



10. Redibujar los campos arreglados dentro de los grupos (registros), cada uno teniendo una llave primaria y sus atributos asociados.

11. Identificar todas las llaves secundarias.

12. Procurar que el modelo resultante sea lo más estable posible.

a) Conviene crear la estructura de los datos teniendo siempre presente cómo van a ser utilizados los datos en el futuro. Esto minimizará aquellos cambios que causarán que los programas de aplicación existentes sean reescritos.

b) Muchos de los cambios en la estructura básica de un registro son porque, suele suceder que, un campo que en un momento dado es atributo, llega a ser después una llave primaria.

Para esto hay que preguntarse con cada campo-atributo: ¿Hay posibilidad de que llegue a utilizarse como llave primaria en el futuro?. Si la hay, convendrá ponerla como llave primaria desde ahora.

c) El proceso de síntesis borra flechas que parecen ser redundantes. Habrá que checar todas las flechas borradas para asegurarse que en realidad eran redundantes.

d) El modelo puede contener algunas relaciones 1:1 . Estas indican una llave candidata. Es decir, si A identifica a B y B identifica a A (A <---> B), entonces A y B son funcionalmente equivalentes.

Ante estos casos, conviene preguntarse: ¿Realmente son A y B funcionalmente equivalentes?, ¿es probable que permanezcan así en el futuro?.

Ejemplo:

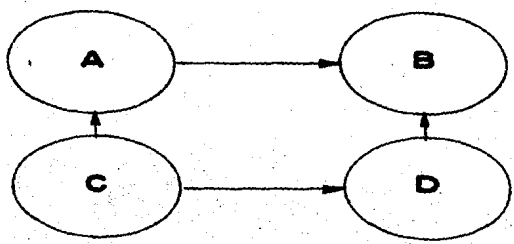
Sea A = "# empleado"
B = "nombre empleado"
tal que A <---> B

En este ejemplo, A y B no son realmente llaves candidatas porque puede darse el caso de que dos empleados tengan el mismo nombre. Por lo que, sólo el campo A = "# empleado" podrá ser utilizado como llave primaria, quedando:

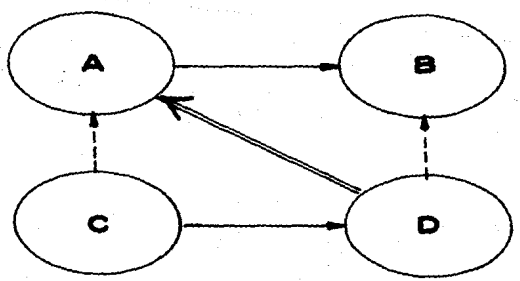
A <<---> B

e) Habrá que examinar los ciclos que, con flechas de una sola punta, se hayan podido formar en el modelo.

Ejemplo:



Esto se puede solucionar agregando una flecha como lo muestra el siguiente dibujo, en donde las flechas punteadas resultan redundantes y se puede romper el ciclo.



13. Finalmente, el modelo canônico resultante deberà acomodarse al esquema que ofrece un paquete de software particular y deberàn hacerse los ajustes que sean necesarios. Una vez diseñado el esquema del software hay que asegurar que todas las vistas de los usuarios originales pueden ser manejadas.

3.4. PASOS A SEGUIR PARA LA CREACION DE UN MODELO DE DATOS CONFIABLE

1ero. DEFINIR LA ESTRATEGIA.

* Determinar quê Bases de datos van a ser diseñadas: Bases de datos subordinadas, de aplicaciones aisladas o de Sistemas de informaciôn.

* Determinar todos los tipos posibles de usuarios finales de una Base de datos dada.

* Seleccionar una herramienta de modelado y un diccionario de datos.

* Seleccionar el software adecuado.

2do. ENTRADA AL PROCESO DE MODELADO DE DATOS.

* Hacer una captura de todos los documentos que serân derivados de la Base de datos o que servirân como entradas a êsta.

* Determinar con los usuarios finales quê tipos de datos quieren obtener de la Base de datos, actualmente y a futuro.

* Determinar a partir de un proceso de anàlisis de sistemas si algunos requerimientos nuevos de documentos o registros son urgentes.

* Planear si los archivos o las Bases de datos actuales podràn coexistir con la nueva Base de datos o si deben ser convertidos. Si pueden coexistir, se tiene que planear el puente entre el viejo sistema y el nuevo.

* Emplear un diccionario de datos para documentar una descripciòn del significado de cada elemento de los datos.

3ero. CREACION DEL MODELO DE DATOS.

* Emplear un tipo de sintesis canònica o en su defecto, asegurar que el modelo se encuentre en una tercera forma normalizada òptima.

* Inspeccionar cada entrada para ver si puede ser simplificada.

* Para cada elemento de la entrada de datos, observar que ningùn otro elemento utilice el mismo nombre.

* Emplear estàndares convencionales para la selecciòn de los nombres de los elementos de entrada de los datos.

* Asegurar que todos los atributos introducidos como entrada sean dependientes en absoluto de la llave con la cual se les identifica.

* Asegurar que los grupos de datos introducidos como entrada no contienen dependencias transitivas.

4to. INSPECCION DE LA SALIDA DEL MODELO DE DATOS.

* Con los usuarios revisar el diccionario de datos para asegurar que todos concuerden con las definiciones de los elementos de datos.

* Revisar con los usuarios el modelo obtenido para asegurar que sus requerimientos de datos pueden ser derivados de él.

* Con los usuarios hacer una estimación de los posibles usos futuros de los datos. Para cualquiera de estos usos, en los cuales el modelo no sirva, crear una nueva entrada al proceso de síntesis.

* Examinar cada campo en el modelo para determinar si podría convertirse en una llave primaria en el futuro.

* Completar el mapeo reverso de cualquiera de los lazos entre llaves para identificar alguna posible liga M:M. La herramienta de síntesis creará una llave concatenada extra en el modelo para prevenir cualquier inserción futura de datos.

* Examinar cualquier liga que haya sido removida en el proceso de síntesis para asegurar que verdaderamente era redundante.

* Si existen llaves candidatas en el modelo resultante hay que confirmar que en efecto permanecerán de esa forma en el futuro.

* Confirmar el tratamiento que se le ha dado a cualquier atributo de intersección para asegurar que se ha seguido

el mejor camino.

* Inspeccionar los ciclos en la salida modelada. Observar si una liga futura puede ser añadida para romper el ciclo.

* Convertir el modelo a un esquema lôgico de software.

* Utilizar un rediseño rápido después de que se hagan cambios con el fin de mantener el interés de los usuarios.

5to. CONSIDERACIONES FISICAS.

* Para cada patrôn de uso en el modelo hay que añadir un volumen de utilizaciôn.

* Para cada liga A ---> B determinar cuàntos valores de B en promedio son asociados con un valor de A.

* Ver si el DBMS permite que nuevos atributos sean añadidos a un grupo existente de datos sin causar que tengan que ser escritos nuevamente los programas que utilizan dicho grupo. Preguntarse si convendrà dejar espacio libre en los registros.

* Ver si el DBMS permite que ligas de llaves secundarias sean añadidas a una Base de datos existente sin ocasionar que los programas que ya existen tengan que ser escritos nuevamente.

* Si algûn patrôn de llave secundaria tiene un volumen de utilizaciôn grande, ver la posibilidad de que el modelo se parta en Bases de datos separadas debido a razones de la màquina.

4. EL dBASE III DE ASHTON-TATE

Uno de los paquetes de Base de datos màs utilizado actualmente es el dBASE III, que es un sistema de manejo de Base de datos relacional.

4.1. ANTECEDENTES DEL dBASE III

dBASE III es un sucesor de dBASE II, la primera Base de datos popular para microcomputadoras.

Un grupo de cientificos del Jet Propulsion Laboratory (JPL) en Pasadena, California, usaban un DBMS para rastrear informaciòn recibida de diferentes satèlites. Cuando se introdujeron las microcomputadoras, Wayne Ratliff, un diseñador de sistemas de software de JPL, diseñò un sistema de Base de datos para su microcomputadora utilizando el sistema JPL como modelo. Una vez terminado, Ratliff decidiò sacar al mercado su manejador de Base de datos ponièndole el nombre de Vulcan.

Vulcan, a pesar de sus limitaciones, fue un manejador de Bases de datos poderoso en su tiempo.

Uno de sus partidarios fue George Tate, un distribuidor de software al cual se le asignò un contrato con derechos para las ventas del programa.

Tate buscò la manera de incrementar las ventas de Vulcan. Comenzò por cambiarle el nombre por el de dBASE II. No habia dBASE I, pero el II ya implicaba una versiòn originaria.

Posteriormente, Tate se asociò con Hal Lashlee y formaron la Compañia Asthon-Tate para distribuir dBASE II. Despuès de varios

años de èxito del dBASE II, los competidores comenzaron a buscar productos con mayores ventajas sobre el dBASE II. Ante èsto, Ratliff y un equipo de diseñadores de Ashton-Tate trabajaron en un nuevo programa: dBASE III.

dBASE III es diseñado para utilizarse de manera amplia en microcomputadoras de 16 bits y es escrito en el lenguaje de programaciòn C. Ofrece un nûmero significativo de mejoras sobre dBASE II.

4.2. CARACTERISTICAS DEL dBASE III

4.2.1. Requisitos del sistema.

dBASE III requiere del sistema operativo MS DOS o PC DOS, es decir, se puede utilizar en computadoras tales como IBM PC, Compaq, Columbia, Corona o Eagle PC.

Se necesita un mínimo de memoria de 256 K, dos unidades de disco flexible o una unidad de disco flexible y otra de disco duro y una terminal de video de 25 x 80 caracteres. Con dBASE III se puede utilizar cualquier impresora que imprima al menos 80 columnas de texto.

4.2.2. Especificaciones.

dBASE III permite utilizar 5 tipos de campos:

* campos de caracteres - incluyen cualquier caracter: letras, nûmeros, símbolos especiales o espacios en blanco.

Un campo de caracteres tiene un tamaño máximo de 254 caracteres.

* campos numéricos - incluyen números con o sin decimales. Pueden tener hasta 19 dígitos.

* campos de fechas - el formato para meter fechas es MM/DD/AA.

* campos lógicos - consiste en una letra que representa un valor falso o verdadero. Las letras T o Y representan un valor verdadero y F o N representan un valor falso

* campos de memoria - dBASE III puede almacenar bloques grandes de texto para cada registro en forma de campos de memoria. Se pueden almacenar hasta 4000 caracteres.

dBASE III puede almacenar hasta:

1 billón de registros
128 campos por registro
4000 caracteres por registro

El lenguaje de programación de dBASE III es una colección de comandos que permiten el acceso a la Base de datos. Los 124 comandos en el lenguaje de programación de dBASE III ofrecen una amplitud de caminos para manejar información.

4.2.3. Ventajas.

dBASE III cuenta con:

* Diccionario de datos que identifica los campos por nombre y define su tipo, tamaño y el número de lugares decima-

les.

* Medios de consulta.
* Documentaciòn y apoyo.
* Ediciòn de pantalla completa.
* Formato de presentaciòn en pantalla.
* Generador de informes.
* Capacidad de reestructuraciòn.
* Manipulaciòn efectiva de errores.
* Capacidad para ordenar archivos. Puede ordenar varios campos simultàneamente.
* Capacidad para trabajar hasta con 10 archivos al mismo tiempo.
* Capacidad para transferir archivos a WordStar, Word Processors, MailMerge, Lotus y otros.
* Lenguaje incorporado.

4.2.4. Limitaciones.

Una de las mayores limitaciones de dBASE III es que muchas operaciones requieren informaciòn almacenada en disco y esto atrasa los programas. El retraso es màs evidente en sistemas basados en discos flexibles que en sistemas con disco duro.

Otra limitaciòn es que dBASE III no està diseñado para ser usado en sistemas multiusuarios.

4.3. CREACION DE UNA BASE DE DATOS CON dBASE III

dBASE III crea una Base de datos mediante el comando CREATE. Este comando realiza tres tareas: crea un archivo de Base de datos, define su estructura y abre el archivo para recibir la información .

Su formato es :

CREATE nombre del archivo

dBASE III automáticamente le asigna al nuevo archivo la extensión .DBF

Para introducir información en la Base de datos existente se debe utilizar el comando APPEND .

4.4. MODIFICACION DE UNA BASE DE DATOS CON dBASE III

Con dBASE III es posible modificar la estructura de una Base de datos y hacer cambios en los registros y campos determinados.

* Comando EDIT.

El modo de edición es un modo de operación con el cual se pueden cambiar los datos contenidos dentro de un registro elegido.

Mediante este comando el registro se muestra de la misma manera que con el comando APPEND .

* Comando BROWSE.

Despliega más de un registro en la pantalla y permite la edición de los mismos.

BROWSE despliega tantos campos como quepan en la pantalla y puede mostrar hasta 23 registros en forma horizontal.

* Comandos DELETE y PACK.

dBASE III usa la combinación de estos 2 comandos para borrar un registro. Con DELETE el registro queda marcado con un asterisco (*) indicando que es un candidato para ser borrado. DELETE da la oportunidad de cambiar de intención con el comando RECALL que elimina dicha marca.

El comando PACK remueve todos los registros marcados en la Base de datos y renumera los registros que quedan rellenando cualquier espacio creado por los registros borrados.

También se pueden borrar archivos dentro de dBASE III con una variación del comando DELETE:

DELETE FILE nombre del archivo

* Comando CHANGE.

Es un comando global, es decir, realiza la operación del comando sobre la Base de datos entera.

El comando CHANGE, primeramente, encuentra el campo y después, pregunta en cada registro sobre la corrección que se quiere hacer.

* Comando REPLACE.

El comando REPLACE opera muy parecido a CHANGE, excepto en que REPLACE no pregunta en cada registro sobre el tipo de cambio que se desea hacer, sino que hay que especificar el cambio dentro del mismo comando.

* Comando MODIFY STRUCTURE.

Este comando permite cambios en la estructura de la Base de datos. MODIFY STRUCTURE copia la Base de datos en un archivo temporal y entonces realiza las modificaciones que le sean indicadas.

4.5. ORDENAMIENTO DE UNA BASE DE DATOS CON dBASE III

dBASE III permite arreglar una Base de datos de diferentes maneras mediante los comandos SORT e INDEX.

* Comando SORT.

Con este comando, dBASE III ordena en forma ascendente o descendente una Base de datos. SORT crea un nuevo archivo con un nombre diferente en donde coloca los registros ordenados.

dBASE III da la oportunidad de ordenar por más de un campo listándolos como parte del comando SORT. El campo que se quiera ordenar primero deberá de listarse primero.

* Comando INDEX.

INDEX crea un archivo que contiene información respecto a la localidad de los registros individuales en la Base de datos original. El orden que sigue siempre es ascendente. La extensión que dBASE III da a estos archivos es .NDX

Al igual que con Sort, se pueden indexar archivos basados en varios campos. Aunque existe una limitación, no se pueden indexar campos múltiples que no tengan el mismo tipo de campo.

Un archivo indexado se puede abrir con cualquiera de los siguientes comandos: USE INDEX Y SET INDEX .

Todos los cambios efectuados en la Base de datos se incorporan automàticamente en los archivos indexados que estèn abiertos.

Una de las ventajas del comando INDEX es que crea archivos màs pequeños que los que crea SORT, ya que INDEX sólo ordena el campo o los campos que se especifiquen, en cambio el comando SORT, reordena toda la Base de datos.

4.6. GENERACION DE REPORTES CON dBASE III

dBASE III ofrece 2 caminos para la impresión de reportes:

* haciendo una combinación de LIST y PRINT para imprimir la información especificada
* utilizando el generador de reportes de dBASE III mediante el comando CREATE REPORT.

CREATE REPORT crea un archivo especial cuya extensión es .FRM Este archivo incluye el formato del reporte.

Dicho reporte se imprime mediante el comando:

REPORT FORM nombre del archivo TO PRINT

El comando MODIFY REPORT permite cambiar el formato de los reportes.

4.7. PROGRAMACION CON dBASE III

Una serie de comandos que ejecuten cierta tarea pueden ser escritos en un archivo especial almacenado por el procesador de palabras de dBASE III. La extensiòn de este tipo de archivos es .PRG

La ejecuciòn del programa almacenado en estos archivos se realiza mediante el comando DO.

Para entrar en el procesador de palabras de dBASE III se uti- III se utiliza MODIFY COMMAND.

Existen varios conceptos asociados con la programaciòn de dBASE III:

* Expresiones.
  Una expresiòn puede ser una combinaciòn de uno o màs campos, funciones, operadores, variables de memoria o constantes.

* Variables de memoria.
  Una variable de memoria es una localidad de memoria dentro de la computadora usada para guardar datos. No puede contener màs de 10 caracteres.

El comando STORE es comùnmente usado para asignar datos a una varible de memoria.

dBASE III permite 4 tipos de variables: caracteres, nùmeros, fechas y variables lògicas.

* Operadores.
  Los operadores son representados por simbolos. dBASE III

tiene 4 tipos de operadores: matemÁticos, relacionales, lôgicos y operadores para manejo de cuerdas.

- Operadores matemàticos.
  Son usados para producir resultados numèricos.
  Los símbolos son:

Exponenciación	* * ô ^
División	/
Multiplicación	*
Substracción	-
Suma	+

- Operadores relacionales.
  Son usados para comparar cuerdas de caracteres con cuerdas de caracteres y nûmeros con nûmeros. Los valores que compara pueden ser constantes o variables.

  Los operadores relaciones son:

<	menor que
>	mayor que
=	igual
< > ô #	distinto
< =	menor o igual
> =	mayor o igual

* Operadores lôgicos.
  Comparan valores del mismo tipo para producir un valor verdadero o falso. Son: .AND. , .OR. y .NOT.

* Operadores para manejo de cuerdas.
  El ûnico operador "cuerda" que se utiliza en dBASE III es el signo + que realiza una concatenación, es decir, combina 2 ô màs cuerdas.

Ejemplo:

Si

ANIMAL = " Pâjaro "
COLOR = " Azul "

entonces

ANIMAL + COLOR = " Pâjaro Azul "

* Funciones.

Las funciones son utilizadas en dBASE III para realizar operaciones especiales complementando a los comandos.

dBASE III tiene 37 diferentes funciones.

Al programar, comûnmente se usan las 2 siguientes funciones:

EOF (end of file)	-	indica cuândo el apuntador de los registros ha llegado al final de un archivo
BOF (beginning of file)	-	indica cuândo dicho apuntador estâ al comienzo de un archivo.

4.8. PRINCIPALES COMANDOS DE dBASE III

dBASE III cuenta con 124 comandos que ofrecen una amplitud de caminos para manejar informaciôn. A continuaciôn se presentan algunos de ellos. (1)

---

(1) Ashton-Tate, dBASE III - Guia de Referencia, 1985.

Ejemplo:

Si

ANIMAL = " Pàjaro "
COLOR = " Azul "

entonces

ANIMAL + COLOR = " Pàjaro Azul "

* Funciones.

Las funciones son utilizadas en dBASE III para realizar operaciones especiales complementando a los comandos.

dBASE III tiene 37 diferentes funciones.

Al programar, comûnmente se usan las 2 siguientes funciones:

EOF (end of file)	-	indica cuândo el apuntador de los registros ha llegado al final de un archivo
BOF (beginning of file)	-	indica cuândo dicho apuntador està al comienzo de un archivo.

4.8. PRINCIPALES COMANDOS DE dBASE III

dBASE III cuenta con 124 comandos que ofrecen una amplitud de caminos para manejar informaciôn. A continuaciôn se presentan algunos de ellos. (1)

---

(1) Ashton-Tate, dBASE III - Guia de Referencia, 1985.

* APPEND.- Agrega registros al final de un archivo.

* ACCEPT.- Almacena en una variable de memoria la secuencia de caracteres introducida.

* AVERAGE.- Calcula la media aritmètica de las expresiones numèricas especificas.

* BROWSE.- Permite editar varios registros a la vez.

* CHANGE.- Modifica el contenido de los campos especificados en el archivo.

* CLOSE.- Cierra los archivos.

* CONTINUE.- Continùa la bùsqueda del siguiente registro que cumpla la condiciòn especificada en el ùltimo LOCATE.

* COPY.- Duplica una parte o la totalidad de un archivo en otro nuevo.

* COPY FILE.- Crea una copia de cualquier tipo de archivo.

* COUNT.- Cuenta el nùmero de registros del archivo que cumplan la condiciòn especificada.

* CREATE.- Define un nuevo archivo.

* DELETE FILE.- Borra un archivo del directorio.

* DIR.- Muestra una parte o la totalidad del directorio del disco.

* DISPLAY.- Muestra contenido de un archivo.

* DO.- Comienza la ejecuciòn de la secuencia de comandos contenida en determinado archivo.

* DO CASE.- Permite seleccionar opciones.

* DO WHILE.- Permite operaciones repetitivas.

* EDIT.- Realiza cambios en el registro especificado.

* EJECT.- Genera un salto de pàgina en la impresora.

* ERASE.- Elimina el archivo especificado del directorio.

* FIND.- Localiza un registro indexado.

* IF.- Permite ejecutar comandos si la condiciòn es verdadera. ELSE define el conjunto de comandos a ejecutar si la condiciòn es falsa.

* INDEX.- Crea un archivo indexado.

* INPUT.- Acepta datos lógicos, numéricos o caracteres.
* JOIN.- Crea un nuevo archivo combinando los campos y registros de dos archivos.
* LIST.- Muestra el contenido del archivo.
* LOCATE.- Encuentra registros que cumplen condiciones especificadas.
* MODIFY COMMAND.- Crea y edita archivos de formato, de programas y de texto.
* MODIFY STRUCTURE.- Modifica la estructura del archivo.
* PACK.- Elimina de forma permanente los registros del archivo.
* QUIT.- Cierra todos los archivos y devuelve el control al sistema operativo.
* RECALL.- Recupera registros que se han marcado para eliminar.
* REINDEX.- Reconstruye archivos indexados.
* REPLACE.- Cambia el contenido de los campos especificados.
* RESTORE.- Recupera las variables de memoria almacenadas.
* RUN.- Ejecuta el mandato del sistema especificado desde dBASE III.
* SAVE.- Escribe las variables de memoria activas en un archivo de memoria.
* SELECT.- Abre un máximo de diez archivos de Base de datos (.DBF) simultáneamente.
* SORT.- Ordena un archivo.
* STORE.- Crea variables de memoria a partir del cálculo de una expresión.
* USE.- Abre un archivo de Base de datos (.DBF) y, opcionalmente, los archivos indexados especificados.
* WAIT.- Suspende la ejecución del programa hasta que se oprima una tecla.
* ZAP.- Elimina todos los registros del archivo.

5. EJEMPLO DE LA APLICACION DE LA METODOLOGIA EN EL DISENO DE UN SISTEMA DE CALIFICACIONES

Este capitulo presenta una aplicación de lo visto anteriormente.

Se comenzará con la creación de un modelo de datos confiable para un sistema de calificaciones siguiendo la metodología señalada en el capitulo tercero.

Finalmente, utilizando dicho modelo y mediante dBASE III, se crea la Base de datos adecuada y se diseña el sistema.

1er. paso: Se define la ESTRATEGIA.

La idea es crear un sistema de calificaciones para la Preparatoria de un Colegio determinado. Actualmente, este trabajo lo realiza ya una Compañía particular, pero lo que se quiere es tener un sistema propio con vistas a que, en un futuro, dicho sistema pueda ampliarse y ser utilizado para todo el Colegio.

Los tipos posibles de usuarios finales de la Base de datos requerida son:

- el coordinador de la sección
- los maestros
- las alumnas

La herramienta de modelado seleccionada es la propuesta por James Martin. (1)

---

(1) Ver Capitulo 3, pág. 28

El software que se utilizarà es dBASE III. Ya que su capacidad y las ventajas que ofrece son convenientes para el sistema que se desea. Además de ser una Base de datos accesible a las posibilidades de cômputo del Colegio que cuenta con computadoras personales.

2do. paso: Se entra al PROCESO DE MODELADO DE DATOS.

Hay que comenzar determinando los documentos que se utilizarán ya sea como entrada a la Base de datos o como derivados de ella, de acuerdo a la informaciòn que los usuarios desean obtener. El diseño de estos documentos se puede ver en los Anexos 1,2 y 3.

El primer anexo muestra la forma de captura de calificaciones en la que cada maestro pondrà la calificaciòn de sus respectivas alumnas en cada evaluaciòn. Està diseñada de tal manera que una misma forma le sirva durante todo el curso para que pueda ir comparando de manera acumulativa las evaluaciones que durante el se realicen.

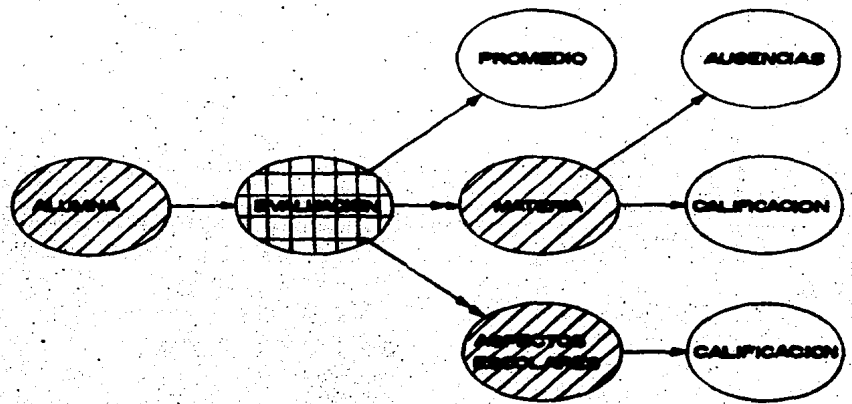
El anexo segundo presenta otra forma de captura de calificaciones pero en la que se evalùan otros aspectos escolares como son la conducta, la integraciòn grupal,el orden y aseo, los retardos y las ausencias. Esta forma serà llenada por el coordinador en cada evaluaciòn.

El tercer anexo es un modelo del reporte de calificaciones que se quiere obtener.

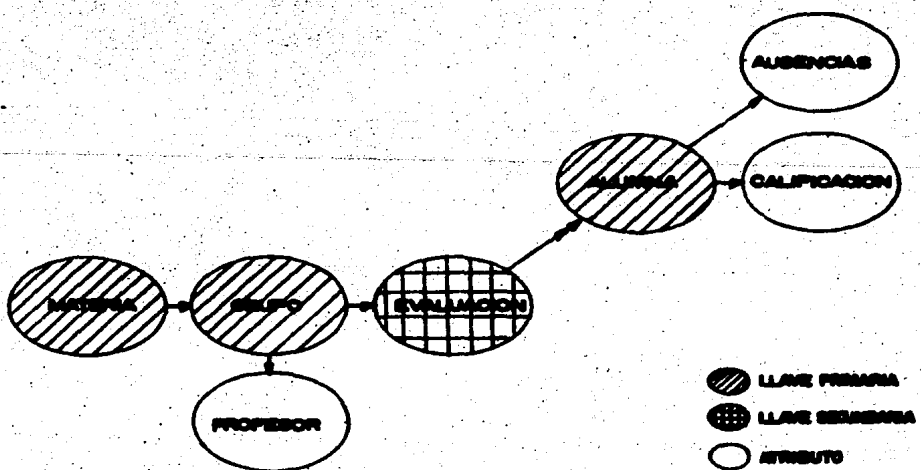
A continuaciòn se forma un DICCIONARIO de DATOS para documentar una descripciòn del significado de cada elemento

llaves primarias, las secundarias y los atributos.

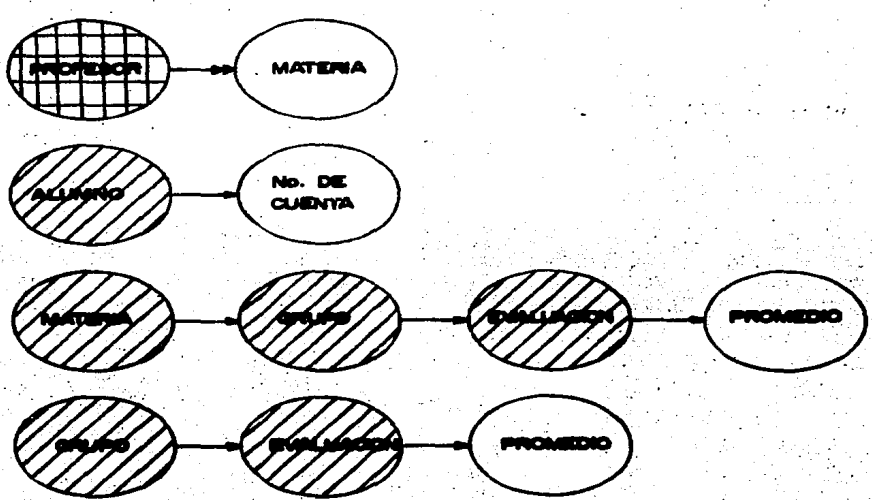
$ Vista de datos de la Alumna:



$ Vista de datos por Materia para cada Profesor:

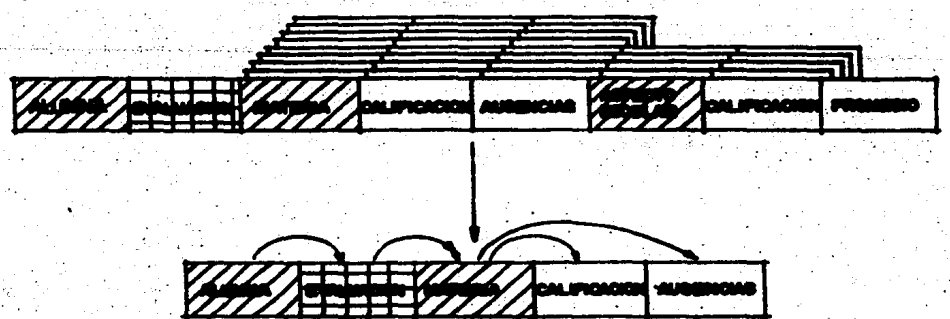


* Vista de datos del coordinador:



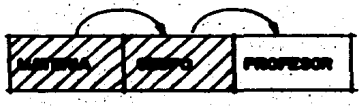
Ahora se NORMALIZA cada vista de los usuarios quedando de la siguiente manera:

* Vista de datos de la Alumna:

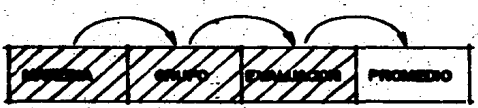


* Vista de datos por Materia para cada Profesor:

NORMALIZANDO



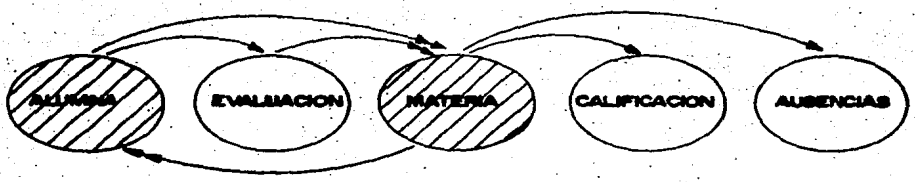
* Vistas de datos del coordinador que, en este caso, se encuentran ya normalizadas:



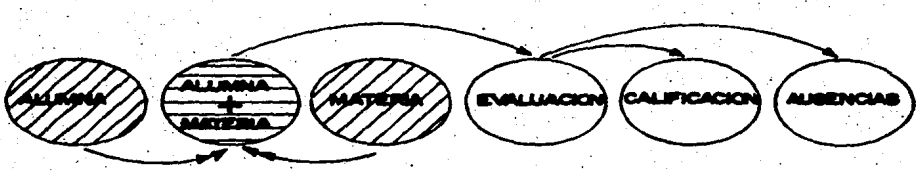
A continuación, se lleva a cabo la SINTESIS de las vistas normalizadas, manejándolas nuevamente como Cartas de Burbuja y tomando en cuenta todas las indicaciones señaladas en el capitulo tercero.

Uno de los pasos de la sintesis canónica indica que para cada asociación entre llaves, hay que agregar la asociación inversa si no está ya en la gràfica. Si resulta una relación M:M y realmente hay posibilidad de que se use en el futuro, entonces habrá que introducir una llave concatenada incorporando los campos llave que estaban ligados.

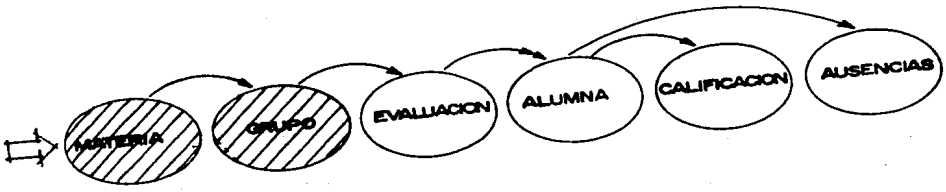
La única vista de datos en las que, al aplicar este paso, resulta una relación M:M entre llaves que seguramente se utilizará en el futuro, es la siguiente:



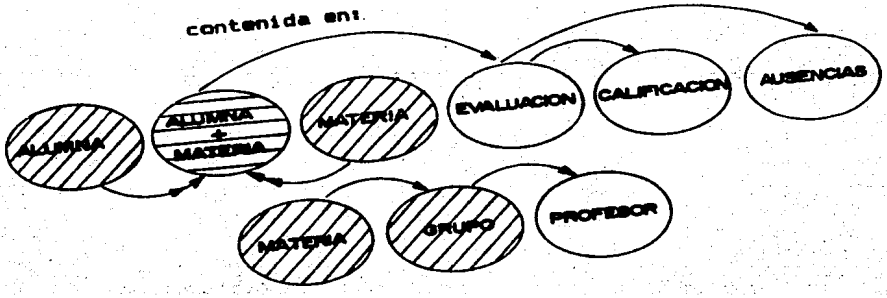
Por lo que, introduciendo una llave concatenada en la que se incorporen los campos ALUMNA y MATERIA, queda de la siguiente forma:



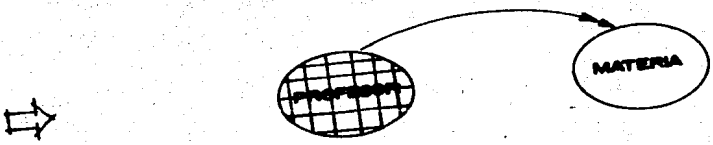
Ahora se examinan las asociaciones existentes y se observa que resultan redundantes las siguientes:



contenida en:



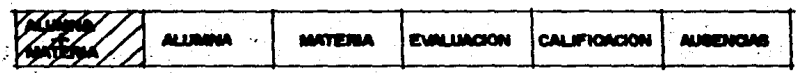
y

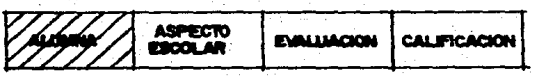


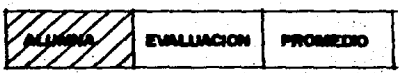
contenida en:

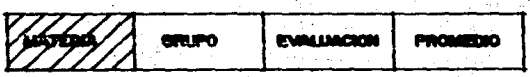
por lo que se eliminan de las demás.

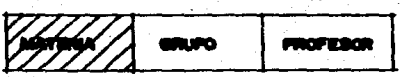
Observando las gráficas que hasta el momento se tienen, se puede ver que no existen atributos aislados ni intersectados, por lo que se redibujan ya los campos arreglados en forma de registros, teniendo cada uno una llave primaria y sus atributos asociados:

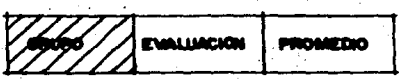












Teniendo presente cómo van a ser utilizados estos datos en el futuro y procurando que el modelo resultante sea lo más estable posible, se llega a lo siguiente:

El campo ALUMNA convendrá manejarlo a manera de clave en la que se incluya la SECCION, el GRADO, el GRUPO y el NUMERO de LISTA.

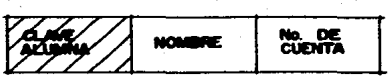
Ejemplo:

La clave -----> B1A15 identifica a la alumna que estudia en la SECCION: Bachillerato
GRADO: 1ero.
GRUPO: A
No. de LISTA: 15

Mediante esta clave se trabajará con las alumnas.

Entonces el nombre de cada una de ellas convendrá incluirlo sólo en el registro:

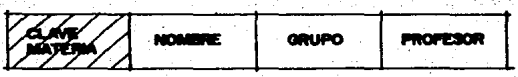
quedando como sigue:



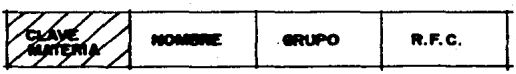
De la misma manera, ya que cada materia tiene una clave determinada, se le utilizará para manejar este campo.

Sólo el nombre de la materia convendrá agregarse al registro:

quedando:

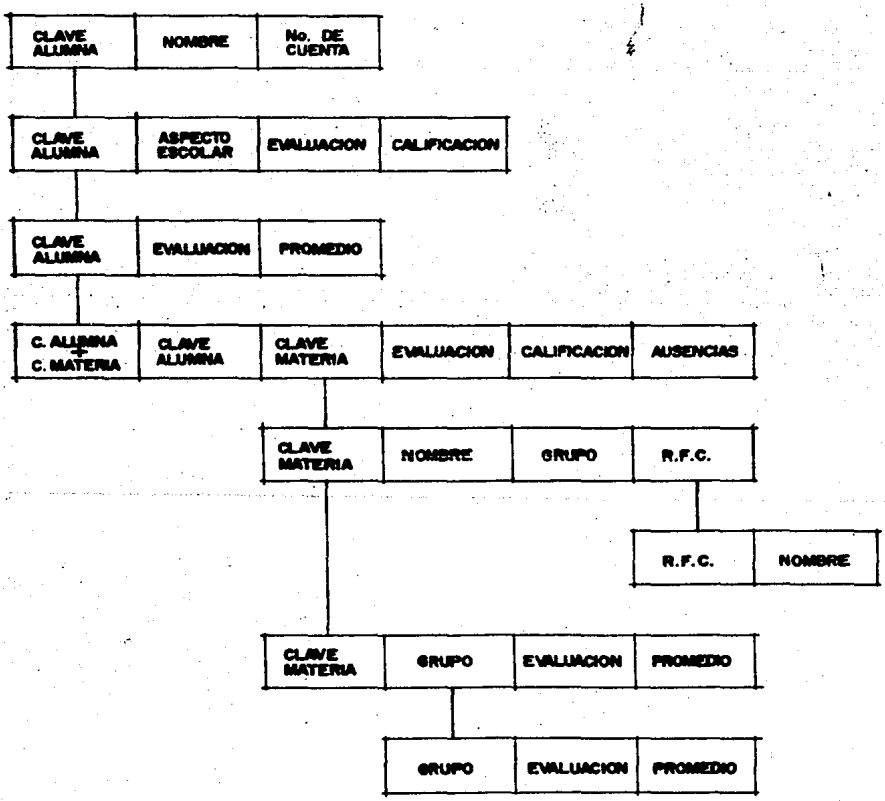


Como última observación, y siguiendo los pasos señalados en este proceso de síntesis, se ve que el campo PROFESOR que en este momento existe sólo como atributo, posiblemente llegue a necesitarse después como llave primaria por lo que convendrá ponerlo así desde ahora. Para esto, podemos identificar a cada profesor mediante su Registro Federal de Causantes:



incluyendo, además, el siguiente tipo de registro:

Una vez realizada la sintesis, con los registros formados se construye el MODELO DE DATOS:



4to. paso: Se inspecciona la SALIDA del modelo de datos.

Creado el modelo, se revisa junto con los usuarios para comprobar que responde a las necesidades que tienen. Se examina cada campo y los lazos que entre ellos han quedado para asegurarse que se les ha agrupado de la mejor manera posible.

Finalmente, se convierte el modelo a un esquema lógico para ser trabajado con el dBASE III creando los siguientes archivos:

ALUMNA
- clave_a
- nombre_a
- num_cta

MATERIA
- clave_m
- nombre_m
- grupo
- rfc

PROFESOR
- rfc
- nombre_p

ALUMCALI
- alummater
- clave_a
- clave_m
- eval
- calif
- ausencias

ALUMASPE
- clave_a
- aspecto
- eval
- calif_a

ALUMPROM
- clave_a
- eval
- promedio

PROMMAT	PROMGPO
. clave_m	. grupo
. grupo	. eval
. eval	. promgpo
. prommat	

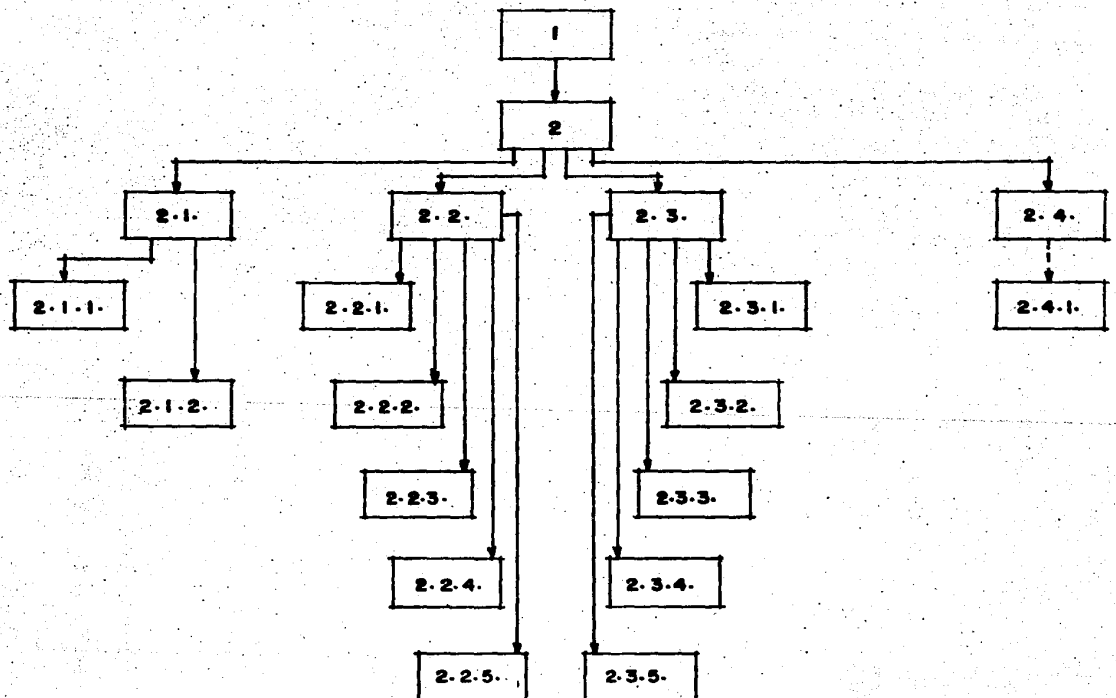
5to. paso: Se hacen las CONSIDERACIONES FISICAS.

Dada la capacidad de la microcomputadora (Denki Corona) que se está utilizando, se vio la necesidad de dividir la información en diferentes discos de la siguiente manera:

Habrá un disco por grupo, cada uno con los 8 archivos que forman la Base da datos.

Una vez creada la Base de datos, se diseña el Sistema valiéndose de la programación que el mismo dBASE III permite.

A continuación se presenta el "diagrama de flujo" que, mediante pantallas, sigue el sistema:


1
2
2·1·
2·2·
2·3·
2·4·
2·1·1·
2·1·2·
2·2·1·
2·2·2·
2·2·3·
2·2·4·
2·2·5·
2·3·1·
2·3·2·
2·3·3·
2·3·4·
2·3·5·
2·4·1·

Descripción del diagrama anterior:

1. Pantalla de presentación del Sistema de Calificaciones. (Anexo 4)

2. Pantalla que muestra el Menú principal. (Anexo 5)

   2.1. Pantalla que muestra el submenú CAPTURA de calificaciones. (Anexo 6)
      2.1.1. Pantalla para la captura de calificaciones.
      2.1.2. Pantallas para la captura de otros aspectos escolares.

   2.2. Pantalla que muestra el submenú del cálculo de PROMEDIOS. (Anexo 7)
      2.2.1. Pantallas para calcular el promedio de la alumna.
      2.2.2. Pantallas para calcular la media del grupo por materia.
      2.2.3. Pantallas para calcular el promedio del grupo.
      2.2.4. Pantallas para calcular la evaluación final del curso.
      2.2.5. Pantallas para calcular el promedio final de la alumna.

   2.3. Pantalla que muestra el submenú de CONSULTAS y MODIFICACIONES. (Anexo 8)
      2.3.1. Pantallas para hacer consultas y modificaciones sobre la alumna.
      2.3.2. Pantallas para hacer consultas sobre la materia y el profesor.
      2.3.3. Pantallas para hacer modificaciones sobre la materia y el profesor.

2.3.4. Pantallas para hacer consultas y modificaciones sobre las calificaciones.
2.3.5. Pantallas para hacer consultas sobre los promedios.

2.4. Pantalla "puente" que indica cómo entrar a la pantalla siguiente. (Anexo 9)
2.4.1. Pantallas para IMPRIMIR las calificaciones. (Anexos 10, 11 y 12)

Como puede verse, el Sistema se presenta al usuario mediante una serie de pantallas que facilitan su manejo.

Este Sistema permite hacer la CAPTURA de calificaciones de cada grupo de alumnas por materia, incluyendo el número de ausencias que durante el bimestre hayan tenido en dicha materia; y capturar también, la calificación que se le da a cada alumna en aspectos tales como: conducta, integración, orden y aseo, retardos y ausencias generales.

Una vez capturadas las calificaciones, se puede hacer el cálculo del PROMEDIO de cada alumna, obtener la media del grupo por materia y calcular el promedio del grupo en cada evaluación.

Al concluir el curso, el Sistema permite también calcular la última evaluación promediando los bimestres con el examen final, ofreciendo la posibilidad de escoger uno de los siguientes criterios:

1ero --> Calcular el promedio considerando la calificación obtenida en los bimestres como el 60% de la nota final y el examen como el 40% restante.

2do. --> Obtener el promedio considerando la calificación de los bimestres como un 50% y el examen final como el otro 50%.

Cuando el Sistema va realizando el cálculo de esta evaluación final, automáticamente guarda una última calificación que resulta de convertir la última nota en otra, siguiendo la escala siguiente:

Sea X = calificación de la evaluación final

si	$X \geq 9$	guarda otra como	10
si	$7.5 \leq X < 9$	guarda otra como	8
si	$6 \leq X < 7.5$	guarda otra como	6
si	$X < 6$	guarda otra como	0

Esta operación se realiza para que, posteriormente, en el proceso de impresión, pueda hacerse una conversión más de la siguiente forma:

Sea Y = calificación extra guardada anteriormente

si	Y =	10	imprimirá	MB
si	Y =	8	imprimirá	B
si	Y =	6	imprimirá	S
si	Y =	0	imprimirá	NA

Y se puede, entonces, obtener un último reporte en el que se tenga la calificación final convertida a la sigla correspondiente.

Una vez obtenido ésto, el Sistema ofrece la posibilidad de calcular el promedio final de la alumna tal y como aparecerá en su certificado, es decir, calcula este último promedio tomando

en cuenta el valor de la sigla obtenida y excluyendo aquellas materias que formalmente no están incluidas dentro del plan de estudios correspondiente.

Otra de las opciones que el Sistema ofrece es la posibilidad de hacer variadas CONSULTAS y MODIFICACIONES sobre la información que se tiene en relación con las alumnas, profesores, materias, calificaciones y promedios obtenidos.

Finalmente, el Sistema permite hacer la impresión del reporte de calificaciones.

Este reporte presenta las calificaciones de cada evaluación en forma acumulativa. Incluye, también, la media del grupo por materia y el promedio grupal como información que le puede ser útil a la alumna.

En la siguiente hoja se presenta un ejemplo:

****

REPORTE DE CALIFICACIONES

INSTITUTO CULTURAL A.C.
MIGUEL ANGEL DE QUEVEDO #1190
MEXICO, D.F.

FECHA: 17/08/87

****

NOMBRE: MARQUEZ ARELLANO MARIA FERNANDA

NO. CUENTA: 86341722-5
EVALUACION: Tercera

SECCION: PREPA
GRUPO: 2o. A
No. de lista: 20

MATERIAS	1o.	2o.	3o.	4o.	5o.	6o.	7o.	8o.	Aus.	M.G.
MATEMATICAS V	9.0	8.0	8.5							8.5
QUIMICA II	7.5	8.0	8.0							7.5
BIOLOGIA IV	8.5	8.5	7.5							8.0
ANAT. FIS. E HIG.	9.0	9.0	10.0							8.5
HIST. MEXICO II	10.0	10.0	9.5							9.0
ETIM. GRECO-LAT.	8.5	9.0	9.0							8.0
ETICA	9.0	8.5	9.5							8.5
ACT. EST. V	10.0	9.5	10.0							9.0
EDUC. FISICA V	10.0	10.0	9.0							9.5
INGLES	10.0	10.0	10.0							8.5
PROMEDIO	9.2	9.1	9.1							
PROMEDIO DEL GRUPO	8.5	8.3	8.7							

Otros aspectos escolares:

CONDUCTA	M.B.	BIEN	BIEN							
INTEGRACION	BIEN	M.B.	M.B.							
ORDEN Y ASEO	BIEN	BIEN	BIEN							
RETARDOS			2							
AUSENCIAS		3								

Observaciones:

Recorte:

MARQUEZ ARELLANO MARIA FERNANDA
PREPA
2o. A

Fecha:

Firma del Padre o Tutor:

Una vez concluido el sistema, se puede comprobar mediante el siguiente cuadro que cada uno de los campos establecidos interviene por lo menos en alguno de los procesos que en el se realizan. La numeración con que se señalan dichos procesos, hacen referencia al diagrama del Sistema descrito anteriormente.

CAMPOS	1.1	2.1	3.1	2.2	3.2	4.2	5.2	1.3	2.3	3.3	4.3	5.3	1.4
CLAVE_AA	XX	XX	XX			XX	XX	XX			XX	XX	XX
CLAVE_M	XX			XX		XX	XX		XX	XX	XX	XX	XX
REF/C.									XX	XX			
NOMBRE_AA								XX					XX
NOMBRE_M									XX	XX			XX
NOMBRE_P									XX	XX			
NUM_CTA								XX					XX
GRUPO				XX	XX				XX	XX		XX	XX
ALUMMATER	XX					XX					XX		XX
EVALUACION	XX	XX	XX	XX	XX	XX	XX				XX	XX	XX
CALIFICACION	XX					XX					XX		XX
AUSENCIAS	XX												XX
ASPECTO		XX									XX		XX
CALIF_A		XX									XX		XX
PROMEDIO			XX				XX					XX	XX
PROMMAT				XX								XX	XX
PROMGPO					XX							XX	XX

PROCESOS

CONCLUSIONES

Me ha parecido interesante y muy provechosa la investigación sobre las Bases de datos que he realizado. Y he podido comprobar de manera práctica la facilidad que éstas ofrecen para el manejo de los datos. Sin duda, el desarrollo continuo que los DBMS van teniendo, está siendo un gran avance en el campo de la informática porque cuenta, cada vez más, con sistemas manejadores de Bases de datos que ofrecen toda una gama de posibilidades para el tratamiento computarizado de cualquier aplicación.

Considero que el dBASE III que en este trabajo utilicé, es un buen manejador de datos, a pesar de sus limitaciones; sólo que hay que tomar en cuenta que para aplicaciones complejas no resultaría conveniente. Pero para el ejemplo que realicé, se me hizo útil. Pienso que, además de la capacidad que ofrece, permite de una manera sencilla y práctica crear archivos y operar con ellos sin dificultad alguna. Me parece una gran ventaja la posibilidad que tiene de trabajar con varios archivos a la vez. En la aplicación que presento, me valgo de ésto para lograr la impresión de las calificaciones ya que en este proceso utilizo información de 7 archivos diferentes.

El inconveniente que encontré, como ya se mencionó algo al hablar de las limitaciones del dBASE III, es que este DBMS realiza muchas operaciones que requieren información en disco y trabajando con discos flexibles el programa todavía se retrasa más.

Por otro lado, la serie de comandos y funciones que el dBASE III ofrece es una gran ayuda para crear programas diversos y muy completos, pero también veo como desventaja el hecho de que el tamaño de estos programas tiene que ser limitado, ya que me vi en la necesidad de dividir varios programas en otros más pe-

queños. A la vez, en ésto pude ver una ventaja del dBASE III, al permitir que la información pueda ser transferida de un programa a otro; pero también considero que dividir un mismo programa en 4 diferentes, como tuve que hacerlo con el que imprime las calificaciones, hace que el tiempo de realización del proceso sea mayor.

Otra observación es que el comando MODIFY REPORT que el dBASE III ofrece, permite crear y modificar informes variados pero utilizando campos de un archivo determinado. Por lo que, dada la situación de que la impresión del reporte de calificaciones tiene un formato especial y se necesita además intercalar varios campos de diferentes archivos a la vez, me pareció mejor hacer la impresión del reporte mediante un programa que vaya consultando cada archivo, realizando las operaciones necesarias y mandando cada línea a imprimir.

Esto, aunado a las limitaciones del dBASE III señaladas anteriormente hace que la impresión de las calificaciones del sistema que propongo llegue a tener un tiempo de duración de hasta 15 minutos por reporte. Por lo que, si en la práctica esta limitación resulta una desventaja considerable, sugiero lo siguiente:

Si se quiere ampliar el sistema a todas las secciones del Colegio incluyendo, además, la posibilidad de obtener estadísticas sobre los resultados de las alumnas, consultas sobre los datos biográficos y familiares de cada una de ellas y agregar además subsistemas tales como el de inscripciones, colegiaturas, etc... Creo que lo más conveniente será, por principio, cambiar la computadora con la que actualmente se trabaja por una que incluya disco duro ya que el manejar tanta información con discos flexibles resultaría poco práctico.

En un segundo momento, si aún así el tiempo que requiere la impresión de reportes es largo, posiblemente convenga pensar en otro DBMS que solucione esta dificultad.

Pero independientemente del DBMS que se utilice, quiero hacer notar la importancia y utilidad que descubrí en el hecho de seguir una metodología para el diseño del modelo de datos con el que se va a trabajar.

En general, la metodología de James Martin aquí seguida, me ha resultado muy sencilla y accesible, sobre todo con la peculiaridad que incluye de manejar los datos mediante Cartas de Burbuja. Aunque considero que ya en la práctica, muchos de los pasos que tan concretamente señala, no es necesario seguirlos uno a uno, o en algunas ocasiones, no en el mismo orden que se indican.

Mi experiencia fue, siguiendo esta metodología, la de sentirme llevada de la mano hasta el modelo de datos final sin dificultad alguna. Llegando a tener desde el principio una estructura clara de cada archivo, las llaves primarias que iba a utilizar y aún las llaves concatenadas que en un futuro llegué a necesitar.

Además, durante el desarrollo del sistema, no encontré ninguna dificultad en cuanto a la manera en que los archivos quedaron creados, al contrario, conforme iba programando me fui dando cuenta del provecho que tuvo crear mi modelo de datos metodológicamente.

Me parece, entonces, que el seguir una metodología para el diseño del modelo de datos a utilizar, resulta muy conveniente para un desarrollo eficaz de cualquier sistema que se realice.

ANEXOS

FORMA DE CAPTURA DE CALIFICACIONES

NOMBRE DEL PROFESOR:

MATERIA:
CLAVE:

GRUPO:

ALUMNA	CALIFICACION								AUSENCIAS							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8

FORMA DE CAPTURA DE OTROS ASPECTOS ESCOLARES

EVALUACION:

GRUPO:

ALUMNA	CONDUCTA	INTEG.	ORDEN ASEO	RETARDOS	AUSENCIAS

REPORTE DE CALIFICACIONES

(Dirección del Colegio) Fecha:

Nombre

No. Cuenta : Sección:
Evaluación : Grupo:
No. de lista:

MATERIAS	1o.	2o.	3o.	4o.	5o.	6o.	7o.	8o.	Aus.	M.G.
Promedio										
Promedio del Grupo										

OTROS ASPECTOS ESCOLARES

Conducta Integración Orden y Aseo Retardos Ausencias										

observaciones

recorte

(Nombre de la alumna)
(Sección)
(Grado y Grupo)
Fecha:

Firma del Padre o Tutor:

SISTEMA DE CALIFICACIONES

Oprime C para continuar:

Menú de opciones

1) CAPTURA
2) PROMEDIOS
3) CONSULTAS Y MODIFICACIONES
4) IMPRESION
5) SALIDA

Opción elegida:

1. Captura de calificaciones
2. Captura de otros aspectos
3. Salida

Opciòn elegida:

1. Promedio de la alumna
2. Media del grupo por materia
3. Promedio grupal
4. Evaluaciòn final
5. Promedio final de la alumna
6. Salida

Opciòn elegida:

1. Consultas y modificaciones sobre la alumna
2. Consultas sobre la materia y el profesor
3. Modificaciones sobre la materia y el profesor
4. Consultas y modificaciones sobre las calificaciones
5. Consultas sobre los promedios
6. Salida

Opciòn elegida:

Para imprimir debes oprimir RETURN para salir al

PUNTITO INDICADOR y teclear DO PROCED4A

```
Dame  SECCION  GRADO  GRUPO:

Dame el No. de Evaluación:

Dame la fecha:                    /  /
```



```
Cuántas calificaciones deseas imprimir:
```


Dame el número de lista:

BIBLIOGRAFIA

- Ashton-Tate, dBASE III - Guía de Referencia, 1985.

- Bartee, Thomas C., Fundamentos de Computadoras digitales, Trad. García Rozo Antonio, México, Ed. Calypso, 1984, pp. 393-410.

- Burch, John G. Jr. y Strater, Felix R. Jr., Sistemas de Información, Trad. Calvet Pérez Ricardo, México, Ed. Limusa, 1985, pp. 564.

- Date. C.J., An Introduction to Database Systems, U.S.A., Addison-Wesley Publishing Company, 1977, pp. 51 - 80.

- Jones, Edward, Using dBASE III, U.S.A., Osborne Mc. Graw Hill, 1985, pp. 220

- Krajewski, Rich, Database Types, BYTE, Vol. 9, No. 11, October 1984, pp. 137 - 142.

- Kruglinski, David, Sistemas de Administración de Bases de datos, España, Osborne Mc. Graw Hill, 1983, pp. 278.

- Kruma, Rob, Understanding and Using dBASE II, U.S.A., Brady Communications Company, Inc., 1984, pp. 304.

- Martin, James, Organización de las Bases de datos, Trad. Di Marco Adolfo, Ed. Prentice-Hall International, 1981, pp. 544.

- Martin, James, Managing the Data-Base Environment, U.S.A., Prentice-Hall, Inc., 1983, pp. 171 ss.

- Schubert, Richard F., Basic Concepts in DataBase Managements Systems, BYTE, Vol. 9, No. 11, Julio 1972.

- Smith, Henry, Database Design: Composing fully normalized tables from A rigorous dependency diagram, Comunications of the ACM, Vol. 28, No. 8, August 1985.