13
2 y

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"ACATLAN"

SISTEMA DE BASE DE DATOS RELACIONAL CON LENGUAJE DE COMANDOS

T E S I S

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A N :

JUAN LUIS VAZQUEZ CAMPOS

ALFONSO RICALDE CURIEL

DIRECTOR DE TESIS:
DR. SERGIO VICTOR CHAPA VERGARA

México, D. F. 1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

"La única persona que jamás comete un error es la que no hace NADA"

Introducción.

De todas las máquinas que ha inventado el hombre, hay una que ha destacado por encima de las demás: la computadora. Durante muchos años se le ha otorgado el título de "cerebro". Sin lugar a dudas, el cerebro humano es mucho más eficiente que cualquier máquina de calcular, por muy sofisticada que ésta sea. En lo único que se ve superado el cerebro por la máquina es en la velocidad de cálculo, probablemente de ahí venga su mistificación.

Lo cierto es que a partir de su nacimiento, la computadora (u ordenador electrónico, como también suele llamarse) ha experimentado una evolución muy rápida en cuanto a sus características físicas y de operación, denominadas hardware y software, respectivamente.

Uno de los principales avances que ha experimentado el ámbito de la información electrónica, es la capacidad de almacenamiento. Esto, aunado a su capacidad de operación, ha incrementado grandemente el poder de aplicación de la informática a, prácticamente, cualquier ámbito de la vida humana.

La informática se ocupa del tratamiento a datos de cualquier índole; ya sea estadística, económica, científica, artística, y un largo etcétera. Es por ello que se está viendo con mayor atención a la información como un recurso vital en las empresas, que a diferencia de hace apenas unas décadas, consideraban primordial el manejo exclusivo de otro tipo de recursos, como el económico y de bienes de consumo. Por supuesto que no se intenta de ninguna forma demeritar la importancia que ellos tienen, pero se está viendo que, dada la diversidad de información que ahora manejan las empresas, es necesario que operen de una forma adecuada sus recursos informativos para hacer frente a las necesidades que se van tornando cada vez más complejas en nuestro mundo moderno. De tal forma que ahora, para manejar eficientemente una gran cantidad de información, es necesaria la ayuda de un medio automatizado, el cual hará la misma labor manual que anteriormente se realizaba en un lapso de tiempo mayor. Aunado a la velocidad, está la confiabilidad de un factor de error reducido, cuya responsabilidad recae en el desarrollo tecnológico que ha conseguido (y continúa consiguiendo) la humanidad.

Pero aunque pareciera que el dominio de grandes cantidades de información fuera competencia exclusiva de las grandes corporaciones e incluso comerciantes de diversa índole y magnitud, en realidad es el hombre, ya sea individualmente o asociado en grupos, quien necesita controlar el cúmulo de información para que pueda servirse de ella. Es incluso en el hogar mismo donde se maneja ya un volúmen considerable de información, referente a los ingresos, gastos, inversiones, planeación de actividades, etc.

Con lo anterior, puede verse que la creación y manejo de los bancos de datos no es ya una utopía, pues parecen ser la mejor opción para satisfacer las necesidades informativas del mundo actual y del futuro. Por ello es importante la realización de un trabajo enfocado a la creación, el manejo y la obtención de información incorporada en conjuntos denominados Bases de Datos (Figura I).

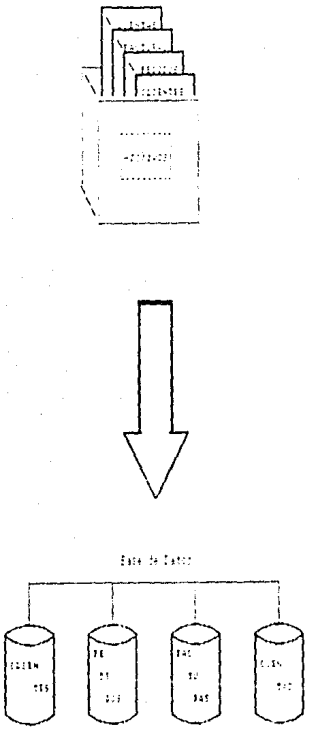


Figura I Conversión de archivos tradicionales a una Base de Datos.

CAPITULO I

"Empieza por hacer lo
necesario, luego lo que es
posible y de pronto te
encontraras haciendo lo
imposible"

San Francisco de Asis

FUNDAMENTOS EN SISTEMAS DE BASE DE DATOS

1.1 Información y Datos.

El campo de las Bases de Datos, dentro de la informática, es amplio y de múltiples aplicaciones, tanto de apoyo a otras áreas, como tema central de desarrollo.

Hay quienes conciben las Bases de Datos como un enorme receptáculo de datos, al que acuden diversos consultadores para obtener la información deseada; hay quienes ven al Sistema de Base de Datos como un conjunto de archivos relacionados entre sí y un grupo de programas para accesar esos datos. Aunque esta última definición es más completa y engloba mejor varios conceptos, puede también definirse una Base de Datos como una colección de datos interrelacionados, almacenados dentro de un conjunto donde se descartan las redundancias innecesarias, tales como la repetición de datos y de subconjuntos de datos; y es concebido para servir de la mejor manera posible a fines predeterminados, además de poderse utilizar en lo futuro con el mínimo mantenimiento posible a su estructura. Los datos almacenados son independientes de los programas de aplicación que los utilizan. La existencia de estos programas es evidentemente indispensable, puesto que sería inútil tener datos eficientemente almacenados pero carentes de una forma adecuada para aprovecharlos en beneficio de metas específicas.

El contar con datos debidamente agrupados, y una estructura a su alrededor bien definida para accesar y controlar el flujo de datos que de ellos se derive, es lo que va conformando aquello que se denomina Sistema de Base de Datos (Figura 1.1).

1.2 Sistemas de Base de datos.

Un Sistema de Base de Datos controla, básicamente, dos aspectos de la Base de Datos: la estructura y el acceso a los datos.

La estructura de una Base de Datos se diseña de acuerdo con el tipo de datos que va a contener y al uso que se les dará. El Sistema de Base de Datos contempla la posibilidad de definir o modificar esa estructura, sin grandes repercusiones en los demás elementos del Sistema. Para ello es indispensable la existencia de un "Diccionario", o Descriptor de Tablas para definir la estructura de los datos almacenados.

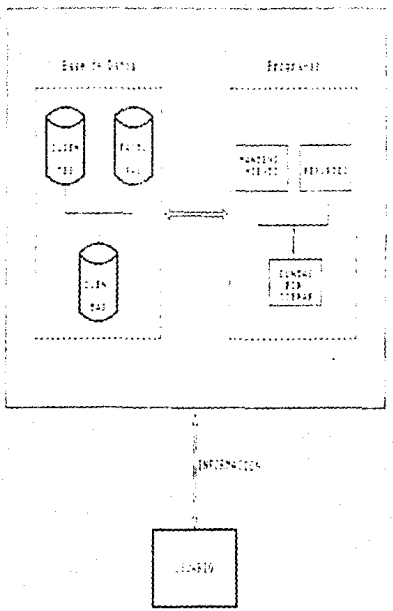


Figura 1.1 Sistema de Base de Datos.

El acceso a una Base de Datos, generalmente, se restringe de acuerdo con el uso y la persona que la accesará. Con ello, se garantiza la integridad de los datos. Un Sistema de Base de Datos posibilita la correcta captación de los datos requeridos, validando la autorización del usuario que hizo el requerimiento, dejando que el usuario se ocupe sólo de las actividades de carácter informativo y el manejo lógico de los datos. El manejo físico de los datos es, en buena medida, responsabilidad del Sistema de Base de Datos.

Los Sistemas de Base de Datos, como todo elemento de la informática, está constituido de cuatro componentes fundamentales. Aunque obviamente no son los únicos, sí constituyen los componentes generales básicos, a través de los cuales puede comprenderse la estructura y funcionamiento general de un Sistema de Base de Datos. Estos componentes son:

- Datos
- Hardware
- Software
- Usuarios

DATOS.- El elemento esencial de un Sistema de Base de Datos son los datos, pero no los datos aislados, sino los que se encuentran lógica y ordenadamente agrupados. Esta agrupación lógica y ordenada es una Base de Datos. Una Base de Datos es una agrupación de datos tanto integrada como compartida. Por integrada, se entiende que la Base de Datos puede verse como una unificación de varios archivos de datos, donde se elimina parcial o totalmente la redundancia entre los mismos. Por compartida se entiende que, partes individuales de los datos puedan consultarse por distintos usuarios, teniendo así acceso a la misma parte de la Base de Datos para propósitos diferentes (Figura 1.2).

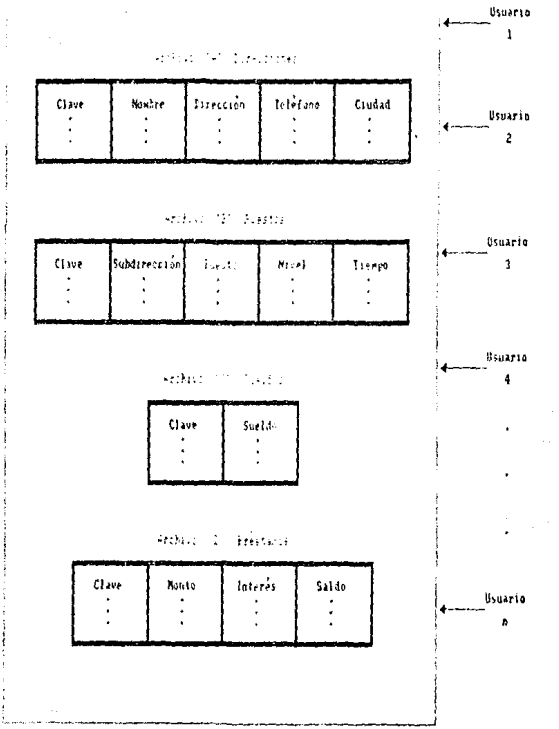


Figura 1.2 Ejemplo de una Base de Datos

HARDWARE.- El Hardware es un elemento fundamental para el desarrollo de un Sistema de Base de Datos, puesto que determina en buena medida el grado de sofisticación que el sistema pueda alcanzar. Consistente en los instrumentos físicos y aparatos electrónicos indispensables para dar inicio a todos los procesos automatizados que explotan la información electrónica, puede verse el hardware como el conjunto que consiste de memorias (principal y secundarias), monitores, teclados, discos duros, diskettes, impresoras, etc. Debido al auge de las computadoras personales y el creciente desarrollo de sus capacidades, es factible disminuir la diferenciación que tradicionalmente se señala, en cuanto a lo que es posible desarrollar en las computadoras grandes, algunas veces denominadas "mainframes", y lo que es posible realizar en las microcomputadoras. Uno de los principales factores es el espacio que ocupa, en código objeto, todo un sistema de base de datos. Esto es determinante, sobre todo, al momento de querer optimizar el manejo del sistema mediante técnicas especiales, las cuales consumen por sí solas una considerable cantidad de recursos físicos. Elementos tales como un administrador de la base de datos, técnicas de manejo de memoria, no forman parte de un sistema de base de datos para microcomputadoras debido a lo antes señalado. A pesar de lo anterior, el futuro de los Sistemas de Base de Datos es alentador; se está observando ya un prometedor acoplamiento entre macros y microcomputadoras, si bien, las micros no suplen en capacidad a las grandes máquinas, constituyen un valioso soporte operacional de apoyo a los Sistemas de Bases de Datos, e incluso ofrecen un buen medio para desarrollarlos, con sus pequeños pero flexibles y potentes dispositivos (Figura 1.3).

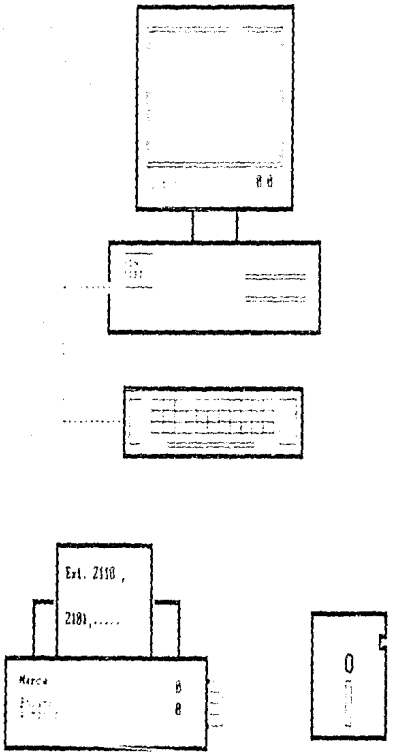


Figura 1.3 Hardware

SOFTWARE.- El software es el nivel intermedio entre quienes utilizan el Sistema de Base de Datos y la información propiamente almacenada. En los equipos grandes, este nivel se engloba en lo que se denomina un Sistema Administrador de Base de Datos (DBMS son las iniciales de su nombre en inglés: Data Base Management System), el cual controla todas las transacciones desde, hacia y dentro del Sistema de Base de Datos. En las microcomputadoras personales, este administrador no se encuentra presente debido sobre todo al gran espacio, en memoria, que ocupa: sin embargo, cada usuario del Sistema de Base de Datos funge como administrador del mismo, lo cual le confiere toda la responsabilidad sobre la integridad de los datos. Por otra parte, en las computadoras personales se brinda al usuario un acceso más directo a las herramientas que conforman el software del sistema. El lenguaje de consultas y manejo de datos es bastante sencillo, para utilizarlo, el usuario formula la acción que desea realizar de un modo simple: llenando una forma predefinida, porque así se disminuye en lo posible el uso de espacio en memoria. Un único usuario ocupa el Sistema de Base de Datos cada vez, debido a que las computadoras personales no ofrecen la posibilidad de compartir usuarios a menos que se conecten en red, para lo cual un sistema de base de datos para micros ya no es viable, sino más bien un sistema para macros. Por otra parte, el mismo usuario es el encargado de la integridad de los datos, realizando los debidos respaldos en forma regular y administrando por sí mismo el uso de sus recursos. Por último, dado que las computadoras personales limitan su uso por la capacidad operativa que tienen, el modelo de datos usado en los Sistemas de Base de Datos para microcomputadoras es el relacional, porque es el más accesible para desarrollar software, por su simplicidad operacional; el software aquí ocupa menos espacio de memoria y es más cercano a la concepción sobre relaciones de datos que se forma el usuario (Figura 1.4).

USUARIOS.- Para poder utilizar una Base de Datos y mantener la integridad de la información almacenada, se necesita estar autorizado, más aun; se necesita tener un nivel de autorización que defina a cuales datos se puede accesar, a cuales modificar, etc. Considerando esta cuestión, los Sistemas de Base de Datos deben contemplar por lo menos tres tipos de usuario:

- Administrador de la Base de Datos.

- Programador de Aplicaciones.

- Usuario Final.

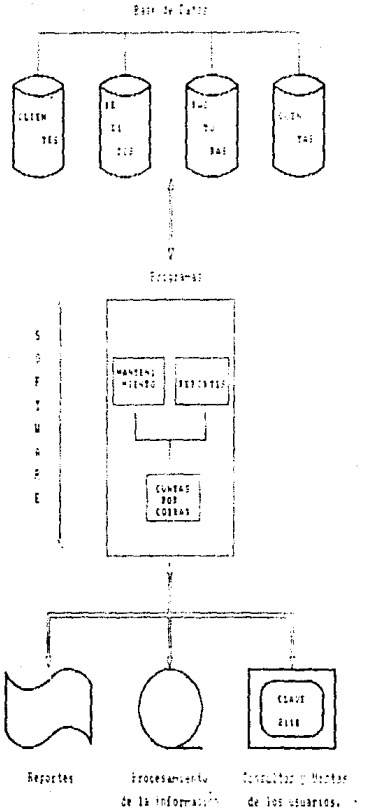


Figura 1.4 Software

Cabe mencionar que estos tres tipos de usuario engloban toda la gama de personas que pueden tener acceso a una Base de Datos.

El Programador de Aplicaciones es el encargado de realizar programas que operen sobre la base de datos para manejar la información de todas las formas requeridas: creación, modificación, y obtención de los datos contenidos en ella.

El Usuario Final es la persona que utiliza las facilidades que brinda el Sistema de Base de Datos, para satisfacer las necesidades informativas que requieren del uso de la tecnología informática actual. Para ello ocupa los programas creados por el programador de aplicaciones a través de un lenguaje de consulta, proporcionado como parte integral del sistema, o desarrollado por el mismo programador.

El Administrador de la Base de Datos, como su nombre lo indica, es la persona responsable de administrar el funcionamiento correcto de la Base de Datos dentro del Sistema. Sus funciones, entre otras, son:

- Definición y creación del esquema de datos de la estructura de almacenamiento y de los niveles de acceso. Es decir, diseñar y crear el esquema lógico de la Base de Datos, la estructura física y autorizaciones de acceso apropiadas para el aprovechamiento óptimo de la Base. Esto se realiza mediante el registro de las definiciones pertinentes, que se incorporan en un conjunto de tablas dentro del Diccionario de Datos.

- Modificación del esquema y de la organización física. Estas acciones son poco frecuentes pero importantes para afrontar los cambios de requerimientos que surgen.

- Concesión de autorización para acceso a los datos. Definiendo tipos de acceso que pueden realizar los diversos usuarios.

En forma esquemática, puede interpretarse el papel de cada tipo de usuario de un Sistema de Base de Datos como se muestra en la figura 1.5.

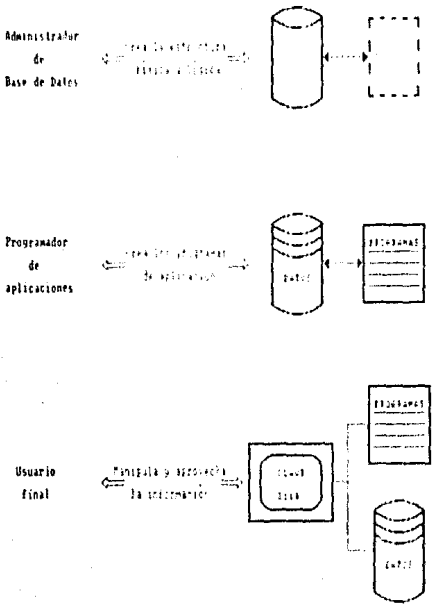


Figura 1.5 Tipos de usuarios.

1.3 Estructura General de un Sistema de Base de Datos.

La estructura general de un Sistema de Base de Datos define sus características teórico-prácticas. Esta estructura es simple, y su presencia ayuda a esclarecer la concepción que se tiene del Sistema.

En un Sistema de Base de Datos para microcomputadoras, esta estructura está básicamente determinada por la que constituye la base de datos a utilizar, debido sobre todo al aspecto personal de su utilización. En las computadoras medianas y grandes, esta estructura abarca los diversos tipos de acceso y manejo de los datos, contemplando así aspectos tales como el tipo de usuario, autorización, nivel de acceso permitido, etc., sin embargo, es en esencia la misma para ambas. Esta estructura se divide en tres niveles generales:

- Nivel Interno.

- Nivel Externo.

- Nivel Conceptual.

El nivel Interno es el más cercano al almacenamiento físico de los datos, el cual atiende a las características de almacenamiento en los dispositivos físicos.

El nivel Externo corresponde al manejo del Sistema por parte del usuario final, determinándose por la manera en que cada usuario utiliza los datos.

El nivel conceptual es el nivel de mediación entre los dos anteriores. Surge a partir de la necesidad de definir un modelo estructural de una base de datos, y es una representación abstracta que permite dominar eficazmente la utilización de los datos (Figura 1.6).

1.4 Organización de un Sistema de Base de Datos.

Un Sistema de Base de Datos se puede dividir en varios módulos funcionales, cada uno de ellos se encarga de una tarea específica del sistema. Algunas de las funciones del Sistema de Base de Datos pueden ser realizadas por el Sistema Operativo. Así, el diseño de la Base de Datos puede incluir la consideración de interfase entre el Sistema de Base de Datos y el Sistema Operativo.

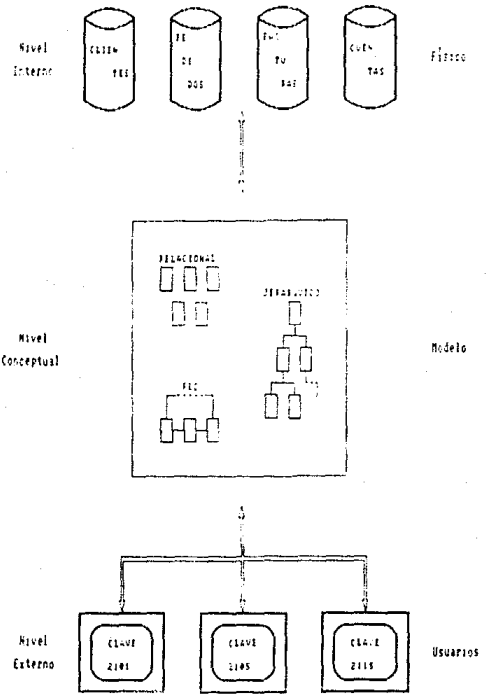


Figura 1.6 Niveles de utilización de un Sistema de Base de Datos.

De entre los componentes funcionales que constituyen un Sistema de Base de Datos, se cuentan:

- El manejador de archivos.- El cual es el encargado de asignar espacio en el disco y las estructuras de datos que se van a emplear para representar la información almacenada.
- El manejador de base de datos.- Que constituye la interfase entre los datos almacenados en la base de datos y los programas de aplicación o las consultas que se hacen al sistema.
- El procesador de consultas.- Que traduce las proposiciones en lenguaje de consulta a instrucciones de bajo nivel, el cual puede entender el manejador de la base de datos. Así también, el procesador de consultas puede convertir la solicitud del usuario a una forma equivalente que sea más eficiente, encontrando una estrategia adecuada para ejecutar la consulta.
- El precompilador del lenguaje de definición de datos.- Que convierte las proposiciones de un programa de aplicación en una serie de llamadas a procedimientos. El precompilador debe interactuar con el procesador de consultas para generar el código optimizado adecuado.
- El compilador del lenguaje de definición de datos.- Que convierte las proposiciones dadas en un conjunto de tablas que constituyen el cuerpo de la base de datos. Dichas tablas se almacenan después en el Diccionario (o Descriptor) de datos.

Además de los elementos antes mencionados, se necesita contar con varias estructuras de datos a fin de constituir el sistema físico:

- Archivos de datos.- Que guardan la base de datos.
- Diccionario de datos.- Que almacena la información relativa a la estructura de la base de datos.
- Indices.- que permiten el acceso rápido a elementos de información, tomados de la base.

Los elementos mencionados anteriormente se esquematizan en la Figura 1.7.

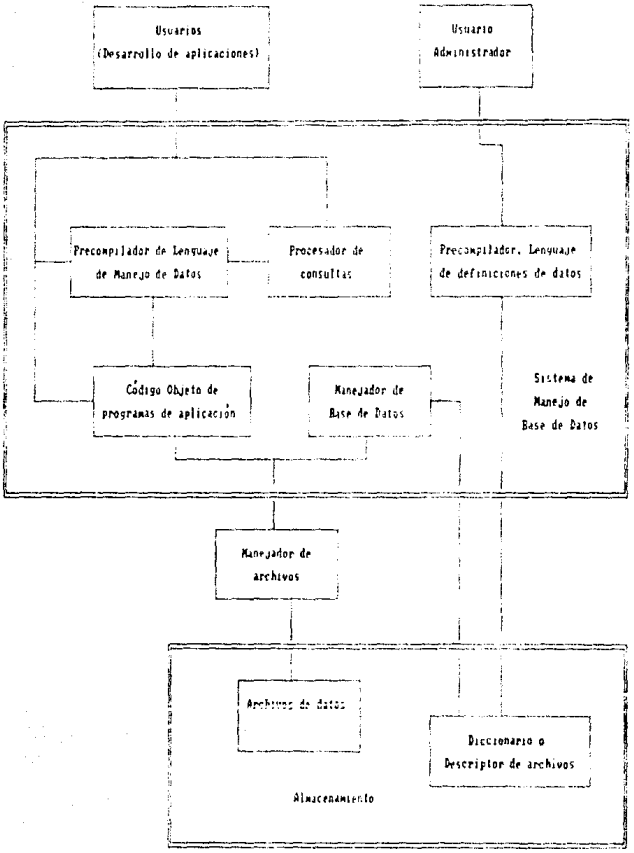


Figura 1.7 Organización de un Sistema de Base de Datos.

CAPITULO II

"Cuando una persona se encierra
en la frontera de lo seguro y
lo cómodo, no tiene desarrollo
ni inspiración; pero si ha
comprobado que con cierta
experiencia se desenvuelve
bien.
¿Por qué no lanzarse
a un nuevo reto?"

MODELOS CONCEPTUALES DE BASE DE DATOS

Introducción.

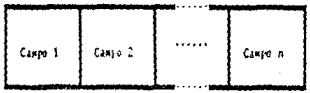
Como se vió en el capitulo anterior, la estructura general de un Sistema de Base de Datos puede dividirse en tres niveles generales, que son el Interno, el Externo y el Conceptual. Este último determina varios aspectos valiosos para el diseno, elaboración y mantenimiento de la información almacenada en la Base de Datos. El manejo interno de datos al que se hace referencia, tiene como punto de partida algún modelo de datos. Existen tres modelos generales de Bases de Datos que se han desarrollado con el objeto de formar los principios adecuados para el avance de la teoría de Base de Datos. Estos tres modelos son :

- El Modelo Jerárquico
- El Modelo de Red
- El Modelo Relacional

A continuación se darán algunas caracteristicas de estos modelos.

2.1 Modelo Jerárquico

Este modelo representa algunas situaciones reales de una manera más conveniente, que se apega a un concepto de jerarquía tipica (organigrama). Donde hay un jefe y un subordinado, el cual a su vez tiene a su cargo otro u otros subordinados, de esta forma se van creando los niveles de flujo de la información ya que un subordinado tiene que reportar a su jefe inmediato, y así sucesivamente. De esta manera el modelo tiene la caracteristica de que la información fluye a través de cada jefe y subordinado, que son puntos de envio y recepción de información, estos se denominan nodos. En este modelo, se toma al registro como el conjunto básico de información o nodo de la estructura. Puede considerarse al flujo de la información como la liga entre dos registros, exclusivamente (Figura 2.1).



Registro

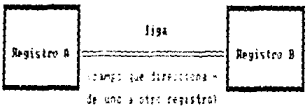


Figura 2.1 Representación gráfica de una liga y un registro.

Para cada archivo que constituye la Base de Datos, se tiene un conjunto de registros agrupados a través de ligas. Estos registros son diferentes entre sí, y están asociados bajo el criterio de niveles de jerarquía, de ahí que se defina con este nombre al presente modelo. Los nodos pueden relacionarse a través de los términos "padre" e "hijo".

Los últimos nodos de la estructura correspondiente se denominan nodos terminales, y al nodo inicial se define como nodo raíz. Este último es la cabeza y punto de inicio al acceso de cualquier nodo subordinado a él. Figura 2.2

En este modelo, la información de cada archivo se representa a través de una sencilla estructura de árbol con raíz.

De esta forma, el esquema de una Base de Datos jerárquica se puede ver como un conjunto de diagramas de árbol. La raíz de cada árbol es un nodo o registro de trabajo para el archivo de datos correspondiente. Figura 2.3

Esta representación tiene dos restricciones básicas:

1) la gráfica no puede contener ciclos.

2) las relaciones entre padre e hijo pueden ser una a una, o una a muchas.

El manejo de esta estructura es siempre a través de su secuencia jerárquica, definida inicialmente. La carga completa requiere que se determine el orden de los datos, antes de ser almacenados; la carga de un registro dado necesita del conocimiento de la ruta que determinará la posición exacta donde se anadirá la información.

En general, las operaciones de actualización son más complicadas que las de recuperación, porque además de la información debe considerarse, en su caso, la actualización de los apuntadores correspondientes, e incluso de los registros sujetos al modificado (este último caso está claramente representado en el caso del borrado de registros no terminales).

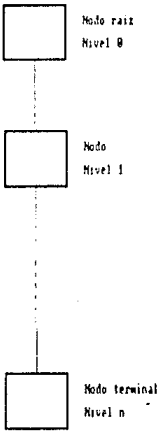


Figura 2.2 Clasificación de nodos

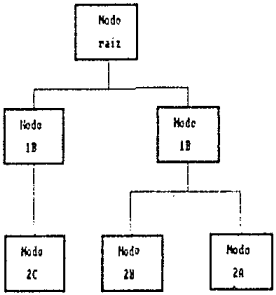


Figura 2.3 Estructura Jerárquica en forma de árbol.

2.2 Modelo de Red

De la misma forma que el modelo jerárquico, el presente modelo utiliza un flujo de información (liga) y un punto de transmisión o recepción de datos (nodo); pero a diferencia del anterior, el concepto de nodo padre y nodo terminal desaparecen debido a que el flujo de la información puede empezar desde cualquier nodo de la estructura y terminar en cualquier otro, incluso el mismo del cual se inicio el flujo, es decir, en vez de jerarquía se tiene una agrupación arbitraria de nodos.

En este modelo, la representación gráfica más simple puede ser como la que se muestra en la figura 2.4.

Esta se puede extender de diversas formas, como la que se muestra en la figura 2.5, donde se tienen tres tipos de registro, para los cuales existen 2, 2 y 3 ocurrencias respectivamente; en este sentido, una red es una estructura más general que una jerarquía. El registro de trabajo para este modelo puede asumir cualquiera de las ocurrencias de 1A a 3C. Pero siempre respetando los caminos determinados a través de los apuntadores.

El manejo de la información contenida en este modelo es más complicado debido a que su organización es arbitraria. Se puede desarrollar un proceso que, partiendo de un nodo específico de la gráfica, se tenga acceso a cualquier punto de la misma en una cantidad mínima de pasos; pero esto es un procedimiento complejo ya que debe evaluar todas las rutas posibles, las cuales son variables en cantidad y forma dependiendo de la red que se utilice.

2.3 Modelo Relacional

En este modelo, a menudo se usan indistintamente los conceptos de "relación", "archivo" o "tabla"; a los renglones, o los registros de los archivos se les denominan tuplas; las columnas o campos se llaman atributos; las tablas se asemejan mucho a un archivo secuencial convencional, donde los renglones de la tabla corresponderían a los registros del archivo, y las columnas a los campos del registro. Figura 2.6

Cada una de estas tablas, en realidad, es un caso especial de la construcción conocida en Matemáticas como relación. La aproximación relacional a los datos se fundamenta en el hecho de que los archivos que obedecen ciertas restricciones se pueden considerar relaciones matemáticas y, por tanto, la teoría elemental de las relaciones se puede aplicar a varios problemas prácticos.

Figura 2.4 Estructura elemental de una Red.

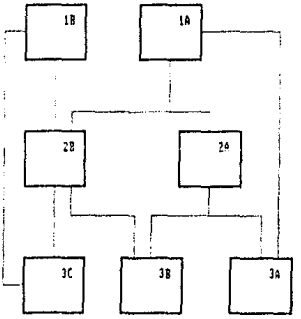


Figura 2.5 Estructura compleja de una Red.

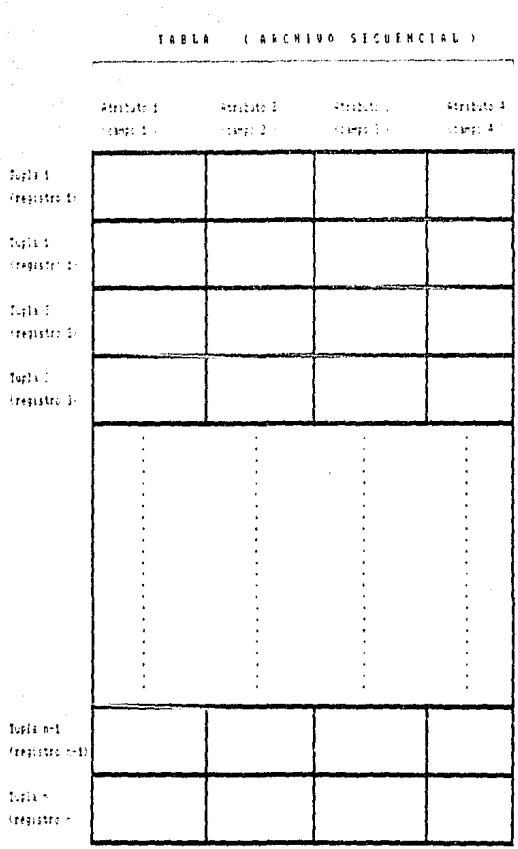


Figura 2.6 Modelo de Archivo Relacional.

Las asociaciones entre tuplas (o renglones), se representan únicamente por valores de datos en columnas sacadas de un dominio común. Un dominio es un depósito de valores del cual se obtienen los que aparecen en una columna específica. De hecho, es una característica del modelo relacional que toda la información de la Base de Datos (tanto las entidades como las asociaciones) se representa de una sola manera uniforme, a saber, en forma de tablas.

Es importante apreciar la diferencia entre el dominio, por una parte y los atributos (o columnas) que se obtienen de ese dominio, por otro. Un atributo representa el uso de un dominio dentro de una relación; para acentuar la distinción, se pueden asignar nombres a los atributos que sean distintos de los dominios subyacentes.

Así, la manipulación de la información en el presente modelo se torna menos complicada debido a que los algoritmos de recuperación y mantenimiento de datos se enfocan a los archivos secuenciales tradicionales que podría tener cualquier lenguaje de alto nivel.

CAPITULO III

"El objetivo de nuestra vida no
es superar a los demás, sino
superarnos a nosotros mismos"

S. Johnson

MODELOS DE ORGANIZACION FISICA

Introducción.

Uno de los principales factores para satisfacer las necesidades del usuario de un Sistema de Base de Datos es su rendimiento. Si el tiempo de respuesta a una consulta es largo, disminuirá el valor del sistema. El rendimiento de un Sistema de Base de Datos depende en parte de la eficiencia de las estructuras que se empleen para representar los datos, y de que el sistema pueda operar en forma eficiente con esas estructuras. Una de esas estructuras de datos son los archivos, los cuales se utilizan para representar y operar los datos. Por ello, el crear una herramienta que manipule en forma eficaz a los archivos, da como resultado el buen rendimiento de un Sistema de Base de Datos.

Por consiguiente, en el presente capitulo se definirá una herramienta denominada **DESCRIPTOR DE ARCHIVOS**, la cual permite de una manera sencilla el control y mantenimiento de los archivos. También se señalará el rendimiento que pueda tener al usarla, en los distintos modelos para Bases de Datos. Figura 3.1

3.1 Descriptor de archivos.

El modelo de organización física de un Sistema de Base de Datos es parte fundamental para el buen funcionamiento de la misma. Entre más sencillo es el comportamiento en memoria de las operaciones de entrada-salida, más eficiente resultará la recuperación de los datos. Para facilitar dichas operaciones, se han definido ciertas estructuras de datos, tales como los **archivos**.

Un archivo está diseñado para procesar registros de manera eficiente. Para la recuperación de los registros de un modo práctico, existen diferentes formas de acceso, la principal y de la cual pueden derivarse otras, es la secuencial. Así, los registros están ordenados en forma subsecuente (uno tras otro), y se permite un manejo sencillo.

Para reducir al mínimo el número de accesos a memoria durante el procesamiento de un archivo secuencial, los registros deberían grabarse físicamente lo más cerca posible uno de otro (fig. 3.2).

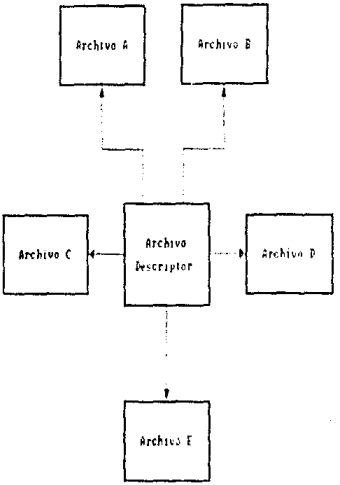


Figura 3.1 Organización de los archivos a través de un archivo de Descriptores.

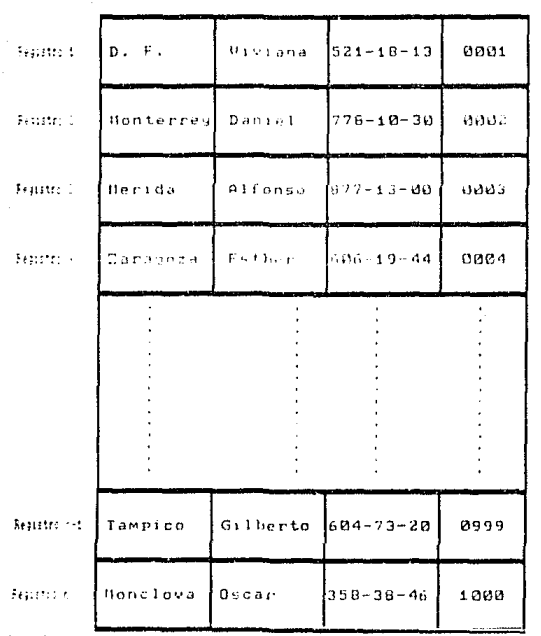


Figura 3.2 Archivo Secuencial.

Por otro lado, la forma en que se agrupan los datos en un registro afecta también el tiempo de recuperación y proceso de los mismos. A veces no todos los registros contienen la misma cantidad de datos, de ahí que los registros puedan manejarse con una longitud fija o variable.

Cuando se maneja una longitud fija, el tiempo de recuperación y procesamiento del registro es eficiente y corto, pero en cuanto a espacio en memoria se refiere, algunas veces habrá desperdicio, pues algunos registros no estarán ocupando todo el espacio que tienen definido (fig. 3.3).

Por otra parte, el uso de registros con longitud variable evita el desperdicio de memoria, pero a cambio implica un procesamiento más lento, debido a que en ocasiones los registros son muy largos y requieren se les procese parcialmente; en otras ocasiones los registros son muy cortos, por lo que hay que tener bien determinado donde empieza y donde termina cada uno (fig. 3.4).

Por tanto, el buen diseño de registros conlleva a un mejor uso de los archivos, permitiendo la disminución del tiempo de entrada-salida que requiere el dispositivo físico para leer o escribir un registro y el tiempo de proceso a los registros, en memoria.

Se ha considerado hasta ahora la forma en que se encuentran almacenados los datos, que es a través de archivos y registros, pero no se ha considerado la forma en que podemos referirnos a ellos, cómo están constituidos, cómo se pueden accesar para darles mantenimiento, etc.

Para poder hacer un buen uso de los archivos, es necesario construir una herramienta denominada **descriptor de archivos**, la cual contendrá toda la información acerca de los mismos.

Un descriptor de archivos puede definirse como un registro de longitud variable, el cual está asociado a los registros de datos que se encuentran en un archivo. La asociación se considera a través de las descripciones del archivo de datos y los apuntadores, que determinan en dónde se encuentran los datos dentro de los registros del archivo en cuestión. Los nombres de los atributos se tienen en el descriptor y se asocian con los valores que se encuentran en los registros del archivo que se está describiendo.

Denominaremos una función VAL, como la función que realiza el acceso a los archivos en forma física y obtiene de una manera precisa los valores de los atributos que se describen en los descriptores de archivos.

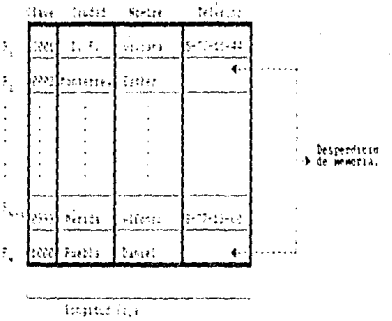


Figura 3.3 Archivo de longitud fija.

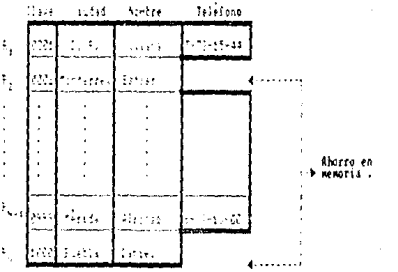


Figura 3.4 Archivo de longitud variable.

La recuperación de los datos que se encuentran en cada uno de los registros del archivo de datos se efectúa con la función VAL, obteniendo así algún valor Vi de un conjunto de valores que se encuentran en el registro

$$(V1,V2,\ldots,Vi,\ldots,Vn)$$

los valores se asocian con su descriptor, en donde se encuentran registrados los nombres de cada uno de los valores (denotados por d1,d2,...,dj...,dn)

A continuación se da una lista con los elementos -dj- que, generalmente, contiene un descriptor de archivos representado en la figura 3.5 :

1) Nombre del archivo.

2) Número de registros.

3) Número de bytes por registro.

4) Tipo de archivo.

5) Tamaño de los campos del archivo de descriptores.

6) Número de campos.

7) Nombre del campo.

8) Apuntador inicial al dato del campo.

9) Longitud en bytes del dato (tamaño del campo).

10) Tipo de dato:

    a) Alfanumérico A

    b) Numérico entero I

    b) Numérico flotante F

    c) Descripción explícita E

    (El tipo de descripción explícita significa que hace referencia a una tabla).

NOMBRE DEL ARCHIVO	NUMERO DE REGISTROS	NUMERO DE BYTES EN REGISTRO	TIPO DE ARCHIVO	FORMATO DE CAMPOS DEL REGISTRO	NUMERO DE ELEMENTOS	NOMBRE DEL ELEMENTO	VALOR INICIAL DE ELEMENTO	VALOR MAXIMO FINAL DE ELEMENTO	TIPO DE VARIABLE DE ELEMENTO

Descriptor del archivo Clientes

Clientes.dat	1 2 5	3 5	S	:	1 2	Nombre del cliente	1	3 2	A

Descriptor del archivo Proveedor

Prove.dat	1 2 0	5 0	S	:	1 1	Nombre del proveedor	1	2 0	A

Figura 3.5 Modelo y ejemplos del descriptor de archivos.

Ahora bien, cada elemento dj del descriptor será parte del argumento que la función VAL, éste le servirá para obtener el Vi dentro de algún registro del archivo de datos. Hacer VAL (d1, d2, ..., dn) = (V1, V2, ..., Vn). Para tomar físicamente ese valor, la función VAL toma dos apuntadores, los cuales delimitan la posición real del valor dentro del registro físico de datos (el segundo apuntador se determina a través de la longitud del elemento).

Así, cada uno de los elementos dj de los descriptores, puede considerarse con varias especificaciones asociadas al valor de algún registro contenido en un archivo de datos.

3.2 Modelos de organización física de datos.

Como se vió anteriormente, el Descriptor de Archivos es una herramienta simple pero de gran utilidad, que permite el acceso a cualquier archivo descrito en él de una manera eficiente. También es posible agrupar una colección de descriptores que contengan la descripción de acceso a diferentes archivos. El agrupamiento de estos descriptores dá como consecuencia la configuración de un archivo más, denominado ARCHIVO DE DESCRIPTORES.

Un archivo de descriptores está constituido de una secuencia de registros, en donde cada uno de estos es un descriptor para un archivo de datos (Figura 3.6); por medio de este concepto se puede elaborar un esquema de consulta a diferentes archivos que se encuentran descritos en él.

La utilidad de esta herramienta se aprecia mejor en el mantenimiento y control de archivos de un Sistema de Base de Datos, como se ejemplifica a continuación:

1) En una Base de Datos se requiere saber si el archivo de "CLIENTES" existe (Figura 3.7).

   - Se busca en el Archivo de descriptores el registro descriptor del archivo que se desea encontrar, en forma secuencial. Esta búsqueda se hace sobre el campo que contiene el nombre del archivo, Buscando en cada registro hasta encontrarlo (si es que existe).

2) Dar de alta el archivo "PROVEEDOR" (Figura 3.8).

   - Se busca el nombre del archivo en el archivo de descriptores, para determinar un intento de duplicación.

   - Se dan las características (nombre, forma de acceso, longitud de registro, número de registros, campos y sus nombres, etc.) que tendrá el archivo en un registro descriptor.

   - Se agrega el nuevo registro descriptor al archivo de descriptores.

3) Dar de baja el archivo "CLIENTES" (Figura 3.9).

   - Se localiza el registro descriptor del archivo que se desea dar de baja, en el archivo de descriptores (si es que existe).

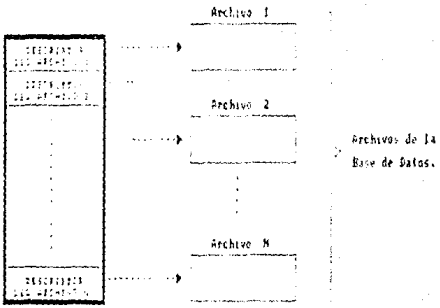


Figura 3.6 Archivo de Descriptores

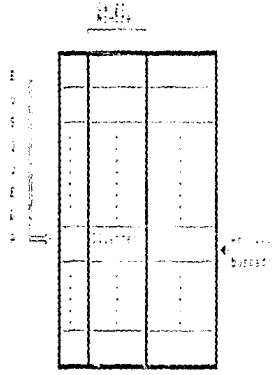


Figura 3.7 Búsqueda de archivo Clientes

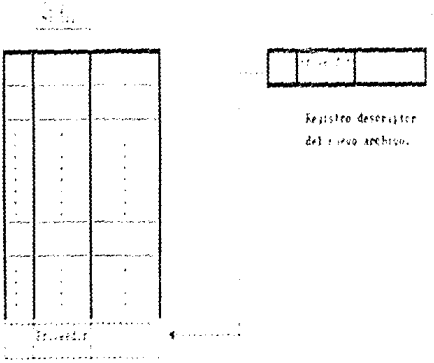


Figura 3.2 Inserción del archivo Proveedor

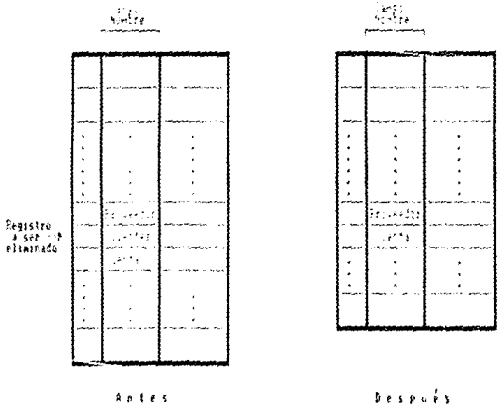


Figura 3.3 Eliminación del archivo Clientes.

- Una vez encontrado el registro descriptor, se elimina del archivo de descriptores. Esta operación implica el borrado físico del archivo.

4) Renombrar el archivo "PROVEEDOR" por "CLIENTES" (Figura 3.10).

- Se localiza el registro descriptor del archivo "PROVEEDOR" en el archivo de descriptores.

- Una vez localizado el registro descriptor, se modifica el campo donde se encuentra el nombre, sustituyéndolo por el nuevo nombre proporcionado.

- Hecha la modificación, se reescribe el registro descriptor en el archivo de descriptores.

5) Del archivo "CLIENTES", cambiar el nombre del campo "ciudad" por el de "localidad" (Figura 3.11).

- Se halla el registro descriptor del archivo que se quiere cambiar, en el archivo de descriptores.

- Una vez localizado el registro, se procede a la búsqueda del campo "ciudad", para modificar su identificación por el de "localidad".

- Realizado el cambio, se reescribe el registro en el archivo de descriptores.

Como se puede apreciar, el uso de un descriptor de archivos facilita en gran medida la manipulación de los datos almacenados, aunque ello no implica necesariamente que al utilizarse el descriptor en los diferentes modelos de Base de Datos, se tenga la misma eficiencia para todos.

Por ejemplo, en la figura 3.12 se tiene un modelo jerárquico basado en archivos, donde podría determinarse un archivo de descriptores para los archivos que son raíz, por cada archivo raíz se definiría otro registro de descriptores que contuviera los archivos que dependen de él, éstos a su vez, tendrán sus archivos descriptores que harán referencia a sus archivos dependientes y así sucesivamente hasta que un archivo no tenga más archivos dependientes (a este archivo se le denomina archivo terminal).

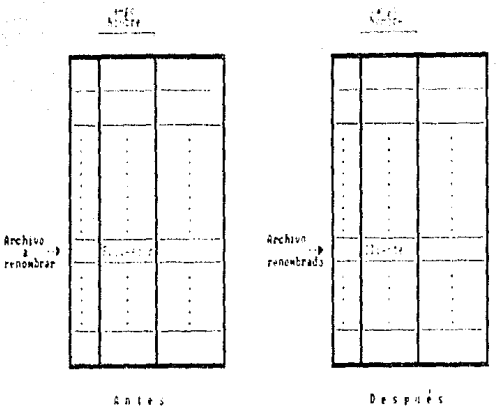


Figura 3.10 Renombrando el archivo Proveedor por Clientes

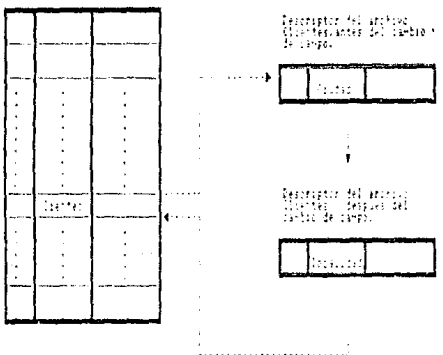
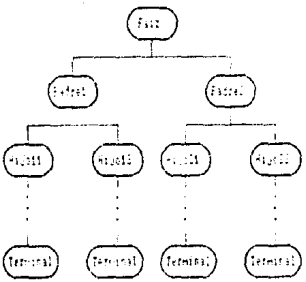


Figura 3.11 Cambio de campos en un archivo de Descriptores.



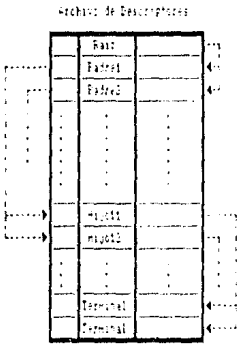


Figura 3.12 Modelo Jerárquico basado en archivos

Como puede notarse, en este modelo y con esta solución podría haber tantos registros descriptores como archivos hubiese en la Base de Datos, por consiguiente, consultas como la (1) se tornarían más lentas debido al gran número de registros descriptores que tendrían que accesarse para saber cuál de ellos apunta al archivo que se desea encontrar.

Por otro lado, si los archivos se tomaran como si cada uno de ellos fuese un árbol (Figura 3.13), se tendría en un mismo archivo a varios registros de diferentes características, los cuales serían definidos en un descriptor muy largo y complejo.

Esto lleva a que un archivo contendrá tantos campos en el registro descriptor como registros diferentes tuviese, dando como resultado un registro descriptor muy grande o implicando con ello un mantenimiento más delicado, porque si se requiere hacer alguna modificación sobre algún archivo, se necesita buscar el registro descriptor de éste y además sus registros dependientes, para poder hacer alteraciones sin que se vea afectada la estructura del archivo.

Hay que resaltar que el modelo jerárquico se basa en una estructura de árbol, la cual no es tan compleja como la de un modelo de red, fundamentada esta última en agrupaciones de registros y ligas ordenadas arbitrariamente.

Si se considera a los archivos como nodos de la red (Figura 3.14), éstos tendrían asociados registros descriptores, que también apuntan a los archivos con los que se pueden comunicar y, a su vez, ellos estarían apuntados en otros registros descriptores que los apuntan. Teniendo así, una comunicación con toda la red.

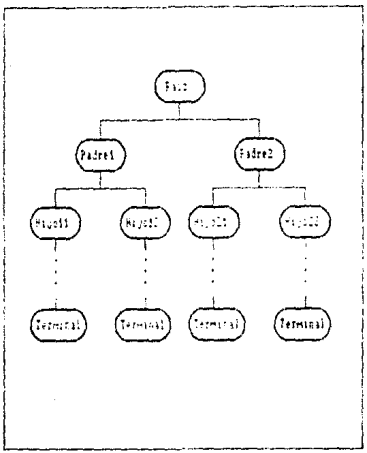
Ahora bien, si los archivos se tratan como una red, en donde los registros son parte de ella (Figura 3.15), la definición de un registro descriptor para ese archivo se tornaría más compleja debido a que:

1o Por cada registro diferente del archivo, se define un campo en el registro descriptor.

2do Cada registro descriptor va a contener información acerca del archivo, y además debe tener datos sobre los registros con los que debe tener comunicación, para saber a cuál apunta y quién lo apunta.

La representación y manipulación de archivos, usando un archivo de descriptores en el modelo de red, resulta más compleja que en el modelo jerárquico.

ARCHIVO JERARQUICO



Descriptor del archivo Jerárquico

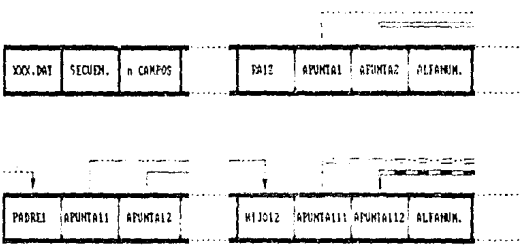
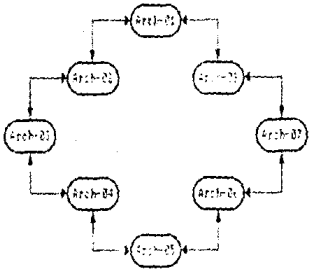


Figura 3.13 Modelo Jerárquico basado en registros.



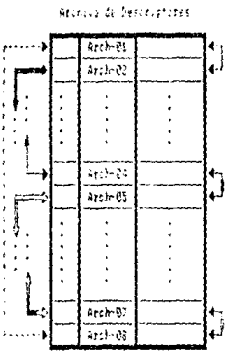
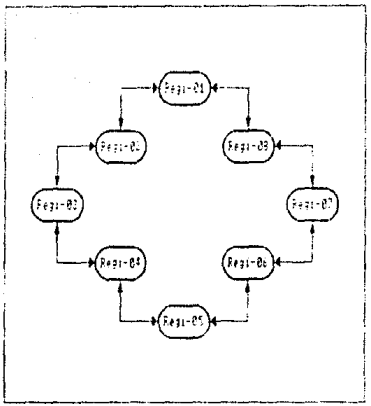


Figura 3.14 Modelo de Red basado en archivos.



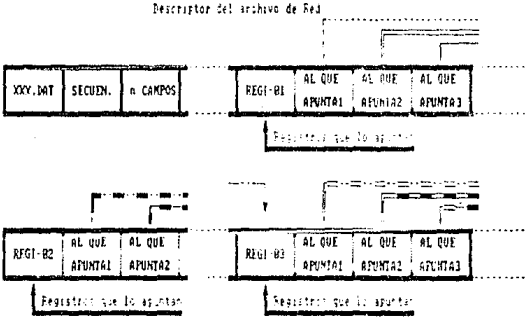


Figura 3.15 Modelo de Red basado en registros.

Hasta ahora, se ha visto que en modelos como el jerárquico y el de red, el uso de un archivo de descriptores en el mantenimiento de archivos se vuelve deficiente. En estos modelos, donde la conceptualización de los archivos no está explícita y la forma en que se definen los datos (mediante registros, generalmente de longitudes variables) ligas y apuntadores; el archivo de descriptores pierde eficiencia en medida que los registros descriptores se van haciendo más complejos.

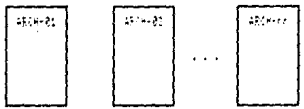
Sin embargo, modelos como el relacional (que se basa en tablas, tuplas y atributos figura 3.16), permiten el uso eficiente del archivo de descriptores. Las tablas pueden representarse por medio de archivos secuenciales, las tuplas por los registros (de longitud fija) de esos archivos y los atributos por los campos de dichos registros. Las características de los archivos pueden ser representadas en una forma sencilla dentro de un registro descriptor y a su vez, todas las tablas pueden estar representadas dentro de un solo archivo de descriptores. De esta manera, las consultas serían muy parecidas a las que se mostraron anteriormente.

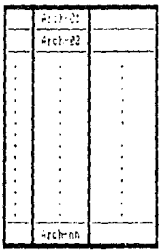
Por ejemplo, para saber si la tabla "CLIENTES" existe, se buscaría en el archivo de descriptores, el registro descriptor de la table que se desea encontrar en forma secuencial, esta búsqueda se hace sobre el campo que contiene el nombre de la tabla, hasta encontrarlo (si existe; si no, basta con señalarlo cuando se llegue al final del archivo).

Así pues, el uso del descriptor de archivos y el archivo de descriptores en el modelo relacional, permiten:

- Gran facilidad en el acceso a los archivos.
- Un manejo independiente en cada uno de los archivos.
- Facilidad y sencillez en el mantenimiento, tanto a la información como a la estructura misma de los archivos.
- Centralización de los archivos (todos se encuentran contenidos en un solo "archivo de descriptores").

Debido a que el archivo de descriptores se manipula como un archivo secuencial, el rendimiento de un Sistema de Base de Datos llegará a ser mejor, porque la secuencial es una estuctura de archivo sencilla para crearla, accesarla, mantenerla y modificarla.





Descriptores de los Archivos Relacionales.

ARCH-01	SECUEN.	n CAMPOS	CAMPO1	NUMERICO	CAMPO2	ALFABE.	CAMPO3

ARCH-02	SECUEN.	n CAMPOS	CAMPO1	ALFANUM.	CAMPO2	NUMERICO	CAMPO3

ARCH-nn	SECUEN.	n CAMPOS	CAMPO1	ALFABE.	CAMPO2	ALFABE.	CAMPO3

Figura 3.16 Modelo Relacional.

CAPITULO IV

"Continúa en ascenso porque las montañas se ven muy altas sólo desde las partes bajas"

Ezequiel Huerta R.

OPERADORES DEL ALGEBRA RELACIONAL

Introducción.

Para la manipulación de los archivos de una Base de Datos es necesario el desarrollo de herramientas u operadores, los cuales pueden actuar sobre los datos de una manera sencilla. Estas operaciones se pueden basar en lo que el Algebra Relacional realiza sobre la Teoría de Conjuntos. Por tanto, el desarrollo de la manipulación de datos de una Base de Datos Relacional se hará a través de operadores del Algebra Relacional, pero ahora sobre archivos de datos.

Como se sabe, en la programación con archivos convencionales, todos los operadores se ocupan, en esencia, de un solo registro cada vez. Sin embargo, muchos problemas se expresan con mayor naturalidad, no en términos de datos individuales, sino en términos de conjuntos.

Así, el Algebra Relacional es un conjunto de operaciones sobre las relaciones (archivos). Cada operación toma uno o más archivos como su operando y produce otro archivo como su resultado. El Algebra, tal como se presenta, se compone de dos grupos de operadores:

- Los operadores tradicionales de la teoría de conjuntos: unión, intersección, diferencia y producto cartesiano.

- Los operadores relacionales de otros términos: proyección, selección, cociente, junta natural y junta con operador de igualdad y desigualdad.

En el Algebra Relacional los operandos y resultados de una operación son archivos; por tanto, los operandos y los resultados se manejan, a continuación, como archivos.

4.1 Unión

Con esta herramienta se realiza la concatenación vertical de los archivos cuya estructura de sus registros es la misma. A partir de dos archivos se construye un archivo resultante cuya información es la suma no duplicada de los registros contenidos en los archivos iniciales, lo que lleva a la siguiente definición.

La unión de los archivos R y S, denotada por **R U S**, es el conjunto de registros que están en R, S, o ambas, y para las cuales, deben tener el mismo número de atributos y del mismo tipo, así el archivo resultante tendrá el mismo número de atributos con el mismo tipo.

Ejemplo : Se tienen los archivos R y S

R

a	b	c
1	2	3
4	1	6
3	2	4

S

d	e	f
2	7	1
4	1	6

Entonces **R U S** es :

a	b	c
1	2	3
4	1	6
3	2	4
2	7	1

Algoritmo de la unión

Entrada : Archivos O1 y O2

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si los archivos de entrada se encuentran
3. Efectuar un empatamiento de patrones entre los dos registros del descriptor
   si son los mismos pasa al paso (4)
   si no, termina
4. Copia las n-adas de O1 en el archivo OS y baja el apuntador al fin de archivo
5. Lee el registro del archivo O2
6. Lo compara con los demás registros de OS
   si no es igual a los demas registros lo inserta en OS
   si es igual, lo omite y pasa a (5)
7. El proceso se repite hasta encontrar el fin de archivo de O2

```
BEGIN
   IF    CALL Busca (Arch1) AND  CALL Busca (Arch2)
     THEN
      BEGIN
       IF #columnas de Arch1 = #columnas de Arch2
         THEN
           BEGIN
             CALL Crea (registro salida en descriptor)
             OPEN (Arch1)
             OPEN (Arch3)
            REPEAT
               READ (Arch1,Registro1)
               WRITE (Arch3,Registro1)
             UNTIL EOF (Arch1)
             CLOSE (Arch1)
             OPEN (Arch2)
            REPEAT
               READ (Arch2,Registro2)
               OPEN (Arch1)
              REPEAT
                 READ (Arch1,Registro1)
                   IF Registro2 <> Registro1
                     THEN WRITE (Arch3,Registro1)
               UNTIL EOF (Arch1)
               CLOSE (Arch1)
             UNTIL EOF  (Arch2)
             CLOSE (Arch2)
             CLOSE (Arch3)
           END
       END
       ELSE
          WRITE ('Algún archivo no fue encontrado')
END.
```

Se observa que la unión permite resumir la información de dos archivos en uno solo. Esto posibilita una depuración de los archivos existentes en la Base de Datos, lo cual no implica necesariamente la pérdida de la información.

4.2 Diferencia

Considerando que se tienen dos archivos, en los cuales hay registros comunes y se desea obtener sólo aquellos que sean exclusivos de un archivo, se necesita obtener un archivo cuyos registros estén sólo en el primero, no los que tambien estén presentes en el segundo. De esta manera se define a la diferencia de los archivos R y S, denotada por R - S, como el conjunto de registros que están en R pero no en S; para lo cual deben tener el mismo número de atributos con el mismo tipo, asi el archivo resultante tendrá el mismo número de atributos con el mismo tipo.

Ejemplo : Se tienen los archivos R y S

R

a	b	c
1	2	3
4	1	6
3	2	4

S

d	e	f
2	7	1
4	1	6

Entonces R - S es :

a	b	c
1	2	3
3	2	4

Algoritmo de la diferencia

Entrada : Archivos O1 y O2

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si los archivos de entrada se encuentran
3. Efectuar un empatamiento de patrones entre los dos registros del descriptor
   si son los mismos, pasa al paso (4)
   si no, termina
4. Copia las n-adas de O1 en el archivo OS y baja el apuntador al fin de archivo
5. Lee el registro del archivo O2
6. Lo compara con los demas registros de OS
   si no es igual a los demás registros pasa al paso (5)
   si es igual a uno de los registros, lo omite de OS y pasa a (5)

7. El proceso se repite hasta encontrar el fin de archivo de O2

Pseudocódigo

```
BEGIN
   IF  CALL Busca (Arch1) AND  CALL Busca (Arch2)
     THEN
      BEGIN
       IF #columnas de Arch1 = #columnas de Arch2
         THEN
           BEGIN
             CALL Crea (registro salida en descriptor)
             OPEN (Arch1)
             OPEN (Arch3)
            REPEAT
               READ (Arch1,Registro1)
               OPEN (Arch2)
               Bandera := 0
              REPEAT
                 READ (Arch2,Registro2)
                   IF Registro2 = registro1
                      THEN Bandera := 1
              UNTIL EOF (Arch2)
              CLOSE (Arch2)
              IF Bandera = 0
                 THEN WRITE  (Arch3,Registro1)
            UNTIL EOF  (Arch1)
            CLOSE (Arch1)
            CLOSE (Arch3)
           END
       END
       ELSE
 WRITE ('Algún archivo no fue encontrado')
END.
```

La utilidad de la diferencia radica en la identificación de los registros que no estén presentes en dos archivos a la vez, lo que permite manejar un nuevo conjunto de registros exclusivos de uno de los archivos.

4.3 Intersección

A veces es necesario identificar los registros comunes a dos archivos, esto lleva a la siguiente definición.

Se define la intersección de R y S, denotada por R ^ S al conjunto de registros que estan en R y tambien en S. Asi, por ejemplo :

Se tienen los archivos R y S

R

a	b	c
1	2	3
4	1	6
3	2	4

S

d	e	f
2	7	1
4	1	6

Entonces R ^ S es :

a	b	c
4	1	6

Algoritmo de la intersección

Entrada : Archivos O1 y O2

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si los archivos de entrada se encuentran
3. Efectuar un empatamiento de patrones entre los dos registros del descriptor
   si son los mismos, pasa al paso (4)
   si no, termina
4. Copia las n-adas de O1 en el archivo OS y baja el apuntador al fin de archivo
5. Lee el registro del archivo O2
6. Lo compara con los demás registros de OS
   si no es igual a los demás registros lo omite de OS y pasa a (5)
   si es igual, pasa a (5)
7. El proceso se repite hasta encontrar el fin de archivo de O2

```
BEGIN
   IF  CALL Busca (Arch1) AND  CALL Busca (Arch2)
     THEN
      BEGIN
       IF #columnas de Arch1 = #columnas de Arch2
         THEN
           BEGIN
             CALL Crea (Registro salida en descriptor)
             OPEN (Arch1)
             OPEN (Arch3)
            REPEAT
               READ (Arch1,Registro1)
               OPEN (Arch2)
              REPEAT
                 READ (Arch2,Registro2)
                   IF Registro2 = Registro1
                      THEN WRITE  (Arch3,Registro1)
              UNTIL EOF (Arch2)
              CLOSE (Arch2)
            UNTIL EOF  (Arch1)
            CLOSE (Arch1)
            CLOSE (Arch3)
           END
       END
       ELSE
 WRITE ('Algún archivo no fue encontrado')
END.
```

De esta manera la intersección permite obtener, de dos archivos, los registros que están presentes en ambos, o sea aquella información que comparten y depositarla en un nuevo archivo.

4.4 Producto cartesiano

Al igual que en la unión, el producto cartesiano es una concatenación de dos archivos, con la diferencia de que se realiza en forma horizontal y además a cada registro del primer archivo se le asocian todos y cada uno de los registros del segundo archivo.

Se asignan los archivos R y S de columnas m,n respectivamente, entonces R x S, es el conjunto de registros t, tales que t es la concatenación de un registro r que pertenece a R y un registro s que pertenece a S. La concatenacion de un registro r=(r1,...,rm) y un registro s=(sm+1,...,sm+n) -en este orden- es el registro t=(r1,...,rm,sm+1,...,sm+n).

Ejemplo : Se tienen los archivos R y S

R

a	b	c
1	2	3
4	1	6
3	2	4

S

d	e	f
2	7	1
4	1	6

Entonces R x S es :

a	b	c	d	e	f
1	2	3	2	7	1
1	2	3	4	1	6
4	1	6	2	7	1
4	1	6	4	1	6
3	2	4	2	7	1
3	2	4	4	1	6

Algoritmo del producto cartesiano

Entrada : Archivos O1 y O2

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si los archivos de entrada se encuentran
3. Se leen los atributos de O1 y O2
4. Se construye un nuevo registro en el descriptor con los atributos de O1 y O2, para el archivo OS
5. Lee el registro del archivo O1
6. Lee el registro del archivo O2
7. Se concatenan los registros de O1 con los de O2 y se insertan en OS
8. Es fin de archivo de O2 ?
   si se cumple, ir al paso (5)
   si no se cumple, ir al paso (6)
9. Se repite hasta encontrar el fin de archivo de O1

Pseudocódigo

```
BEGIN
   IF  CALL Busca (Arch1) AND  CALL Busca (Arch2)
     THEN
      BEGIN
           CALL Crea (Registro salida en descriptor)
           OPEN (Arch1)
           OPEN (Arch3)
           REPEAT
             READ (Arch1,Registro1)
              OPEN (Arch2)
              REPEAT
                 READ (Arch2,Registro2)
                 Registro3 := Concatena Registro1  con  Registro2
                 WRITE(Arch3,Registro3)
              UNTIL EOF (Arch2)
              CLOSE (Arch2)
           UNTIL EOF (Arch1)
           CLOSE (Arch1)
           CLOSE (Arch3)
       END
       ELSE
         WRITE ('Algún archivo no fue encontrado')
END.
```

De esta forma, el producto cartesiano determina todas las posibles combinaciones en la asociación de los registros de dos archivos, obteniendo mayor información en un solo archivo.

4.5 Proyección

Cuando se pide la información contenida en un archivo, se obtiene generalmente a través de los registros, es decir; con la información "horizontal". Pero sucede que también se llega a requerir toda la información existente sobre un campo específico.

Este operador, utiliza un archivo R, de donde se toma alguna o algunas columnas y/o se ordenan. Se produce un subconjunto de ocurrencias acerca de los campos seleccionados. Si R es un archivo de k columnas, se asigna P i1,...,im(r), donde los ij's enteros distintos , en el rango de 1 a k, denota la proyección de R sobre los componentes i1,...,im, que es el conjunto de m-registros a1,...,am tales que hay algunos k-registros b1,...,bk en R, para los cuales aj = bij para j = 1,...m.

Ejemplo : Se tiene el archivo R

R

a	b	c
1	2	3
4	1	6
3	2	4

Entonces P a,c (R) es :

a	c
1	3
4	6
3	4

Algoritmo de la proyección

Entrada : Archivo O1

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si el archivo de entrada se encuentra
3. Se leen las columnas a proyectar
4. Se construye un nuevo registro en el descriptor con las columnas de O1 que se van a proyectar, para el archivo OS
5. Lee el registro del archivo O1
6. Se incluyen las columnas a proyectar en el registro de Os
7. Se repite hasta encontrar el fin de archivo de O1

```
BEGIN
   IF CALL Busca (Arch1)
     THEN
      BEGIN
       READ COL1...COLn ( columnas a proyectar )
       CALL Crea ( Registro salida en descriptor con
                   base a las columnas seleccionadas )
       OPEN (Arch1)
       OPEN (Arch2)
       REPEAT
           READ (Arch1,Registro1)
           Registro2 := COL1+COL2+...+COLn (Registro1)
           WRITE (Arch2,Registro2)
       UNTIL EOF (Arch1)
       CLOSE (Arch2)
       CLOSE (Arch1)
       END
      ELSE
       WRITE('El archivo no fue encontrado')
END.
```

Es de observarse que con la proyección se facilita la obtención de algún o algunos campos, con todas sus respectivas ocurrencias, generando otro archivo que es subconjunto del inicial.

4.6 Selección

Como el operador proyección, el operador selección tiene un operando que puede ser un archivo de datos; se denota el operador selección como un subconjunto horizontal de una relación especifica, es decir, el subconjunto de los registros de la relación dada, para la cual se cumple una condición. esta condición involucra :

- operandos que son constantes o componentes numéricos

- operadores de comparación aritmética (<,>,=,<>,etc.)

- operadores lógicos (AND,OR y NOT).

Sin embargo, la condición no puede incluir operadores aritméticos como +,-,* y /. No se usarán operaciones básicas dentro de este operador, porque se podría confundir con operadores del álgebra relacional.

Sea T la selección sobre el archivo R y X la condición que involucra la relación de un campo con un valor constante k, para el cual debe cumplirse la condición X.

Ejemplo : Se tiene la relación R

R

a	b	c
1	2	3
4	1	6
3	2	4

Entonces **T X (R)**,

donde X : b = 2, es :

a	b	c
1	2	3
3	2	4

Algoritmo de la selección

Entrada : Archivo O1

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si el archivo de entrada se encuentra
3. Se lee la columna con la que se va a comparar
4. Se lee el comparador aritmético que se va utilizar
5. Se lee el valor con el que se va a comparar el campo antes seleccionado
6. Se lee (si lo hay)el comparador lógico
7. Lee el registro del archivo O1
8. Se compara el campo seleccionado con el valor dado usando el comparador selecionado
   si cumple, lo escribe en OS
   si no, va al paso 7
9. Se repite hasta encontrar el fin de archivo de O1

Pseudocódigo

```
BEGIN
     IF CALL Busca (Arch1)
       THEN
       BEGIN
         READ COL
         READ COM_AR (comparador aritmetico: 1 = '>',2 = '<' y 3 = '=' )
         READ VALOR
         READ COM_LOG (comparador lógico -si lo hay-)
         OPEN (Arch1)
         REPEAT
           READ (Arch1,Registro1)
           CASE COM_AR
             1 :  IF  COL > VALOR
                      THEN WRITE (Arch2,Registro1)
             2 :  IF  COL < VALOR
                     THEN WRITE (Arch2,Registro1)
             3 :  IF  COL = VALOR
                      THEN WRITE (Arch2,Registro1)
           END CASE
         UNTIL EOF (Arch1)
         CLOSE (Arch2)
         CLOSE (Arch1)
       END
       ELSE
          WRITE ('El archivo no fue encontrado')
END.
```

La selección es la herramienta que trae de un archivo dado, y de acuerdo a una condición especifica, todos los registros que satisfacen dicha condición. Así se puede manejar un subconjunto de registros del archivo utilizado.

4.7 Cociente

Hay requerimientos que piden la obtención de la información de una forma muy especifica, por lo que la utilización directa de los operadores básicos del Algebra Relacional no es suficiente. Situaciones como esta, llevan a desarrollar herramientas que satisfagan especificamente dichos requerimientos. Uno de esos casos es el que pide conocer el o los campos que forman pareja con el campo o conjunto de campos identificados en el registro. Para este efecto se desarrolla el operador del Algebra Relacional conocido como división.

Se tienen los archivos R y S, con r y s columnas respectivamente, además r < s, s <> 0 y S es subconjunto de las columnas (r - s) + 1, (r - s) + 2, ..., r de R. Como primer paso, se proyectan los (r - s) primeras columnas de R. A este resultado se aplica el producto cartesiano asociado con el archivo S, quedando como resultado un archivo denominado V.

$$V = P\ (r - s)\ (R) \times S$$

Se obtiene la diferencia V - R para proyectar las (r - s) primeras columnas de esta diferencia, denominada W.

$$W = P\ (r - s)\ (V - R)$$

Teniendo así que los valores de W no hacen pareja con los valores presentes en S. Por lo tanto, se observa que la diferencia

$$P\ (r - s)\ (R) \ - \ W$$

es la división buscada R / S.

$$R / S = P\ 1,\ldots,r\text{-}s(R) - P\ 1,\ldots,r\text{-}s((P\ 1,\ldots,r\text{-}s(R) \times S) - R)$$

R

a	b	c	d
1	2	3	4
1	2	5	6
2	3	5	6
5	4	3	4
5	4	5	6
1	2	4	5

S

e	f
3	4
5	6

R / S

x	y
1	2
5	4

Algoritmo del cociente

Entrada : Archivos O1 y O2

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si los archivos de entrada se encuentran
3. Se compara si el # de columnas de O1 es mayor que el # de columnas de O2
   si cumple, ir a (4)
   si no cumple, termina
4. Proyectar las columnas de O1 conforme a la expresión:
   Pi 1,...,#columnas O1 - #columnas O2 (O1)
   Obteniendo un archivo temporal T1
5. Hacer el producto cartesiano de:
   T1 x O2
   Obteniendo un archivo temporal T2
6. Hacer la diferencia de:
   T2 - O1
   Obteniendo un archivo temporal T3
7. Proyectar las columnas de T3 conforme a la expresión:
   Pi 1,...,#columnas O1 - #columnas O2 (T3)
   Obteniendo un archivo temporal T4
8. Hacer la diferencia de:
   T1 - T4
   Obteniendo el archivo OS que es :
   R / S

Pseudocódigo

```
BEGIN
   IF  CALL Busca (Archl) AND  CALL Busca (Arch2)
     THEN
      BEGIN
       IF  #columnas de Archl > #columnas de Arch2
         THEN
           BEGIN
              lim := #columnas de Archl - #columnas de Arch2
              CALL  Proyecta (lim,Archl,Archtem1)
              CALL  Producto cartesiano (Archtem1,Arch2,Archtem2)
              CALL  Diferencia (Archtem2,Archl,Archtem3)
              CALL  Proyecta (lim,Archtem3,Archtem4)
              CALL  Diferencia (Archtem1,Archtem4,Archl)
           END
       END
       ELSE
          WRITE ('Algún archivo no fue encontrado')
END.
```

La división del Algebra Relacional es similar a la división matemática, donde se define que entre el dividendo y el divisor existe un número tal que determina la cantidad de particiones del dividendo, necesarias para corresponder en partes iguales a cada unidad del divisor. La diferencia con el Algebra Relacional es la siguiente: un archivo debe ser subconjunto del otro para poder repartir todas las ocurrencias que pueda tener el subconjunto especificado.

4.8 Junta natural

La junta natural, que se define R <> S, solo es aplicable cuando los dos archivos tienen al menos un campo con el mismo nombre y tipo, este operador, en términos de los operadores básicos de conjuntos, se puede evaluar como sigue:

- Evaluar R x S
- Por cada atributo A que está tanto en R como en S, seleccionamos de R x S aquellos registros cuyos valores concuerden para R.A y S.A; renombramos R.A, que es el nombre de la columna R x S correspondiente a la columna A en R, S.A es definida análogamente
- Para cada atributo A, proyectamos la columna S.A

R

a	b	c
1	2	3
4	2	3
2	2	6
3	1	4

S

b	c	d
2	3	4
2	3	5
1	4	2

Entonces R <> S es :

a	b	c	d
1	2	3	4
1	2	3	5
4	2	3	4
4	2	3	5
3	1	4	2

Algoritmo de la junta natural

Entrada : Archivos O1 y O2

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si los archivos de entrada se encuentran
3. Se leén los atributos de O1 y O2 que son iguales
4. Se construye un nuevo registro en el descriptor con los atributos de O1 y O2, para el archivo OS
5. Leer el registro del archivo O1
6. Leer el registro del archivo O2
7. se comparan todos los atributos iguales de O1 y O2
   si son iguales, se concatenan los registros de O1 con los de O2 y se insertan en OS
   si no, se lee el siguiente registro de O2
8. Es fin de archivo de O2 ?
   si se cumple, ir al paso (5)
   si no se cumple, ir al paso (6)
9. Se repite hasta encontrar el fin de archivo de O1

Pseudocódigo

```
BEGIN
   IF  CALL  Busca (Arch1) AND  CALL  Busca (Arch2)
     THEN
      BEGIN
            READ atributos iguales de Arch1 y Arch2
            asignarlos al vector
            CALL  Crea (registro salida en descriptor)
            OPEN (Arch1)
            OPEN (Arch3)
            REPEAT
               READ (Arch1,Registro1)
               OPEN (Arch2)
                 REPEAT
                  READ (Arch2,Registro2)
                  IF  vector(Registro1) = vector(Registro2)
                    Registro3 := ( Registro1 || Registro2)
                    THEN WRITE (Arch3,Registro3)
                 UNTIL EOF (Arch2)
              CLOSE (Arch2)
            UNTIL EOF (Arch1)
            CLOSE (Arch1)
            CLOSE (arch3)
      END
      ELSE
         WRITE ('Algún archivo no fue encontrado')
END.
```

En algunas ocasiones el producto cartesiano obtiene campos y ocurrencias con el mismo valor. En esas situaciones puede necesitarse identificar las repeticiones.

6.9 Junta con operador de igualdad y desigualdad

Este operador simplifica muchas consultas que requieren un producto cartesiano. Normalmente, una consulta que utilice algún producto cartesiano incluye una operación de selección sobre el resultado del producto cartesiano. Así, éste permite combinar la selección y el producto cartesiano en una sola operación. Se indica por x@, donde x es el símbolo del producto cartesiano y @ se sustituye con el predicado de selección. El operador forma el producto cartesiano de sus dos argumentos y después lleva a cabo una selección mediante el predicado @.

R

a	b	c
1	2	3
4	2	3
2	2	6
3	1	4

S

x	y	z
2	3	4
2	3	5
1	4	2

Entonces R x@ S es :

donde @ = a < y

a	b	c	x	y	z
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	1	4	2
1	2	3	1	4	2
2	2	6	2	3	4
2	2	6	2	3	5
2	2	6	1	4	2
3	1	4	1	4	2

Algoritmo de la junta con operador
de igualdad y desigualdad

Entrada : Archivos O1 y O2

Salida : Archivo OS

1. Abrir el descriptor de archivos
2. Explorar si los archivos de entrada se encuentran
3. Se leen los atributos de O1 y O2 que se van a comparar
4. se les el comparador aritmético a utilizar
5. Se construye un nuevo registro en el descriptor con los atributos de O1 y O2, para el archivo OS
6. Lee el registro del archivo O1
7. Lee el registro del archivo O2

8. se comparan todos los atributos seleccionados de acuerdo al comparador seleccionado
   si cumple con la condición, se concatenan los registros de O1 con los de O2 y se insertan en OS
   si no, se lee el siguiente registro de O2
9. Es fin de archivo de O2 ?
   si se cumple, ir al paso (6)
   si no se cumple, ir al paso (7)
10. Se repite hasta encontrar el fin de archivo de O1

Pseudocódigo

```
BEGIN
   IF  CALL  Busca (Arch1)  AND  CALL  Busca (Arch2)
     THEN
      BEGIN
            READ Arch1 (atri1)
            READ Arch2 (atri2)
            READ COM_AR ( comparador aritmético ( 1 = '>',2 = '<' y 3 = '=' )
            CALL  Crea (registro salida en descriptor)
            OPEN (Arch1)
            OPEN (arch3)
            REPEAT
              READ (Arch1,Registro1)
              OPEN (Arch2)
              REPEAT
                READ (Arch2,Registro2)
                  CASE  COM_AR
                      1 :  IF  atri1 > atri2
                               THEN Registro3 := (registro1 || Registro2)
                               WRITE (Arch3,Registro3)
                      2 :  IF  atri1 < atri2
                               THEN Registro3 := (registro1 || Registro2)
                               WRITE (Arch3,Registro3)
                      3 :  IF  atri1 = atri2
                               THEN Registro3 := (registro1 || Registro2)
                               WRITE (Arch3,Registro3)
                  END CASE
              UNTIL  EOF (Arch2)
              CLOSE (Arch2)
            UNTIL  EOF  (Arch1)
            CLOSE  (Arch1)
            CLOSE  (Arch3)
      END
      ELSE
         WRITE ('Algún archivo no fue encontrado')
END.
```

En otros casos del producto cartesiano, se necesita identificar aquellos registros cuyos campos (tomados tanto de un archivo y de otro) cumplan entre sí una condición dada.

CAPITULO V

"No te desanimes ¡insiste!
porque normalmente la última
llave que pruebas es la que abre
la puerta"

OPERADORES UTILITARIOS

Introducción.

El buen uso de una Base de Datos implica no sólo la manipulación de sus datos, sino primordialmente el mantenimiento que se le dá a su estructura, e incluso a los datos mismos. Por ejemplo, si es necesario actualizar la dirección de alguna persona en un archivo que contenga direcciones, es necesario modificar directamente el dato, sin alterar la estructura u otras características del archivo; se necesita así, de un operador que haga esta función. Así como ésta, hay algunos otros operadores cuya función afecta sólo a los datos, o incluso a la estructura (como lo son la creación, la captura, modificación, etc). Por lo anterior, es necesario contar con operadores que puedan dar el adecuado mantenimiento a la Base de Datos.

5.1 Creación y Eliminación.

Acciones de uso necesario son la creación y eliminación de archivos pertenecientes a una Base de Datos, lo cual implica la necesidad de tener herramientas que realicen estas acciones cotidianas. Para ello se tienen las siguientes.

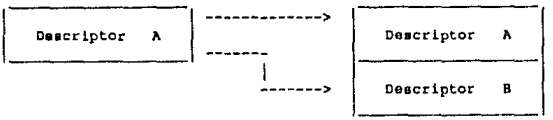
Creación

Este operador agrega un registro a los ya existentes en el Archivo de Descriptores, con la descripción de un nuevo archivo; además lo crea externamente como nuevo y vacío.

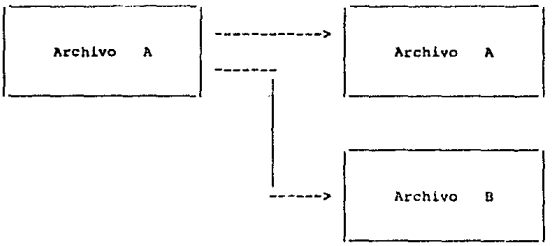
Un ejemplo esquemático es el que se expone a continuación:

ANTES **DESPUES**

(Archivo de Descriptores)



(Archivos Externos)



Algoritmo de la creación de archivos.

Entrada : Registro Descriptor.

Salida : Archivo Descriptor.

1. Abrir el Archivo de Descriptores
2. Capturar el nombre del nuevo archivo
3. Buscar un Registro Descriptor con el nombre leido
   Si existe, ir al paso 8.
4. Leer el número de campos que contendrá el archivo

5. Capturar, para cada registro, los siguientes datos:
   Nombre del campo
   Tamano del campo
   Tipo del campo
5a. Calcular los apuntadores, inicial y final, en base al tamano del campo
6. Anadir el Registro Descriptor leido, al Archivo de Descriptores
7. Asignar un Archivo externo (crearlo vacio), relacionado al Descriptor elaborado
8. Cerrar el Archivo de Descriptores

Pseudocódigo de la creación de archivos

```
BEGIN
 LEER(Archivo)
 OPEN(Descriptor)
 IF CALL Existe(Archivo) THEN
    BEGIN
    WRITE('El archivo ya existe')
    END
 ELSE
    BEGIN
    READ(Número de campos)
    FOR 1 TO Número de campos DO
        BEGIN
        READ(Nombre del campo)
        READ(Tamano del campo)
        CALL Calcula(Apuntador Inicial y Apuntador Final)
        READ(Tipo del campo)
        END
    ASSIGN(Archivo)
    END
 CLOSE(Descriptor)
END
```

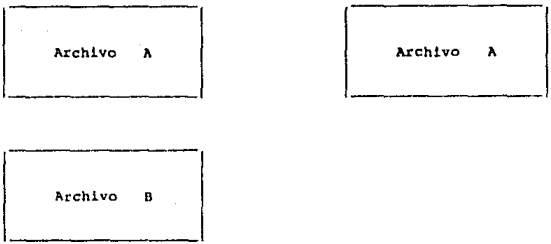
Eliminación

A través de este operador se suprime, del archivo de descriptores y de la Base de Datos, el archivo seleccionado. Es pertinente observar que debe emplearse con suma precaución para evitar la pérdida de información. Gráficamente, se puede entender esta herramienta como sigue.

ANTES DESPUES

(Archivo de Descriptores)

(Archivos Externos)



Algoritmo de la eliminación de archivos.

Entrada : Registro Descriptor.

Salida : Archivo Descriptor.

1. Abrir el Archivo de Descriptores
2. Capturar el nombre del archivo a eliminar
3. Buscar un Registro Descriptor con el nombre leido
   Si no existe, ir al paso 6.
4. Borrar, fisicamente, el archivo seleccionado.

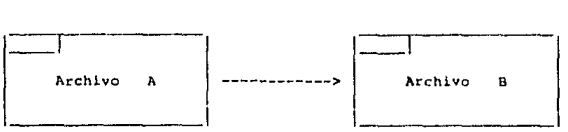
5. Borrar del archivo de descriptores, el registro correspondiente.
6. Cerrar el Archivo de Descriptores

Pseudocódigo de la eliminación de archivos

```
BEGIN
 LEER(Archivo)
 OPEN(Descriptor)
 IF NOT CALL Existe(Archivo) THEN
    BEGIN
    WRITE('El archivo no existe')
    END
 ELSE
    BEGIN
     DELETE(Archivo)
     ERASE (Descriptor)
    END
 CLOSE(Descriptor)
END
```

5.2 Copia

En ocasiones, es conveniente repetir la información de un archivo para obtener un respaldo de seguridad o para generar un duplicado del archivo en cuestión a fin de transformarlo mediante el uso de otras herramientas, lo que lleva a la creación de una herramienta que permita generar una imagen exacta de un archivo. Esta se denomina copia, la que crea un archivo con la misma estructura e información de otro, definido con anterioridad.



Algoritmo de la copia

Entrada : Archivo de datos A

Salida : Archivo de datos B

1. Abrir el Archivo de Descriptores
2. Buscar el archivo a copiar (A)
3. Si no está en el Descriptor, ir al paso 11
4. Buscar el archivo a generar (B)
5. Si existe, ir al paso 11
6. Abrir archivos
7. Leer registro de A
8. Escribir el registro en B
9. Repetir pasos 7 y 8 hasta encontrar fin de archivo de A
10. Cerrar archivos A y B
11. Cerrar el archivo de Descriptores.

Pseudocódigo de la copia

```
BEGIN
 OPEN (Descriptor)
 READ (A, B)
 IF CALL Existe (A) AND NOT (Existe (B) ) THEN
   BEGIN
    OPEN (A, B)
    FOR 1 TO Número-de-registros (A) DO
        BEGIN
         READ (A, Registro)
         WRITE (B, Registro)
        END
    CLOSE (A, B)
   END
 ELSE
   BEGIN
    WRITE ('Conflicto: archivo A no existe, ó
            archivo B ya existe')
   END
 CLOSE (Descriptor)
END.
```

5.3 Mezcla, Partición y Factor

Mezcla

La mezcla de dos archivos es, como su nombre lo indica, la unión intercalada de los registros pertenecientes a dos archivos.

Ejemplo:

Archivos de entrada

Archivo A

1	
2	
3	

Archivo B

A	
B	
C	

Archivo de salida

Archivo C

1	
A	
2	
B	
3	
C	

Algoritmo de la mezcla

Entrada : Archivos A y B

Salida : Archivo C

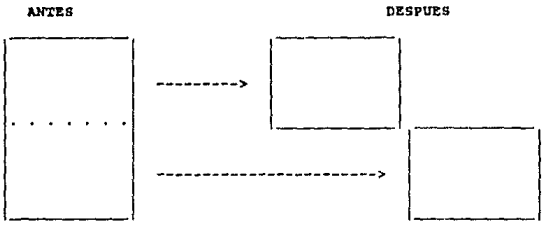
1. Abrir el Archivo de Descriptores
2. Buscar los archivos a mezclar A y B
3. Si no existe alguno, ir al paso 11
4. Efectuar un empatamiento de patrones entre A y B
5. Si no son los mismos, ir al paso 11
6. Abrir A, B y C
7. Si no hay fin de archivo en A, leer un registro y escribirlo en el archivo C
8. Si no hay fin de archivo en B, leer un registro y escribirlo en el archivo C
9. Repetir pasos 7 y 8 hasta que haya
   fin de archivo en A y
   fin de archivo en B
10. Cerrar archivos A, B y C
11. Cerrar el Archivo de Descriptores

Pseudocódigo de la mezcla

```
BEGIN
  OPEN (Descriptor)
  IF CALL Existe (A) AND CALL Existe (B)
     AND CALL Empat (A, B) THEN
     BEGIN
      OPEN (A, B, C)
      REPEAT
        IF NOT EOF (A)
          BEGIN
           READ (A, Registro)
           WRITE (C, Registro)
          END
        IF NOT EOF (B)
          BEGIN
           READ (B, Registro)
           WRITE (C, Registro)
          END
      UNTIL EOF (A) AND EOF(B)
      CLOSE (A, B, C)
     END
  ELSE
     BEGIN
      WRITE ('Conflicto: no existe algún archivo y/o no
              son compatibles para mezcla')
     END
  CLOSE (Descriptor)
END.
```

Partición

El manejo de la información de un archivo, puede requerir la utilización de un mismo archivo en dos subconjuntos de registros



determinados a partir de un registro especifico. También puede requerirse anadir al final de un archivo, los registros pertenecientes a otro, para tener identificada la suma de la información en un solo archivo. De igual forma, la suma de la información puede hacerse en forma aleatoria, es decir; de dos archivos se unen los registros en forma intercalada. Estas operaciones las realizan las siguientes herramientas.

Esta herramienta realiza la separación del conjunto de registros en dos conjuntos diferentes, esto es; un archivo se divide en dos partes para poder ser manipulado en forma independiente.

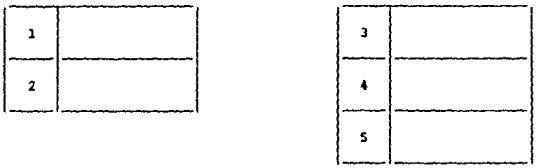
Ejemplo:

Archivo de entrada

Archivo A

1	
2	
3	
4	
5	

Archivos de salida



Algoritmo de la partición

Entrada : Archivos A y temporal T, N número de registro a partir del cual se particiona

Salida : Archivos A y B

1. Abrir el Archivo de Descriptores
2. Buscar archivo A
3. Si no existe, ir al paso 16
4. Buscar el archivo B
5. Si existe, ir al paso 16
6. Abrir archivos A, B y T
7. Leer registro de A
8. Escribir el registro en T
9. Repetir pasos 7 y 8 hasta N - 1
10. Leer registro de A
11. Escribir registro en B
12. Repetir pasos 10 y 11 hasta encontrar fin de archivo de A
13. Copiar archivo T, en A
14. Cerrar A y B
15. Borrar T
16. Cerrar Archivo de Descriptores

Pseudocódigo de la partición

```
BEGIN
 OPEN (Descriptor)
 IF CALL Existe (A) AND CALL NOT Existe (B) THEN
  BEGIN
```

```
   OPEN (A, B, T)
   FOR I TO N - 1 DO
    BEGIN
     READ (A, Registro)
     WRITE (T, Registro)
    END
   REPEAT
    READ (A, Registro)
    WRITE (B, Registro)
   UNTIL EOF (A)
   CLOSE (A, B, T)
   OPEN (A, T)
   REPEAT
    READ (T, Registro)
    WRITE (A, Registro)
   UNTIL EOF (T)
   CLOSE (A, T)
  END
 ELSE
  BEGIN
   WRITE ('Conflicto: algún archivo ó ambos, no existen')
  END
 CLOSE (Descriptor)
END.
```

Factor

Es la herramienta que anade, al final de una tabla, los renglones de otra, para lo cual deberán ser compatibles en cuanto a las caracteristicas de sus campos.

Ejemplo:

Archivos de entrada

Archivo A

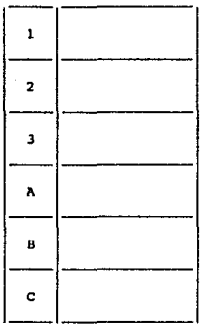
1	
2	
3	

Archivo B

A	
B	
C	

Archivo de salida

Archivo A



Algoritmo del factor

Entrada : Archivos A y B

Salida : Archivo A

1. Abrir el Archivo de Descriptores
2. Buscar los archivos A y B
3. Si alguno no existe, ir al paso 12
4. Si no son compatibles, ir al paso 12
5. Abrir los archivos A y b
6. Posicionar al final del archivo A
7. Leer registro de B
8. Escribir registro en A
9. Repetir pasos 7 y 8 hasta encontrar fin de archivo en B
10. Cerrar archivos A y B
11. Borrar archivo B
12. Cerrar el Archivo de Descriptores

Pseudocódigo del factor

```
BEGIN
 OPEN (Descriptor)
 IF CALL Existe (A) AND CALL Existe (B) THEN
  IF CALL Compatibles (A, B) THEN
   BEGIN
    OPEN (A, B)
    CALL Posiciona (A, EOF(A))
    REPEAT
     READ (B, Registro)
     WRITE (A, Registro)
    UNTIL EOF (B)
    CLOSE (A, B)
    DELETE (B)
   END
  ELSE
   BEGIN
    WRITE ('Conflicto: archivos no compatibles para factor')
   END
 ELSE
  BEGIN
   WRITE ('Conflicto: alguno ó ambos archivos no existen')
  END
 CLOSE (Descriptor)
END.
```

5.4 Captura, Actualiza y Supresión.

Como se vió anteriormente, existen herramientas que definen la estructura de los archivos, pero no brindan la posibilidad de dar mantenimiento a la información contenida en ellos. Este mantenimiento es, generalmente, el que se dá a través de la inserción, modificación o eliminación de registros.

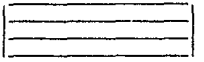
El mantenimiento realizado a la información contenida en los archivos, se desarrolla a través de las herramientas elaboradas a continuación.

Captura.

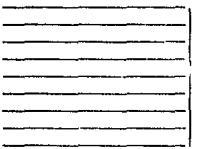
La captura de registros a un archivo ya existente es para anadirle datos, no importando si el archivo en cuestión es nuevo o si contenía previamente algunos registros. Los datos agregados al archivo en cuestión se incorporan al final del mismo.

Ejemplo:

Archivo A antes



Archivo A después



Algoritmo de la captura.

Entrada : Archivo de Datos.

Salida : Archivo de Datos.

1. Abrir el Archivo de Descriptores
2. Buscar el archivo a capturar
3. Si no está en el Descriptor, ir al paso 7
4. Leer la descripción de los campos, del Descriptor
5. Capturar el detalle de los campos
6. Cerrar el archivo de datos
7. Cerrar el archivo de Descriptores.

Pseudocódigo de la captura

```
BEGIN
 READ (Archivo)
 OPEN (Descriptor)
 IF CALL Existe(Archivo) THEN
    BEGIN
     OPEN(Archivo)
     FOR 1 TO Descriptor(Número de Campos) DO
        BEGIN
         READ(Archivo(Campo))
        END
     CLOSE(Archivo)
    END
 ELSE
    BEGIN
    WRITE(El archivo no existe)
    END
 CLOSE(Descriptor)
END
```

Actualiza

Mediante la inspección a todo un archivo existente, se puede realizar con esta herramienta una modificación directa a los campos de cada registro. Opcionalmente, puede ser útil implementar la modificación dirigida a los registros seleccionados a través de una condición señalada en forma previa.

Ejemplo:

Entrada.- Archivo A Modificaciones a c/registro

0	AAABBCCCCDD	<----	1	AAAAAABBBBB
2	HHHHHHHHHHH			
4	XXXXXYZZZZZ	<----	3	

Salida; Archivo A actualizado

1	AAAAAABBBBB
2	HHHHHHHHHHH
3	XXXXXYZZZZZ

Algoritmo de la actualización

Entrada : Archivos A y T

Salida : Archivo A

1. Abrir el Archivo de Descriptores
2. Buscar el archivo A
3. Si no existe, ir al paso 16
4. Abrir A y T

5. Leer registro
6. Desplegar campos
7. Leer, si hay, nuevos valores para los campos
8. Escribir el registro en T
9. Repetir 5, 6, 7 y 8 hasta encontrar fin de archivo en A
10. Cerrar A y T
11. Abrir A y T
12. Leer registro de T
13. Escribir registro en A
14. Repetir 12 y 13 hasta encontrar fin de archivo en T
15. Cerrar A y T
16. Cerrar el Archivo de Descriptores

Pseudocódigo de la actualización

```
BEGIN
 OPEN (Descriptor)
 IF CALL Existe (A) THEN
  BEGIN
   OPEN (A, T)
   REPEAT
    READ (A, Registro)
    CALL Modifica (Registro)
    WRITE (T, Registro)
   UNTIL EOF (A)
   CLOSE (A, T)
   OPEN (A, T)
   REPEAT
    READ (T, Registro)
    WRITE (A, Registro)
   UNTIL EOF (T)
   CLOSE (A, T)
  END
 ELSE
  BEGIN
   WRITE ('Conflicto: el archivo no existe')
  END
 CLOSE (Descriptor)
END.
```

Supresión

Borra los registros identificados a través de una inspección a todo el archivo en cuestión. Opcionalmente puede implementarse la inspección sólo al conjunto de registros que cumplen con alguna condición especificada

Ejemplo:

Condición.- Campo < 6

Entrada; Archivo A

3	
4	
9	
2	
8	

Salida; Archivo A

9	
8	

Algoritmo de la Supresión

Entrada : Archivo A, Archivo Temporal T, Campo C, Condición D

Salida : Archivo A

1. Abrir el Archivo de Descriptores
2. Buscar el archivo A
3. Si no existe, ir al paso 14
4. Buscar en A, a C
5. Si no existe, ir al paso 14
6. Abrir A y T
7. Leer registro de A
8. Extraer de A, el campo C
9. Si cumple con D, escribir el registro en T
10. Repetir pasos 7, 8 y 9 hasta encontrar fin de archivo de A
11. Copiar T en A
12. Cerrar A y T
13. Borrar T
14. Cerrar el Archivo de Descriptores

Pseudocódigo de la supresión

```
BEGIN
 OPEN (Descriptor)
 IF CALL Existe (A) THEN
  IF CALL Existe (A (C)) THEN
   BEGIN
    OPEN (A, T)
    REPEAT
     READ (A, Registro)
     CALL Extract (Registro, C)
     IF CALL Cumple (C, D) THEN
       WRITE (T, Registro)
    UNTIL EOF (A)
    CLOSE (A, T)
    OPEN (A, T)
    REPEAT
     READ (T, Registro)
     WRITE (A, Registro)
    UNTIL EOF (T)
    CLOSE (A, T)
   END
  ELSE
   BEGIN
    WRITE ('Conflicto: el archivo no contiene el campo
           requerido')
   END
 ELSE
  BEGIN
   WRITE ('Conflicto: no existe el archivo')
  END
 CLOSE (Descriptor)
END.
```

5.5 Ordenación.

Tener un archivo con los datos necesarios es útil, pero de mayor utilidad es tener esa información debidamente ordenada a través de ciertos campos. Ello agiliza la operación de algunas actividades tales como:

- Búsqueda directa de registros.
- Obtención de los valores máximo y mínimo del campo ordenado.
- Generación de reportes en forma ascendente y descendente.

En términos generales, posibilita un uso más ágil de la información contenida en el archivo.

Este operador organiza los registros contenidos en un archivo, en forma ascendente o descendente, tomando como base un campo, el cual es la llave de referencia para hacer dicha reorganización.

Ejemplo de ordenación ascendente:

Archivo A de entrada

5	
3	
1	
6	
2	
4	

Archivo A de salida

1	
2	
3	
4	
5	
6	

Algoritmo de la ordenación

Entrada : Archivo A, Archivo Temporal T, Campo C
Tipo de ordenación (Ascendente/Descendente)

Salida : Archivo A

1. Abrir el Archivo de Descriptores y archivos A y T
2. Buscar el archivo A
3. Si no está en el Descriptor, ir al paso 12
4. Buscar el campo a ordenar
5. Si no existe el campo, ir al paso 12
6. Si el tipo de ordenación es ascendente, buscar el registro con el campo C menor, en A.
   Si el tipo de ordenación es descendente, buscar el campo mayor
7. Escribirlo en el archivo temporal T
8. Eliminarlo del archivo A
9. Repetir pasos 6, 7 y 8 hasta encontrar fin de archivo de A
10. Copiar archivo temporal T en A
11. Cerrar el archivo temporal T y borrarlo
12. Cerrar el archivo de Descriptores

Pseudocódigo de la ordenación

```
BEGIN
 OPEN (Descriptor)
 READ (A, C, TipOrd)
 IF CALL Existe (A) THEN
   BEGIN
    IF CALL Existe (A(C)) THEN
      BEGIN
         OPEN (A, T)
         REPEAT
           IF TipOrd = Ascendente THEN
              BEGIN
                CALL BuscaMenor (A(C))
                WRITE (T, Registro Menor)
                CALL BorraMenor (A (Registro Menor) )
              END
           IF TipOrd = Descendente THEN
              BEGIN
                CALL BuscaMayor (A(C))
                WRITE (T, Registro Mayor)
                CALL BorraMayor (A (Registro Mayor) )
              END
         UNTIL EOF (A)
         CLOSE (A, T)
         OPEN (A, T)
         REPEAT
          READ (T, Registro)
          WRITE (A, Registro)
         UNTIL EOF (T)
         CLOSE (A, T)
      END
```

```
    ELSE
      BEGIN
       WRITE ('Conflicto: el archivo no tiene el campo
              requerido')
      END
   END
  ELSE
    BEGIN
     WRITE ('Conflicto: el archivo no existe')
    END
 CLOSE (Descriptor)
END.
```

5.6 Permuta de llave

Rotar el campo que se encuentra en el registro de un archivo, posibilita alterar el orden de los caracteres que forman parte de ese campo. Esto es útil cuando se tienen campos que están conformados por claves, puesto que al rotar o permutar sus caracteres, éstos conservan un sentido lógico como claves, una vez que haya sido definida la nueva presentación del campo utilizado.

Para tal efecto, la permuta de llave efectúa la rotación de un campo en un registro dado. Esta se realiza al nivel de sus caracteres.

Ejemplo de permuta a una posición:

Archivo A de entrada

HOLA	
1234	
XXXY	

Archivo A de salida

AHOL	
4123	
YXXX	

Algoritmo de la permuta

Entrada : Archivo A, Campo C, N (número de caracteres a permutar

Salida : Archivo A

1. Abrir el Archivo de Descriptores
2. Buscar el archivo A
3. Si no existe, ir al paso 13
4. Buscar en A, a C
5. Si no existe, ir al paso 13
6. Abrir A
7. Leer registro de A
8. Extraer el campo C
9. Rotar C, N posiciones a la derecha
10. Reescribir el registro en A
11. Repetir pasos 7, 8, 9 y 10 hasta encontrar fin de archivo de A
12. Cerrar A
13. Cerrar el Archivo de Descriptores

Pseudocódigo de la permuta

```
BEGIN
 OPEN (Descriptor)
 IF CALL Existe (A) THEN
  IF CALL Existe (A (C)) THEN
   BEGIN
    OPEN (A)
    REPEAT
     READ (A, Registro)
     CALL Extract (Registro, C)
     CALL Permuta (C, N)
     CALL Reescribe (A, Registro)
    UNTIL EOF (A)
    CLOSE (A)
   END
  ELSE
   BEGIN
    WRITE ('Conflicto: el archivo no tiene el campo
           requerido')
   END
 ELSE
  BEGIN
   WRITE ('Conflicto: el archivo no existe')
  END
 CLOSE (Descriptor)
END.
```

5.7 Búsqueda de Máximo y Mínimo.

De la información contenida en un archivo, puede requerirse aquel registro que sea el mayor o el menor, esto con base en un criterio señalado por el contenido de un campo específico. Las herramientas que identifican esos registros son las siguientes.

Búsqueda de Máximo

Determina el renglón que tiene el campo mayor a todos los demás renglones del archivo.

Ejemplo:

Archivo A

3	
4	
9	
2	
8	

El registro con el campo mayor es el número : 3

Algoritmo de búsqueda del máximo

Entrada : Archivo A, Campo C

Salida : N; número de registro

1. Abrir el Archivo de Descriptores
2. Buscar el archivo A
3. Si no existe, ir al paso 13
4. Buscar en A, a C
5. Si no existe, ir al paso 13
6. Abrir A
7. Leer registro de A

8. Extraer el campo C como el máximo
9. Leer registro de A
10. Comparar C con el máximo,
si es mayor, hacerlo el nuevo máximo
11. Repetir pasos 9 y 10 hasta encontrar
fin de archivo de A
12. Cerrar A
13. Cerrar el Archivo de Descriptores

Pseudocódigo de búsqueda del máximo

```
BEGIN
 OPEN (Descriptor)
 IF CALL Existe (A) THEN
  IF CALL Existe (A (C)) THEN
   BEGIN
    OPEN (A)
    READ (A, Registro)
    CALL Extract (Registro, C)
    Máximo:= C
    REPEAT
     READ (A, Registro)
     CALL Extract (Registro, C)
     IF C > Máximo THEN
       Máximo:= C
    UNTIL EOF (A)
    CLOSE (A)
   END
  ELSE
   BEGIN
    WRITE ('Conflicto: el archivo no contiene el campo
           requerido')
   END
 ELSE
  BEGIN
   WRITE ('Conflicto: el archivo no existe')
  END
 CLOSE (Descriptor)
END.
```

Búsqueda de Mínimo

Determina el renglón que tiene la columna menor a todos los renglones del archivo.

Ejemplo:

Archivo A

3	
4	
9	
2	
8	

El registro con el campo menor es el número : 4

Algoritmo de búsqueda del mínimo

Entrada : Archivo A, Campo C

Salida : N; número de registro

1. Abrir el Archivo de Descriptores
2. Buscar el archivo A
3. Si no existe, ir al paso 13
4. Buscar en A, a C
5. Si no existe, ir al paso 13
6. Abrir A
7. Leer registro de A
8. Extraer el campo C como el mínimo
9. Leer registro de A
10. Comparar C con el mínimo,
    si es menor, hacerlo el nuevo mínimo
11. Repetir pasos 9 y 10 hasta encontrar
    fin de archivo de A
12. Cerrar A
13. Cerrar el Archivo de Descriptores

Pseudocódigo de búsqueda del mínimo

```
BEGIN
 OPEN (Descriptor)
 IF CALL Existe (A) THEN
  IF CALL Existe (A (C)) THEN
   BEGIN
    OPEN (A)
    READ (A, Registro)
    CALL Extract (Registro, C)
    Minimo:= C
    REPEAT
     READ (A, Registro)
     CALL Extract (Registro, C)
     IF C < Minimo THEN
       Minimo:= C
    UNTIL EOF (A)
    CLOSE (A)
   END
  ELSE
   BEGIN
    WRITE ('Conflicto: el archivo no contiene el campo
            requerido')
   END
 ELSE
  BEGIN
   WRITE ('Conflicto: el archivo no existe')
  END
 CLOSE (Descriptor)
END.
```

CAPITULO VI

"Nunca es más negra la noche,
que cuando está cerca el
amanecer"

DISEÑO DEL LENGUAJE DE COMANDOS

Introducción.

Para el óptimo manejo de los elementos que conforman una Base de Datos, es necesaria la presencia de un lenguaje que el usuario pueda utilizar en una forma sencilla pero a la vez potente.

Generalmente, el usuario se encuentra en la situación de que la utilización del sistema de Base de Datos le resulta compleja, debido sobre todo a la cantidad de elementos con los que cuenta. Por ello es necesario brindarle un lenguaje sencillo, a través del cual pueda manejar aquellos elementos de manera eficiente. Este lenguaje tiene que cumplir dos requisitos básicos: el primero es que sea sencillo en su estructura, para que de esta forma se explote fácilmente; el segundo requisito es que debe ser lo suficientemente capaz de albergar una amplia gama de aplicaciones, constituyéndose así su carácter sólido y eficiente. Este lenguaje debe permitir que el usuario vaya formulando sus requerimientos de forma conversacional, para realizar así su petición como más le convenga.

6.1. Diseno del Lenguaje de Comandos.

El lenguaje de manipulación de datos, es decir, el conjunto de operadores suministrados para manipular los datos que se representan en forma relacional es al menos igualmente importante que la estructura física de la Base de Datos.

La uniformidad de la representación de los datos conduce a una misma uniformidad en el conjunto de operadores; puesto que la información se representa de una sola manera, se necesita tan sólo un tipo de operador para las funciones básicas que se deseen realizar (esto contrasta con la situación de estructuras más complejas, donde la información puede representarse de varias maneras y, por tanto, se requieren varios tipos de operadores).

Por ejemplo, para la recuperación de datos, el operador básico que se necesita es "obtenga el siguiente", el cual traerá el renglón siguiente de una tabla que se esté ocupando ("siguiente" se interpreta en relación a la posición actual, como el posterior o como el primer renglón en caso de dar inicio a la recuperación de los datos).

A continuación se explicarán brevemente cada una de las herramientas y utilitarios que se definen en este trabajo.

ELIMINA.- Suprimir del archivo de descriptores y de la base de datos, el archivo seleccionado.

ACTUALIZA.- Modificacción directa a los campos de cada registro.

CAPTURA.- Anadir datos en el archivo seleccionado.

SUPRESION.- Borrado de todos los registros a través de una inspección al archivo en cuestión.

ORDENA.- Organiza los registros en forma ascendente o descendente, del archivo en turno.

COPIA.- Crea un archivo con la misma estructura e información definida con anterioridad.

FACTOR.- Anade, al final de un archivo, los registros de otro.

PERMUTA.- Rota los caracteres del campo seleccionado, en el registro de un archivo.

MAXIMO.- Determina el registro que tiene el campo mayor a todos los demás registros del archivo.

MINIMO.- Determina el registro que tiene el campo menor a todos los demás registros del archivo.

SELECCION.- Determina de un archivo dado, y de acuerdo a una condición especifica, todos los registros que satisfagan la condición.

UNION.- Concatenación vertical de los archivos cuya estructura es la misma.

DIFERENCIA.- Obtener de dos archivos uno que contenga los registros del primero que no estén repetidos en el segundo.

INTERSECCION.- Determina los registros que son comunes a dos archivos.

PRODUCTO CARTESIANO.- Concatenación horizontal de dos archivos, asociando todos y cada uno de los registros del primero con el segundo.

COCIENTE.- Identifica el o los campos que forman pareja con el campo o conjunto de campos identificados en el registro de dos archivos.

JUNTA NATURAL.- Obtiene de dos archivos con campos en común, las ocurrencias repetidas del campo del primer archivo con todas las ocurrencias del campo del segundo archivo (puede ser uno o varios campos).

MEZCLA.- Concatenación intercalada de los registros de dos archivos.

PARTICION.- Separación de un archivo en dos archivos diferentes.

CREACION.- Agrega un registro descriptor con las caracteristica de un nuevo archivo.

JUNTA CON OPERADOR.- Determina los registros de dos archivos con campos en común, que satisfagan una condición del primer archivo con todas las ocurrencias del segundo archivo.

PROYECCION.- Obtiene los campos de todas las ocurrencias seleccionadas en forma vertical.

El diseno del lenguaje de comandos se lleva a cabo a través de los siguientes ejemplos.

EJEMPLO 1. Se quiere borrar el archivo de becas.

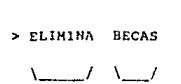
Para la resolución de este ejemplo se puede utilizar un comando con la siguiente estructura;

donde:

operando.- es la herramienta ó utileria que especifica qué acción se desea ejecutar.

operador.- es la entidad a la cual se aplicará la acción. En este caso es la tabla que se va a procesar.

el comando que se utilizaria para la solución de este ejemplo es 'ELIMINA', quedando especificado el comando de la siguiente manera:

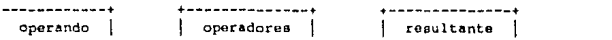


Esta estructura contiene la manera más sencilla de formular un comando, porque se requiere un número mínimo de elementos. Una variante de esta estructura se aprecia en la siguiente consulta.

EJEMPLO 2. Actualizar el archivo histórico de empleados a través del archivo mensual.

Considérese que el método para actualizar un archivo a través de otro es mediante la adición de registros de uno al final del otro

La estructura del comando que puede resolver esta consulta, es como sigue:



donde:

operando.- es la herramienta ó utilería que especifica qué se hará con las tablas.

operadores.- son las tablas que se van a procesar.

resultante.- es la tabla donde se escribirán los resultados.

la aparición de 'operadores' lleva a la situación de que los comandos pueden manejar dos tablas (realizándose así operaciones binarias), las cuales al procesarse , dan como resultado una sola tabla, denominada aquí 'resultante'.

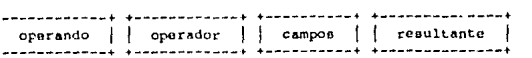
Así, el comando que se utilizaría es la 'UNION', el cual se escribe de la siguiente forma:

```
> UNION   HISTORICA   MENSUAL   RESULTADO
  \___/   \_____________/   \_____/
  opa--       operadores        resultante
  rando
```

Otro tipo de estructura, parecida a la anterior, es la que se aplica a las consultas como la siguiente.

EJEMPLO 1. Listar los nombres de los empleados de la tabla de empleados.

La estructura del comando sería así:



donde:

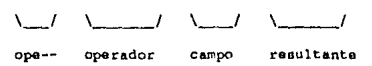
operando.- es la herramienta ó utilería que especifica qué se hará con la tabla.

operador.- es la tabla que se va a procesar.

campos.- es el campo ó conjunto de campos que se van a evaluar.

resultante.- es la tabla donde se escribirán los resultados.

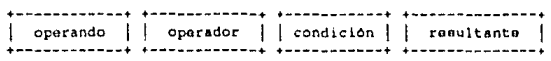
al agregar el elemento 'campos' a la estructura, ésta se torna más compleja puesto que 'campos' puede contener un número variable de elementos a evaluar, aunque para este ejemplo solo es uno, quedando la consulta como a continuación aparece;



Una estructura más sería la que se aplica a consultas del tipo que se muestra a continuación.

EJEMPLO 4. Dar los nombres de empleados, de la tabla correspondiente, cuya clave de puesto sea T-001.

La estructura es como la siguiente:



donde;

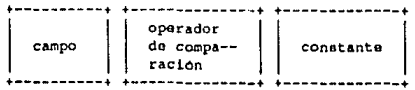
operando.- es la herramienta ó utilería que especifica qué se hará con la tabla.

operador.- es la tabla que se va a procesar.

condición.- implica un predicado, el cual al ser evaluado puede resultar falso ó verdadero.

resultante.- es la tabla donde se escribirán los resultados.

la variante de esta estructura es el elemento 'condición', que implica un predicado, el cual consta de:



donde;

campo.- es el campo a evaluar.

operador de comparación.- este puede ser alguno de los que se muestran a continuación:

```
>       ("mayor que")

<       ("menor que")

=       ("igual a")

>=      ("mayor o igual a")

<=      ("menor o igual a")

<>      ("diferente de")
```

constante.- es el valor contra el cual se compara cada campo seleccionado de la tabla.

por consiguiente, la consulta requerida quedaria como sigue:

```
> SELEC   EMPLEADOS   CVE-PU = 'T-001'   RESULTADO
  \___/   \_______/   \_______________/   \_______/
  opera-  operador       condición         resultante
  dor
```

Como se observa, todos los tipos de estructura presentados son similares entre si. De esta manera puede deducirse una estructura general o básica para los comandos, puede ser la que se muestra a continuación:

esta estructura se adapta a las herramientas y utilitarios que se utilizan en este sistema de Base de Datos, como lo muestra la figura 6.2 .

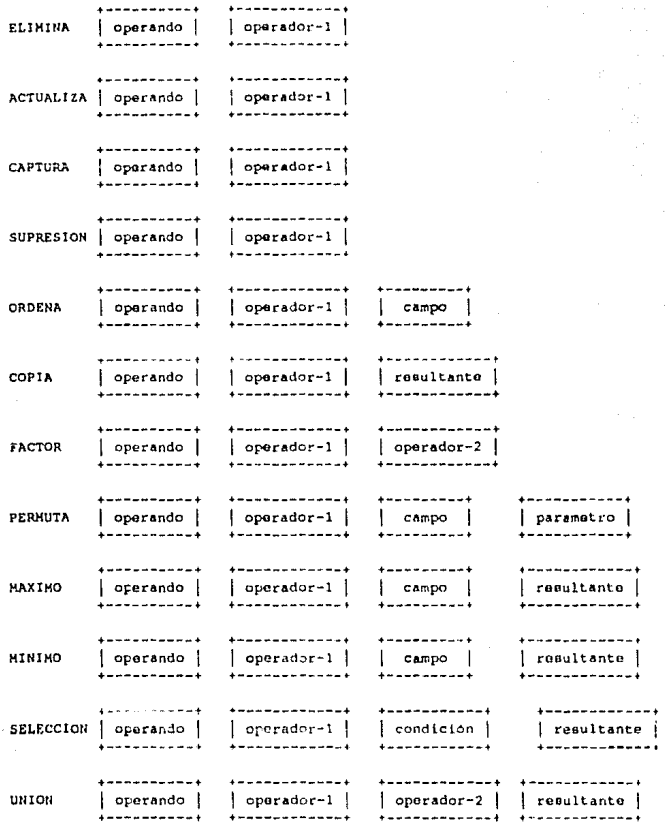


figura 6.2. Tabla de Operadores para una Base de Datos.

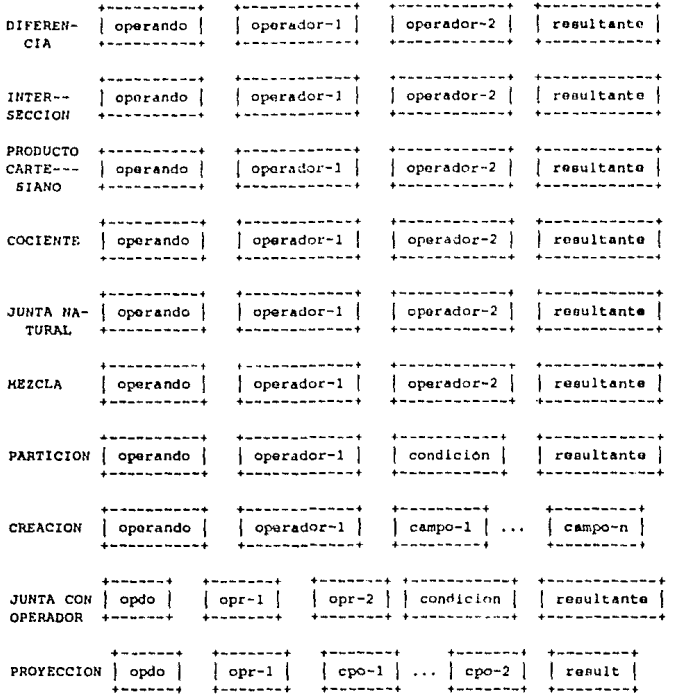


figura 6.2. Tabla de Operadores para una Base de Datos. (continuación)

6.2. APLICACION DEL LENGUAJE

Para la mejor comprension del lenguaje de comandos se darán unos ejemplos, estos se basan en las tablas siguientes:

T A B L A D E E M P L E A D O S

CVE-EMP	NOMBRE	CVE-PUESTO	ANTGDAD	HORARIO
E-001	Alfonso Ricaldo Curiel	P-001	2	MATUTINO
E-002	Juan Luis Vázquez Campos	P-005	2	ABIERTO
E-003	Teodoro Jimenez Castro	P-007	1	NOCTURNO
E-004	Cristobal Garcia Torres	P-004	2	MATUTINO
E-005	Eduardo Briseno Garcia	P-003	1	VESPERTINO
E-006	Teresa Ramirez Cárdenas	P-006	4	NOCTURNO
E-007	Javier Mosqueda Ramos	P-001	9	MATUTINO
E-008	Enrique Chavarria Lomeli	P-002	4	ABIERTO
E-009	Alfredo Maldonado Aceves	P-006	5	NOCTURNO
E-010	Jesus Alvarado Méndez	P-003	2	MATUTINO
E-011	Lilia Guerrero Martinez	P-008	1	ABIERTO
E-012	Julio Francisco Curiel Cardenas	P-005	4	NOCTURNO

T A B L A D E B E C A R I O S

CVE-BCA	NOMBRE	CVE-PUESTO	ANTGDAD	HORARIO
B-001	Andrea López Guerrero	T-001	2	MATUTINO
B-002	César Gutiérrez Carmona	T-005	2	MATUTINO
B-003	Everardo Sánchez Hernández	T-007	1	MATUTINO
B-004	Manuel Torres Lorenzo	T-004	2	MATUTINO
B-005	Carlos Bucio Albarrán	T-003	1	MATUTINO
B-006	Himelda Castro Millán	T-006	4	MATUTINO
B-007	Hermenegildo López Díaz	T-001	9	VESPERTINO
B-008	Doroteo Villa Alcázar	T-002	4	VESPERTINO
B-009	Sebastian Perez Gómez	T-006	5	VESPERTINO
B-010	Marco Antonio Salinas Arellano	T-003	2	VESPERTINO
B-011	Gustavo Pliego Cortés	T-008	1	VESPERTINO
B-012	Mario Chávez Soto	T-005	4	VESPERTINO

NOTA: la antigüedad para los empleados se contabiliza en anos, mientras que para los becarios se realiza en meses.

T A B L A D E P U E S T O S

CVE-PUESTO	DESCRIPCION	SUELDO
P-001	Operador	550,000
P-002	Analista	1'050,000
P-003	Programador	700,000
P-004	Capturista	400,000
P-005	Supervisor	1'100,000
P-006	Contador	1'000,000
P-007	Secretaria	600,000
P-008	Director	3'000,000

T A B L A D E B E C A S

CVE-PUESTO	DESCRIPCION	SUELDO
T-001	Teleproceso	350,000
T-002	Base de Datos	650,000
T-003	Sistemas Operativos	500,000
T-004	Análisis de Algoritmos	200,000
T-005	Calidad Total	900,000
T-006	Matemáticas Financieras	800,000
T-007	Taquigrafía	100,000
T-008	Toma de Decisiones	1'000,000

A continuación, se ejemplifican algunas consultas, que serán resueltas a través de las herramientas desarrolladas.

CONSULTA 1. Encontrar los becarios que tengan 5 ó más meses en la empresa, y promoverlos como empleados de planta.

Se toma como hecho que ya se decidió incluir a los becarios con la antigüedad señalada. Si se asume la existencia de dos archivos que contienen la información sobre los empleados y sobre los becarios, respectivamente, puede deducirse que la solución a este requerimiento se dá con la inclusión al archivo de empleados de todos aquellos becarios que cumplen con el requisito deseado. Supóngase además que una norma establecida señala que se realice un respaldo de seguridad para cada archivo que se va a modificar. Todo esto implica la realización de copias (los respaldos) y la utilización de otras herramientas que posibiliten la actualización del archivo de empleados a través de los becarios seleccionados. Esta solución puede dividirse en los siguientes pasos:

A) Duplicar la información de la tabla de 'BECARIOS' en 'RBECARIOS', para tener un respaldo de la tabla de becarios.

```
COPIA BECARIOS RBECARIOS
```

B) Determinar qué becarios tienen cinco o más meses en la empresa, generando una nueva tabla, llamada 'BECA5'.

```
SELEC BECARIOS ANTGDAD >= '5 M' BECA5
```

C) Elaborar una lista con los becarios que serán promóvidos como empleados.

```
PROYE BECA5 NOMBRE
```

D) Generar la tabla 'REMPLEADOS' a través de 'EMPLEADOS', para tener un respaldo de la información.

```
COPIA EMPLEADOS REMPLEADOS
```

E) Adicionar los becarios seleccionados al final de la tabla de empleados.

```
UNION EMPLEADOS BECA5 EMPLEADOS
```

F) Eliminar los becarios que fueron promovidos como empleados, en la tabla de becarios.

```
DIFER BECARIOS BECA5 BECARIOS
```

CONSULTA 2. Encontrar los becarios de 'Toma de Decisiones', además los capturistas que trabajan en el turno matutino.

Esta consulta puede resolverse de la siguiente manera:

A) Identificar en la tabla de becarios a los que están en el curso de 'Toma de Decisiones', cuya clave es T-008.

SELEC BECARIOS CVE-BECA = 'T-008' TRAB1

B) Determinar en la tabla de empleados, aquellos que sean capturistas, a través de la clave 'P-004'.

SELEC EMPLEADOS CVE-PUESTO = 'P-004' TRAB2

C) Asociar las tablas 'TRAB1' y 'TRAB2' para tener los registros de los empleados y los becarios seleccionados en una sola tabla: 'TRAB3'.

UNION TRAB1 TRAB2 TRAB3

D) Determinar, en la tabla 'TRAB3', a las personas que trabajan en el turno matutino.

SELEC TRAB3 HORARIO = 'MATUTINO' RESULTADO

CONSULTA 3. Encontrar los empleados y becarios que ganan menos de $700,000.00 y determinar qué puesto o beca tienen, para un posible aumento.

Para darle solución a este requerimiento, supóngase que se desea obtener una lista, que contenga los datos de los empleados y de los becarios, para lo cual es necesario brindar una solución como la que se presenta en los siguientes incisos:

A) Identificar de la tabla 'PUESTOS', los sueldos que sean menores de 700,000.00 .

```
SELEC PUESTOS SUELDO < 700,000 TRAB1
```

B) Identificar de la tabla 'BECAS' los sueldos que sean menores de 700,000.00 .

```
SELEC BECAS SUELDO < 700,000 TRAB2
```

C) Asociar las tablas 'TRAB1' y 'EMPLEADOS', para tener los registros de los empleados requeridos en la tabla 'TRAB3'.

```
JUNTA TRAB1 EMPLEADOS TRAB3
```

D) Asociar las tablas 'TRAB2' y 'BECARIOS' para tener los registros de los becarios solicitados en la tabla 'TRAB4'.

```
JUNTA TRAB2 BECARIOS TRAB4
```

E) Sumar las tablas 'TRAB3' y 'TRAB4', para tener los registros que darán la solución a la consulta requerida.

```
UNION TRAB3 TRAB4 RESULTADO
```

CONSULTA 4. Encontrar los empleados que tienen más de tres anos de antigüedad y ganan más de $700,000.00 , pero que no sean directores o analistas.

Esta consulta puede resolverse a través de los siguientes pasos:

A) Identificar de la tabla 'PUESTOS' los sueldos mayores de 700,000.

```
SELEC PUESTOS SUELDO > 700,000 TRAB1
```

B) Identificar de la tabla 'PUESTOS' las descripciones que sean diferentes a 'ANALISTA'.

```
SELEC PUESTOS DESCRIPCION <> 'ANALISTA' TRAB2
```

C) Identificar de la tabla 'PUESTOS' las descripciones diferentes a 'DIRECTOR'.

```
SELEC PUESTOS DESCRIPCION <> 'DIRECTOR' TRAB3
```

D) Confrontar las tablas 'TRAB2' y 'TRAB3', para obtener los puestos con la descripción requerida.

```
INTER TRAB2 TRAB3 TRAB4
```

E) Confrontar las tablas 'TRAB1' y 'TRAB4' para satisfacer todas las condiciones relacionadas a la tabla de puestos.

```
INTER TRAB1 TRAB4 TRAB5
```

F) Asociar la tabla 'TRAB5' con 'EMPLEADOS', pero sólo con los empleados cuya antigüedad sea mayor a 3 anos. Para así obtener el resultado solicitado inicialmente.

```
JUNTOP TRAB5 EMPLEADOS ( ANTGDAD > 3 ) resultado
```

CONCLUSIONES

"Este es el fin del principio"

- La utilización de la teoría de Base de Datos relacional es útil para toda área de aplicación, no únicamente para el área de las matemáticas, ya que por su sencillez se aplica a cualquier modelo de operación informativa.

- El manejo de longitudes fijas en archivos de tipo secuencial, es una capacidad que comparten todos los lenguajes de programación, por lo que el modelo de Base de Datos relacional es capaz de explotarse a través de una amplia gama de lenguajes, que van desde los más básicos hasta los complejos.

- A veces, las Bases de Datos no se explotan a través de lenguajes sencillos, pudiendo limitar así su aprovechamiento. Si bien, ellas se manipuladan a través de lenguajes muy poderosos, son generalmente poco asimilables por el usuario. Ahora bien, la sencillez en el lenguaje puede auxiliarlo en la óptima comprensión del mismo y como consecuencia, en su mejor aprovechamiento y del sistema de Base de Datos en general.

- El lenguaje de comandos desarrollado en este trabajo, pudiera parecer muy elemental, pero en realidad es lo suficientemente potente para permitir una confiada accesibilidad y, en consecuencia, un buen aprovechamiento de la Base de Datos utilizada. Precisamente es su sencillez la que posibilita el óptimo manejo que haga el usuario sobre el mismo; esta persona puede ser alguien familiarizado o no con el manejo de lenguajes de aplicación.

- La presentación que se hace del lenguaje es en forma teórica, puesto que no es objetivo del presente trabajo enfocarse sobre la teoría o desarrollo de un compilador para el reconocimiento de comandos. Trata, sin embargo, de exponerse el lenguaje que posibilite una óptima utilización de los elementos del sistema de Base de Datos.

BIBLIOGRAFIA

Autor : Date, C. J.

Titulo : Introducción a los Sistemas de Base de Datos

Editorial : Addison-Wesley Iberoamericana.

Autor : Kernighan Brian W. / Plauger P. J.

Titulo : Software Tools

Editorial : Addison-Wesley Publishing Company

Autor : Ullman, Jeffrey D.

Titulo : Principles of Database Systems

Editorial : Computer Science Press

Autor : Chapa Vergara, Sergio V.

Titulo : Herramienta para Consulta y Captura basada en el Descriptor de Archivos

Editorial : Centro de Investigación y Estudios Avanzados del I.P.N., Depto. de Ingeniería Eléctrica. Reporte Técnico # 15, 1985.

Autor : Borland International Inc.

Titulo : Turbo Pascal 3.0

Editorial : Scotts Valley Ca., U.S.A.
Borland International Inc., 1987.

Autor : Henry F. Korth / Abraham Silberschatz

Titulo : Fundamentos de Bases de Datos

Editorial : McGraw-Hill

Autor : varios

Titulo : Enciclopedia de Informática (4 tomos)

Editorial : Nueva Lente / Ingelek. España

Autor : Ullman, Julian

Titulo : A Pascal Database Book

Editorial : Oxford Applied Mathematics and Computing Science Series

Autor : Lyon, John K.

Titulo : The Database Administrator

Editorial : John Wiley and Sons

Autor : Wiederhold. Gio

Titulo : Diseno de Base de Datos

Editorial : McGraw-Hill

Autor : Delobel, Claude and Adiba, Michel

Titulo : Relational Database Systems

Editorial : North Holland