Universidad Nacional Autónoma de México

FACULTAD DE CIENCIAS

BASE DE DATOS. HISTORIA.

TECNOLOGIA Y UNA APLICACION.

T E S I S

QUE PARA OBTENER EL TITULO DE:

ACTUARIO

P R E S E N T A:

VICTOR ILDEFONSO CAMPOS CORONA

1982

I N D I C E

Febrero de 1982.

I N T R O D U C C I O N

Base de Datos es una colección organizada de datos acerca - de "algo". Por ejemplo un inventario de materiales, una biblioteca reflejan este concepto.

El desarrollo de las Bases de Datos colectivas ha sido durante los últimos años una de las actividades más importantes en el campo de la Informática.

A finales de los sesentas y principios de los setentas se - desarrollaron grandes aportaciones a la Tecnología del Manejo de Datos en virtud del enorme crecimiento de los usuarios de computadoras, quienes demandaban por una herramienta cada vez mas poderosa, eficiente y completa en los Sistemas de Manejo de Datos.

Es impresionante observar como crecen en volúmen e importancia los archivos de datos que utilizan las computadoras, por lo mismo los fabricantes de la rama, vendedores, investigadores etc. se empezaron a dar cuenta de la importancia de desarrollar una - serie de programas de apoyo que optimizaran las siguientes funciones:

- El manejo de programas que fueran independientes tanto de su almacenamiento como de su acceso.
- La flexibilidad al relacionar registros.
- Evitar la redundancia de información.
- Eficiencia en la ejecución.
- Seguridad en el uso.

fué entonces que surgieron los Sistemas de Manejo de Base de Datos, y es en este punto donde este trabajo tiene su razón de ser.

La idea fundamental de esta Tesis es mostrar por medio de - una aplicación, las grandes facilidades que ofrece uno de estos-

modernos sistemas en el desarrollo de una base de datos. va dirigido a la enorme población de técnicos y profesionistas principiantes en el área de la informática.

A traves de la experiencia vivida en terreno laboral y estudiantil, tanto a nivel de mis compañeros como a nivel personal la gente que se inicia en el extenso campo de la informática, tiene un problema inmediato que enfrentar; "los manueles". este problema existe por dos razones fundamentales:

a) La dificultad de interpretarlos correctamente, lo que origina un concepto desvirtuado de las instrucciones.

b) Cuando en la práctica se presenta alguna anomalía, la posible solución sugerida por el manual no siempre es acorde con la realidad, lo que ocasiona grandes dolores de cabeza y una gran pérdida de tiempo en la averiguación de la solución real.

En el caso particular de un manual de operación de un manejador de base de datos, encontraremos ejemplos en el tercer capítulo que nos permitirán la mejor comprensión de sus instrucciones, lo que pretende llegar a ser de utilidad para alguna persona que desarrolle una base de datos.

Una facilidad que pretende ofrecer este trabajo es la independencia de información en los capítulos para efectos de consulta. En virtud de ser el tercero de estos el principal componente trataremos de situarlo con mayor detenimiento.

Autores famosos en la rama tales como James martin y David Kronke concuerdan en que es posible y probable que una organización pudiera tener diversas bases de datos, cada una de estructura separada, pero disponible en cualquier combinación posible y requerida para dar soporte a una aplicación dada.

En vez de pensar en una Base de Datos para una empresa, — sería más apropiado pensar en el ambiente de una Base de Datos- como una colección de Bases de Datos (cada una, una colección - de datos orientados a un tema) en contraposición con la clásica colección de archivos (cada uno orientado a una aplicación - específica). El problema que encara la mayoría de las organi - zaciones es como evolucionar, desde este último modo, al ante - rior.

Pues bien pensando en un ambiente de Base de Datos, nues - tro trabajo se acopla como parte de una colección de Base de Da tos, orientada a un tema específico que sería el Control de Requisiciones y Pedidos de una empresa, desde luego considerando- esta colección como la meta ideal para conjuntar un sistema de- información administrativo complejo que hasta el momento no ha- sido posible desarrollar al 100% en ninguna institución, debido en primer lugar a que la mayoría de las organizaciones encuen - tran que deben soportar simultáneamente programas de aplicación nuevos y convertidos, que satisfagan el ambiente de la Base de- Datos, al igual que programas anteriores de aplicación, que de- pendan de la que corresponde a archivos científicos. En segun- do lugar, los costos adicionales de utilizar un Sistema de Ma - nejo de Base de Datos son apreciables a pesar de los costos con tinuamente menores de la potencia de computo.

Al encarar estas dificultades, algunas organizaciones dife rirán la decisión de comprometerse a un ambiente de Base de Da- tos, (como hasta la fecha ha ocurrido en grandes empresas) y en ves de ello utilizarán en forma selectiva un Sistema de Manejo- de Base de Datos como herramienta para implementar las aplica - ciones en su medio de datos actual, y es en este contexto como- se desarrolló nuestra Base de Datos.

CAPITULO I

ORGANIZACIONES TRADICIONALES DE DATOS

Existen varios tipos de organización de datos que han sido usados tradicionalmente, este hecho se debe por un lado a que los usuarios de computadoras quienes almacenan, recuperan y actualizan datos se enfrentan en general a dos situaciones; una es cuando los datos deben ser recuperados rapidamente pero su actualización puede efectuarse en una fase mas lenta, esta situación se dá en los sistemas de recuperación de información, la otra situación se dá cuando es necesario almacenar rapidamente grandes volúmenes de datos, los cuales son recuperados más tarde en una etapa mas lenta como se efectúa en registro de datos en Tiempo Real. Cada tipo de situación puede utilizar una diferente organización de datos tal que satisfaga las necesidades de la aplicación y al mismo tiempo conserve el gasto en un mínimo.

por otro lado la existencia de más de una organización de datos se debe a que las características de los dispositivos en los cuales los datos están almacenados no son siempre compatibles con las demandas de la aplicación. por lo tanto diferentes organizaciones de datos son utilizadas para cerrar la brecha entre los requerimientos lógicos del usuario y las características físicas de estos dispositivos de almacenamiento.

Es conveniente notar que todos los métodos de organización de datos son construídos a partir de tres organizaciones básicas.

1.- SECUENCIAL
2.- RANDOM
3.- LISTA

Veremos como éstas organizaciones son la base de organizaciones más complejas, las cuales son frecuentemente más deseables, tomando ésta afirmación como resultado de la exposición.

ORGANIZACION SECUENCIAL

Quizá la organización de datos mas conocida sea la secuencial, donde los registros son almacenados en posiciones relativas a otros registros de acuerdo a una secuencia específica.

Para ordenar los registros en una secuencia, un atributo común de los registros es elegido. Cuando es un campo dentro del registro, el atributo seleccionado para ordenar los registros en el archivo se le llama llave.

Es posible almacenar registros sin llaves, esto ocurre cuando los registros son almacenados en orden de su llegada al sistema, en tal caso cada registro está posicionado secuencialmente siguiendo al registro precedente. En ambos casos el orden lógico de los registros en el archivo y el orden físico de los registros en los dispositivos de grabación es el mismo

Un archivo de organización secuencial puede estar compuesto de registros de diferentes tipos, pero los registros son agrupados en un archivo porque tienen un propósito funcional común.

La ventaja de esta organización es el acceso rápido durante recuperación de información. Si elmecanismo de acceso se activa para recuperar un registro particular, puede rapidamente accesar el siguiente registro en la estructura de los datos de acuerdo a la relación establecida cuando los datos fueron almacenados.
Cuando los registros son almacenados en secuencia, en disco o en memoria de alta velocidad, es posible efectuar una búsqueda binaria. Esta consiste en partir los datos por en medio, eliminando la mitad de los casos en una comparación; la mitad remanente es entonces dividida a la mitad y así sucesivamente hasta llegar a una búsqueda secuencial de una pequeña porción del archivo y se establece si el registro deseado existe o no existe en el archivo.

La ventaja de poder accesar rápidamente el registro siguiente se convierte en desventaja cuando en un archivo se efectuará una búsqueda hasta que un registro con un valor de llave particular es encontrado. Aquí el primer registro es examinado, si la llave no es la correcta, se examina el siguiente y el proceso continúa hasta que el registro correcto es encontrado.

Actualizar valores en un registro almacenado en una organización secuencial es también dificil. Si el nuevo registro es mas chico o mas grande que el registro original, se corre el riesgo de que registros adyacentes en el archivo pueden ser destruídos. Cuando agrupamos registros para integrar una unidad de transmisión de datos denominada bloque, con el propósito de conservar el espacio de almacenamiento e incrementar la eficiencia en el acceso y durante el proceso decimos que los registros estan bloqueados, la actualización en estos registros es imposible a menos que el bloque entero sea reescrito.

Por éstas razones un archivo con organización secuencial es usualmente actualizado mediante la copia de registros de un archivo a otro, haciendo tantas actualizaciones en los registros como las necesiten. Se vuelve muy caro actualizar un registro en una organización secuencial, usualmente la actualización ocurre sólo cuando varios registros se van a alterar.

Es difícil insertar nuevos registros o remover registros viejos de un archivo con organización secuencial. El proceso de inserción requiere que los registros que ya están almacenados empujen para hacerle lugar a un nuevo registro, lo que implica la copia del archivo entero. La conversión es verdadera para remover registros en tal caso los registros existentes son empujados juntos. Se pueden agregar registros fuera de secuencia, y en una subsecuente actualización del archivo se clasifican en la secuencia correspondiente.

Las Organizaciones de Listas, las cuales describiremos más adelante, proveen una más fácil opción de inserción y remoción- y evitan los problemas de copia de la organización secuencial.

A menudo es deseable almacenar y recuperar registros usando más de una llave. Un caso en punto medio es un registro de- un archivo de personal, el cual será almacenado y accesado ya - sea por el nombre o la ocupación de la persona que describe al- registro. Este es más facilmente establecido, mediante el alma cenamiento del dato en archivos duplicados, así en el primer - archivo se efectuó una búsqueda si la primera llave es conocida y en el segundo archivo se efectúa la búsqueda si la segunda -- llave es conocida. Sin embargo esta duplicidad utiliza dos -- veces el espacio de almacenamiento y duplica el costo de mante- nimiento, debido a que los registros que van a actualizarse -- deben ser accesados y cambiados en ambos archivos. En resumen, La Organización Secuencial siempre permite acceso rápido al si- guiente registro en el archivo, pero plantea dificultades en la- actualización del archivo y en la recuperación de registros -- fuera de secuencia. Para corregir esta desventaja, una segunda técnica de manejo de datos ha sido desarrollada; es llamada -- Organización Random.

ORGANIZACION RANDOM

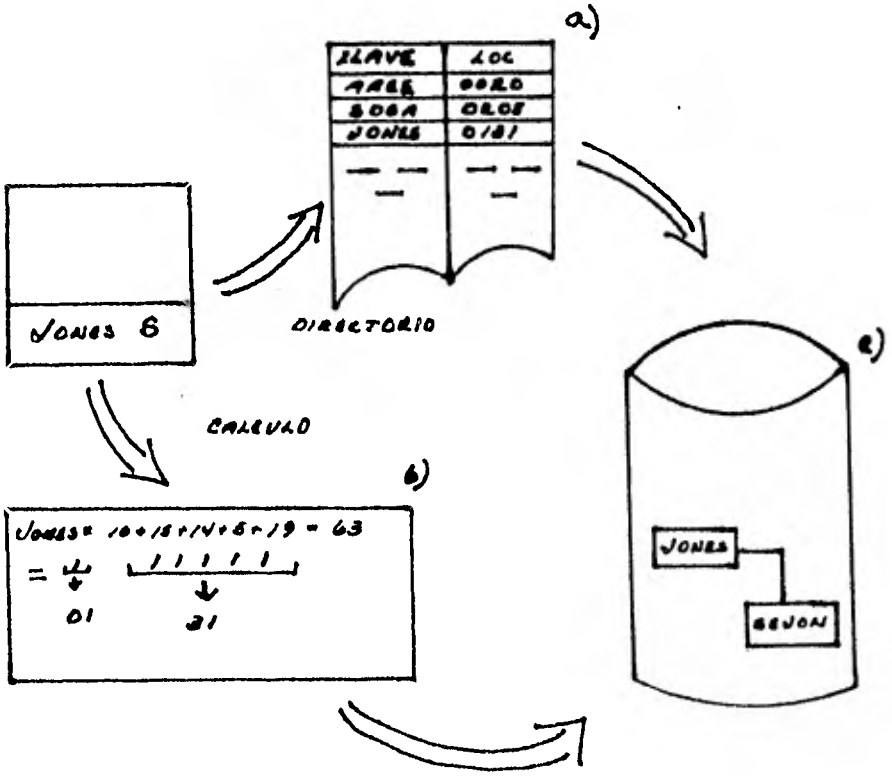
En esta organización, los registros son almacenados y recu perados en la base de una relación predicible entre la llave -- del registro y la dirección directa del lugar donde están alma- cenados los registros. La dirección es usada cuando los regis- tros son almacenados y utilizada otra vez cuando los registros- son recuperados.

En virtud de que la relación entre la llave y la dirección del registro es muy importante para el manejo de datos organizados en forma random, examinaremos los 3 métodos para acceso de registros en los dispositivos de la dirección directa que son: Dirección Directa, Directorio y Cálculo.

DIRECCION DIRECTA

En la forma mas simple, la Dirección Directa es conocida por el programador y es suministrada en tiempos de almacenamiento y recuperación. El Hardware (equipo físico, todos los dispositivos mecánicos, magnéticos o electrónicos componentes de una computadora) entonces utiliza esta dirección para accesar el registro en los dispositivos de almacenamiento.

La siguiente figura nos ilustra los metodos de Directorio y Cálculo por medio de los cuales se obtiene la dirección directa de un registro antes del almacenamiento o de la recuperación.



DIRECTORIO

Cuando este método es utilizado, ambos la llave de registros y su dirección directa son almacenados en un directorio tal como lo mostramos en la figura anterior inciso a).

Cuando un registro es almacenado o recuperado, la llave se encuentra en el directorio y la correspondiente dirección directa es utilizada. Por ejemplo la llave "Jones" es comparada con el directorio y la dirección directa 0131 es escogida. Entonces esta dirección se utiliza para almacenar o recuperar al registro.

El uso de un directorio implica que cada registro tiene una dirección única. Sin embargo, para hacer esto el directorio debe ser suficientemente largo para incluir todas las direcciones directas potenciales y esto puede ocupar tanto espacio como los datos mismos. También la busqueda secuencial paso a paso en un directorio puede echar por la borda las ganancias obtenidas al tener direcciones de registros únicas. Si los directorios son muy largos, son con frecuencia accesados por medio de una búsqueda binaria, o bien tener una organización de árbol (descrita más adelante) para reducir el tiempo de la búsqueda.

CALCULO

Este método se encarga de convertir la llave de un registro en una dirección directa, la cual no es necesariamente única. En nuestra figura un método muy simple es utilizado. Cada letra de la llave es reemplazada por un número. La J por el número 10 (debido a que la J es la décima letra del alfabeto), la O por 15, la N por 14, la E por 5 y la S por el número 19. Estos números son sumados y el resultado se convierte a un número binario, cuyos cinco bitts más a la derecha forman dos dígitos octales, los cuales son los dos dígitos más a la derecha-

de la dirección, con los bits remanentes se forman los dígitos de la parte izquierda de la dirección. El resultado es la dirección directa del registro "Jones". Se muestra en el inciso b) de nuestra figura.

Los metodos de dirección directa y directorio siempre determinan una única dirección. Sin embargo el método de cálculo puede causar que dos diferentes llaves al calcularse tengan la misma dirección directa. En nuestra figura el registro que tenga la llave "Jones" y el registro que tenga la llave "Sejon" -- tienen la misma dirección directa debido al algoritmo de cálculo empleado (inciso c). Esto causa colisión y puede ser manejado mediante el uso de un apuntador en el primer registro en una dirección particular para señalarlo del registro en estado de colisión, el cual está en algún otro lugar del dispositivo de almacenamiento. El apuntador de este registro está en la dirección directa del siguiente registro. En este ejemplo tanto "Jones" como "Sejon" tienen la dirección 0131 en el cálculo. Los mecanismos de almacenamiento o recuperación accesarán la dirección 0131 y examinarán la llave del registro ahí almacenada. Si la llave es aquella del registro deseado, el registro es recuperado, si la llave no es aquella del registro deseado, el apuntador del registro en estado de colisión es utilizado para recuperar otro registro con la misma dirección calculada. La llave de este registro es examinada para ver si se trata del registro correcto y el proceso continúa hasta que el registro correcto es encontrado. Otros métodos de cálculo pueden ser usados para reducir la ocurrencia de colisiones y para producir una más uniforme distribución de registros en los dispositivos de almacenamiento. La ventaja de utilizar la organización random es que -- cualquier registro puede ser recuperado en un sólo acceso. Otros registros en el archivo no tienen que ser accesados y sus lla -

ves examinadas, como en el método secuencial, cuando se trata de recuperar un registro particular. Registros individuales pueden ser almacenados, recuperados y actualizados sin afectar otros registros en los dispositivos de almacenamiento. Sin embargo, todos los registros son generalmente de una longitud uniforme.

Este requisito es algunas veces impuesto por las direcciones de hardware que referencian el dispositivo de almacenamiento y también permite el manejo mas fácil de registros en estado de colisión.

Las organizaciones Random son frecuentemente usadas en forma conjunta con la dirección directa de los dispositivos de almacenamiento, y la dirección directa de un registro corresponde a una dirección de hardware en el dispositivo de almacenamiento. Aunque las organizaciones Random permiten un acceso rápido de un registro particular con una llave conocida, no son demandadas para accesar rapidamente un número de registros. Esta limitación es impuesta por el tiempo que se toma el mecanismo de acceso del hardware en localizar un registro. Otras dificultades son:

1) Manejo del problema de colisiones cuando utilizamos calculos para obtener la dirección directa.
2) Manejo de largos y exhaustivos directorios cuando utilizamos el método de directorio.

ORGANIZACION DE LISTA

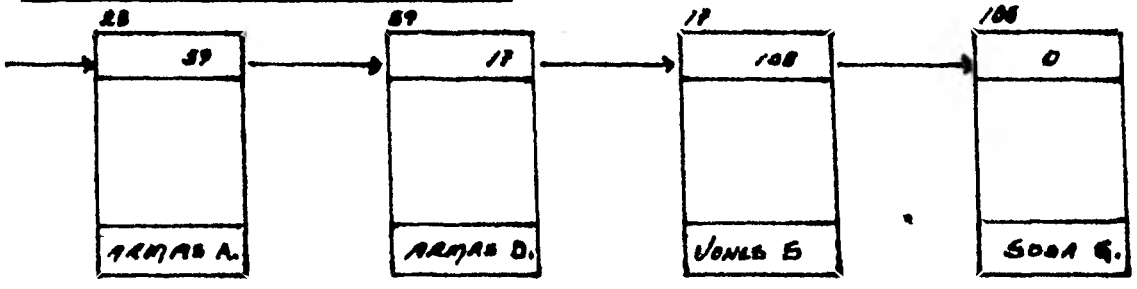
El uso de apuntadores para manejar el problema de colisiones en una organización Random sugiere una tercera forma de organización de datos, llamada organización de Lista. Hay tres tipos de organización de Lista:

a) Lista simple
b) Lista invertida

c) Anillo (Ring)

El concepto básico de una Lista es que los apuntadores son usados para separar la organización lógica de la organización física. En una organización secuencial el siguiente registro - lógico es también el siguiente registro físico. Sin embargo si incluímos con cada registro un apuntador al siguiente lógico, - los arreglos físico y lógico pueden se completamente diferentes. Un apuntador puede ser cualquiera, el cual permitirá al meca — nismo de acceso localizar un registro. Si los registros son — almacenados en la media de la dirección directa, esta es la dirección directa del registra; si los registros están en memoria central esta es la dirección central en el registro.

ESTRUCTURA DE LISTA SIMPLE



Esta figura es un ejemplo de Lista Simple en la cual cuatro registros son almacenados en un órden lógico secuencial. Sin em bargo, no están en el mismo órden físico. El primer registro - está en el lugar 23, el segundo en el 59, y el tercero y cuarto- en los lugares 17 y 105 respectivamente. La organización lógica secuencial es obtenida por el uso de los apuntadores. Inicial - mente, hay un apuntador al primer registro. Este primer regis - tro contiene un apuntador, el 59, al segundo registro, el cual - contiene un apuntador, el 17, al tercer registro, y así hasta - completar la lista. El último registro en la lista está desig -

nado por un símbolo especial en este caso, el 0.

En virtud de que cualquier campo en un registro puede ser - considerado como llave, muchas listas pueden pasar através de un sólo registro.

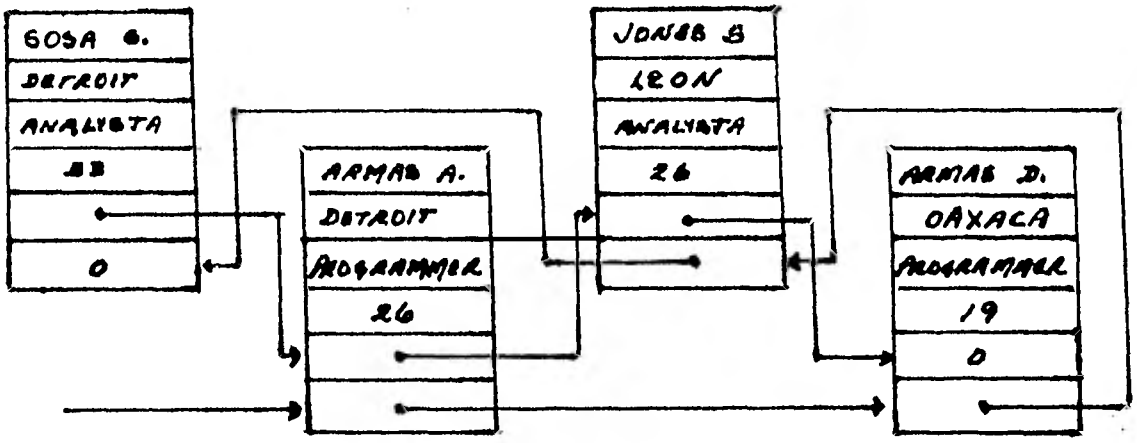


FIGURA 2. Dos listas navegando através de un registro.

Esta figura nos muestra varios registros. Cada registro - contiene cuatro campos elementales de datos: nombre, ciudad, ocupación y edad. Cada registro es también un miembro de dos listas. En la Lista superior los registros están lógicamente relacionados por el nombre de la ciudad, y en la Lista inferior, -- están lógicamente relacionados por el nombre de la persona. En cada una de estas listas, la ordenación es alfabética. Esto permite a los registros estar en múltiples Listas, y así la duplicidad es evitada.

Si un registro va a ser actualizado, ésta función puede efectuarse en ambas listas automáticamente, mediante la actualización en sólo una de estas listas. Los registros pueden estar en

cualquier lugar dentro de una lista mediante el cambio del apuntador del registro precedente al nuevo registro e insertando un- nuevo apuntador en el registro que acaba de ser dado de alta.

Por ejemplo si quisieramos insertar un registro entre los - registros "Jones" y "Sosa" en la figura 1, el apuntador en el registro "Jones" será puesto señalando al nuevo registro, y al nuevo registro se le dará el apuntador 105, el cual entonces apunta al registro "Sosa". La remoción de un registro de una lista requiere cambiar el apuntador en el registro precedente, para que- ahora apunte hacia el registro siguiente del que fué removido. Cuando un registro participa en una sola Lista, la remoción no - es difícil porque el registro es accesado inicialmente del registro precedente. Sin embargo, si un registro es un miembro de -- varias Listas, se dificulta en encontrar los registros precedentes de todas las Listas, y apuntadores extras deben ser almace - nados para permitir que tanto el precedente como el siguiente registro de la Lista puedan ser encontrados con mayor facilidad.

Con archivos grandes las Listas tienden a ser largas y si - la longitud de estas no se controla de algún modo pueden ser requeridas extensas búsquedas. Donde la longitud de la Lista no - está restringida, hay un sólo punto de partida para cada Lista.- Donde la longitud de la Lista es restringida, la solución es -- crear sublistas, cada una de las cuales tiene su propio punto de partida. Esto reduce la búsqueda a unos índices expandidos de - puntos de partida.

La figura 3 ilustra un archivo parcialmente invertido en el cual los datos de cada registro han sido puestos dentro de un -- índice (Index). Cada entrada en el Index tiene un punto de partida a la Lista de registros que tienen el mismo valor de la llave en el Index;"Jones" y "Sosa" son "analistas" y"Armas A" y --

"Armas D" son programadores. Observe que los datos han sido removidos de cada registro y residen en un lugar en el Index con - el gasto de agregar un apuntador a cada registro. Si el apuntador es más chico que el campo reemplazado, el registro ocupa menos espacio.

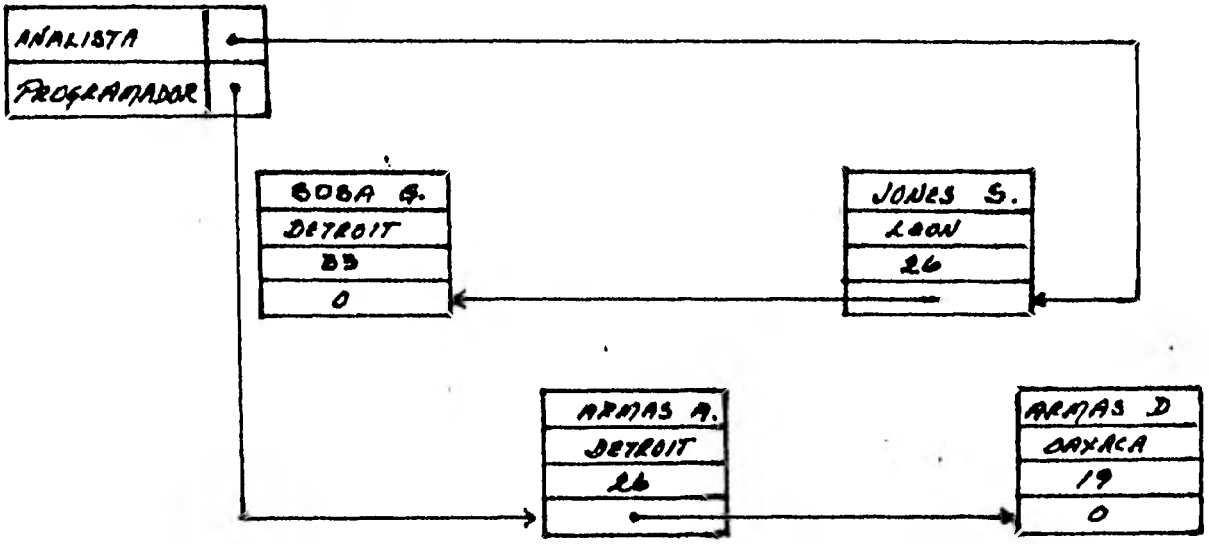


FIGURA 3.- LISTA PARCIALMENTE INVERTIDA

ANALISTA PROGRAMADOR	105. 17 23. 89
19 26 23	89 23. 17 105
DETROIT LEON OAXACA	23. 105 17 89

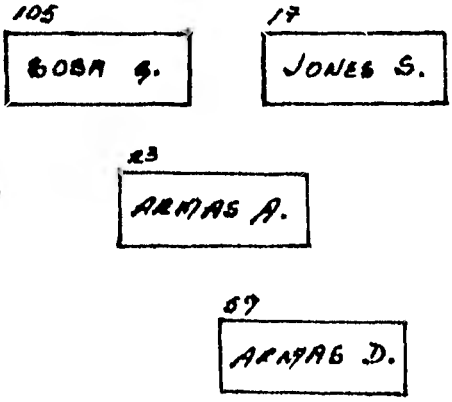


FIGURA 4.- ESTRUCTURA DE LISTA INVERTIDA

ESTRUCTURA DE LISTA INVERTIDA

Cuando la restricción en la longitud de la Lista es tomada - como última conclusión, la longitud de la Lista está restringida- a uno y cada llave aparece en el Index. Este así apunta directamente al registro y no se requiere de apuntadores. La Lista se - ha convertida ahora en invertible. La figura 4 es un ejemplo de- una lista invertida.

Aqui todos los apuntadores son guardados con el Index. Si - por ejemplo, se desean todos los registros que tengan la llave - "Analista", la palabra "Analista" es encontrada en el Index y los apuntadores 105 y 17 son obtenidos. Este apunta a los registros- de datos actuales mostrados en la parte derecha de la figura 4.

Mediante la conservación de los apuntadores con el Index, un requerimiento de información puede ser obtenido mediante la pri - mera manipulación de apuntadores en el Index. Una vez que los -- apuntadores correctos de los datos son encontrados, son usados -- para recuperar los datos del archivo. Por ejemplo, asumamos que- se requiere un dato de información de todos los analistas que viven en León. La entrada "Analista" en el Index muestra que los - registros con apuntadores 105 y 17 satisfacen la primera parte -- del requerimiento. Un exámen de la entrada "León" en el Index -- muestra que el registro que contiene un apuntador de valor 17 satisface el requerimiento. Ahora comparando las entradas bajo estos dos índices, se concluye que sólo el registro que contiene el apuntador 17 satisface el requerimiento inicial. Se hace enton - ces un acceso al archivo y el registro que contiene el nombre -- "Jones" es recuperado. Mediante esta conservación de apuntadores con el Index, es posible procesar un requerimiento sin tener que- ir al archivo. Por ejemplo, procesemos el requerimiento de recuperar los nombres de los programadores que viven en "León". La - entrada "Programador" en el Index muestra que los apuntadores 23

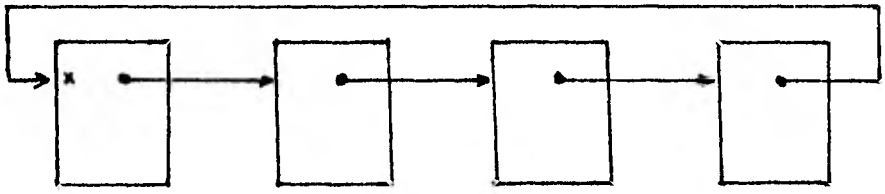
y 59 son relevantes, mientras que la entrada "León" en el Index-muestra que el apuntador 17 es relevante. Comparando estas dos-Listas de apuntadores llegamos a la conclusión de que no hay información en el archivo sobre programadores que vivan en "León"- y el acceso al archivo nunca se lleva a cabo.

La Organización de Lista Invertida hace disponible cada -- campo de dato como una llave. Tal organización requiere un diccionario de todos los valores de datos en el sistema conteniendo las direcciones de todos los lugares donde tales valores ocurren. El directorio puede ser tan largo o aún más largo que los datos-mismos. La virtud de tal sistema es que permite acceso a todos-los datos con igual facilidad. Consecuentemente es más conve -- niente para situaciones en que los requerimientos de recupera -- ción de datos son menos predecibles, por ejemplo, toma de deci - siones y planeación de funciones, más que funciones de proceso - específico. Así la aproximación de Lista Invertida se presta a-fácil recuperación, almacenamiento y actualización de datos es - más difícil, debido al mantenimiento de largos diccionarios. Es mejor combinar una organización de Lista Invertida con cualquiera de las dos siguientes: Random ó Secuencial. De este modo -- los registros son invertidos en sólo una o dos llaves más bien - que en todas las llaves. Los diccionarios se reducen y es todavía posible accesar todos los registros en el archivo.

ESTRUCTURAS DE ANILLOS

Los Anillos son una extensión de una organización de Lista. Como observamos en la figura 1 el fín de la lista está distinguido con ún 0, valor para el último apuntador. En un anillo, el - último registro apunta hacia el primer registro de este anillo y un símbolo especial es puesto y conservado en el primer registro para designar que este es el primero, o el punto de partida del-

anillo, tal como lo vemos en la siguiente figura:



Es posible seguir el anillo hasta encontrar cualquie anillo, ya sea el registro precedente, el registro siguiente, ó el registro inicial del anillo. Es posible como lo muestra la figura 6 - que multiples anillos puedan pasar através de un anillo. El anillo primario va de izquierda a derecha; sin embargo, un segundo - anillo va de derecha a izquierda también conectando cada registro. Es por consiguiente posible recoger el siguiente ó el precedente- registro mediante la obtención y uso del apuntador adecuado. Apuntadores extras pueden también ser almacenados en cada registro -- para apuntar al punto de partida del anillo. La estructura Coral mostrada en la figura 7 hace uso de una estructura de anillos en- la cual cada registro apunta al siguiente registro en el anillo.- Todos los registros del anillo excepto el registro inicial, tie - nen un segundo apuntador, el cual apunta ya sea hacia el registro inicial o es usado como apuntador hacia atrás. Los apuntadores - de regreso apuntan al elemento previo más cerca con un apuntador- de regreso, así forman un anillo completo. En virtud de que los- anillos son procesados más frecuentemente en la dirección delan - tera, alternar los apuntadores de regreso menos útiles y los apuntadores de registros iniciales retienen la ventaja ambos de estos apuntadores menos útiles en la mitad del espacio.

La Estructura de Anillos resulta ser muy poderosa, ya que - abastece de una facilidad para recuperar y procesar todos los registros en cualquier anillo mientras se ramifique en cualquier -- registro, o en cada uno de los registros para recuperar y proce -

sar otros registros los cuales están logicamente relacionados.

Tales registros quedarán también almacenados en una estruc - tura de anillo y en su oportunidad permitirán la misma facilidad: este anidamiento es transportado através de cualquier nivel re - querido por las relaciones lógicas en los datos. Estas organi - zaciones más complicadas forman lo que es llamada una Organiza - ción Jerárquica de Almacenamiento.

La necesidad para un almacenamiento jerárquico puede ser -- ilustrado, si examinamos la estructura de datos que la figura 4- ilustra, donde los registros son accesados después de procesar - los apuntadores en un Index. Que asume que un registro es recu- perado por la llave "Analista". Una vez que el registro es en - contrado, no es posible obtener cualquier otra cosa del registro, que no sea el nombre. La edad de los individuos y la ciudad en- la cual viven están guardados en el Index y los registros no tie nen apuntadores de regreso a las entradas del Index que contie - nen esta otra información vital. El archivo puede ser reestruc- turado dentro de una organización jerárquica, como lo muestra la figura 8 y así que esa información relacionada puede ser obteni- da. La nueva estructura es usada de la siguiente manera. Supon gamos que se desea toda la información de los registros que tie- nen la llave "Analista". El anillo de "Ocupación" es buscado -- primero hasta que el registro con la llave "Analista" es encon - trado. Este registro es el punto de partida de otro anillo que- conecta todos los registros de los individuos que son analistas, en este caso "Jones" y "Sosa". Una vez que los registros de -- "Jones y "Sosa" son recuperados, otra información puede ser -- obtenida si seguimos el anillo C a su punto de partida y recupe ramos el nombre de la ciudad del registro inicial. El anillo C pasa através de "Jones", entonces determinamos que la ciudad en la cual vive "Jones" es "León".

Las edades pueden ser obtenidas de una manera similar. Esta forma de estructura jerárquica permite a los mecanismos de alma - cenamiento y recuperación del Sistema de Manejo de Datos empezar- con cualquier registro en el archivo y subir la jerarquía.

Es más fácil de insertar y remover registros en los anillos- colocados en puntos arbitrarios que ejecutarlos con cualquier -- otra estructura de Lista. Esto es posible porque el registro ad- yacente cuyos apuntadores son modificados durante la operación, - pueden ser encontrados por la navegación del anillo hasta que el- registro deseado es encontrado. En contraste, una Lista simple - debe ser examinada siempre a partir del primer registro, debido a que, dado un punto arbitrario en la mitad de la Lista no es posi- ble encontrar el registro precedente.. La principal desventaja -- de todos los métodos de Organizaciones de Listas es el gasto del- espacio que causa el uso de los apuntadores. En Listas inverti - das, los apuntadores reducen el número de ocurrencias de una lla- ve y esto puede ocasionar un ahorro en el espacio. El costo de - los apuntadores debe ser ponderado contra las ventajas de almace- namiento y recuperación cuando contemplamos el uso de organizacio nes de Lista.

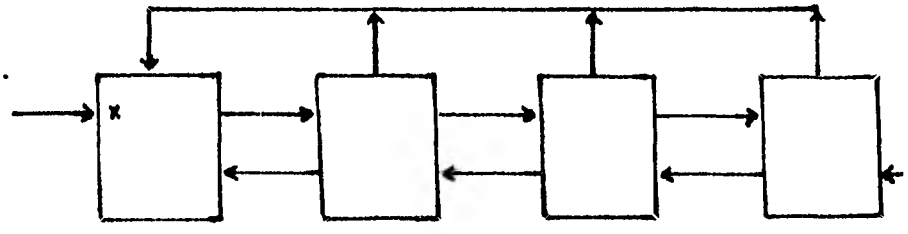


FIGURA 6 .- APUNTADORES EXTRA EN UNA ESTRUCTURA DE ANILLO

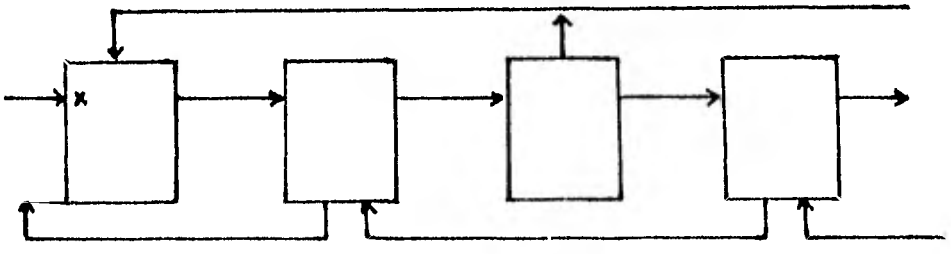


FIGURA 7.- ESTRUCTURA DE ANILLOS CORAL

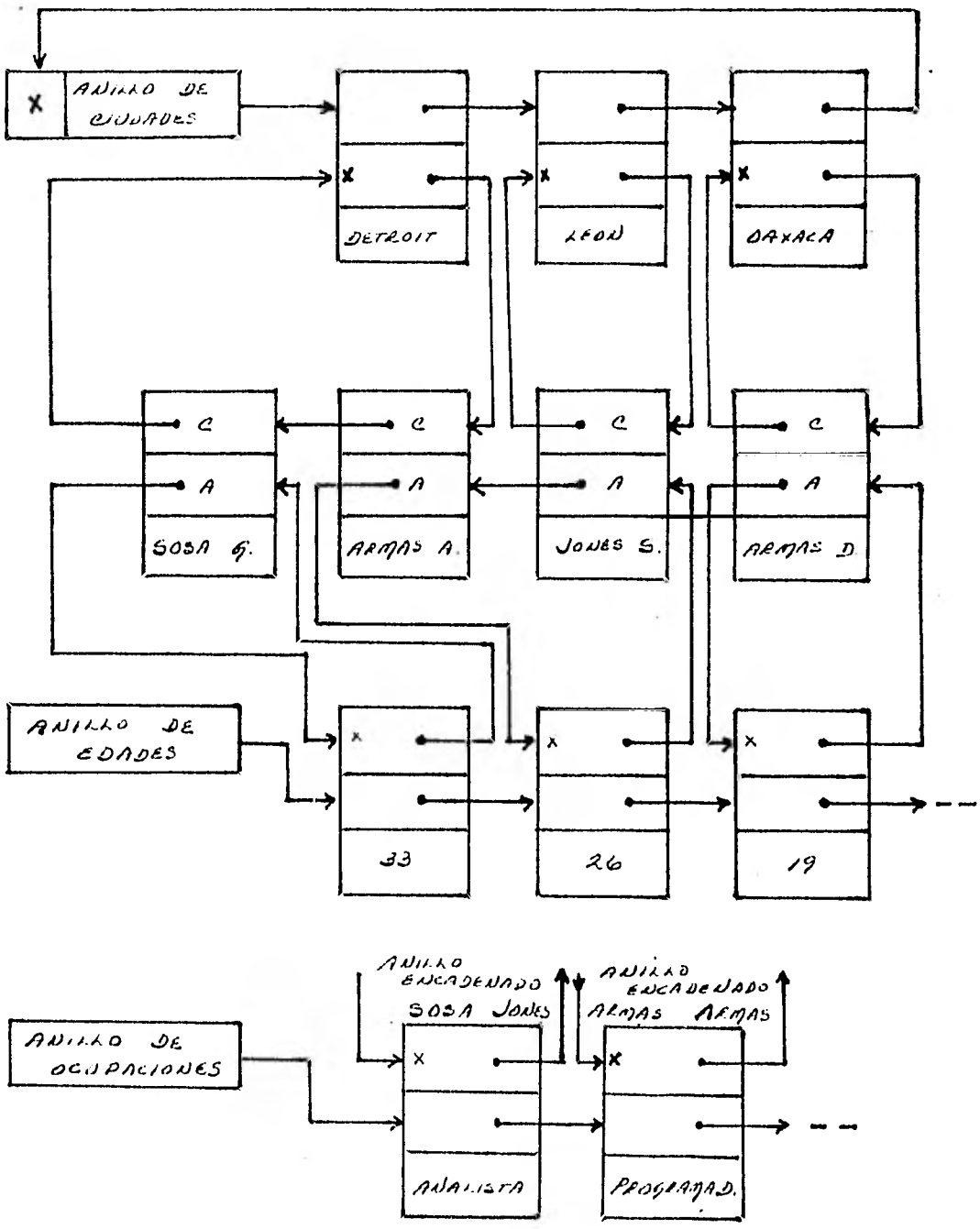


FIGURA 8.- ESTRUCTURA JERARQUICA

MANEJO DEL ESPACIO DE ALMACENAMIENTO

Cuando se almacenan nuevos registros en una lista o en un archivo organizado Random, el espacio que no está siendo usado por algunos otros registros debe ser encontrado para los nuevos registros. Cuando se remueven registros de un archivo el espacio que ellos ocupaban puede hacerse disponible para volverse a usar. En ambos casos un mecanismo encargado del manejo del espacio es invocado para seguir su rastro en el dispositivo de almacenamiento. Varios factores gobiernan la operación de este mecanismo:

1) El registro nuevo debe ser localizado de tal suerte que para recuperarlo, el tiempo de acceso sea mínimo. Por ejemplo los registros en estado de colisión en una organización Random en disco, deben ser colocados de modo que los brazos del disco no tengan que moverse para localizar el siguiente registro. Registros en un archivo multilista (que describiremos más adelante) deben ser localizados en páginas en las cuales el cruzamiento de sus límites mientras se busca una lista sea mínimo.

2) Si los registros son de longitud variable, el espacio debe ser de tal modo que el dispositivo de almacenamiento no sea fragmentado en pequeñas piezas de espacio libre que serán inutilizables para requerimientos futuros de espacio.

3) Compactar piezas de espacio libre dentro de bloques muy largos debe tomar lugar en intervalos suficientemente frecuentes para que grandes bloques de espacio sean puestos disponibles para volverse a usar (esto ayuda a reducir el problema de fragmentación). Al mismo tiempo debe-

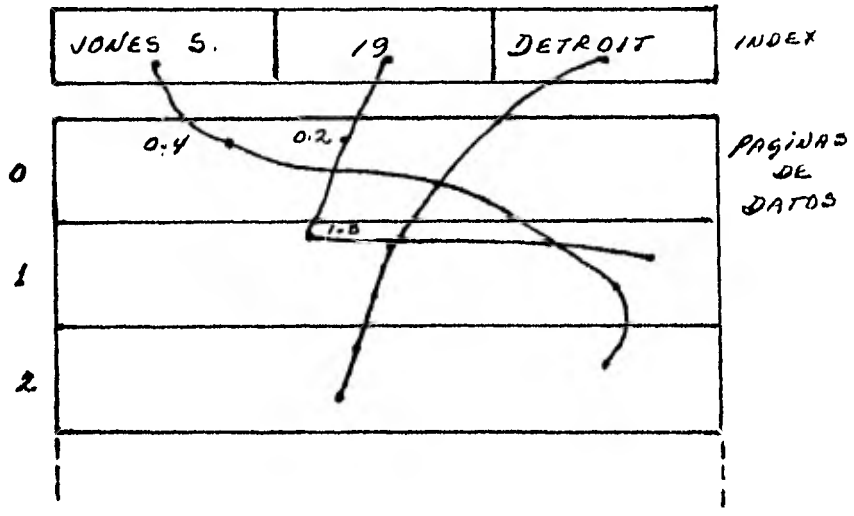
reconocerse que existe un gasto asociado con el espacio-compactado, el cual debe tratar de conservarse en un mínimo.

Encontrando estos tres criterios puede presentarse un pro - blema intratable cuando hablamos de Organizaciones de Lista y -- Random.

Se han desarrollado un buen número de estrategias para tratar con el manejo del espacio. Algunas técnicas reclaman diná - micamente el espacio ocupado por registros removidos. Si es manejado apropiadamente el espacio no representa un problema serio. Sin embargo, presenta un pequeño gasto cuando almacenamos y remo vemos registros en ambas organizaciones, Random y de Lista.

Una vez que hemos examinado los métodos de organización dedatos: Secuencial, Random y de Lista, pasaremos a ver algunas de las más sofisticadas técnicas de manejo de datos y observaremos- como se implementan através del uso de las estructuras antes men cionadas.

ARCHIVO MULTILISTA



Esta figura ilustra la organización de un archivo Multilista. En este archivo un directorio secuencial contiene los valores de las llaves por los cuales los registros son indexados. Asociado con cada valor de llave está un apuntador a la Lista de registros que tienen ese valor de la llave. Los registros son almacenados en blocks de tamaño fijo llamadas celdas ó páginas. Están direccionados por un número de página y un número de registro dentro de la página; el identificador del registro-0,4 referencía el cuarto registro en la página 0 y el identificador 1,3 referencía el tercer registro en la página 1. El agrupamiento de registros dentro de páginas es un concepto importante usado en el almacenamiento de estructuras de Listas en la media de la dirección directa. Más bien que usando la dirección directa de un registro como un valor en un apuntador, un número de página y un número de registro son especificados. El número de página es usado como una dirección directa para recuperar la página y el número de registro es usado para localizar el registro dentro de la página. De este modo un grupo de registros puede ser recuperado a un tiempo, y el tiempo de acceso por registro se reduce.. Se obtiene el ahorro sólo si se desea un número de registros en la página ya que es ineficiente recuperar una página entera para procesar sólo un registro. En el uso de Archivos Multilistas es por consiguiente preferible almacenar registros relacionados dentro de una página común. Para eliminar acceso innecesario de páginas, puede ser deseable limitar la longitud de la Lista para que todos los registros en una Lista se ajusten en una sola página.

En resúmen, un Archivo Multilista consiste de un directorio organizado secuencialmente con cada entrada del directorio apuntando a una Lista de registros.

Los registros en el archivo son bloqueados dentro de unidades lógicas llamadas páginas. Una página entera es accesada -- cuando un registro es almacenado, actualizado o recuperado.

ARCHIVO MULTILISTA CELULAR

Cuando trabajamos en el Archivo Multilista, a menudo varias páginas son accesadas para recuperar un registro. Una organización de datos que limita las Listas a que no crucen los límites de la página como lo muestra la figura 10 es llamada una Multi - lista Celular porque cada Lista está contenida en una y sólo una celda (página). Esta estructura es procesada en una manera similar a la usada para una Lista invertida, los valores de los índices son recuperados y sólo aquellas páginas que contienen registros de la Lista satisfaciendo los criterios de la búsqueda - son recuperados. En la figura 10 observamos que el directorio - está organizado como una Lista mientras que en la figura 9 está- organizado de una manera secuencial. Esto es sólo para demos -- trar que cualquier tipo de organización en el directorio es factible. En la Multilista Celular la longitud de la Lista está -- restringida así como el ajuste dentro de una página, si la lon - gitud de la Lista está limitada a 1, el archivo se convierte en- invertido tal como lo muestra la figura 11.

Mientras que el Archivo Multilista es más simple de programar y actualizar, el tiempo de recuperación es más largo que el- encontrado con la Lista invertida ó la Multilista Celular. Los - tiempos más largos son causados por recuperar páginas excesivas- mientras se sigue una Lista de página en página en persecusión - de un registro particular.

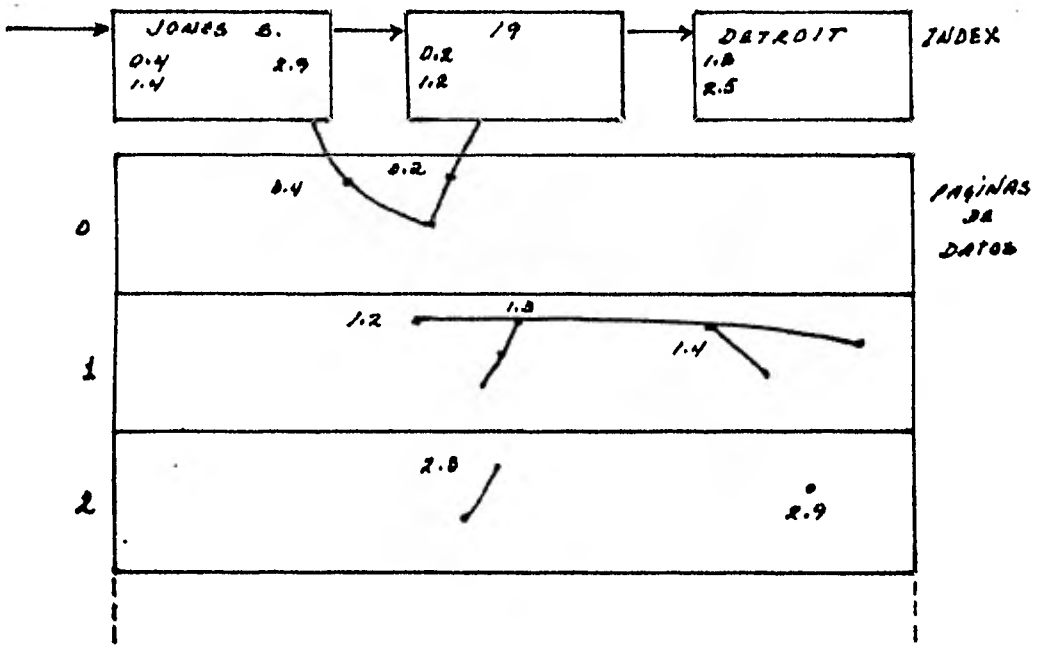


FIGURA 10.- ARCHIVO MULTILISTA CELULAR

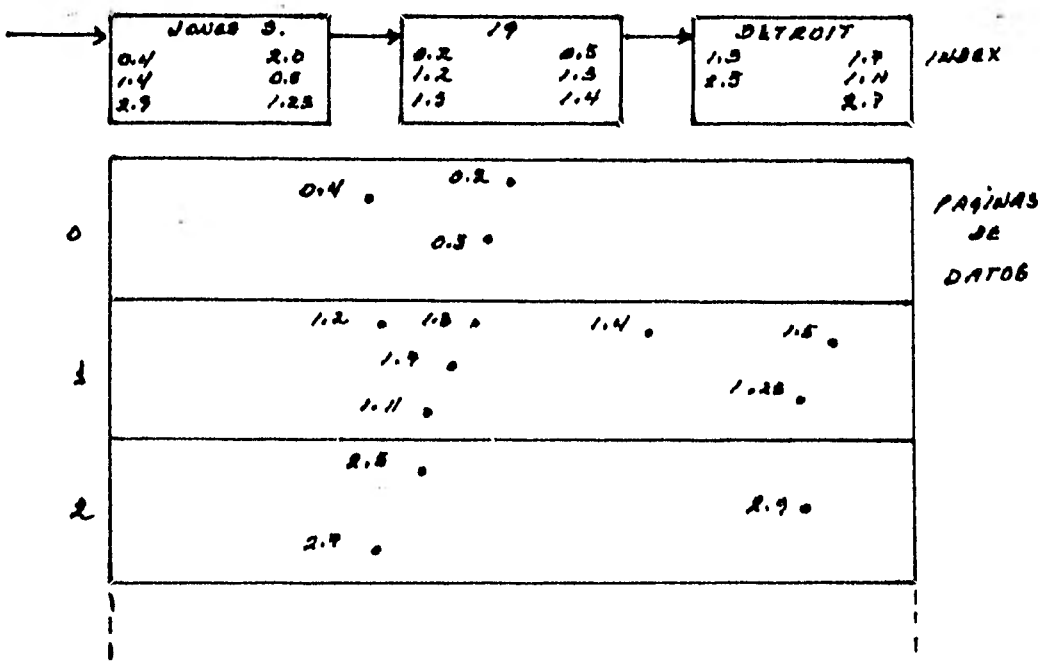
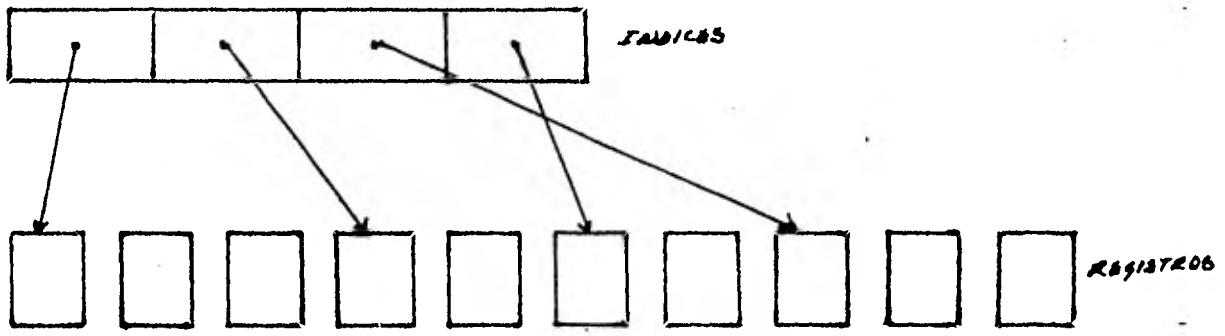


FIGURA 11.- ARCHIVO INVERTIDO

ARCHIVO INDEX SECUENCIAL

Una organización de archivos Index Secuencial se muestra en la siguiente figura:



Este archivo tiene la propiedad de que los registros pueden ser accesados ya sea por el uso de índices ó bien de un modo secuencial. En la parte superior de la figura 12 observamos como los índices están secuencialmente contiguos. Cada Index contiene la dirección de un registro en el archivo. Si un registro va a ser recuperado mediante el Index, los índices son buscados -- hasta que la llave deseada es encontrada y el apuntador del índice al registro es usado como la dirección directa para recuperar el registro. Los registros son también almacenados en la media de almacenamiento de un modo secuencial, permitiendo a los registros ser accesados en órden secuencial.

Los índices en un archivo Index Secuencial están general -- mente orientados a las características de la media de almacenamiento. En algunos sistemas hay un Index en forma de cilindro - así como un Index rastreador para cada campo de dato almacenado en el archivo.

ORGANIZACION DE ARBOL

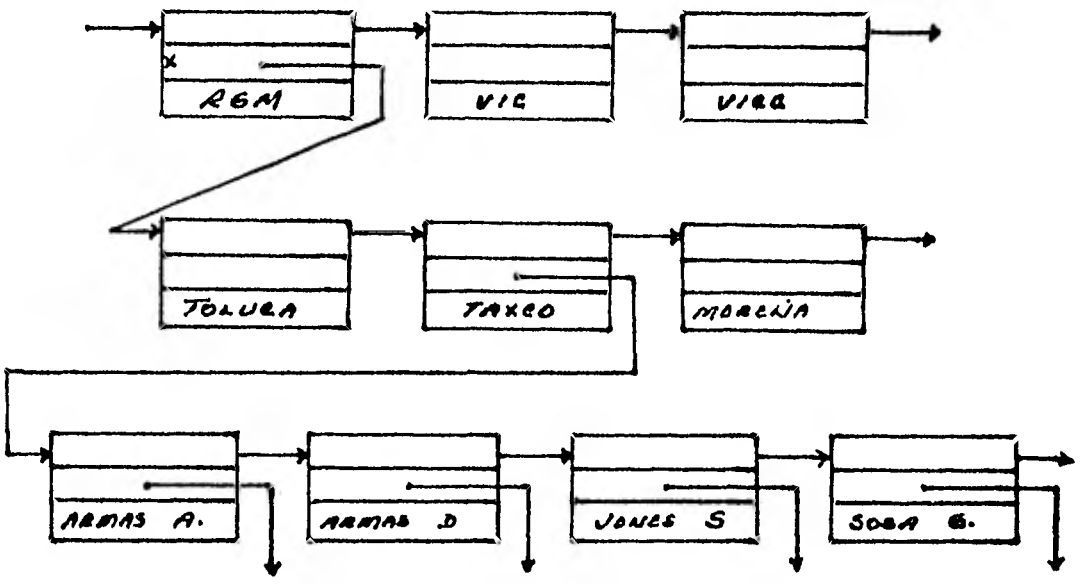
Una Organización de Arbol de datos es aquella en la cual se usan varios niveles de índices. El Index en el primer nivel --

apunta a una colección de índices en el segundo nivel. Uno de estos índices de segundo nivel se utiliza para encontrar una colección de índices a un tercer nivel, y así sucesivamente hasta que los valores actuales de los datos han sido encontrados en el fondo de una rama particular de este árbol. La figura 13 es un ejemplo de un directorio multinivel que tiene una organización de árbol. En esta figura los índices en los varios niveles están organizados en Listas.

Si el registro identificado como "RGM. TAXCO . JONES" se desea recuperar, las siguientes acciones toman lugar:

- Se busca en la primera Lista hasta que un registro con el valor de llave "RGM" es localizado. De este registro un apuntador a la lista de índices de segundo nivel es obtenido.
- Los registros en la lista de segundo nivel son examinados secuencialmente hasta que la llave " TAXCO " es localizada, y el apuntador en el registro " TAXCO " es usado para accesar la Lista de tercer nivel.
- Las llaves de los registros en este tercer nivel son buscadas hasta que el registro que contenga la llave "JONES" sea localizado. En este registro está la dirección directa del registro identificado como "RGM. TAXCO .JONES". Desde luego los indices pueden ser organizados secuencialmente así como en Lista.

La organización de Arbol conviene usarla en el mantenimiento de grandes directorios. Pueden ser agregadas, removidas o cambiadas partes del directorio sin disturbar el resto de la organización.



DISPOSITIVOS DE ACCESO SECUENCIAL

Son aquellos cuyos mecanismos de acceso pueden localizar -- solo el siguiente registro, es decir uno físicamente adyacente - al lugar común del mecanismo de acceso. Ejemplos son cintas magnéticas o cintas perforadas las cuales se mueven de un modo se - cuencial a través del mecanismo de lectura ó de escritura. Cuando la cinta está en un lugar determinado, el siguiente registro de datos es localizado mediante el avance de la cinta una posición. Algunos mecanismos permiten la lectura del dispositivo en cualquier dirección. En este caso ya sea el prioritario (inmediatamente precedente) o el siguiente (inmediatamente después) - registro puede ser accesado. Los mecanismos de acceso secuencial sobresalen en la lectura o escritura del siguiente registro en el dispositivo que está siendo usado por el mecanismo y son - usados generalmente con archivos secuenciales.

DISPOSITIVOS DE ACCESO DIRECTO

Son aquellos en los cuales se utiliza una dirección de --

hardware para posicionar el mecanismo de acceso. Esta dirección es utilizada cuando los datos son escritos dentro del dispositivo y otra vez cuando son recuperados. Memoria central es un ejemplo obvio de un dispositivo de acceso directo, en virtud de que cada posición en la memoria tiene una dirección. Los discos magnéticos y los tambores son también considerados como dispositivos de acceso directo, en virtud de que cada parte de la superficie grabada puede ser direccionada. Sin embargo, una vez que el mecanismo de acceso esta posicionado en una dirección, los datos son leídos o escritos de una manera secuencial dentro de un rastreador de grabación que pasa por debajo del mecanismo de acceso. Los dispositivos de acceso directo sobresalen en localizar un registro de datos particular que tenga una dirección conocida. Son también usados para recuperar una página de registros en una organización multilista, donde el número de página corresponde a la dirección de la página en el dispositivo de acceso directo.

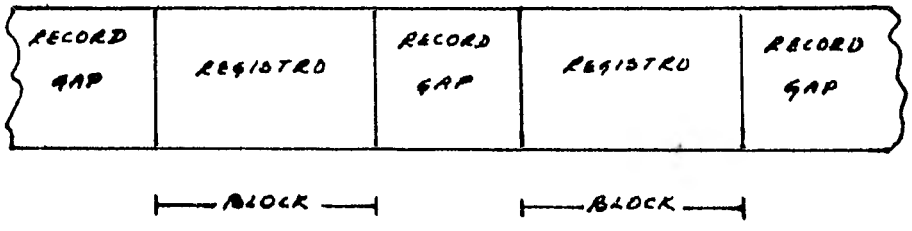


FIGURA 14.- REGISTROS SIN BLOQUEAJE

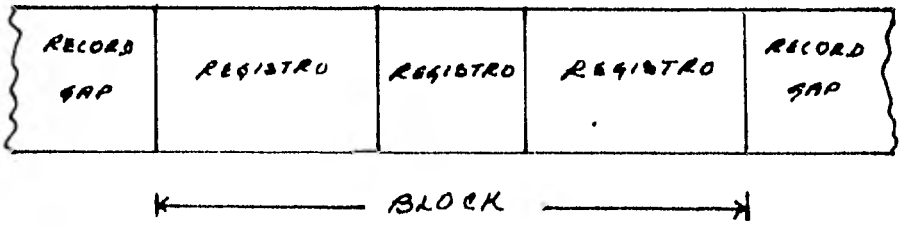


FIGURA 15.- REGISTROS BLOQUEADOS

La figura 14 ilustra la apariencia de los datos fisicamente los cuales han sido grabados dentro de cualquier dispositivo de almacenamiento excepto en memorias magnéticas. Los record gaps- (un intervalo de espacio asociado con un registro para indicar- o señalar el fin del registro) son puestos en el dispositivo de- almacenamiento por el hardware. Se utilizan para separar las - áreas en las pistas y permiten ejecutar ciertas funciones de e - quipo cuando el gap se mueve a través del mecanismo de acceso -- del dispositivo. Dentro de cada bloque de datos grabado está un registro de datos, el cual puede ser de cualquier longitud. Es- a menudo ventajoso agrupar varios registros dentro de un bloque- como vemos en la figura 15. El bloqueaje mejora la velocidad con la cual los registros pueden ser leídos o escritos porque el gas to involucrado en localizar un bloque se distribuye entre el ac- ceso y el uso de todos los registros dentro del bloque. También ahorra el espacio de record gaps, en virtud de que los gaps en- tre registros dentro de un bloque son eliminados.

FACILIDADES DE LENGUAJES PARA PROGRAMAR SISTEMAS DE MANEJO DE DATOS

Existen muchos lenguajes de programación exclusivamente -- orientados al manejo de una organización de datos particular. Los lenguajes de programación más comunes, tales como PL/I, -- Fortran y Cobol, tienen mecanismos para tratarcon archivos orga- nizados secuencialmente. Por medio del postulado READ el siguien te registro es recuperado, y el postulado WRITE almacena un re - gistro en el siguiente lugar en un archivo organizado secuencial mente. En Fortran el archivo puede ser retrasado por medio de - la declaración BACKSPACE y regresar a su punto de partida por el postulado REWIND. PL/I y Cobol no tienen la declaración de BACKS PACE: por medio del postulado CLOSE el archivo es regresado a su

a su punto de partida.

Algunos lenguajes de lista, tales como LISP e IPL-V fueron inicialmente desarrollados para experimentar en las áreas más - cientificas y matemáticas de la computación, tales como problemas heurísticos, resolución de problemas generales, máquinas organizadas por si mismas y teoría de autómatas. No fueron orientadas al manejo de datos y para la aritmética son utilizadas - operaciones muy simples dificultando el manejo de datos en expresiones. En algunas implementaciones el programador debe desarrollar su estructura para el manejo de listas en memoria -- principal, puede ocurrir que las listas sean mas grandes que la capacidad de la memoria y esto le hará manejarlas en memorias - auxiliares.

PL/I permite la generación de apuntadores, facilitando con esto al programador la localización dinámica de espacio para un registro y optimiza el almacenamiento de apuntadores entre re - gistros creando así su propia estructura de lista y anillos. Los registros son almacenados en una área (similar a una página) y esta puede ser leída de y escrita en dispositivos externos de almacenamiento usando las facilidades de entrada/salida provistas por PL/I.

PL/I provee al programador de instrucciones de entrada/salida para construir nuevos tipos de organizaciones y listas, -- por tal razón en PL/I se requiere el mantenimiento de apuntadores. SLIP es un lenguaje para procesar listas por medio del -- cual se manejan estructuras de anillos de dos modos y existe -- como un conjunto de subrutinas las cuales son posibles de invocar desde fortran.

Quizá el rasgo generalmente menos provisto de todos los -- lenguajes de programación es la habilidad para manejar organiza

ciones Random. Cobol provee algunas declaraciones rudimentarias. las cuales no están bien implementadas por ningun diseñador. PL/I tiene una serie de declaraciones para leer, escribir y reescribir registros en dispositivos de acceso directo. Sin embargo estas declaraciones son limitadas para trabajar con los elementos que provee el lenguaje de control de trabajo y no permiten al programador tener acceso a todas las capacidades del hardware.

Existen dos lenguajes los cuales permiten al programador organizar sus datos en una estructura de anillo. APL consiste de seis proposiciones las cuales han sido agregadas al lenguaje -- PL/I e IDS tiene un número similar de proposiciones, las cuales también han sido añadidas al lenguaje Cobol. Ellos permiten al programador definir las relaciones de anillos entre registros de datos, crear nuevos registros de datos dentro del conjunto de datos, insertar registros de datos en anillos, y buscar en anillos hasta encontrar registros de datos que tengan valor particular de datos, sin tener que volver a complicar a los mecanismos de manejo de apuntadores. Después que los registros han sido recuperados, los valores de datos pueden ser modificados o usados en cualquier cálculo aritmético. Declaraciones complementarias para remover registros de anillos y quitar registros del archivo son tambien provistos. No hay un lenguaje que sea el mejor adaptado para programar todos los sistemas de manejos de datos. Sistemas para manejo de datos han sido programados en lenguaje ensamblador y JOVIAL, también como con los otros lenguajes descritos anteriormente. Sólo un estudio de las facilidades de los -- lenguajes, los tipos de hardware que serán usados, los tipos de requerimiento de entrada y la manera en que serán sometidos al sistema, las capacidades en los dispositivos en que se almacenará la información y las eficiencias del compilador dictaminarán cual lenguaje utilizar para programar en un sistema particular de manejo de datos.

RESUMEN

Hemos hecho una breve exposición de los propósitos de los sistemas de manejo de datos. Se ha dado una descripción de las técnicas para manejo de datos (Secuencial, Random y Lista). La organización secuencial ha hecho de los requerimientos de almacenamiento físicos y lógicos una conjunción, permitiendo rápido acceso al siguiente registro lógico en el archivo. Las organizaciones Random son caracterizadas por el acceso rápido de datos basado en una llave que indica el lugar de los datos en el dispositivo de almacenamiento. Las organizaciones de lista separan las relaciones lógicas de los mecanismos físicos de almacenamiento por medio de apuntadores, permitiendo a registros de datos -- tomar parte eficientemente en multiples relaciones de archivos.

Se han considerado los mecanismos de como estructuras más complicadas pueden ser construídas a partir de usar las organizaciones antes mencionadas. Anillos, listas invertidas, árboles, jerarquías y otros tipos de estructuras de datos han sido explicadas. Los dispositivos de almacenamiento fueron discutidos y se observó que el dispositivo de acceso secuencial es adaptable a las organizaciones secuenciales, mientras que el dispositivo de acceso directo es utilizado para las organizaciones random y de lista. Lenguajes de programación que tratan con sistemas de manejo de datos fueron examinados y se concluyó que ningun lenguaje puede manejar satisfactoriamente el universo de problemas.

C A P I T U L O II

TECNOLOGIA DE BASE DE DATOS

CRONOLOGIA Y RELACIONES FUNCIONALES DE SISTEMAS DE MANEJO DE DATOS

La aportación principal de la entonces incipiente Tecnología del manejo de datos fué el desarrollo de Cobol por parte de CODASYL alrededor del año 60. Entre sus principales beneficios se encuentran la identificación y la separación de:

a) Manipulación de Datos
b) Descripción de los Datos y Archivos a nivel de programa.

Las facilidades de la manipulación de datos son aquellas - usadas para el proceso de datos en memoria. OPEN, CLOSE, READ, WRITE y cualquier otro postulado para ENTRADA/SALIDA de datos - entre el almacenamiento principal y el almacenamiento externo,- algunas veces y otras no son incluidos en esta categoría. En - un creciente número de casos debido al reporte presentado por - CODASYL en el año de 1971 denominado DBTG (Data Base Task Group) el lenguaje de manipulación de datos es considerado como desa - rrollado unicamente para postulados de ENTRADA/SALIDA. Las facilidades de la descripción de datos son aquellas que permiten- describir las estructuras de datos en memoria y las estructuras de datos externas y sus inclusiones como archivos externos. El desarrollo de lenguajes de alto nivel, especialmente COBOL para el manejo de datos, es verdaderamente uno de los más significativos desarrollos en la historia de la tecnología de la computación.

A las implementaciones hechas a COBOL a principios de los- setentas, fundamentalmente a los reportes que eran la base de - estas implementaciones, se les vió la necesidad de incluirles - algunas más. Las primeras de estas fueron los paquetes SORT y- REPORT WRITER.

El paquete SORT es un módulo muy especializado, que permite mediante un algoritmo adecuado, la clasificación de registros en un archivo grande, y la clasificación y fusión de varios archi - vos.

El paquete REPORT WRITER es una utilería que provee amplias facilidades para editar salidas, formatear, para contar y algu - nas herramientas más que permiten generar reportes complejos; -- estas facilidades van más allá de las primeras definidas como -- parte clásica de un lenguaje. Algunos vendedores han sacado al- mercado estos paquetes como complemento o enlace a varias imple- mentaciones de lenguajes. Paquetes de esta índole que juegan el papel de enlace o complemento son nombrados a menudo utilerías.- A mediados de los años sesentas CODASYL revisa COBOL e incluye - SORT como un nuevo verbo del lenguaje regular COBOL.

Antes y durante los principios de COBOL se desarrollaron un buen número de los llamados "generadores de programas de repor - tes". Hoy en día, el principal es RPG (programa generador de re portes) el cual incluye facilidades primarias de ambos, COBOL y REPORT WRITER más algunos atractivos implementos para recupera - ción y actualización de archivos, no utilizables en COBOL. RPG- fué y es usado significativamente en pequeños sistemas computa - lizados para aplicaciones que tratan con archivos menos comple - jos.

A finales de los sesentas GFMS(Sistemas generalizados de ma nejo de archivos) fueron desarrollados. Estos sistemas integran varias facilidades de COBOL, REPORT WRITER, RPG y paquetes SORT- además de un número considerable de otras útiles facilidades en- el manejo de datos dentro de su propio sistema. Fundamentalmente, como lo muestra la figura A, pueden ejecutar la mayoría de los - postulados que COBOL ejecuta más muchos más en áreas donde COBOL es débil, nombrados ENTRADA/SALIDA y REPORT WRITING. Estos in -

tentan reemplazar COBOL y cualquier otro lenguaje de programa -- ción en aplicaciones de ENTRADA/SALIDA y en comunicación directa con las facilidades básicas del manejo de archivos en el sistema operativo para:

1.- Definición de registros y archivos
2.- Creación de archivos
3.- Recuperación de archivos
4.- Actualización de archivos
5.- Manipulación de archivos
6.- Mantenimiento de archivos
7.- Reports Writtings

El lenguaje de la mayoría de los GFMS es un lenguaje de formas orientadas para programación de alto nivel con poderosos -- operadores que permiten ejecutar facilmente estas funciones sin-incurrir en la carga de la programación en los lenguajes convencionales. En órden jerárquico podríamos poner, a la cabeza los-GFMS, luego COBOL y después lenguajes de ensamblador.

Informatics introduce el primero de estos GFMS a mediados - de los sesentas su nombre es MARK IV el cual es usado en cerca - de mil instalaciones. ASI-ST de Applications Software Inc es -- otro de los GFMS presentados a finales de los sesentas y concu - rrentemente es segundo a MARK IV en mímero de usuario. Estos y-otros GFMS usualmente vienen en tres versiones: uno que es solamente un poderoso REPORT WRITER, un sistema intermedio y el sistema completo.

Los GFMS operan como un programa de aplicación (muchos son-escritos en Fortran o en lenguaje de ensamblador y Fortran) dentro del sistema operativo y utilizan los métodos de acceso dis - ponibles. Através de los lenguajes de GFMS, el programador y --

también el usuario (el cual puede no tener contacto alguno con - lenguajes) pueden especificar, por ejemplo la recuperación de registros sobre la base de cualquier expresión lógica, usando cualquier campo de la estructura del registro, sin las limitaciones, carga e inquietud de una secuencia determinada (si los registros estan almacenados como archivos secuenciales) o en acceso determinado por una llave (si los registros estan almacenados en archivos random, o en Index Sequential o en dispositivos de acceso directo).

La productividad en reportes aumenta (en términos de la programación para implementar y mantener aplicaciones así como obtener el reporte deseado) usando GFMS en lugar de COBOL, FORTRAN y PL/I en aplicaciones orientadas a ENTRADA/SALIDA en el órden - de cuatro a uno hasta quince a uno dependiendo de la aplicación.

Como se muestra en la figura A funcionalmente, GFMS no hace todo lo que lenguajes como COBOL pueden hacer. El mejor GFMS incluye entre 60% y 80% de las facilidades que COBOL provee, y - excluye algunos comandos. Los GFMS son designados para cubrir - entre el 60% y 90% de las necesidades en el manejo de archivos - complejos durante el proceso comercial de datos. Así, los GFMS- no pretender ser un total reemplazo de los lenguajes de alto nivel. Los GFMS son más fuertes donde COBOL es más débil (manejo- de archivos), y son más débiles donde COBOL es más fuerte (aritmética compleja y proceso de registros en memória).

La mayoría de los GFMS proveen puntos de interface con programas escritos en lenguajes de programación para procesos más- complejos. Por ejemplo ASI-ST tiene su "propio módulo de código" que permite hasta 25 salidas a programas escritos en COBOL, PL/I, FORTRAN ó IBM 360/370 en su lenguaje de ensamblador.

Los principales GFMS, hoy en día son muy eficientes y en -

muchos casos pueden tomar menos tiempo de máquina que un programa equivalente en un lenguaje convencional.

Nuestro intento en este punto es establecer las relaciones funcionales entre los varios participantes en la escena del manejo de datos.

A finales de los sesentas, la necesidad de integrar archivos muy grandes en una Base de Datos y la necesidad para muchos usuarios de ciertos programas de aplicación que les permitieran efectiva y eficientemente accesarla, dieron lugar a los primeros esfuerzos en el desarrollo de los sistemas de manejo de Base de Datos. Las estructuras convencionales fueron inadecuadas para tales demandas y resultaron en cargas durante el tiempo de proceso así como en excesivos requerimientos de almacenamiento auxiliar. Los GFMS, limitados por las estructuras convencionales de un "sólo archivo" en sus sistemas operativos, fueron también -- inadecuados.

Varias organizaciones grandes desarrollaron sistemas "en-casa" usando estructuras de archivos del tipo de los descritos en el capítulo anterior e interconectando tales archivos para formar Bases de Datos. Los conceptos pioneros y las facilidades -- fueron introducidas por el sistema de General Electric llamado - IDS (Integrated Data Store) a mediados de los sesentas. Este extiende el COBOL para permitir la definición e interconectividad de cadenas agregadas de datos y archivos. Fué el primer modelo de redes de datos. Existe otro, que es considerado como pionero de los GDBMS actuales se trata de TDMS (sistema de manejo de datos de tiempo compartido).

IBM y NORTH AMERICAN AVIATION se enfrentaron con el problema de manejar la vasta información relacionada con el vuelo a la luna (Apollo) a finales de los sesentas, comenzaron el proyec

to que se convertiría en el primer Sistema de DB/DC (Data Base/ Data Communications) llamado IMS (Information Management System) Un sistema DB/DC es uno en el cual las facilidades para la Base-de Datos y la Comunicación de Datos (o teleproceso), han sido -- altamente desarrolladas en unas bases integradas superiores a - aquellas de un sistema operativo. Otros sistemas notables del - tipo DB/DC son TOTAL de Cincom, éste junto con IMS dominan hoy - en día el mercado de los DB/DC (con 2000 ó 3000 instalaciones -- hasta 1976).

La figura A (4)* resume las principales relaciones funcionales de los Sistemas de Manejo de Datos en los setentas y ya -- dentro de los ochentas:

1.- Programas de aplicación en un lenguaje de programación-convencional que permita comunicarse con él GDBMS ó con un sistema DB/DC mediante un lenguaje apropiado de in - terface provisto por el sistema.

2.- Programas de aplicación en un lenguaje de alto nivel -- del GFMS que permita comunicarse con el GDBMS ó con un-sistema DB/DC mediante una interface apropiada generalmente provista por el GFMS.

3.- Un lenguaje query de muy alto nivel particular a cada - sistema de DB/DC ó GDBMS es generalmente muy útil para-accesar un alto volúmen de datos rapidamente.

En suma, las relaciones funcionales expuestas anteriormente (ilustradas en la figura A (1)*, (2)*, (3))* continúan dominando. Así, como un ejemplo del avanzado estado de flexibilidad, un programa de aplicación desarrollado en ASI-ST puede comunicarse mediante las interfaces correspondientes implementadas por el mismo ASI-ST con:

* ver lista de figuras a partir de la hoja #93

1.- Un programa de aplicación en COBOL, FORTRAN, PL/I ó BAL

2.- Los métodos convencionales de manejo de archivos del -- sistema operativo (IBM 360/370).

3.- TOTAL ó IMS

Los sistemas DB/DC y los GDBMS generalmente dependen de un- lenguaje, convencional externo, ó de un lenguaje de GFMS, para - ejecutar la manipulación de datos en memoria. Sin embargo, los sistemas que han sido extendidos a partir de COBOL (como IDS y - DBTG) pueden considerarse como provistas de sus propias facili - dades para manipular datos. Practicamente todos los GDBMS y los DB/DC tiene alguna capacidad para manejar datos en memoria, sin- embargo cada vez puede ser menor. Por ejemplo, un número de fun ciones o módulos de proceso son agregados a la mayoría de los -- GDBMS para computar, por ejemplo: máximos, mínimos, promedios, - desviación standard, etc. de los registros deseados.

Los DB/DC y los GDBMS permiten el uso de lenguajes orienta- dos a Transacciones, así como de lenguaje query. La función de- lenguaje de interface soporta comunicación del GDBMS o de DB/DC- con lenguajes convencionales de programación, posiblemente con - lenguajes de GFMS y con su propio lenguaje query. La interface- query y su lenguaje asociado query son diseñados para:

1.- Recuperación de información en línea.
2.- Proveer de facilidades para utilizar el sistema.
3.- Manejar grandes volúmenes de datos.
4.- Para accesar subconjuntos de la Base de Datos que satis fagan un conjunto de requerimientos especificados por - el usuario.
5.- Satisfacer requerimientos de ENTRADA/SALIDA.

La interface de transacción está diseñada para permitir el uso de un lenguaje convencional de programación para:

1.- Recuperar información predecible.
2.- Soportar requerimientos orientados al programador.
3.- Manejar largos volúmenes de datos de ENTRADA/SALIDA
4.- Satisfacer requerimientos de ENTRADA/SALIDA (BATCH)

Para ser consistentes de hoy en adelante usaremos el término GDBMS, en el cual estarán incluidos tanto los sistemas DB/DC como los Sistemas Generalizados para el Manejo de Base de Datos. Donde sea necesario diferenciaremos el término DB/DC del término GDBMS.

CONCEPTOS Y OBJETIVOS DE LA TECNOLOGIA DE BASE DE DATOS

Introduciremos ahora los principales conceptos y objetivos de los Sistemas de Manejo de Base de Datos. Estos objetivos son derivados de las necesidades urgentes surgidas en la comunidad de procesamiento de datos. Estos son:

1.- Independencia de Datos
2.- Compartir Datos
3.- No Redundancia
4.- Relación de Datos (Relaciones)
5.- Integridad
6.- Flexibilidad de acceso
7.- Seguridad
8.- Ejecución y Eficiencia
9.- Administración y Control

Existen varios enfoques de estos objetivos. Han sido estudiados y debatidos en gran medida por una población cada vez más numerosa. Los sistemas de Base de Datos disponibles llevan a cabo estos objetivos en varios grados y formas. No existe una industria estándar, aunque CODASYL en su reporte DBTG en el año de 71 hace el intento de definirla.

INDEPENDENCIA DE DATOS

El concepto de la independencia de datos es básico en la construcción de Bases de Datos. Denota independencia o aislamiento de un programa de aplicación con respecto a la organización lógica ó la organización física y las consideraciones de almacenamiento de las estructuras de datos y archivos que utilizan. Cualquier cambio en la organización lógica ó física de los datos, no implica que repercuta en cambios considerables en los progra-

mas de aplicación y cambios en los programas de aplicación no deberán afectar las estructuras de los archivos.

La independencia de datos provee la opción de hacer cambios en los programas de aplicación en:

1.- La localización de los datos
2.- Representación física (como se representan actualmente los datos en los dispositivos de almacenamiento)
3.- Organización física de los datos (por ejemplo organización invertida contra árbol doblemente encadenado)
4.- Familias disponibles (trayectorias de acceso)
5.- Dispositivos particulares de almacenamiento utilizados.
6.- Otros programas independientes compartiendo cualquiera de los archivos de datos que utilicen otro programa.

Los programas no deben estar sujetos a modificarse por factores externos a ellos. La habilidad de cambiar la organización física sin que los programas de aplicación se conviertan en inservibles se denomina Independencia Física de los Datos. La habilidad de un programa de aplicación para continuar ejecutandose correctamente usando su propia visión de la Base de Datos a pesar de cambios en otras partes de la Base de Datos lógica es denominada Independencia Lógica de los Datos. El término independencia de datos se refiere generalmente a las dos mencionadas con anterioridad: Independencia Lógica y Física. Todavía, existen muchos grados de independencia de datos. No hay escala alguna para medir o interpretar el grado de la independencia de datos.

Todo manejo de datos disponibles en sistemas operativos convencionales mediante lenguajes de programación (Secuencial, ISAM, RANDOM) han sufrido por la falta de la independencia de datos. Específicamente, en los ejemplos relatados a continuación,

hay que mencionar que no se trata de una lista exhaustiva e inclusive estamos en posibilidad de ampliarla de acuerdo a nuestras propias experiencias:

1.- La modificación o reedificación de niveles de registros en cualquiera de los archivos usados por un programa de aplicación requiere de las dos operaciones siguientes:

a) generar nuevamente el archivo

b) Modificar y recompilar el programa específico y cualesquier otro que ocupe el archivo.

por ejemplo, este es el caso si sólo la posición de una subestructura o grupo de campos son cambiados en la descripción del registro al que pertenecen. Esto puede ser considerado una reedificación en la estructura lógica pero no en el contenido lógico del registro.

2.- La modificación del nombre, tipo de dato o longitud decualquiera de los campos elementales de registros en un archivo requiere nuevamente la generación tantas veces como modifiquemos las descripciones de los registros en cada programa que utilice el archivo.

3.- La tendencia "un archivo - un programa". Un cambio en la descripción lógica del archivo involucra generar nuevamente el archivo entero y modificar y recompilar cada programa que utilice el archivo. Asimismo, un cambio en la descripción del registro a nivel de programa requiere cambiar y generar nuevamente el archivo involucrado. Por ejemplo, un programa de aplicación nuevo que necesite para su operación la visión y el acceso de solamente un subconjunto determinado de campos elementa

les en el archivo, debe conformarse de una y la única descripción de registro establecida por el programa que carga el archivo. Todos los programas de aplicación deben tener exactamente la misma visión del archivo aún si algunos de ellos no necesitan ó no deben tener acceso a todos los campos.

4.- Las adiciones de más de un caso de campos elementales o subestructuras en un registro requiere recargar el archivo y modificar y recompilar todos los programas que usan el archivo.

5.- Extender el número de casos máximo permitido en un arreglo de campo o de un grupo repetitivo, más allá del número indicado cuando el archivo fué cargado requiere que el archivo sea recargado y todos los programas modificados y recompilados.

6.- La adición de un nuevo dato al registro implica recargar el archivo y modificar y recompilar todos los programas de aplicación que usan ese archivo.

7.- Cambiar el nombre de la llave de acceso en postulados de ENTRADA/SALIDA en los programas de aplicación con respecto a archivos RANDOM ó ISAM, requiere ya sea recargar el archivo para tener el nuevo nombre de la llave, o bien cambiar la lógica o estrategia de acceso del programa de aplicación.

Muchos de estos retrocesos son el resultado del cambio. Estas clases de cambios ocurren a menudo en la práctica actual debido a requerimientos imprevistos o a cambios de parecer. Siempre ocurren con el tiempo, si no es que frecuentemente. Estos cambios ocurren completamente independientes de cualquier

máquina o de consideraciones de eficiencia.

Un gran número de cambios en la estructura física de un archivo pueden también ser contemplados para consideraciones de eficiencia y para integración de otros programas con el programa que utiliza el archivo. Por ejemplo, cambiando el factor de bloqueaje, cambiando la representación de un número decimal a un número binario, cambiando la longitud física fija de los registros a una longitud variable, etc. Estos generalmente requieren cambiar el archivo y/o la descripción del registro en los programas de aplicación y recompilar. El cambio de la llave para el SORT en el archivo o de la llave de acceso en ISAM u otra organización directa desbaratan los programas de aplicación asociados mucho más.

Lo grave de estos problemas se vuelve intolerable cuando están involucrados múltiples sistemas de información interconectados

Una complicación adicional la cual ha sido experimentada y se está viviendo en muchos lugares es que los programadores de aplicaciones y sistemas no han amparado suficientemente la lógica y los trabajos de los programas de aplicación de la estructura física y de las estructuras de almacenamiento.

Hasta ahora en muchos sistemas de aplicación un cambio en la existencia de un apuntador a nivel de estructura de dato, pondrá fuera de orden los programas que hacen uso explícito de este apuntador.

La tecnología de Base de Datos se esfuerza por proveer tanta independencia de datos como sea posible. Hay varios grados de independencia de datos provista por los GBDMS.

Es necesario acentuar que no hay independencia de datos --

completa en virtud de que un programa de aplicación no puede estar protegido de todos los cambios posibles. El especialista del manejo de datos debe estar enterado de como es la independencia en un sistema particular de manejo de datos. Debe conocer ¿que cambios en la organización física del archivo impactan los programas de aplicación?, debe saber también ¿que cambios en los formatos de datos, estructuras, trayectorias de acceso y métodos básicos de acceso implican modificaciones en los programas de aplicación? y por último ¿de que clase o que tan extensa será esta modificación?.

El estar modificando declaraciones de datos y procesar comandos dependientes de ellas en un programa de aplicación y/o declaraciones de datos en un programa que genere un archivo puede ser de consecuencias fatales.

Los costos del desarrollo inicial de los programas son bastante altos. Peor aún es reprogramar, implica entender en detalle el programa que va a modificarse, en cambio en una pequeña porción del programa puede involucar cambios en otras partes no contempladas. El programador que ejecuta la modificación puede no ser el programador original y así puede tener que enfrentar una lógica distinta a la suya; la documentación del programa tiene que ser cambiada, etc. Los costos de mantenimiento son sorprendentes en términos de horas-hombre, esta reprogramación también ocasiona retrasos, rompimiento de otros procesos que dependen de lo bien que se reprogramen y pérdidas de oportunidades (que sería la incapacidad de derivar beneficios de nuevas aplicaciones que podrían implementarse si la reprogramación de estos viejos programas no estuviera ocurriendo). La mayor parte del presupuesto destinado al procesamiento de datos en una instalación va fundamentalmente al software, pero no directamente al hardware. Desde los sesentas una cantidad cada vez mayor ha sido gastada desafor

tunadamente en programas de mantenimiento, en añadir habilidades a sistemas existentes o desarrollar nuevos sistemas, con esto observamos que la necesidad de alcanzar un alto grado de independencia de datos es evidente.

COMPARTIR DATOS

El problema de que varios usuarios con propositos distintos puedan tener acceso a un mismo conjunto de datos se remonta pormuchos años en la literatura. La idea es poder tener diferentes programas de aplicación accesando independientemente una base de datos integrada. Algunas aplicaciones pueden ser planeadas hasta después de la instalación de otras aplicaciones que operan con datos comunes. Tales programas requieren la habilidad de operar de improviso sin esperar la existencia de otros. Algunas de las necesidades con las que debemos contar son: la facilidad decontar con acceso a un mismo conjunto de datos, el control sobre este acceso, el control sobre interferencias entre programas independientes, mecanismos eficientes de acceso hacia diferentes subconjuntos de campos.

La ausencia de estas necesidades son el resultado de la dependencia de datos y originan en gran parte la pérdida de eficiencia en el manejo de archivos tradicionales cuando pretendemos compartir datos. Por ejemplo, la incapacidad de un programa para ver y accesar solo un subconjunto de elementos de datos enun archivo usado por otro programa. A medida que una base de datos crece y su número de usuarios se incrementa, es más necesario tener acceso eficáz solo a aquellos subconjuntos de interés para un determinado programa.

NO REDUNDANCIA

El modo tradicional de hacer las cosas, como capturar y codificar datos para programas específicos y con eso generar datos más o menos permanentes y exclusivos para aquellos programas, ha dado como resultado una redundancia muy costosa de registros y archivos através de los sistemas de aplicación involucrados (por ejemplo, un sistema de inventarios, un sistema de cuentas).
Esta forma tradicional de operar con la correspondencia "Un archivo- Un programa" (pésima), ha sido el resultado de un buen número de retrocesos en la tecnología, particularmente en la pérdida de independencia de datos.

Tracemos ahora el típico escenario en el que se observa esta costosa redundancia de datos.

Una compañía desarrolla un sólo sistema de aplicación, digamos un sistema de pagos denominado AP. Este utiliza tres archivos de datos, F1 que contiene a los elementos a, b y c; F2 que contiene a los elementos d, e y f; y F3 que contiene a los siguientes elementos g, h, i y j.(figura C)*

Ahora la compañía desea desarrollar un segundo sistema, digamos un sistema de control de ventas denominado PO. Este utiliza dos archivos F1 que contiene a los elementos a, b y c; F4 que contiene a los elementos e, f, i y k. La gente encargada de desarrollar el sistema PO negocian el uso del archivo F1 con aquellos elementos cargados por el sistema AP quienes tienen supropio archivo F1. Notemos que F4 incluye los elementos e, f los cuales ya están cargados en el archivo F2 y el elemento i cargado en F3. La posibilidad de formar un archivo nuevo FN que contenga todos los elementos de F2 y F3 más k para ser accesados por ambos sistemas (AP y PO) no surte efecto en virtud de la

* Ver lista de figuras a partir de la hoja #93

gran cantidad de dolores de cabeza que eso implica.

Primero, si FN está formado, entonces los programas de a -- plicación del sistema AP tienen que ser modificados para refle - jar la estructura entera del nuevo archivo FN. Segundo, la gente de apoyo del sistema AP no quiere guardar huella de cualquier dato extraño, tal como k el cual resulta ser el elemento problema en la carga de PO. Tercero, el ciclo de uso del archivo F4 - es diferente al ciclo correspondiente de los archivos F2 y F3 y- algunos conflictos aparecen con respecto a la coordinación para- el uso de FN por dos aplicaciones tales como: responsabilidad - de control de integración y seguridad de los datos, etc. Como - observamos los problemas son muchos, por lo tanto el soporte del sistema PO desarrolla el archivo F4 todo para sí mismo y así los elementos e, f e i están ahora cargados en dos diferentes luga - res. Pronto en un lapso de tiempo muy pequeño, los valores de - e, f e i pueden no ser los mismos en ambos lugares debido a di - ferentes usos dados por AP y PO, como diferentes ciclos de actua lización.

Una tercera aplicación, digamos un sistema de inventarios - denominado RM es desarrollado. Situaciones similares aparecen - como antes y al final puede ser que se requiera la generación de un nuevo archivo F5 con elementos a, b, l y m, duplicando con -- esto a y b que están contenidos en el archivo F1. Otras aplica- ciones pueden ser desarrolladas cayendo en el mismo problema.

Después de computarizar algunas aplicaciones con las prác - ticas convencionales, la experiencia muestra que hay un tremendo problema de redundancia, la cual invariablemente causa inconsis- tencia en la exactitud de los datos, en el formato, nombre ... - etc. Aunque todos los problemas administrativos inherentes a -- los esfuerzos iniciales para integrar sistemas fueran eliminados los problemas que implican la reprogramación de aplicaciones y -

su respectiva redocumentación y la programación para controlar y coordinar el uso de una Base de Datos compartida por muchos programas demandantes son gigantescos.

La redundancia tiende a crecer. Su gasto es muy grande para cualquier compañía. El peor rasgo es que a la larga una compañía sigue el patrón tradicional y conserva nuevos programas - adheridos y redundantes archivos de datos, estructurados específica y únicamente para aquellos programas, el remedio debe darse cuando finalmente se ensamblan todos los datos en un único sistema de manejo de Base de Datos.

La meta de los sistemos de Base de Datos es permitir y fomentar la centralización e integración de todos los datos de un conjunto de aplicaciones dentro de un Banco de Datos, esto es, - cambiar la situación vista en la figura C dentro de la figura D* mostrando la remoción de redundancia. Todos los elementos de datos usados por un grupo de programas y/o programadores deben ser centralizados y accesados con control propio.

La remoción de redundancia es un camino que tiene que ver - con la idea de compartir datos. Cuando dos diferentes aplicaciones necesitan el mismo dato y no lo comparten, se dá la redundancia. Sin embargo, en un medio ambiente de Base de Datos puede - ser necesaria alguna redundancia independiente de los problemas de compartir datos. para evitar un tiempo excesivo en el acceso de los registros es conveniente generar alguna redundancia de -- campos invertidos. Sin embargo los programas de mantenimiento - evitan desafortunadamente que esa información redundante sea consistente. La idea que pretendemos dar es la eliminación de la - redundancia tanto como sea posible por medio de la Base de Datos, y donde esta sea justificable generarla con el objeto de obtener una mayor eficiencia o por razones internas del G D B M S entonces

* Ver lista de figuras a partir de la hoja # 93

ces los programadores de aplicaciones y los usuarios estan protegidos a partir del conocimiento y la recuperación de la misma. En el caso de respaldo (back-up) el programador de la aplicación debe contar con una imagen redundante de todas las operaciones a un tiempo determinado, así como una copia total de la base de datos en ese mismo tiempo, para poder efectuar el respaldo y que la posterior recuperación de los datos no sea un problema.

RELACION DE DATOS

La relación de datos es un término utilizado con frecuencia, desafortunadamente, no con el mismo significado en todos los casos. Sin embargo, en la gran mayoría de estos se utiliza para denotar la propiedad de la existencia de relaciones entre diferentes registros lógicos. Un registro está constituido de una serie de atributos, por ejemplo en el caso de un registro de empleados sus atributos correspondientes podrían ser: nombre, dirección, departamento, salario etc. Las relaciones existentes entre registros son por ejemplo, los hijos de un padre, los grados escolares de los alumnos, las habilidades de un empleado. Las relaciones son tan importantes que deben ser identificadas claramente, se dan entre cualquier registro y sus atributos de datos, y deben estar definidas y manejadas por un sistema de base de datos en el cual puedan derivarse otras asociaciones, por ejemplo, empleados contra una habilidad. La figura E * muestra por medio de las flechas punteadas las relaciones que pueden existir en una base de datos de empleados que contiene cuatro tipos de registros.

La facultad de definir relaciones entre registros a nivel lógico tan convenientemente como definir registros es un objetivo mayor en la tecnología de base de datos. Discutiremos subse-

* ver lista de figuras a partir de la hoja # 93

cuentemente varios tipos de relaciones y maneras de representarlas en los niveles lógicos y físicos.

INTEGRIDAD

Con muchos usuarios diferentes compartiendo los mismos datos y ahora con todas las relaciones involucradas, es imposible para cada usuario ser responsable de la consistencia de los valores en la Base de Datos y del mantenimiento de las relaciones de los campos de datos que el ve con todos los otros campos de datos, algunos de los cuales pueden ser desconocidos para él, o -- bien que esté prohibido su acceso para él mismo. Una herramienta mayor de un sistema de Base de Datos es mantener control y -- preservar la integridad de la Base de Datos.

El término integridad se ha desarrollado para referirse a la coordinación del acceso de datos por diferentes programas, -- propagación de actualización de valores para otras copias y valores dependientes, y garantizar la validéz de los datos (tales como pruebas de consistencia, edición de entrada y salida).

La preservación de la integridad incluye también mantenimiento denominado audit trail, es decir una bitácora o un registro de todos los accesos y cambios a cada campo de datos en cualquier punto del tiempo en el cual un acceso o cambio toma lugar y de la interacción de programas y datos, así esa integridad puede ser recuperada si un error es detectado más tarde.

FLEXIBILIDAD DE ACCESO

La flexibilidad de acceso se refiere a la propiedad de poder fácil y eficientemente accesar datos en una variedad de for-

mas. Es un término suelto que incluye la facilidad de accesar - datos sobre la base de cualquier llave de acceso y calificación- lógica (en lugar de estar restringido a un fácil acceso por la- vía del SORT o bien por los métodos básicos de acceso, o por el- acceso por llave) esta facilidad se obtiene mediante el uso del- lenguaje QUERY (lenguaje orientado a personas sin conocimientos- de programación, con instrucciones de fácil comprensión y a la - vez poderosas para navegar eficientemente a través de la Base de Datos en un modo de acceso inmediato). Este término incluye -- también la habilidad en un programa de un lenguaje convencional- de programación para usar la Base de Datos y accesar eficiente - mente cualquier subconjunto de datos, y finalmente la facilidad- de poder utilizar mecanismos que permiten el control del acceso- y la administración de la Base de Datos.

SEGURIDAD

Un sistema de manejo de Base de Datos debe tener sus pro -- pios mecanismos para asignar control, y remover los derechos de - acceso (leer, escribir, insertar, dar de baja, cambiar) de cual quier usuario para cualquier campo. El sistema debe garantizar - la seguridad de los datos. Ciertos campos,combinaciones de cam - pos o selección de campos pueden ser sensitivos y requerir altos niveles de autorización para accesarlos. Personas diferentes -- pueden estar restringidas para ver diferentes conjuntos de los- datos; por ejemplo, datos del personal excepto datos médicos o -- campos de salarios. Control de acceso sobre la base de contenido de datos puede ser necesario en tiempos; por ejemplo, permiso pa- ra leer solamente aquellos salarios donde éste sea menor o igual- a $ 30,000.00.

Como la cantidad de datos compartidos y el número de progra- mas de usuario se incrementa, la herramienta del sistema que nos-

dá la certeza de la seguridad se incrementa. Un campo de dato - debe estar completamente protegido de una intromisión no autorizada, esta puede ser accidental o maliciosa, por cualquier usuario o programa de aplicación. El administrador de la Base de Datos debe tener facilidades para asignar y controlar el acceso de los usuarios.

EJECUCION Y EFICIENCIA

La buena ejecución y eficiencia en un sistema de Base de Datos es un requisito primordial en virtud de los grandes tamaños de las Bases de Datos. Cada facilidad que debe ser añadida para satisfacer los objetivos que han sido planeados, es una carga -- adicional al diseño, organización, implementación y ejecución en un sistema de manejo de Base de Datos. La viabilidad de una característica que satisface las necesidades de un usuario depende de la ejecución y eficiencia (costo) involucrados.

La necesidad de contar con una organización física eficáz - y un adecuado manejo de datos es muy grande debido al incremento de datos almacenados en la Base de Datos, así como nuevas relaciones y nuevos programas de aplicación. A medida que crece la Base de Datos y el número de usuarios mayor es la probabilidad - de que cualquier subconjunto por pequeño que sea de la Base de - Datos sea de interés para cualquier usuario. Este crecimiento - puede ser en forma individual a través de terminales usando lenguaje natural o bién por nuevas aplicaciones y cuando utilizamos las facilidades convencionales para el manejo de datos caemos en la clásica ineficiencia y las búsquedas exhaustivas.

El avance en la tecnología de la organización de datos ha - permitido implementar organizaciones físicas eficientes que soportan requerimientos lógicos complicados. Por ejemplo, el pro-

ceso secuencial de subconjuntos de la base de datos debe ser tan eficáz como el que se obtiene al procesar un archivo secuencial. Las organizaciones invertidas están orientadas para obtener una- gran eficiencia en el acceso de datos y son la estructura básica de la gran mayoría de los sistemas de manejo de base de datos ac tuales.

Cuando pretendemos efectuar cambios en una base de datos -- los problemas de ejecución son mayúsculos. Los cambios en el -- contenido de la base de datos son permanentes, por esto el tiem- po de acceso, el espacio de almacenamiento y todo el soporte en- general sufren con el cambio de una manera mucho más compleja y- más difícil de entender que cuando se trata de un simple archivo. En la discusión de la independencia de datos mencionamos varios- grados de independencia entre un programa y los datos almacena - dos, si debido a un cambio solamente en la naturaleza de las ope raciones que pudiera ser accesar registros utilizando diferentes llaves sin modificar los programas de aplicación, o bién utili - zar campos que no se encuentran en los datos almacenados, incre- mentamos el tiempo de acceso en la ejecución lo cual desgraciada mente ocurre en los sistemas de manejo de Base de Datos enton - ces aparentemente incurrimos en una violación de la independen - cia de datos. Aparentemente por lo siguiente: el programa de -- aplicación todavía trabajará pero el tiempo de acceso a los da - tos es demasiado costoso.

Un sistema de base de datos idealmente debe tener la habi - lidad de modificarse, reorganizarse o reinicializar parcial o to talmente la base de datos, así como mantener un nivel aceptable- de ejecución y eficiencia bajo cualquier prioridad y demanda.

El reporte GUIDE/SHARE publicado en 1971 establece que un - sistema de base de datos debe poder medir por si mismo tanto el-

tiempo que ocupa en reorganizarse así como para mantener un nivel optimo de ejecución para cada usuario. Esta medida automática es una meta ideal pero rechazable desafortunadamente en muchos años-venideros, debido a que existen un gran número de parámetros de - ejecución y de interrelaciones involucradas que son necesarias de entender primero por los diseñadores de los sistemas de manejo de Base de Datos. Por mencionar alguno, muchos usuarios tienen di - ferentes requisitos sobre los mismos datos, lo cual causa conflic tos en la ejecución (ejemplo, un grupo de usuarios puede necesi - tar accesar un número de datos unicamente para leer, mientras que otro grupo necesita actualizar este mismo número de datos rapida-mente), en esta medida sólo los de total provecho serán optimiza-dos originando con esto que algunas aplicaciones sufran en su eje cución para el bien de otras.

ADMINISTRACION Y CONTROL

Un ingrediente crucial en la introducción de conceptos de Ba se de Datos y software es la administración y control de una Base de Datos. La responsabilidad para el control y la descripción de-los datos no debe ser difundida entre los muchos programadores de aplicaciones y los analistas. Deben ser centralizadas. Las des -- cripciones y el control de los datos almacenados deben estar des-conectados de los programas de aplicación (una noción básica de - la independencia de datos) y deben estar declarados de una manera común o compatible a todos los sistemas de aplicación y a los len guajes comunes de programación. El diseño total de la Base de Da-tos, las declaraciones de datos, y el camino a los usuarios para-usar los datos son ahora total y absoluta responsabilidad de la - función de administración de la Base de Datos. La determinación-de los niveles de ejecución y eficiencia deben tomar en cuenta el provecho total y así residir en las funciones de la administración.

BENEFICIOS AL ALCANZAR LOS OBJETIVOS DE LA BASE DE DATOS

Como ahora entraremos en detalle en las siguientes secciones aseguraremonos de que no hemos perdido huella de los beneficios que todos los objetivos de la tecnología de la Base de datos buscan en un sentido completo: mayor productividad y mayores beneficios por menor costo.

El desarrollo, integración y mantenimiento de los sistemas de aplicación (por ejemplo, un sistema de órdenes de entrada, un sistema de control de inventarios etc.) es un punto fundamental. Muchos sistemas no existirían hoy en día si no fuera por los sistemas de manejo de Base de Datos. Los beneficios de la inversión son mucho más altos cuando la tecnología de Base de datos es utilizada juiciosamente.

La diferencia entre poder y no poder desarrollar sistemas de aplicaciones integrados y eficientes puede ser la tecnología de Base de Datos.

ESQUEMAS, SUBESQUEMAS Y RELACIONES

Fundamental y comun a todos los sistemas de Base de Datos es la habilidad para:

1.- Definir la estructura lógica de cada caso de dato que -- forman la Base de Datos. Esta estructura lógica es llamada el esquema o la Base de Datos Lógica(usaremos estos dos términos através de esta sección) y está definido mediante un lenguaje especial de definición de esquemas -- por un diseñador de Base de Datos o un administrador. - Pueden ser definidas varias Bases de Datos independientes.

2.- Definir y controlar el acceso a cualquier subconjunto - lógico de la Base de Datos. La estructura lógica de este subconjunto es comunmente llamada subesquema. Es un- programa de aplicación o una visión del programador de - la descripción lógica de sus archivos o Base de Datos, - mientras en realidad es un subconjunto del esquema global compartido por un número de programas. Cualquier -- número de subesquemas pueden ser definidos en un esquema; cualquier número de programas puede compartir un subes - quema; y diferentes subesquemas pueden traslaparse.

La figura F* muestra esto conceptualmente. Los términos es - quema y subesquema fueron originalmente introducidos por CODASYL- en el reporte DBTG en el año de 1971. Un buen número de diseña - dores y vendedores usan diferentes términos para conceptos similares. Por ejemplo, el término usado para el esquema por el IMS- de IBM es Base de Datos Lógica; a un subesquema se le denomina la sensibilidad de un programa de aplicación particular o de un programador. El término Base de Datos Lógica es usado tan frecuen -

*Ver lista de figuras a partir de la hoja # 93

temente como el término esquema en la práctica actual. Usaremos cualquiera de los términos antes mencionados en esta sección.

Ahora ilustraremos un esquema, subesquemas y otros conceptos importantes con un ejemplo específico. La figura G muestra como un esquema, un conjunto de registros de diferentes estructuras aparecen interconectados por líneas sólidas. Este tipo de diagrama es comunmente utilizado para denotar esquemas y subesquemas. Desde luego, este tiene que estar mapeado individualmente dentro de la definición formal equivalente en el lenguaje de descripción de datos. Cada tipo de registro es llamado también una entidad. Cada entidad contiene un número arbitrario de campos elementales también llamados atributos. Una línea sólida representa la relación entre un registro y otro. Como se planteo antes, una relación es una indicación definitiva de como un registro está relacionado con otro. La relación es tan importante y tan definible como cualquier atributo o registro. En la terminología usada por el reporte DBTG de CODASYL en 1971, una relación es denominada un conjunto. El registro DOCTOR es conectado al registro EMPLEADO por medio de la relación:

EMPLEADO-PACIENTES-DE-DOCTOR

El registro EMPLEADO está por su parte conectado a sus HIJOS mediante la relación HIJOS-DE-EMPLEADO.

Notemos que el registro SEGUROS-CO esta todo por sí mismo sin una explicita línea solida de conexion a otros registros. Sin embargo, el registro EMPLEADO está definitivamente relacionado a él por medio del atributo SEGUROS-CO-NOMBRE cargado dentro del registro.

En general, cargando los identificadores de cada caso de una entidad X como atributos de un registro Y, una relación está

efectivamente establecida entre X y Y. Uno puede así representar la relación entre cualesquier dos registros en dos formas como -- ejemplificamos en la figura H*. Este ejemplo muestra, de la manera mas sencilla, la diferencia de ver esquemas o subesquemas en - tre los sistemas convencionales existentes y los sistemas tabulares o relacionados más recientemente avocados. La figura H*(2) - muestra la visión del programador de aplicación de una relación ó bandera del archivo equivalente. Más en la relacional contra la- convencional vista más adelante.

La figura I* muestra dos de los muchos posibles subesquemas - del esquema de la figura G*. Otra vez cada subesquema representa- el esquema particular de un programa dado de aplicación o usuario. Cuando llegamos a un subesquema podemos hacer lo siguiente:

1.- Omitir uno o más registros o entidades del esquema.
2.- Omitir una o más relaciones del esquema.
3.- Omitir uno o más campos de datos de un registro o entrada.
4.- Reedificar el órden relativo de los campos de datos dentro del registro que lo contienen, e introducir estruc - turas adicionales dentro del registro.

Más allá de estos pasos, no todos los sistemas de Bases de - Datos pueden ir. Por ejemplo, sólo algunos pueden permitir el -- cambio en la declaración del tipo de dato de un campo de dato en- el subesquema. Otra vez, la herramienta compleja de los sistemas de manejo de Base de Datos es para soportar subesquemas y encon - trar los objetivos expuestos en la sección anterior.

RELACIONES

Desde el punto de vista convencional de los esquemas, una re lación entre dos entidades puede ser de cuatro tipos:

*Ver lista de figuras a partir de la Hoja # 93

1.- De uno a uno
2.- De uno a N
3.- De N a uno
4.- De M a N

La figura J muestra ejemplos de cada tipo, en la relación -- entre DOCTOR y EMPLEADO. Notemos que a este nivel lógico, se pretende no estar sujeto a la máquina o a consideraciones de almacenamiento, el acercamiento relacional no se preocupa por si la relación es de cualquiera de estos cuatro tipos o no (figura H). En la tecnología convencional uno debe estar enterado de los cuatro tipos, así procura cuidar las consideraciones de almacenamiento. Los tipos de relaciones que pueden ser definidas, como pueden estar definidas, como pueden navegar dentro de la Base de Datos, etc., varia de sistema a sistema así como de grupo de interés en grupo de interés (por ejemplo, CODASYL VS GUIDE/SHARE).

REGISTROS, GRUPOS REPETITIVOS, SEGMENTOS, ...

Como hemos dejado asentado, la terminología en la tecnología de Base de Datos no está estandarizada, y puede ser muy confusa. Cada competidor en la tecnología de Base de Datos implementada y propuesta por desgracia tiende a acarrear mucha terminología particular solamente a él. En esta Tésis tendemos a usar términos cercanos al convencional CODASYL y al mundo de COBOL. La figura K sumariza algunos de los términos principales y sus sinónimos, o al menos su equivalencia más cercana, entre varios implementadores y grupos competitivos.

Algunos GLBMS no hacen una distinción entre grupos repetitivos y registros. Por ejemplo el IMS de IBM y CICS se refieren a un grupo dirigido particularmente de campos de datos elementales como un segmento; los postulados de ENTRADA/SALIDA recuperan seg-

mentos enteros directamente, pero no campos individuales dentro-- del segmento. Los sistemas CODASYL recuperan registros enteros- o directamente entidades, y sólo después se pueden accesar en memoria grupos repetitivos o campos de datos elementales de este - registro.

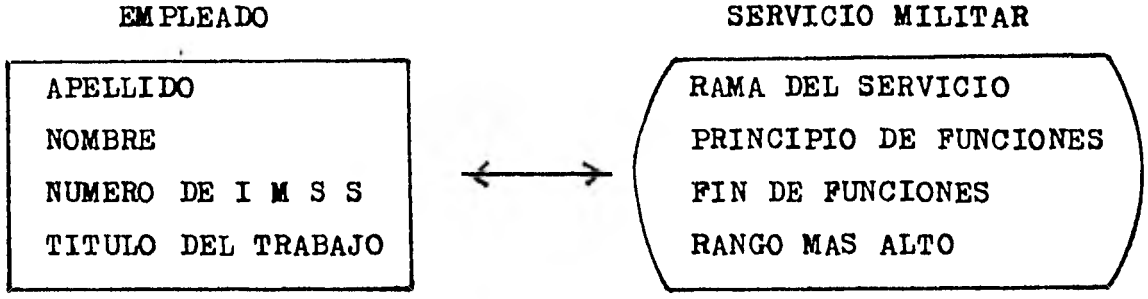
GRUPO DE RELACIONES

UNO-A-UNO, UNO-A-MUCHOS y MUCHOS-A-MUCHOS

Cuando dos grupos están relacionados entre sí, ellos pueden relacionarse en cualquiera de las tres siguientes formas:

1) Uno-a-Uno.

Los dos grupos ilustrados enseguida muestran la relación - uno-a-uno.



En virtud de que el registro de cada servicio militar pertenece solamente a un empleado y ningún empleado tiene más de un - registro de servicio militar decimos que existe una relación uno a-uno. Esto se indica con una sola punta de flecha con puntas - cercanas al fin de la linea conectora de los dos grupos ( ) Las relaciones uno-a-uno son menos comunes en las definiciones - finales de las Bases de Datos que las otras dos relaciones que - veremos más adelante. La aplicación usual de una relación uno - a-uno entre dos grupos es el ahorro de espacio. La información-

militar ocurrirá solamente una vez para cada persona. Debido a esto la información militar puede ser parte del grupo EMPLEADO - como se muestra a continuación:

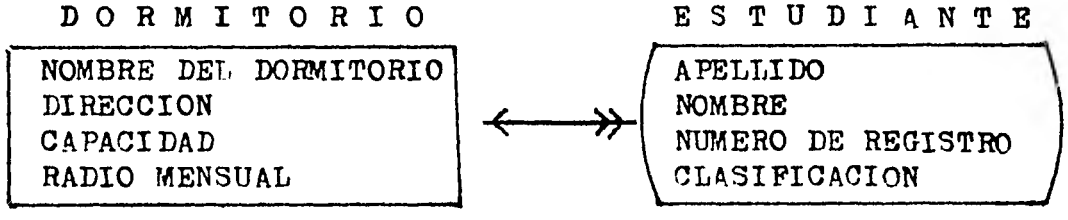
E M P L E A D O

APELLIDO
NOMBRE
NUMERO DEL IMSS
TITULO DEL TRABAJO
RAMA DEL SERVICIO
PRINCIPIO DE FUNCIONES
FIN DE FUNCIONES
RANGO MAS ALTO

La razón de hacer del servicio militar un grupo separado es que no tome espacio en los registros del servicio para empleados que no tiene registro del servicio.

2) Uno-a-Muchos.

Los grupos en la siguiente ilustración nos muestran la relación uno a muchos

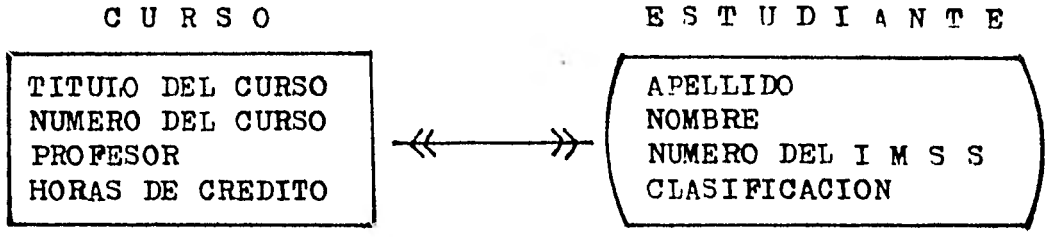


Cada estudiante vive en solamente un dormitorio, pero cada-dormitorio alberga a muchos estudiantes. La relación uno-a-muchos en este caso es un dormitorio para muchos estudiantes. Esto lo indicamos mediante una punta de flecha apuntando al grupo-que ocurre solamente una vez; ponemos dos puntas de flecha apuntando al grupo que ocurre más de una vez ( ). Las-relaciones uno-a-muchos son muy comunes y forman la base prima -

ria para construir jerarquías.

3) Muchos-a-Muchos.

Los grupos ilustrados enseguida forman una relación muchos a-muchos :



Un estudiante toma muchos cursos. Cada curso tiene muchos-estudiantes enrolados. Así existe una relación muchos-a-muchos-entre el grupo de cursos y el grupo de estudiantes... muchos cursos por estudiantes y muchos estudiantes por curso. Puntas de - flechas dobles en cada fin de la línea conectora de grupos indican la relación muchos-a-muchos (⟪——⟫).

EL USO DE PUNTAS DE FLECHA EN LOS DIAGRAMAS DE DISEÑO PRELIMINAR DE UNA BASE DE DATOS.

El uso de puntas de flecha para indicar:

UNO-A-UNO (<——>)
UNO-A-MUCHOS (<——⟫)
MUCHOS-A-MUCHOS (⟪——⟫)

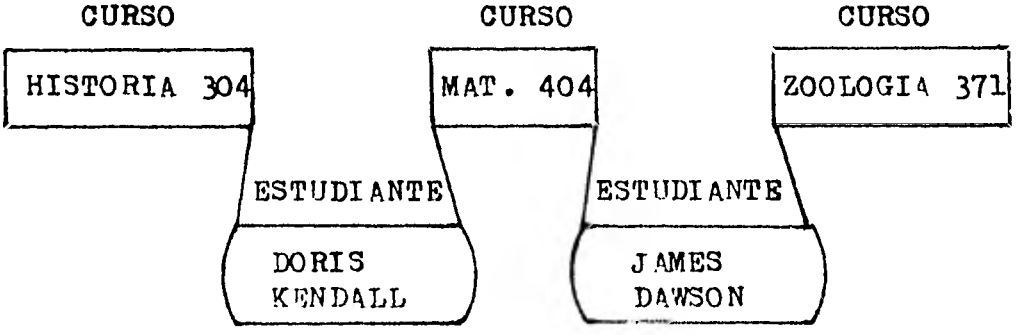
es muy importante en el proceso de diseño ya que cada línea co - nectora entre dos grupos debe claramente indicar que tipo de relación existe entre los grupos.

RELACIONES RECONOCIDAS. DONDE HAY MAS DE UN "PADRE"

Las puntas de flechas dibujadas en el diagrama son importantes en grupos reconocidos con más de un "PADRE". Los diagramas-alternos deben también incluir flechas apropiadas.

SYSTEM 2000 soporta directamente estructuras jerárquicas. - En una jerarquía nunca hay más de un "PADRE". SYSTEM 2000 tam - bién soporta estructuras de redes (network) indirectamente. En- una estructura de redes, puede haber más de un sólo "PADRE". - SYSTEM 2000 puede soportar las redes ya sea desmontandolo dentro de una jerarquía o por medio del uso de un verbo perteneciente - al PLI (lenguaje de programación de interface) llamado LINK. - Es importante reconocer la existencia de las relaciones con más- de un sólo "PADRE", así que la elección apropiada para trabajar- en redes puede ser ejecutada.

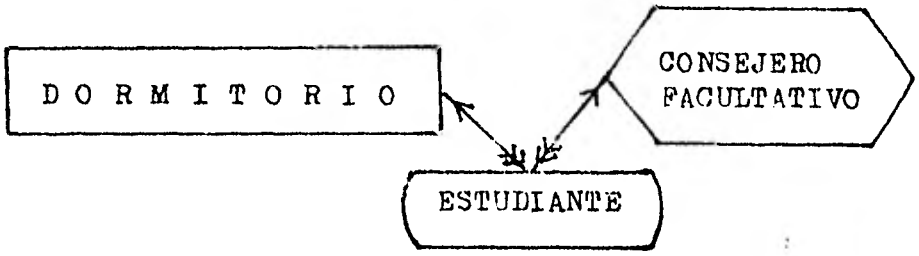
La relación muchos-a-muchos siempre implica más de un "PA - DRE". En nuestro ejemplo para relaciones muchos-a-muchos había- muchos estudiantes por curso y también muchos cursos por estu -- diante. Los datos podrían parecerse a lo siguiente:



Cada uno de los estudiantes tiene más de un curso como "PA- DRE". Si invertimos el diagrama y ponemos a los estudiantes -- arriba MAT. 404 tendría todavía más de un "PADRE". Cada estu --

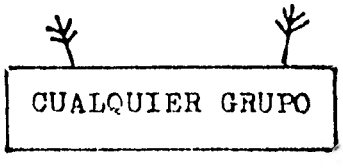
diante enrolado podría ser un "PADRE" para el curso. Una relación muchos-a-muchos siempre implica más de un "PADRE" y con esto una estructura de redes.

Si un grupo dado es el "HIJO" en más de una relación uno-a-muchos, este grupo tiene más de un "PADRE". Suponganos la siguiente relación:



Cada estudiante vive en solamente un dormitorio y cada estudiante tiene solamente un consejero facultativo. Todavía cada dormitorio alberga muchos estudiantes y cada consejero orienta a muchos estudiantes. Así un estudiante puede tener al dormitorio como "PADRE" y también un consejero facultativo como "PADRE". En una jerarquía solo puede haber un "PADRE", así que seguramente usaremos el verbo LINK de PLI para las estructuras de redes de estudiantes, dormitorios y consejeros.

Para resumir: una relación muchos-a-muchos implica más de un "PADRE". Así la notación  muestra la presencia de más de un "PADRE". También un grupo tiene más de un "PADRE" si es el objeto de más de una relación uno-a-muchos. Así la estructura :



significa más de un "PADRE".

ORGANIZACIONES LOGICAS: JERARQUICAS O DE ARBOL, REDES Y RELACIONALES

Un archivo bandera es aquel en el cual cada registro tiene un número similar de campos (figura H (2)). Los archivos que constituyen la estructura lógica en el convencional COBOL sin grupos repetitivos son archivos banderas. La nueva tecnología de Base de Datos ha introducido estructuras lógicas más complejas en Base de Datos através de registros interconectados. Estos pueden ser clasificados dentro de tres categorías:

1.- Arboles, los cuales incluyen los llamados archivos jerárquicos y las estructuras de árboles DBTG.

2.- Redes, las cuales incluyen las estructuras de redes DBTG.

3.- Relacionales.

De estas, sólo las relacionales mantienen la simplicidad lógica de los archivos bandera. Arboles y redes no son banderas. En los parrafos siguientes introduciremos y discutiremos sus diferencias, y aquí recurriremos a diagramas de bloque para mostrar graficamente las estructuras lógicas y sus diferencias.

Observemos que son tres alternativas de ver los datos en un nivel lógico, independiente de las implementaciones de máquinas y almacenamiento. Arboles y redes predominan en todos los sistemas comerciales desde 1976. De estos dos, las redes son las que más abarcan, debido a que los árboles son un caso especial de una estructura de redes. Cualquier estructura de redes puede ser convertida en un árbol o en un conjunto de árboles, aunque valores de datos redundantes pueden tener que incluirse. Tam --

bién, estas dos complicadas Bases de Datos pueden romperse en archivos bandera por medio de la inclusión de campos de datos redundantes. Estos archivos bandera de dos dimensiones son llamados relaciones.

La naturaleza de las organizaciones y las relaciones de datos en la mayoría de los casos prácticos es tal que ya sea árboles o redes pueden manejar la mayoría de estas situaciones sin dificultad.

Ha habido considerables debates acerca de los prós y contras de árboles y redes desde el principio de la tecnología de Bases de Datos en los setentas. A mediados de los setentas el acercamiento relacional estaba siendo comparado y favorecido en el nivel lógico por muchos investigadores y diseñadores de GDBMS sobre los árboles y aún sobre las redes. Sin embargo, todos los sistemas comerciales disponibles en Bases de Datos a partir de 1976 aceptan y manejan datos asumiendo un árbol o una red solamente. La expectativa de muchos individualistas es que estos sistemas también proveerán en el futuro una relacional por medio de una interface para usuarios.

ARBOLES, JERARQUIAS Y CONJUNTOS CODASYL

Las estructuras de arboles han sido usadas por muchos años. Los arboles son usados tanto en la organización y descripción de datos lógica como en la física. En las descripciones de datos lógicas son útiles en la notación de la estructura lógica de los datos, desde la estructura más simple de un sólo registro de archivo hasta más complejas organizaciones multiarchivos como se enfoca en los siguientes párrafos. Los arboles han sido usados en gran medida en la organización física de los datos; por ejemplo, la organización de árbol doblemente encadenada. Varios ti-

pos de arboles han sido discutidos por numerosos autores, particularmente en la organización física. Nuestro interés en los arboles aquí es en la descripción lógica de la Base de Datos.

Para conveniencia del lector, revisemos la naturaleza de los arboles. Un árbol está compuesto de un número de nodos arreglados en una jerarquía (figura L). En esta figura se muestra un árbol. Cada nodo representa un elemento de dato. Cada nodo está relacionado a otro nodo al siguiente nivel más alto; el que está más arriba es llamado el nodo "PADRE". Cada "PADRE" puede tener uno o más elementos en el nivel inferior; estos son llamados "HIJOS", y la conexión con un "PADRE" es llamada RAMA. Un nodo "HIJO" no puede tener más de un "PADRE". La altura del árbol es el número de niveles jerárquicos involucrados. El nodo en la cima es llamado la RAIZ y los elementos en los niveles inferiores los cuales no tienen "HIJOS" son llamados HOJAS. Así revisando el COBOL convencional o la descripción de registro de PL/I, el nombre del registro es la RAIZ , los datos elementales son las HOJAS y un nombre de grupo de campos o un nombre de subestructura es un "PADRE" de uno o más campos de datos. Un árbol binario es un ejemplo de un árbol en el cual un "PADRE" puede tener sólo dos "HIJOS".

Nuestro interés en los términos de la Base de Datos lógica es observar cada nodo de un árbol como un registro, esto es, un árbol debe ser ahora una jerarquía de registros. Esto ha dado como resultado el término archivos jerárquicos refiriendose a un árbol de archivos escalonados. IMS utiliza el acercamiento jerárquico donde cada nodo es un segmento o conjunto de campos. CODASYL introdujo en su reporte DBTG el término conjunto para una relación que denota un árbol de registros de dos niveles. Un número arbitrario de casos de un conjunto del mismo tipo es

un archivo jerárquico de dos niveles. El registro "PADRE" es - llamado el PROPIETARIO. El propietario puede tener uno o más registros "HIJOS", cada uno de los cuales es llamado un registro - MIEMBRO. Cada ocurrencia de un conjunto debe contener una ocurrencia de su propio registro propietario y puede contener un número arbitrario de ocurrencias de cada uno de sus registros miembros. La relación propietario-miembro puede ser 1:1 y 1:N pero no M:N. Un conjunto, que sigue las reglas de los arboles, puede tener solamente un registro propietario dentro del conjunto dado. Un registro propietario y un registro miembro pueden ser estructuralmente el mismo. El conjunto está dando un nombre y debe -- ser definido mediante el lenguaje de declaración de datos. La - figura M muestra un conjunto DBTG.

Una Base de Datos está descrita por un esquema compuesto -- de uno o más conjuntos arreglados de una manera de árbol multi - nivel. Así un registro puede jugar el papel de miembro en un -- conjunto y propietario de otro conjunto. Por ejemplo, en la figura G el registro EMPLEADO es un miembro del conjunto cuyo propietario es DOCTOR y también es el propietario del conjunto con el miembro HIJO.

REDES

La estructura clásica de árbol no permite que un "HIJO" ó - registro miembro en una relación dada tenga más de un "PADRE" ó- propietario. Una estructura de redes es aquella que sí lo permite y es así una organización de datos más general. Es importante darse cuenta que en la aproximación DBTG un miembro puede - tener uno o más propietarios tan largos como cada propietario -- sea en un conjunto diferente. Así el DBTG de CODASYL propone --

capacidades para definir directamente las estructuras de redes,- excepto aquellas que involucran relaciones M:N dentro del mismo- conjunto.Veamos la figura M si dos registros EMPLEADO, digamos - esposo y esposa, cada uno tiene los mismos HIJOS, entonces esposo e hijos deben ser un caso de conjunto diferente al conjunto - esposa e hijos. El grupo CODASYL fué influenciado por el trabajo pionero de Bachman y su GDBMS llamado IDS (Integrated Data - Store) basado en estructuras de redes. El acercamiento a las redes ha sido avocado desde los primeros esfuerzos de Base de Datos a mediados de los sesentas hasta la tecnología de hoy en día.

La figura N muestra dos ejemplos de redes. La agradable visión jerárquica de arboles tiende a perderse ahora, aunque los - niveles pueden ser conservados en un alto grado en la mayoría de los casos con un arreglo apropiado de las redes. Comparemos el- significado lógico de las figuras N(1) y N(2). La figura N(1) - muestra el mismo tipo de registro HIJO en dos relaciones: HIJO-DE-EMPLEADO e HIJO-DE-DOCTOR . Pero en la figura N(2) el - significado es diferente. El registro HIJO ahora está conectado con el DOCTOR como la relación DOCTOR-DE-HIJO. Esto es llamado- un ciclo, que es, uno en el cual un "HIJO" de un nodo se convierte en "PADRE" de un antecesor de dicho nodo. Algunos sistemas - comerciales de Base de Datos no pueden representar ciclos.

Un ciclo quizá pudiera contener solamente un tipo de registro, esto es, el tipo de registro "HIJO" y el tipo de registro - "PADRE" que serían uno y el mismo. Este pudiera ser el esquema- para una Base de Datos con familia de arboles. Muchos sistemas- comerciales y la propuesta del reporte DBTG no permiten tales -- ciclos particulares.

TRANSFORMACION DE REDES EN ARBOLES

Cualquier estructura de redes puede ser transformada dentro de un solo árbol o varios arboles mediante el uso de nodos redundantes. La transformación es fácil y no se necesita agregar -- nuevos niveles de árbol, si la relacion no es M:N. Para este -- tipo de relaciones, la transformación es más compleja e involucra la introducción de un nuevo nivel de árbol.

La figura O ilustra como una red puede ser transformada en- uno o más arboles mediante la introducción de nodos duplicados.

La transformación de cada relación M:N en una red a un árbol vincula el agregar otro nigel. Por ejemplo, consideremos la relación M:N DOCTOR-PACIENTE de los registros DOCTOR Y PACIENTE- de la figura P. La relación M:N es en realidad una red por sí - misma. Cualquier registro DOCTOR puede estar asociado con uno - o varios registros PACIENTE, y cualquier registro PACIENTE puede estar asociado con uno o varios registros DOCTOR.

IMPLEMENTACIONES PARA SOPORTAR ARBOLES Y REDES

La mayoría de las organizaciones físicas orientadas para soportar con eficiencia estructuras de arboles no pueden soportar- estructuras de redes directamente. Así, algunos GDBMS manejan - solamente arboles pero no redes. Sin embargo, en los párrafos - anteriores citamos como puede transformarse una red en una es -- tructura de árbol, así que si es posible usar un GDBMS orientado a arboles para manejar una red. La mayoría de los GDBMS orientados a estructuras de redes serán idudablemente más eficientes- en el proceso de una Base de Datos de redes que lo que sería un- GDBMS orientado a arboles en manejar Bases de Datos de arboles -

que es equivalente de la red. Sin embargo, como hemos visto anteriormente el tiempo de acceso en la ejecución de organizaciones de datos física es enteramente dependiente y altamente sensitiva a muchos factores, incluyendo requerimientos de proceso y contenidos de la Base de Datos, así que las generalidades son peligrosas.

Las implementaciones de dos GDBMS no son semejantes. La diferencia en el acercamiento de la implementación comienza al incrementar el momento en que los access paths (familias) o las organizaciones secundarias de datos empiezan a ser consideradas. Muchas diferencias existen también desde el punto de vista funcional del nivel lógico. Algunos sistemas tienen la capacidad de relacionar solamente un tipo de registro a otro, tales como la situación del registro maestro-a-detalle. Otros pueden manejar relaciones entre un registro "PADRE" y un largo número de tipos de registro "HIJOS". Algunos pueden manejar relaciones M:N. Estas son solamente algunas de las muchas diferencias que se encuentran en las capacidades funcionales en sistemas implementados.

COMPONENTES Y FUNCIONAMIENTO DE SISTEMAS GENERALIZADOS DE MANEJO DE BASE DE DATOS (GDBMS)

Tracemos que pasos generales ocurren en el proceso de un - postulado de ENTRADA/SALIDA que requiere el acceso a un registro de un GDBMS. La figura Q ejemplifica las funciones que se desarrollan:

1.- El postulado de ENTRADA/SALIDA (por ejemplo, DELETE EMPLEADOS DONDE EDAD ≥ 65 AND DEPARTAMENTO = ECONOMICO) es detectado y analizado por el GDBMS. El compilador - del lenguaje del programa que contiene el postulado de- ENTRADA/SALIDA no procesa cualquiera de los postulados- de ENTRADA/SALIDA.

2.- El GDBMS se encarga de ver si el subesquema asociado -- con el postulado de ENTRADA/SALIDA está definido en su- directorio, si está soportado por el GDBMS y puede ser- accesado por el resultado particular del comando de EN- TRADA/SALIDA (Es decir, el password correcto, el per - miso para utilizar DELETE y el acceso en la base de --- EDAD y DEPARTAMENTO permitido).

3.- El GDBMS basado en el análisis del requerimiento, uti - liza sus mecanismos de access paths (es decir, mecanis mos invertidos) de la manera más optima para determi - nar cuales registros califican bajo la pregunta o cua - les son necesitados en órden de responder la pregunta,- y exactamente donde se localizan en el almacenamiento - físico. Los bloques físicos que necesitan ser accesa - dos son determinados.

4.- Los beneficios del GDBMS en los comandos físicos de ENTRADA/SALIDA hacia el sistema operativo para accesar -- los registros físicos específicos o los bloques que se- necesiten que contengan los registros.

5.- El sistema operativo busca los dispositivos secundarios de almacenamiento y accesa los registros requeridos.

6.- El sistema operativo transfiere datos del almacenamiento secundario al área de buffer del sistema en memoria, accesible solamente al GDBMS. El área de buffer del -- sistema es compartido por todos los programas de aplica ción bajo el control de GDBMS.

7.- Los registros particulares requeridos por el programa - de aplicación son derivados por el GDBMS de todos los - datos que tuvieron que ser transferidos a los buffers - del sistema. La transformación de datos y el procesa - miento necesitado por el subesquema particular son ejecutados por los GDBMS así que los programas de aplica - ción pueden ver solamente esos registros específicos.

8.- Los GDBMS transfieren los registros requeridos del área de buffer del sistema al área de buffer de los progra - mas de aplicación.

9.- Los GDBMS proveen el estado de la información del pro - grama solicitado en el resultado de esta solicitud.

10.- Los programas de aplicación ahora pueden manipular en - memoria los registros dados a ellos por los GDBMS.

Esto es obviamente un esquema genérico y muy global de lo--

que ocurre. Los GDBMS disponibles difieren en sus acercamientos de implementación, técnica y pasos en la realización de sus me - tas.

En los sesentas los sistemas de software basados en estructuras multilistas, organizaciones de arboles, etc. fueron desa - rrollados "en casa" con el intento de proveer mayor acceso efectivo a subconjuntos de datos en un medio ambiente creciente de - archivos y para realizar muchas de las metas de compartimiento,- independencia de datos etc. mostradas en la figura C. Los archivos tuvieron que ser eslabonados e integrados. Las facilidades- convencionales en el manejo de datos en los sistemas operativos- no satisfacieron las necesidades. Así, un cuerpo de software de manejo de datos empezó a crecer hacia la cima de compiladores regulares y sistemas operativos. Los intentos fueron hechos para- generalizar las construcciones especiales de software, por ejemplo, definir y cargar facilmente cualquier estructura de regis - tro en un número de archivos, para interconectar cualquier de un número limitado de archivos, para indexar cualquier campo etc. - La mayoría de estos esfuerzos "en casa" fueron muy caros, estu - vieron en un estado constante de flujo y la mayoría de ellos llevaron a cabo solamente en forma parcial sus metas. Estas necesidades y problemas hicieron que los vendedores y las casas de -- software desarrollaran los GDBMS de hoy en día . Es interesante notar que un buen número de estos sistemas "en casa" dieron rú - brica y son usados satisfactoriamente pero sólo "en casa", aun - que la mayoría de ellos serían difíciles si no imposible de reemplazar con un verdadero GDBMS sin extensiva o prohibitiva reprogramación de los programas de aplicación, reformateo de los da - tos, etc.

Los GDBMS pueden ser clasificados dentro de dos categorías:

1.- Aquellos denominados GDBMS .

2.- Aquellos llamados DB/DC.

Los sistemas DB/DC incluyen facilidades para comunicación de datos ó teleproceso y para operaciones de Base de Datos. - Veámos ahora los componentes de los sistemas DB/DC, particularmente desde que la mayoría de los GDBMS pueden ser la interface con un procesador externo de comunicación de datos en el sistema operativo. La supresión de todos los comentarios en comunicación de datos reduce los siguientes parrafos a una corta descripción de lo que hacen los GDBMS.

COMPONENTES DE LOS SISTEMAS DB/DC Y GDBMS

Los sistemas DB/DC son aquellos en los cuales las facili - dades para el manejo de la Base de Datos y el manejo de la comunicación de datos han sido altamente desarrolladas como un sistema integrado. Los primeros ejemplos son: el IMS de IBM y el TOTAL de CINCOM. En pocas palabras, la comunicación de datos - hace a la Base de Datos disponible, así que puede ser accesada- y/o modificada através de terminales. Estas pueden ser teletipo o bien pantalla denominada CRT las cuales están conectadas - al computador por medio de líneas telefónicas privadas o bién - por arrendamiento. Estas terminales pueden estar localizadas - en lugares remotos o en lugares cercanos, en cualquier caso las facilidades de las comunicaciones de datos permiten a un buen - número de usuarios interactuar con la Base de Datos Central a - un mismo tiempo.

Los GDBMS que no son considerados como verdaderos sistemas DB/DC pueden usarse también desde terminales remotas de teclado

y terminales de pantalla CRT si una interface al monitor estandar de teleproceso (TP) del sistema operativo está disponible y la propia programación del momitor esté hecha por el programador de los sistemas de aplicación. Por ejemplo las características del monitor de teleproceso de SYSTEM 2000 permite el uso del conjunto completo de habilidades de SYSTEM 2000 para los teletipos modelo 33/35, para las terminales IBM 2471 y para las terminales CRT de IBM 2260. El sistema ha sido puesto en interface con TCAM, ALPHA, HIPERFASTER y BEST que componen el grupo de monitores de teleproceso comerciales disponibles. Sin embargo los GDBMS utilizan teleprocesadores los cuales no aportarán mayores beneficios, debido a las facilidades altamente desarrolladas y productivas en la comunicación de datos que integran un sistema DB/DC.

Podemos entender mejor las necesidades que los sistemas DB-DC satisfacen si examinamos su evolución. Los sistemas de teleproceso (TP) instalados a principios y a mediados de los años sesentas fueron relativamente simples. Usualmente involucraban una sola aplicación y tenían archivos y terminales las cuales estaban dedicadas a esa aplicación. Si una segunda aplicación se desarrollaba, también tenía sus propios recursos (archivos y terminales). Estos programas de aplicación en teleproceso (TP) tenían interface directamente con el sistema operativo y estaban escritos normalmente en un lenguaje básico de ensamblador porque no había alternativa práctica. En la mayoría de los casos, los acercamientos de la aplicación fueron enteramente elementales y fueron limitados a menudo a una simple pregunta. El programador también tuvo que desarrollar su propio programa de control de comunicaciones, generalmente con lenguajes especiales muy básicos y difíciles de usar tales como BTAM (Basic Telecomunications Access Method) y QTAM.

El medio ambiente inicial de TP requería destreza especial, entrenamiento e ingenuidad para programar lo cual era muy costoso. El proceso de datos necesitaba incluir corridas de muchas - aplicaciones en linea concurrentemente, todas compartiendo la - misma terminal y la misma Base de Datos. En otras palabras lo - que se necesitaba era un sistema DB/DC. Los sistemas operativos de los sesentas no soportaron las nuevas necesidades y como un - resultado muchas instalaciones intentaron resolver los problemas "en casa" así como establecer el nuevo proceso deseado. Muchos- de estos esfuerzos resultaron ser excesivamente caros y muchos - se colocaron para implementaciones parciales e implementaciones- en estado de flujo; la mayoría no eran suficientemente modulares para permitir al programador expanderse facilmente o cambiar su- sistema. Estos esfuerzos pioneros dieron con la tecnología de - los sistemas DB/DC de nuestros días, diseñados para aliviar significativamente los problemas.

La figura R muestra los componentes globales de un sistema- DB/DC que son:

1.- La función del manejo de la Base de Datos.
2.- La función del manejo de Terminales.
3.- La función del Sistema de Control.
4.- La función del Lenguaje de Interface.
5.- La función del Sistema operativo.

Los objetivos y características de la función del manejo de la Base de Datos ha sido discutido en este texto anteriormente. La función del manejo de terminales defiende al programador de - aplicaciones de las complejidades de la red de comunicaciones. Como un resultado, el puede operar sin la carga de programación- para efectuar encuestas, direccionar, cuestionar, convertir de -

de código, y otros factores encontrados en el medio ambiente de- (TP), mientras que un GDBMS el programador tiene toda la carga - de la programación y planeación del monitor de TP, Los progra - madores de aplicaciones necesitan solamente un entendimiento basico conceptual del TP para poder hacer su trabajo. Como su nombre lo indica, la función del manejo de terminales facilita a -- estas terminales operar juntas eficientemente así que cualquier- terminal en la red puede comunicarse con otra terminal o con -- cualquier programa de aplicación en el sistema DB/DC. Esta función también maneja procesos interactivos, respuesta rapida a -- preguntas y seguridad de la terminal. Fomenta la modularidad de los sistemas, así que los programas y cadenas de cambios puedenser manejados sin el mayor rompimiento de las operaciones requeridas.

Como se puede ver de los servicios provistos, la función -- del manejo de terminales es mucho más que un método de acceso TP. Se construye sobre el método de acceso (tal como BTAM y QTAM de- IBM) usando sus líneas y su código de control de la terminal como una base para liberarse, más facil de usar y con servicios -- más comprensivos. Esto es enteramente análogo a la función de - manejo de Base de Datos la cual se contruye en la cima de los métodos de acceso básicos.

La presencia de módulos independientes modulares para el manejo de Base de Datos y el manejo de terminales dieron con la necesidad de un eficiente sistema de control. Es aquí donde la -- función de sistema de control entra en acción. Esta función -- efectúa la interface de los sistemas DB/DC con los sistemas operativos y también coordina la multitud de otros elementos. Estafunción es esencial, se trate de un GDBMS o de un sistema DB/DC. La función de sistema de control es un administrador inteligente

que tiene que optimizar el uso del almacenamiento del sistema y los recursos de proceso y los planes de operación de los programas de aplicación de acuerdo a propiedades definidas por el usuario. Un sistema completo incluye muchas facilidades requeridas para soportar Bases de Datos en operaciones diarias. Estas facilidades son también nombradas utilerías o rutinas de soporte del sistema y son una parte de la dificil función de sistema de control. En seguida se enlistan:

1.- Respaldo, edición e impresión de rutinas.
2.- Rutinas de carga de la Base de Datos.
3.- Rutinas de colección de basura.
4.- Rutinas de análisis y recolección de estadísticas.
5.- Rutinas de reorganización y afinación.
6.- Rutinas de intervención en el proceso.
7.- Rutinas de recuperar, checar puntos y restaurar.
8.- Rutinas para empezar y para terminar.
9.- Rutinas de error en el manejo.
10.- Comprensión de los datos.

Otros servicios necesarios del sistema deben ser también -- provistos. Es en estas áreas donde cada sistema comercial dis - ponible difiere en gran medida. No han sido propuestos estándares suficientemente amplios que deban ser suministrados al usuario o programador. Esto ha sido el reino de implementadores individuales.

Un GDBMS grande o un sistema DB/DC tiene implicitamente el derecho al trato de los elementos y complejidades de un sistema operativo. Esto es evidente cuando uno considera un sistema -- como IMS cargando con sus propios métodos básicos de acceso -- (HSAM, HDAM, HISAM, HIDAM) y sus propios métodos de acceso de -

TP, más allá de archivos básicos y métodos de acceso de TP del - sistema operativo. Todos los GDBMS y los sistemas DB/DC usan un sistema operativo modificado (algunas veces altamente).

La función del lenguaje de interface tiene la tarea de simplificar al encargado del desarrollo de la aplicación el uso del sistema. Por medio de este se permite el uso de transacciones - orientadas a lenguajes tales como COBOL y PL/I así como el lenguaje QUERY para efectuar interface fácil y directamente con el- sistema DB/DC (figuras B y R). Las interfaces han sido también construidas por los vendedores de sistemas generalizados -- para el manejo de archivos, para permitir a sus sistemas la comunicación y el uso de todas las facilidades del sistema DB/DC,- justamente como cualquier otro lenguaje de programación. Estas- interfaces son módulos del GFMS y son vendidas por el vendedor - como una opción extra. Por ejemplo la interface ASI-ST de IMS - y de TOTAL y la interface MARK IV con IMS cuyos precios varían - entre 10 y 15 mil dolares.

SISTEMA DE FLUJO - EL SISTEMA DB/DC IMS DE IBM

Veamos ahora el flujo normal de acciones involucradas en un sistema específico. Se trata del IMS de IBM primero como un sistema DB/DC y después como un GDBMS (excluyendo la función DC).

Sistema de Teleproceso.

Una vez que la región o partición que contiene el programa- de control IMS/VS y una o más regiones que se utilizarán para -- los mensajes del proceso han sido inicializadas por las facili - dades del manejo de trabajo del sistema operativo, ocurre el siguiente flujo del sistema: (figura S)

1.- La facilidad de telecomunicaciones (evento 1) solicita restaurar instrucciones de la terminal maestra. Des -- pues de efectuarse la restauración, la terminal maestra permite la comunicación de todos los usuarios de terminales (evento 2).

2.- Cuando un mensaje de entrada o un mensaje de segmento - está recibiendose (evento 2) la facilidad de telecomunicación invoca la facilidad de servicio común (evento- 3) y el mensaje de entrada es registrado (evento 4) - y requerido (evento 5).

3.- Cuando hay mensaje de entrada pendiente de proceso, y - una región o partición del proceso de mensaje de la clase requerida está disponible para proyectarse, el con - trol es transferido a la facilidad de proyección para - determinar el programa de aplicación de proceso de mensajes que será proyectado. El programa de aplicación - es cargado dentro de la región o partición B y se le dá el control.

4.- El programa de aplicación subsecuentemente hace requerimientos para los mensajes de entrada y/o referencias - de la Base de Datos (evento 6). El control se transfiere a la facilidad del lenguaje de datos para cual -- quier referencia de mensaje (evento 7) o para referencia de la Base de Datos (evento 8). La referencia del mensaje es llevada a cabo através de la facilidad de -- servicio común.

5.- Durante la ejecución del programa de aplicación, se pueden hacer modificaciones a la Base de Datos (evento 8) y/o mensajes de salida pueden ser requeridos (eventos - 5 y 7).

6.- Cuando el programa de aplicación termina o requiere otro mensaje de entrada, todos los mensajes de salida requeridos son transmitidos a la terminal de salida designada (eventos 2 y 3)

Proceso Batch de Teleproceso de Bases de Datos.

Una vez que la región o partición IMS/VS asociada con el teleproceso ha sido inicializada por el sistema operativo, entonces una región o partición BATCH puede ser inicializada. El programa de aplicación en la región o partición batch está controlado por el manejo de trabajo del sistema operativo. Esta región batch puede contener un programa de aplicación para proceso en contraste con el teleproceso de Base de Datos. La facilidad de IMS/VS llamada lenguaje de datos es utilizada para referenciar y actualizar la Base de Datos. Ver figura S. Cualquier referencia de datos está inicializada por el programa de aplicación batch (evento 9).

Proceso Batch de la Base de Datos del Lenguaje de Datos.

Ya sea que existan o no las habilidades de teleproceso de IMS/VS dentro de los trabajos del sistema operativo, la facilidad de IMS/VS llamada lenguaje de datos puede ser usada en un medio ambiente solamente batch como se sigue en la figura T :

1.- El programa de aplicación para proceso de la Base de Datos solo por batch, es inicializado através de la rutina de manejo de trabajo del sistema operativo (evento 1).

2.- La facilidad del lenguaje de datos es invocada por el programa de aplicación (evento 2). El módulo más alto de lenguaje de datos analiza el requerimiento que lla-

ma a la Base de Datos. Dependiendo de la función ENTRADA/SALIDA requerida através de ese llamado se invoca: - la inserción (evento 3), la recuperación (evento 4) o - el reemplazo y supresion (evento 5). Estas funciones subsecuentemente invocan funciones únicas ya sea a la - organización jerárquica secuencial (evento 6) o a la- organización jerárquica directa (evento 7). Estas -- funciones subsecuentemente invocan módulos de métodos - de acceso para ISAM (evento 8), SAM (evento 9), OSAM (evento 10) o VSAM (evento 11).

LENGUAJES DE DESCRIPCION DE DATOS, MANIPULACION DE DATOS Y QUERY

Seis tipos de lenguajes o facilidades de comandos estan involucrados en la Tecnología de Base de Datos:

1.- Los lenguajes de programación de procedimiento normal.

2.- Lenguajes de entrada/salida ó lenguajes de manipulación de datos.

3.- Lenguaje Query.

4.- Lenguaje de descripción de subesquemas.

5.- Lenguaje de descripción de esquemas.

6.- Lenguaje de descripción de datos físicos.

Los describiremos enseguida:

LENGUAJES DE PROGRAMACION DE PROCEDIMIENTO

El programador de aplicación todavía utiliza un lenguaje estándar de programación tal como COBOL para todos los procesamientos de datos en memoria (incluyendo buffers de programa) excluyendo ENTRADA/SALIDA. A menudo, tal lenguaje es también llamado un lenguaje de manipulación de datos, incluyendo en tiempos todos los postulados de ENTRADA/SALIDA para comunicarse con un sistema de Base de Datos. COBOL, FORTRAN, PL/I son los lenguajes con mayor aceptación en los sistemas de Base de Datos.

Como hemos discutido en secciones anteriores el lenguaje altamente no procedural de un sistema generalizado de manejo de archivos (GFMS) puede desplazar los lenguajes convencionales de

programación altamente procedurales. Así varios GFMS son utilizados como interface para comunicarse con un sistema de Base de-Datos. Este no modifica ni ejecuta postulados que no son de ENTRADA/SALIDA de los lenguajes de programación de los GFMS.

LENGUAJES DE ENTRADA/SALIDA O LENGUAJES DE MANIPULACION DE DATOS.

Un programa de aplicación comunica sus necesidades de datos al sistema de Base de Datos mediante un lenguaje de ENTRADA/SALIDA, referido como un lenguaje de manipulación de datos, DML por-el DBTG de CODASYL. Los comandos de DML y los comandos del lenguaje host son mezclados en un programa de aplicación. Este lenguaje de ENTRADA/SALIDA puede ser visto como:

1.- Una extensión del lenguaje de programación host.

2.- Un lenguaje de ENTRADA/SALIDA separado.

Como una extensión al lenguaje de programación, sería concebible independiente del sistema particular de manejo de Base de-Datos utilizado. Como una facilidad del sistema de manejo de -- Base de Datos, podría ser independiente de un lenguaje particular de programación.

Se sigue que lo mejor en algunos sistemas de manejo de Base de Datos sea una interface bastante común a cualquier lenguaje - de programación, es decir en lugar de ver varios lenguajes que - hagan interface con el usuario se ve esencialmente uno. CODASYL, como parte de sus motivos de estandarización, definió en su reporte DBTG en 1971 una extensión a COBOL (introduciendo el término lenguaje de manipulación de datos, DML) con el intento de - obtener una interface estándar a cualquier sistema de manejo de-Base de Datos. Se ha levantado mucha controversia en el terreno en que el DML de DBTG fué construido, ya que viola la independen

cia de datos; por ejemplo los postulados de DML pueden referirse a aquellas áreas que tengan significado físico. El tipo DBTG de GDBMS generalmente soporta este DML.

La posición de la gente especializada en la rama,en 1976 -- fué que un programa de aplicación debe tener todos los comandos- de un lenguaje de ENTRADA/SALIDA ajustados a las expectativas -- del GDBMS particular. Es bien conocido que en muchos casos los- programas de aplicación pueden volverse tan limitados al sistema inicial de manejo de Base de Datos que para cambiar sistemas se- involucra desafortunadamente una extensa reprogramación.

Las diferencias en las clases de modelos de datos lógicos - empeoran la tarea de transferir programas aún más. En alguna -- sección anterior los modelos jerárquicos, de redes y relaciona - les fueron brevemente introducidos. Diferentes comandos de EN - TRADA/SALIDA y particularmente diferentes grados de independen - cia de datos acompañan cada tipo de modelo. La función de Trans ferencia de los programas de aplicación de un sistema de manejo- de Base de Datos a otro usando un modelo diferente de datos in - volucra completa reprogramación (es decir cambiar una aplica -- ción en COBOL de IDS a IMS o viceversa).

LENGUAJE QUERY

Practicamente cada sistema de manejo de Base de Datos ofre- ce su propio lenguaje Query. No existe o no se ha propuesto un- lenguaje query estandar. Sin embargo los lenguajes son bastante similares, la mayoría son no procedurales y complementados con - un lenguaje natural. El lenguaje query pretende lo siguiente:

1.- Manejar información espontánea en línea para recupera - ción o mantenimiento.

2.- Proveer de un lenguaje conveniente sin procedimientos - y sin programación orientada a usar el sistema.

3.- Manejar bajos volúmenes de datos.

4.- Accesar subconjuntos de la Base de Datos que satisfagan un conjunto de calificaciones de datos.

Los comandos del lenguaje query son generalmente diseñados- para no estar relacionados y son procesados individualmente por- el sistema de manejo de Base de Datos. En algunos sistemas los- comandos del lenguaje query pueden aparecer intercalados en un - programa escrito en un lenguaje de programación.

LENGUAJE DE DESCRIPCION DE SUBESQUEMA

El sistema de manejo de Base de Datos debe tener los medios para describir los subesquemas lógicos permitidos a cada progra- ma o usuario, tipos de datos, permisos de acceso etc. Al mismo- tiempo, el lenguaje query de usuario y el programador de la apli cación deben tener los medios para describir su visión de los da tos. El término lenguaje de descripción de datos de subesquema, DDL, introducido por CODASYL, se refiere a estos medios. Como - ya se estableció,otros términos son también usados a menudo, --- tales como la "sensibilidad" de un programa en el medio ambiente de IMS.

Justo como en el caso de DML, el DDL puede ser considerado- ya sea como una extensión del lenguaje de programación host o -- bien como un lenguaje separado de definición de datos. Sin em - bargo, porque cada sistema de lenguaje de programación tiene sus propias convenciones y particularidades, el DDL de subesquema -- tiende a estar dedicado a un lenguaje particular de programación. El DDL de subesquema del reporte DBTG es para COBOL. Funcional-

mente, un DDL de subesquema debe incluir la habilidad para:

1.- Seleccionar los tipos de registro o campos dentro del - tipo de registro del esquema para formar el subesquema.

2.- Establecer la correspondencia entre las descripciones - de campos elementales de datos y organizaciones de re - gistros de esquema y subesquema.

3.- Describir la estructura, el formato, la representación- y otras características generales de los datos consis - tentes de la Base de Datos con las facilidades de des - cripción del lenguaje host.

El modelo de datos del sistema de manejo de Base de Datos - es un primer determinante del particular DDL de subesquema. Las descripciones de los datos de un subesquema para un sistema de - Base de Datos jerárquico difiere de aquellas para un sistema de Base de Datos de redes; la transferencia de programas de un sistema a otro entraña reespecificar las descripciones de datos de- la Base de Datos.

Las reglas para correspondencia entre campos de datos y datos agregados declarados para un registro en el esquema y campos de datos y datos agregados en subesquema no estan estandarizadas. El reporte DBTG propone las reglas para COBOL.

LENGUAJE DE DESCRIPCION DE ESQUEMAS

El lenguaje de descripción de datos de un esquema, DDL, es- el lenguaje usado por el administrador de la Base de Datos para- describirla logicamente toda. El administrador es el individuo- o individuos según sea el caso, que tienen la tarea de desarro - llar una Base de Datos común para un número de usuarios. Una ta-

rea mayor del diseño de proceso de la Base de Datos es definir - la Base tal que se acople a todos los intereses de los usuarios, algunos de los cuales pueden tener necesidades conflictivas.

Es importante tener un solo camino y conjunto de convenciones para especificar un esquema, independiente de los lenguajes particulares de programación y los GFMS que pueden estar usando subconjuntos de la Base de Datos. Cada sistema de manejo de -- Base de Datos tiene su propio lenguaje DDL, reflejando el modelo lógico particular para tener una visión de los datos en los cuales el mundo de información de los usuarios va a estar organizado.

Un DDL de esquema no debe incluir referencias ya sea a la - organización física o dispositivos físicos o espacio medio. Un esquema escrito en DDL es la descripción lógica de la Base de Datos y no esta afectado por la organización física o los dispositivos y la media usada para almacenar los datos. La Base de Datos puede así estar almacenada en cualquier mezcla de dispositivos externos de almacenamiento los cuales están soportados por - el sistema. Algunos dispositivos, tales como cintas, debido a - su naturaleza secuencial y limitaciones, no puede tomar ventajas completas de las facilidades en el DML.

El reporte DBTG propone un esquema DDL el cual nace del medio ambiente de COBOL. En las secciones anteriores hemos estado dando ejemplos los cuales nos permiten conocer todos los conceptos del reporte DBTG.

En la práctica corriente hay muchas diferencias en el DDL - y particularmente en las estructuras de almacenamiento subrayadas, através de los sistemas de manejo de Base de Datos. Esto - es ya de mucho interés. En muchas organizaciones el cometido de algunos de los sistemas más complejos ha sido tan largo que mu -

chos administradores de Bases de Datos reconocen que para cam -- biar a un diferente sistema de manejo de datos será altamente -- costoso y prohibitivamente disturbante para los programas de a - plicación. Así en el sentido más general la independencia de da tos es nula, aunque dentro del reino de un sistema de manejo de- Base de Datos dado hay un grado variante de independencia de da- tos.

LENGUAJE DE DESCRIPCION DE DATOS FISICOS

Una Base de Datos debe estar organizada fisicamente y ma -- peada en dispositivos de almacenamiento. En secciones anterio - res hemos discutido con gran detalle los dispositivos de alma -- cenamiento, las salidas de los datos, los metodos de acceso bá - sicos y el más alto nivel de organizaciones físicas (archivos in vertidos, cadenas de arboles etc.). Muchas alternativas comple- jas aparecen para cada uno de estos niveles arquitectónicos de - organización física de datos. Aqui examinaremos estos niveles.- Un lenguaje de descripción de datos físicos será uno que descri- ba tales aspectos. El reporte DBTG de CODASYL inventó el térmi- no DISPOSITIVOS/LENGUAJE DE CONTROL DE LA MEDIA para el lenguaje que le permite asignar, por ejemplo, datos a dispositivos, y pa- ra especificar el buffer, traslapes. Sin embargo, CODASYL deja- esto en las lides de implementaciones individuales (aunque un co mité CODASYL ha estado desarrollando un lenguaje de datos alma - cenados o de descripción de datos físicos). Cada sistema de ma- nejo de Base de Datos disponible hoy en día utiliza ampliamente- diferentes medios y convenciones mediante los cuales los diseña- dores de Bases de Datos especifican las familias (access paths) y los mapeados lógicos y físicos. En algunos casos, las especi- ficaciones de ciertos aspectos de las familias aparecen a nivel-

de descripción de esquema, por ejemplo, la especificación de los nombres de llaves para invertir en sistemas invertidos.

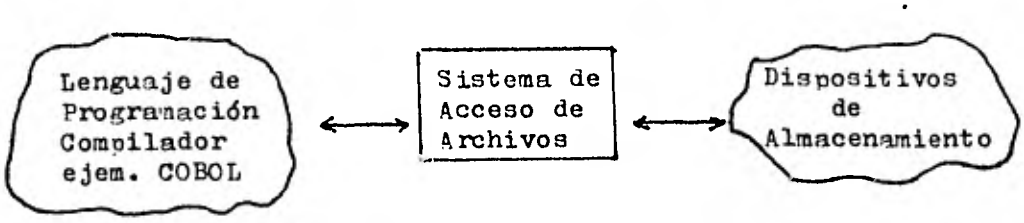
La meta ideal es, desde luego, desarrollar suficiente inteligencia en los sistemas de manejo de Base de Datos para que la elección de las familias y todos los mapeos lógicos y físicos -- sea ejecutada automáticamente y con ello reduzca significativamente los costos. Así, el administrador-diseñador de la Base de Datos no será agobiado por descripciones de datos físicos y no tendrá los muchos dolores de cabeza que tiene con los sistemas corrientes. Pero estamos muy lejos de tener tal necesidad con capacidad automatizada. Esto puede ser el mayor desafío para -- los investigadores y diseñadores de los sistemas de Bases de Datos en los próximos años.

LISTA DE FIGURAS

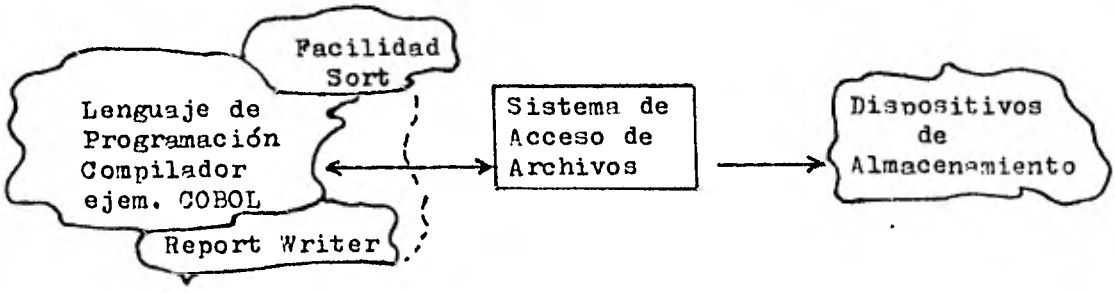
CAPITULO II

FIGURA A CRONOLOGIA Y RELACIONES FUNCIONALES DE LA TECNOLOGIA DE MANEJO DE DATOS.

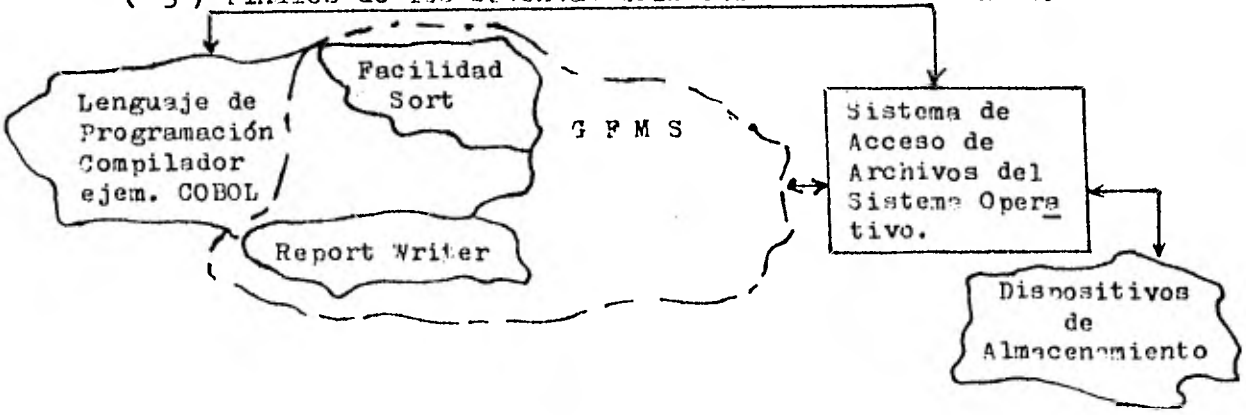
(1) Principios de los sesentas:



(2) Mediados de los sesentas:



(3) Finales de los sesentas principios de los setentas:



(4) Mediados de los setentas

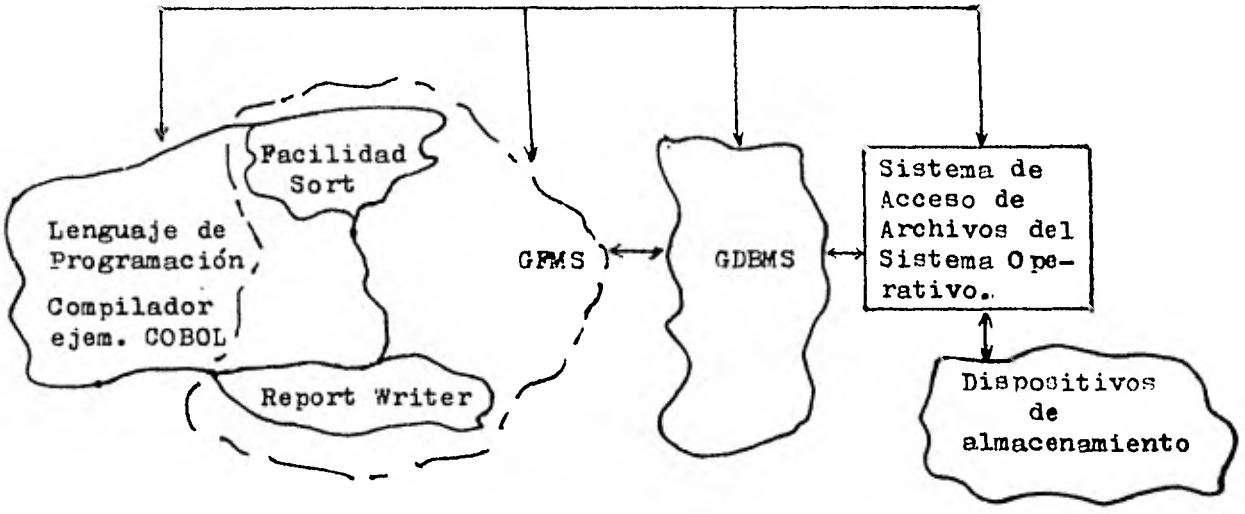
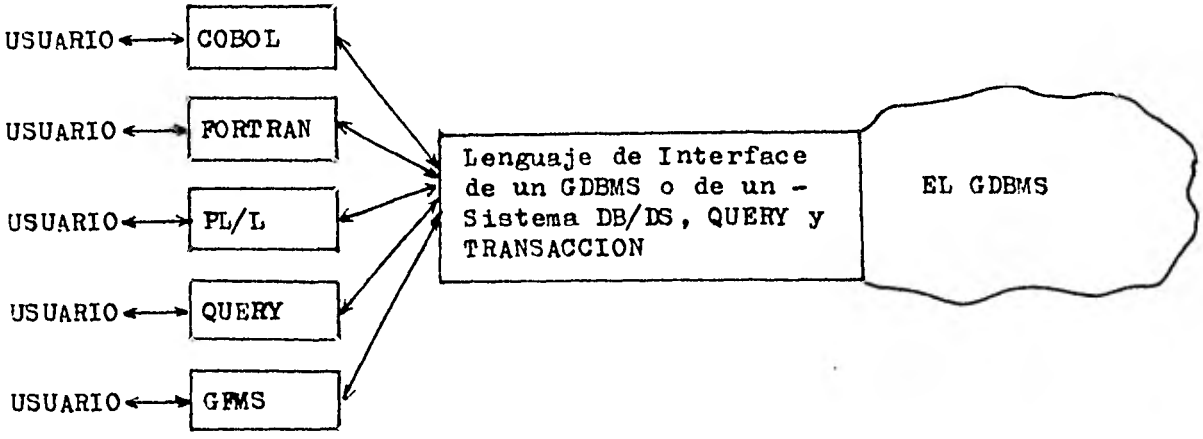


FIGURA B EL LENJUAJE DE INTERFACE DE UN GDBMS



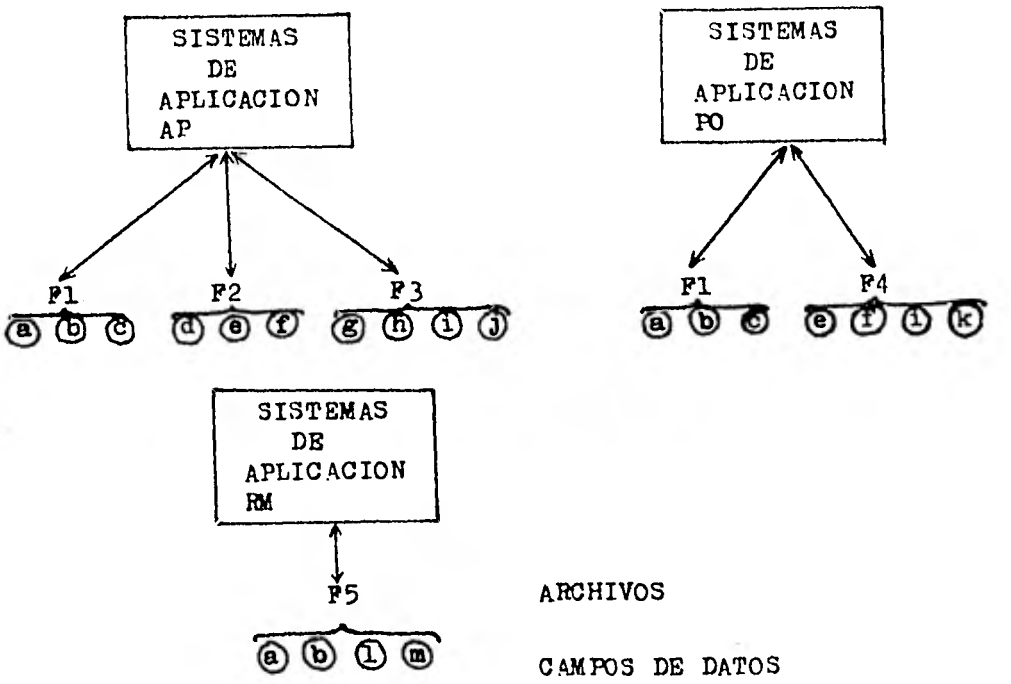
LA INTERFACE QUERY PARA:

Necesidades de información espontánea
Orientada a no programadores
Accesar bajos volúmenes de datos
Requerimientos no repetitivos

LA INTERFACE DE TRANSACCION PARA:

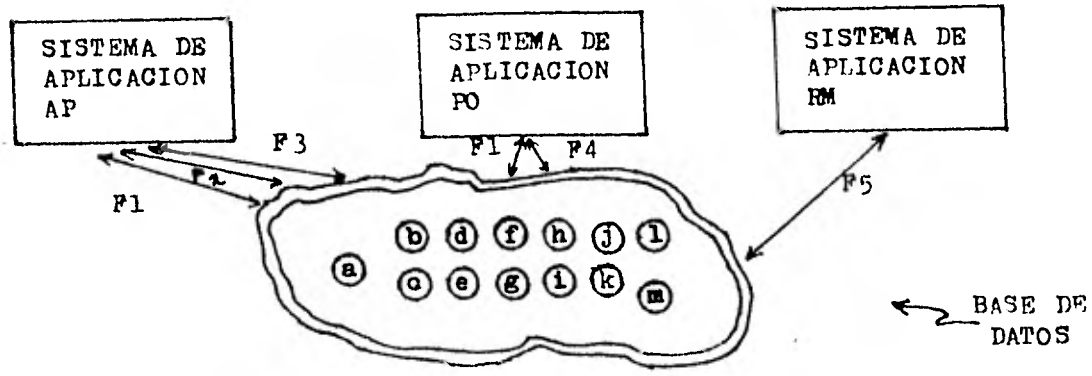
Necesidades de información predecible
Orientada a programadores
Accesar altos volúmenes de datos
Transacciones repetitivas

FIGURA B.- EL LENGUAJE DE INTERFACE DE UN GDBMS



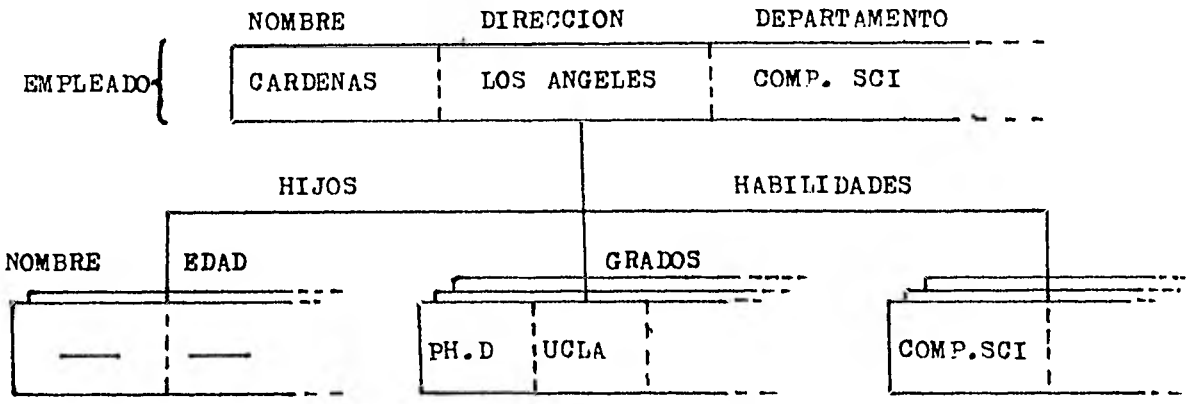
REDUNDANCIA: a,b,e,f,i DOS VECES

FIGURA C.-REDUNDANCIA DE DATOS ALMACENADOS



REDUNDANCIA: PEQUEÑA

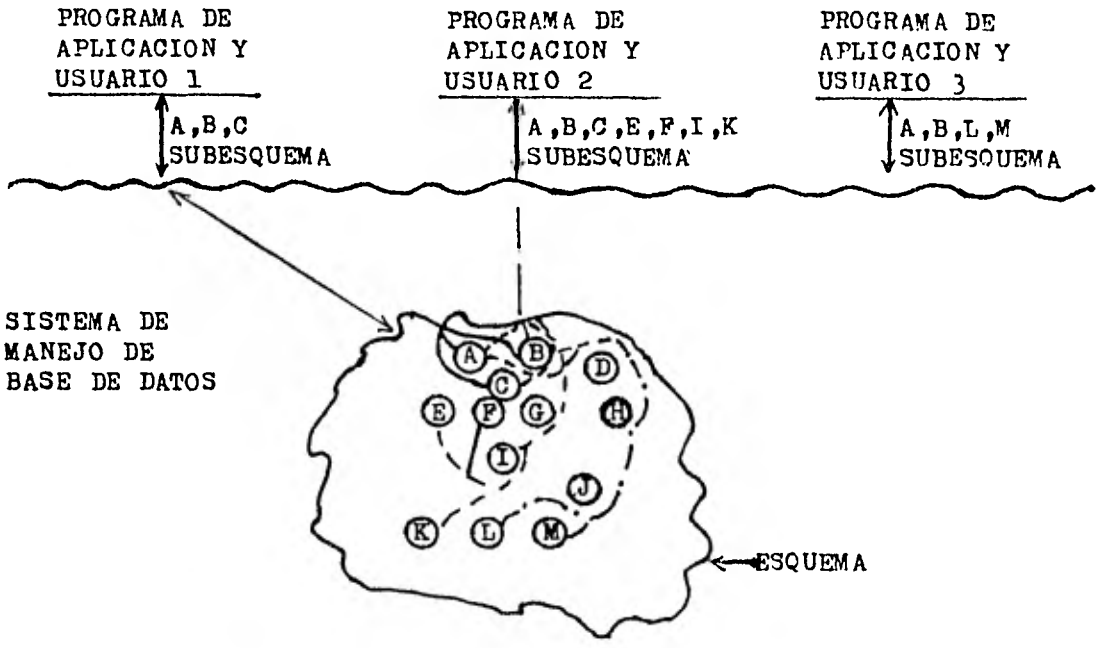
FIGURA D.- EL ACERCAMIENTO DE BASE DE DATOS PARA EVITAR MUCHA REDUNDANCIA DE DATOS ALMACENADOS.



REGISTROS: EMPLEADO (NOMBRE,DIRECCION,DEPARTAMENTO..)
HIJO (NOMBRE, EDAD...)
EDUCACION (GRADOS,UNIVERSIDAD, ...)
HABILIDAD (MAYOR,CERTIFICADO DE DATOS..)

RELACIONES: HIJOS
GRADOS
HABILIDADES

FIGURA E.- RELACIONES EN UNA BASE DE DATOS



CAMPOS DE DATOS ELEMENTALES: A, B, C ...

ESQUEMA: EL CONJUNTO DE TODO A,B,C...COMO CAMPO

SUBESQUEMA: UN SUBCONJUNTO DE A,B,C ...

BASE DE DATOS FISICA: EL CONJUNTO DE TODOS LOS CASOS DE A,B,C,... FISICAMENTE ALMACENADOS.

FIGURA F.- ESQUEMAS Y SUBESQUEMAS

DOCTOR

Nombre del Doctor	Nombre de la Clínica	Direc. Clínica	Especialidad del Doctor

PACIENTES-EMPLEADOS-DE-DOCTOR

EMPLEADO

Clave del Empleado	Nombre Empleado	Dirección Empleado	Departa-mento	Edad	Nombre de Esposa	Nombre Cía. de Seguros	% Adeudo al Doctor

HIJOS-DE-EMPLEADO

HIJO

Nombre	Edad	Año Escolar

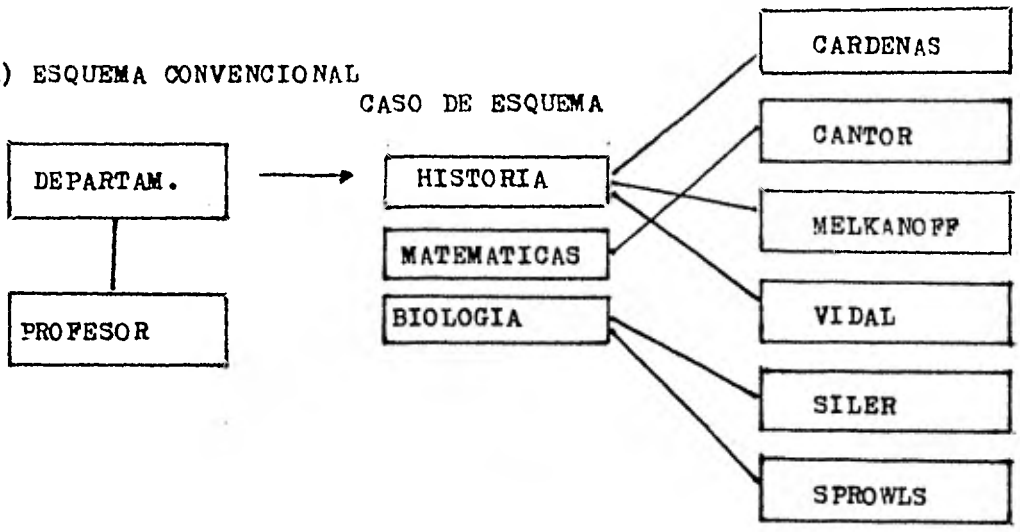
COMPAÑIA DE SEGUROS

Nombre	Dirección	Vigencia

EL ESQUEMA CONTIENE: CUATRO REGISTROS-DOCTOR, EMPLEADO, HIJO, CIA.SEGUROS
DOS RELACIONES EXPLICITAS-PACIENTES-EMPLEADOS-DE-DOCTOR Y HIJOS-DE-EMPLEADO
UNA RELACION IMPLICITA- ENTRE EMPLEADO Y CIA. DE SEGUROS.

FIGURA G.-EJEMPLO DE UN ESQUEMA

1) ESQUEMA CONVENCIONAL



2) ESQUEMA TABULAR, RELACIONAL O BANDERA

DEPARTAMENTO	PROFESOR

CASO DE ESQUEMA

HISTORIA	CARDENAS
HISTORIA	MELKANOFF
HISTORIA	VIDAL
MATEMATIC.	CANTOR
BIOLOGIA	SILER
BIOLOGIA	SPROWLS

FIGURA H.- FORMAS EQUIVALENTES DE VISUALIZAR RELACIONES ENTRE DOS REGISTROS.

EMPLEADO-SEGURO

No. de Empleado	Nombre de Empleado	Dirección de Emplead	Nombre Cía Seguros	Adeudo al Doctor	Vigencia

1) SUBESQUEMA COMO ES VISTO POR UN USUARIO O UN PROGRAMA DE APLICACION A.

CLINICA - EMPLEADO

Nombre de la Clinica	Direcc. de Clinica	No. de Empleado	Nombre de Empleado	Dirección de Emplead.	Nombre de Cía.Seguros

HIJOS - DE - EMPLEADO

HIJO

NOMBRE	AÑO ESCOLAR	EDAD

2) SUBESQUEMA COMO ES VISTO POR UN USUARIO O UN PROGRAMA DE APLICACION B.

FIGURA I.- DOS POSIBLES SUBESQUEMAS DEL ESQUEMA DE LA FIGURA G.

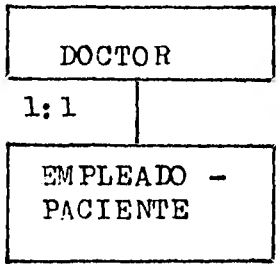
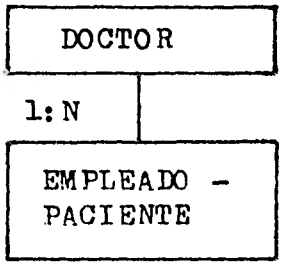
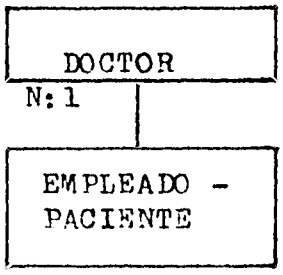
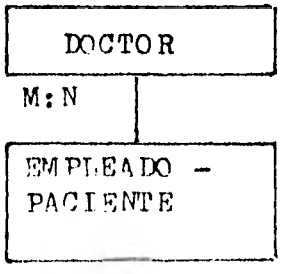
TIPO DE RELACIONES ENTRE DOC. Y EMPLEADO - PACIENTE	DIAGRAMAS	EXPLICACION
UNO-A-UNO		UN DOCTOR TIENE SOLAMENTE UN EMPLEADO-PACIENTE Y VICEVERSA.
UNO-A-MUCHOS		UN DOCTOR TIENE VARIOS EMPLEADOS-PACIENTES PERO LOS EMPLEADOS-PACIENTES TIENEN SOLO UN DOCTOR.
MUCHOS-A-UNO		VARIOS DOCTORES TIENEN EL MISMO EMPLEADO-PACIENTE Y VICEVERSA UN EMPLEADO-PACIENTE TIENE VARIOS DOCTORES.
MUCHOS-A-MUCHOS		UN DOCTOR TIENE VARIOS EMPLEADOS-PACIENTES Y UN EMPLEADO-PACIENTE TIENE VARIOS DOCTORES.

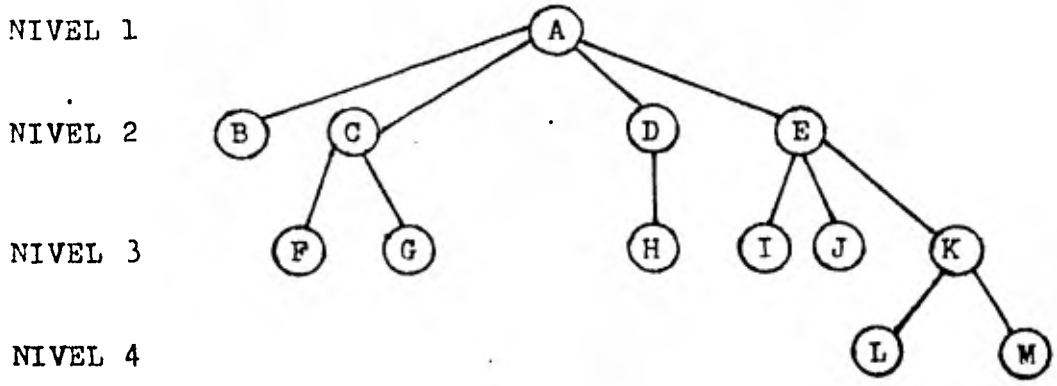
FIGURA J.- POSIBLES RELACIONES ENTRE DOS ENTIDADES.

	DATO INDIVIDUAL	DATOS AGREGADOS	REGISTRO	SUBESQUEMA	ESQUEMA O BASE DE DATOS
COBOL CONVENCIONAL	CAMPO DE DATO ELEMENTAL	GRUPO	REGISTRO	ARCHIVO	ARCHIVO
DBTG DE CODASYL	CAMPO DE DATO ELEMENTAL	GRUPO	REGISTRO	SUBESQUEMA	ESQUEMA
IMS DE IBM	CAMPO	SEGMENTO	SEGMENTO	SENGEMTOS SENSITIVOS: BLOCK - DE COMUNICACION DE PROGRAMA.	BASE DE DATOS (FISICA, LOGICA)
MARK IV DE INFORMATICS	CAMPO	SEGMENTO	SEGMENTO	ARCHIVO	ARCHIVO MAESTRO
IDS DE HONEYWELL	CAMPO DE DATO	GRUPO	REGISTRO	ARCHIVO	BASE DE DATOS
ADABAS DE AG	CAMPO	GRUPO DE CAMPOS	REGISTRO	ARCHIVO O ARCHIVO DE - PROGRAMA	ARCHIVOS ACOPLADOS O -- BASE DE- DATOS
EN USO POR VARIOS INDIVIDUALISTAS	ATRIBUTO	GRUPO DE ATRIBUTOS	ENTIDAD	CONJUNTO DE ENTIDADES.	TODOS LOS CONJUNTOS DE ENTIDADES.

Continúa Hoja 2...

	DATO INDIVIDUAL	DATOS AGREGADOS	REGISTRO	SUBESQUEMA	ESQUEMA O BASE DE DATOS
SYSTEM 2000 DE MRI	ELEMENTO	GRUPO REPETITIVO	REGISTRO	PROGRAMA DE APLICACION - DE LA BASE DE DATOS; SUBESQUEMA.	ESQUEMA O BASE DE - DATOS.
SISTEMAS RELACIONALES DE BASE DE - DATOS	DOMINIO	RELACION O TABLA	RELACION O TABLA	BASE DE DATOS DE USUARIO, SUBMODELO DE DATOS.	BASE DE DATOS, MODELO DE DATOS.

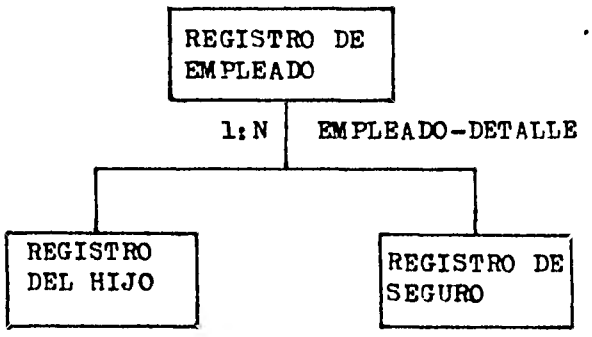
FIGURA K.- TERMINOS COMUNES Y EQUIVALENCIAS EN EL MUNDO DEL MANEJO DE DATOS.



ARBOL: 4 NIVELES
13 NODOS INCLUYENDO, 8 NODOS HOJAS (B,F,G,H,I,J,L,M) Y 5 DATOS AGREGADOS (A,C,D,E,K)

NOTA: EN LA DESCRIPCION CONVENCIONAL DE UN REGISTRO EN COBOL UN NODO HOJA ES UN CAMPO ELEMENTAL DE DATOS;OTROS NODOS SON SOLAMENTE DATOS AGREGADOS DE NODOS HOJAS.- EN LA TECNOLOGIA DE BASE DE DATOS CADA NODO REPRESENTA UN REGISTRO.

FIGURA L.- UN ARBOL(NINGUN ELEMENTO O NODO TIENE MAS DE UN "PADRE").



REGISTRO PROPIETARIO
EMPLEADO INVOCADO

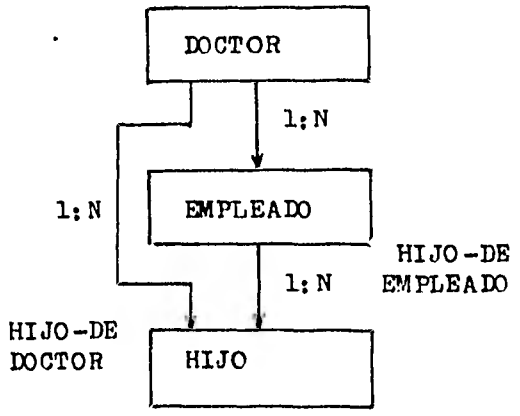
CONJUNTO INVOCADO
EMPLEADO-DETALLE
RELACION 1:N

DOS REGISTROS MIEMBROS: HIJO, SEGURO CUALQUIER NUMERO ARBITRARIO DE OCURRENCIAS DE CADA -- TIPO DE REGISTRO MIEMBRO PARA CADA OCURRENCIA DE TIPO DE REGISTRO PROPIETARIO

CONJUNTO DBTG DE CODASYL : UN ARBOL DE REGISTROS DE DOS NIVELES CON UN REGISTRO PROPIETARIO Y UNO O MAS REGISTROS MIEMBROS. LA RELACION M:N NO ESTA PERMITIDA DENTRO DE UNA OCURRENCIA DEL CONJUNTO.

FIGURA M- EJEMPLO DE UN CONJUNTO DBTG DE CODASYL.

1) UNA RED

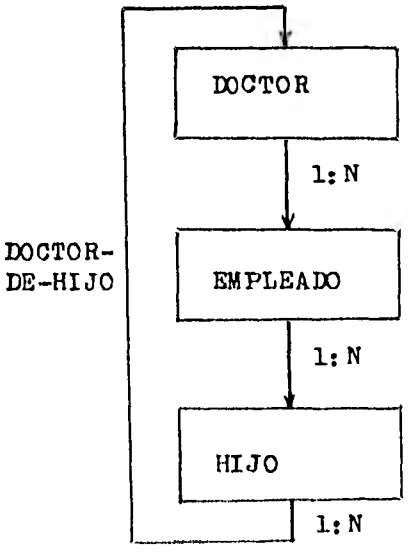


} RELACION: EMPLEADO-PACIENTE-DE DOCTOR.

} DOS RELACIONES: HIJO-DE-EMPLEADO
HIJO-DE-DOCTOR

} EL HIJO TIENE DOS PROPIETARIOS

2) UNA RED LA CUAL ES UN CICLO



NOTA: ALGUNOS SISTEMAS DE BASE DE DATOS PUEDEN REPRESENTAR CICLOS, OTROS NO TIENEN ESTA POSIBILIDAD.

FIGURA N.- EJEMPLOS DE REDES

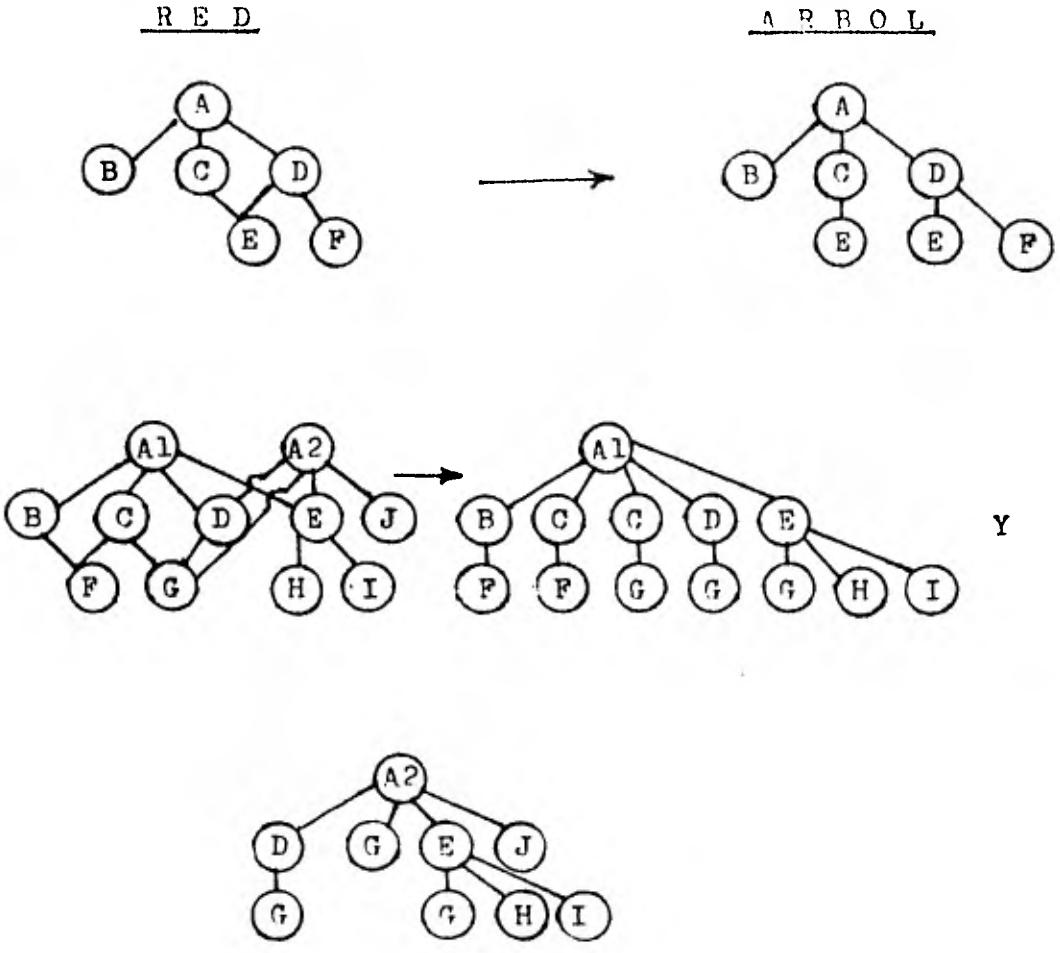
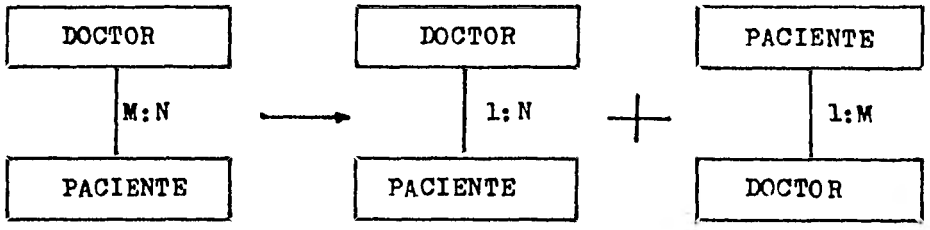


FIGURA 0.- TRANSFORMACION DE REDES EN ARBOLES MEDIANTE LA DUPLICIDAD DE NODOS. LA RELACION M:N NO ES PERMITIDA.

1) ESQUEMA



2) CASOS DE UN ESQUEMA

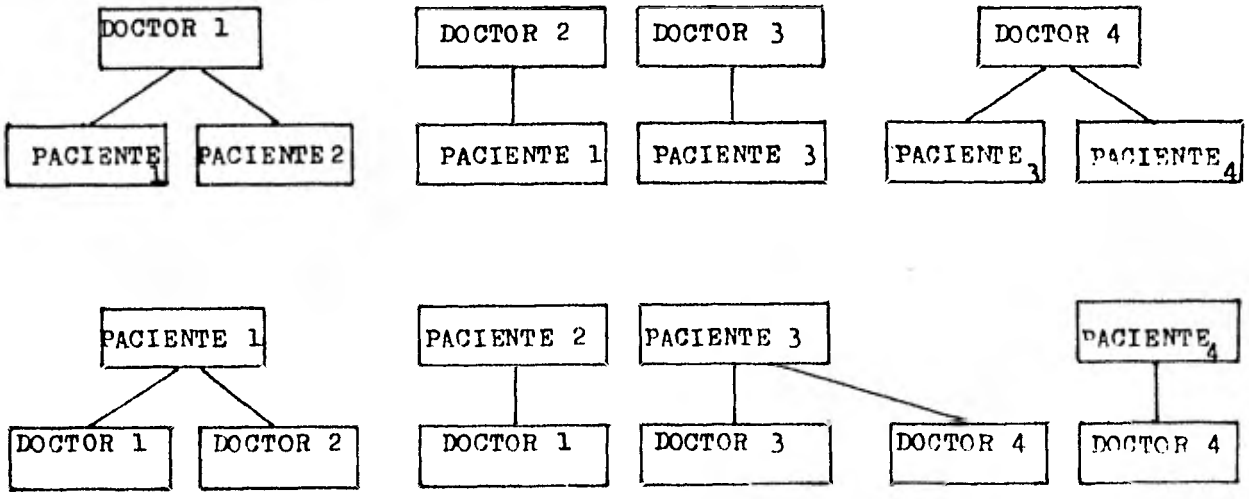


FIGURA P.- UNA RELACION M:N VISTA COMO DOS RELACIONES DE ARBOL DE DOS NIVELES.

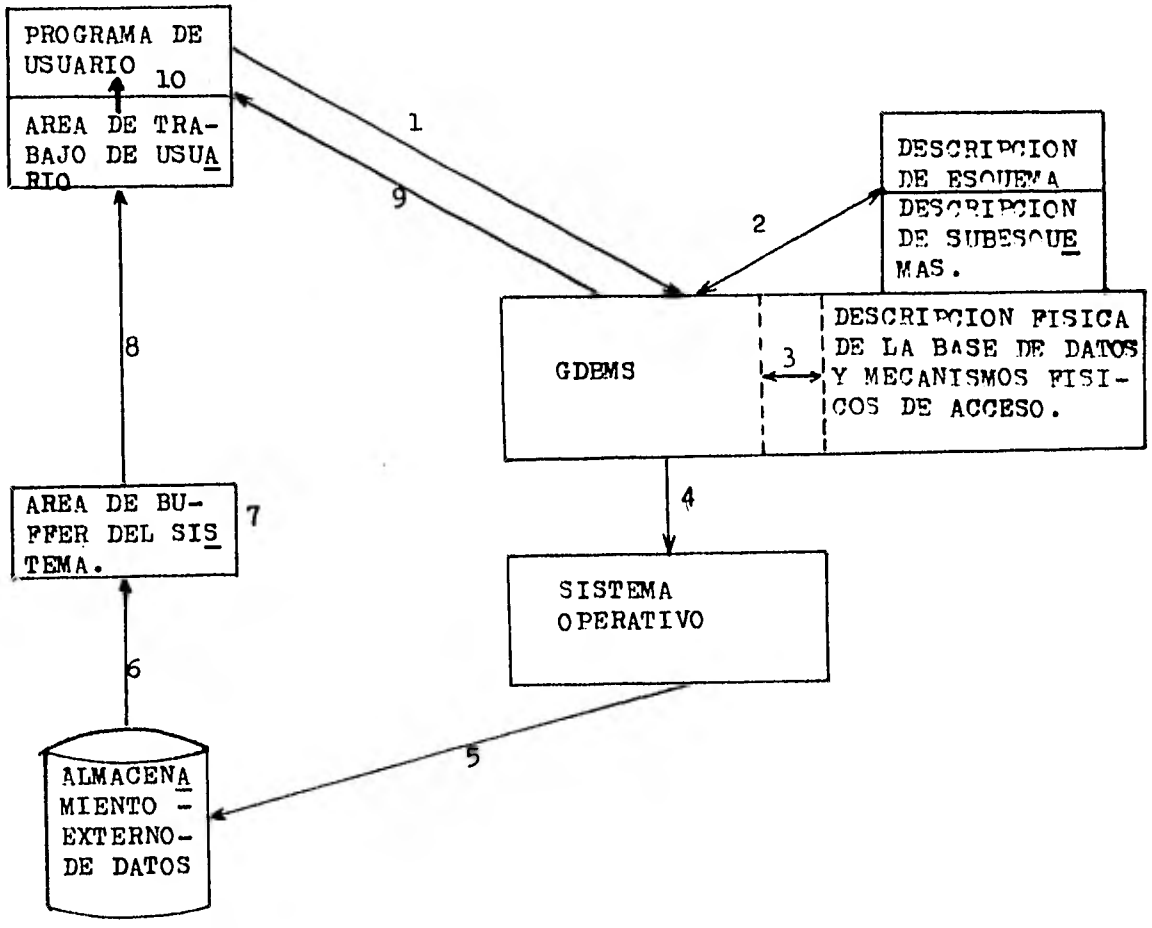
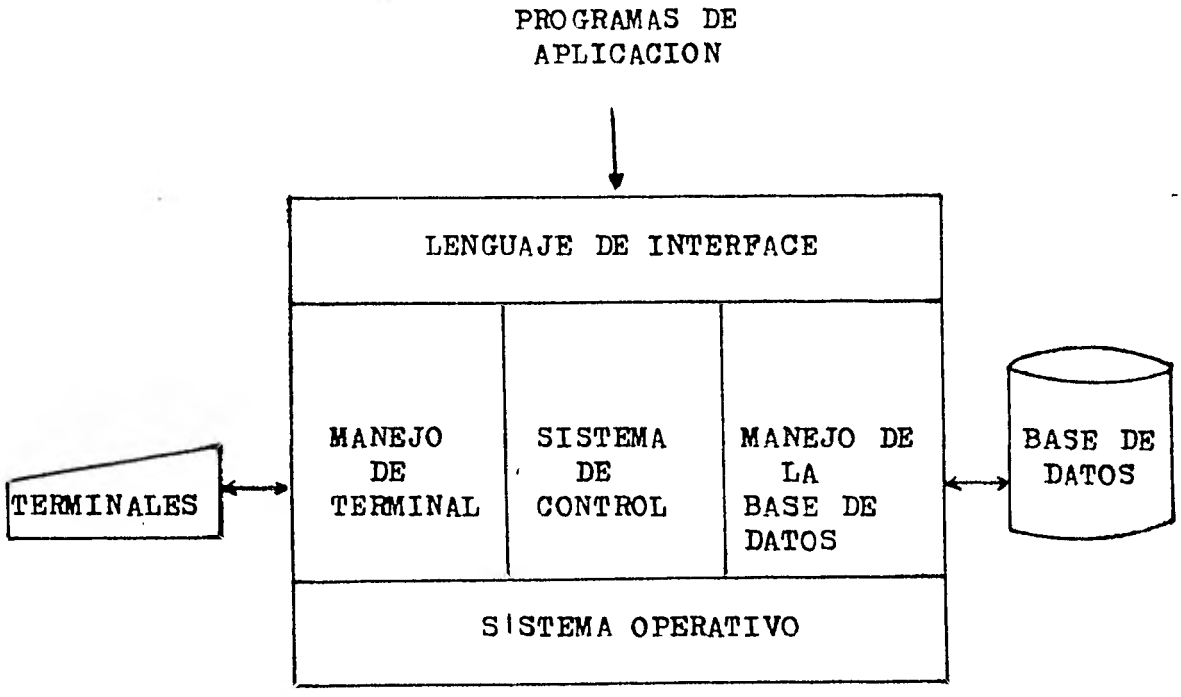


FIGURA Q.- SISTEMA CONCEPTUAL DE UNA BASE DE DATOS Y SECUENCIA DE EVENTOS PARA SATISFACER UN REQUERIMIENTO DE ENTRADA/SALIDA.



EL SISTEMA DB/DC INCLUYE: MANEJO DE LA BASE DE DATOS
MANEJO DE TERMINAL
SISTEMA DE CONTROL
LENGUAJE DE INTERFACE

FIGURA R.- COMPONENTES GLOBALES DE UN SISTEMA DB/DC.

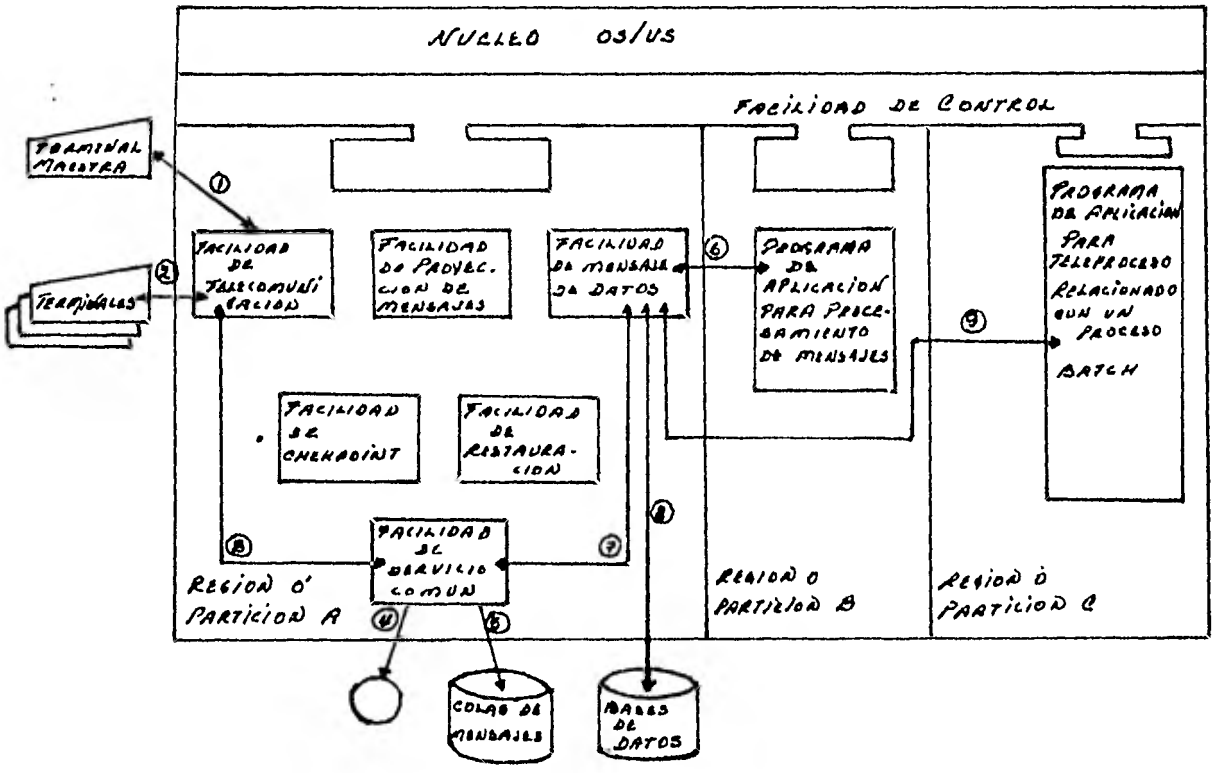


FIGURA 5.- FLUJO NORMAL DE ACCIONES INVOLUCRADAS EN EL SISTEMA IMS/VS EN TELEPROCESAMIENTO Y EL BATCH RELACIONADO.

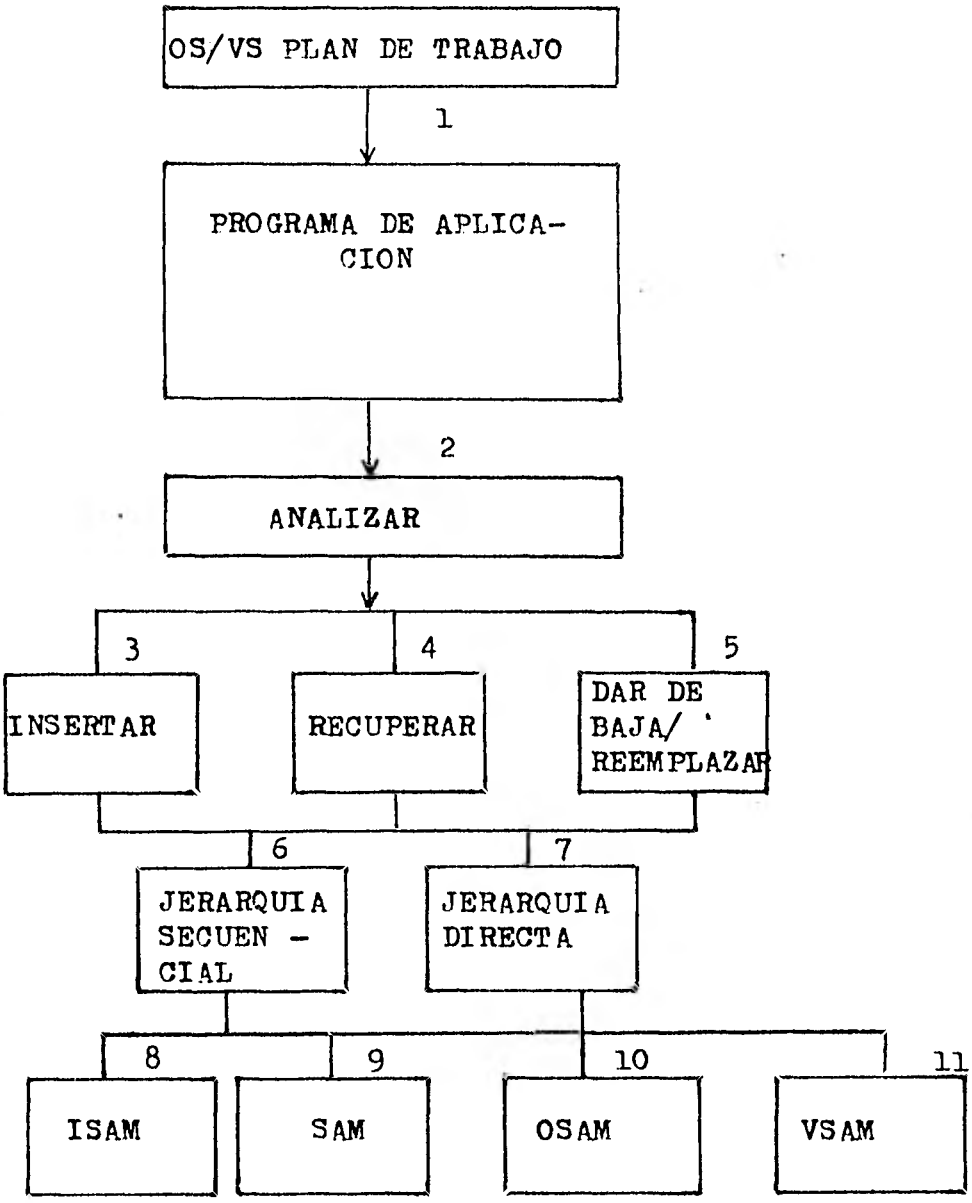


FIGURA T.- MEDIO AMBIENTE BATCH DE LENGUAJE DE DATOS.

GRADOS DE IMPLEMENTACION ORIENTACION Y DEPENDENCIA

No hay orientación a Dispositivos de-Computación.	Nivel 4	Los niveles más altos de organización de datos, visión lógica es decir vistas tabulares o relacionales, visión de un conjunto de entidades.
Interface física-lógica		
Alguna orientación a la implementación	Nivel 3	Organizaciones de familias - generales "lógica-física" para realzar la ejecución. Organizaciones de archivo se - riales, random y secundarias es decir archivos invertidos arboles encadenados y multi-listas.
Alta orientación a la implementación	Nivel 2	Organización general física-plan de manipulación y almacenamiento, orientado a la - implementación pero independiente de la máquina (varias alternativas, por ejemplo, - cada organización secundaria de archivos)
Interface de implementación independiente-dependiente		
Alta dependencia a dispositivos de -- computación	Nivel 1	Acceso básico de datos y -- metodos de manejo (secuen - cial, index secuencial, directo) y mapeo de almacenamiento utilizado en siste - mas operativos específicos- y dispositivos de almacenamiento.

FIGURA U.- ARQUITECTURA EN UN SISTEMA DE BASE DE DATOS.

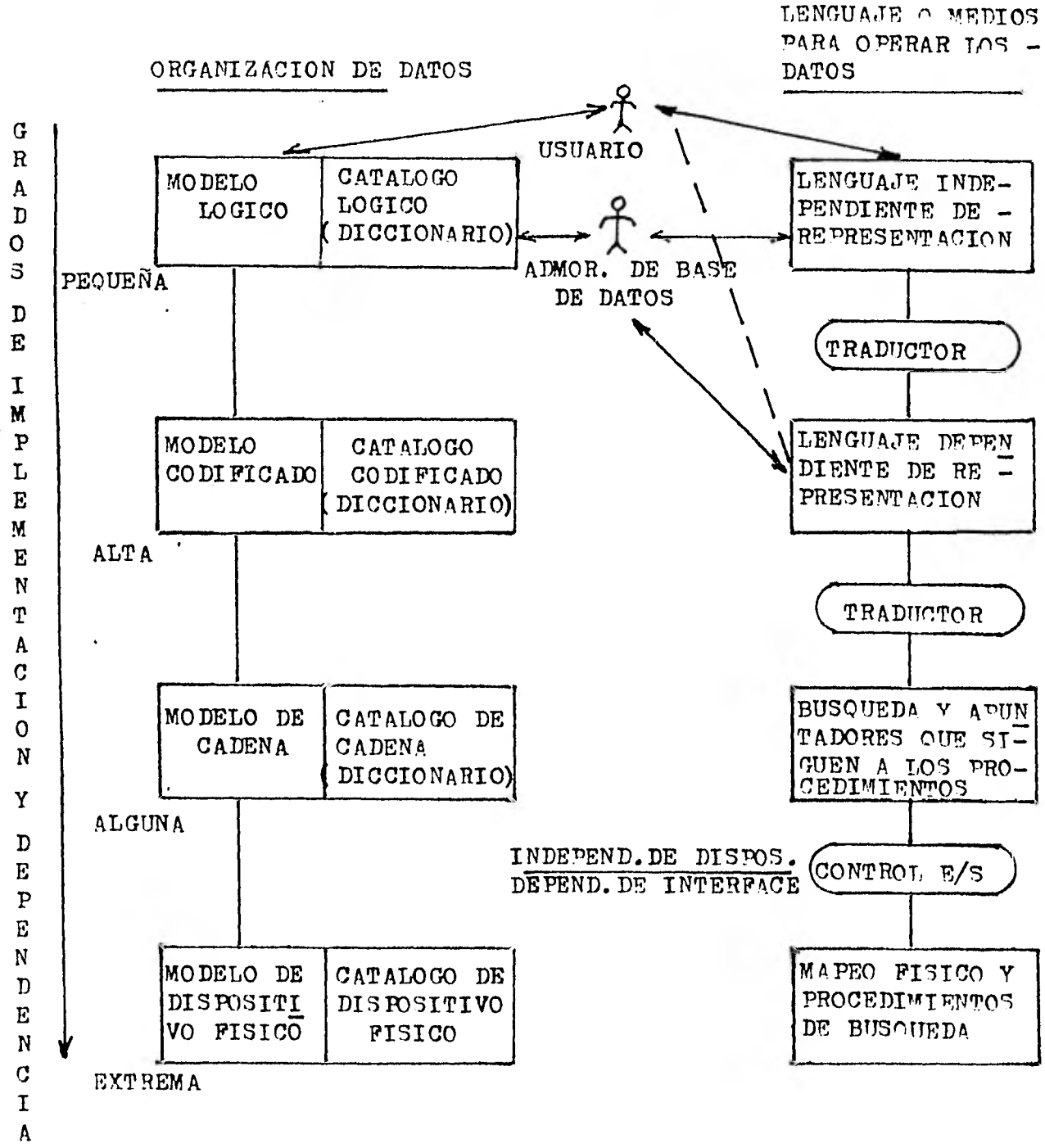


FIGURA V.- NIVELES DE ORGANIZACION DE DATOS EL DIAM (MODELO DE ACCESO DE DATOS INDEPENDIENTES).

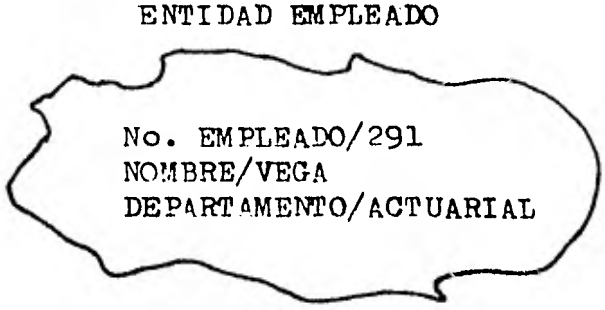


FIGURA W.- UNA ENTIDAD FORMADA DE UNA COLECCION DE PARES DE NOMBRES DE ATRIBUTOS/VALORES DE ATRIBUTOS.

C A P I T U L O III

SISTEMA DE APLICACION EN SYSTEM 2000

BREVE EXPOSICION SOBRE SYSTEM 2000

SYSTEM 2000

Es un GDBMS que opera con computadoras de las series 360 - y 370 de IBM, las series 1100 de UNIVAC y las series CDC 6000 y CYBER. El SYSTEM 2000 básico, con características opcionales - selectas, provee la base para desarrollar sistemas de informa - ción a la medida de los requerimientos de la aplicación y el -- usuario.

SYSTEM 2000 BASICO

Provee al usuario de un conjunto de habilidades fáciles de interpretar para manejar la Base de Datos. Este incluye la habilidad para definir nuevas Bases de Datos, modificar la definición de Bases de Datos existentes, y recuperar y actualizar valores en aquellas Bases de Datos.

Los componentes básicos de las definiciones de la Base de- Datos son: Los elementos de datos y los grupos repetitivos. Los valores son almacenados en los elementos de datos. Los grupos- repetitivos describen una estructura para almacenar conjuntos - múltiples de valores de datos y también sirven para encadenar - niveles jerárquicos de la definición. (Esto se muestra más adelante en nuestro esquema de la Base de Datos)

Los valores para cada elemento y entrada lógica (registro) pueden variar en longitud. El usuario puede especificar sin - restricción cuales elementos en la Base de Datos se invertirán- y con ello se convertirán en campos de llaves, y que relación - jerárquica tendrá un elemento con otros elementos en la Base de Datos. La seguridad en los datos es mantenida por el password- de control de la Base de Datos y por un password adicional de -

control para cada componente.

El SYSTEM 2000 básico provee la opción de copiar los ar -- chivos que componen la Base de Datos, así como guardar una audi toría de los cambios hechos a la Base de Datos. Es capaz de re construir una Base de Datos por medio de la aplicación de la -- auditoría, parcial o completa, a un archivo que sea copia de la Base de Datos.

LA FACILIDAD DEL LENGUAJE PROCEDURAL

Esta facilidad permite al usuario manipular datos en una - Base de Datos en SYSTEM 2000 por medio de un lenguaje de progra mación tales como COBOL, FORTRAN ó ENSAMBLADOR. Esta facilidad provee el mecanismo para direccionar cualquier parte de la Base de Datos de interés para el programa particular, y para actua - lizar la Base de Datos através del programa. Interelaciones - entre dos ó más Bases de Datos se pueden establecer mediante la definición de estructuras de datos de redes.

La selección en la Base de Datos es ejecutada mediante el- uso de un archivo de índices, facilitando la rápida separación- de los datos de interés. La recuperación de los datos puede -- ser clasificada en una o más llaves antes del regreso del pri - mer conjunto de datos al programa. Una colección completa de - estados de banderas está provista en los programas que permiten un fácil seguimiento de condiciones anormales de terminación ó- aborto.

LA FACILIDAD DE ACCESO INMEDIATO

Esta facilidad provee un lenguaje orientado al usuario con el cual una persona que carezca de conocimientos en programa - ción puede expresar su requerimiento para recuperar o actuali -

zar la Base de Datos. Este lenguaje es muy facil de aprender.- Incluye un conjunto completo de mensaje de diagnósticos facil - mente de interpretar y es muy adecuado para uso interactivo des de terminales remotas.

LA FACILIDAD DE REPORT WRITER

Esta facilidad permite al usuario preparar definiciones de reportes en los cuales se puede generar punto de partida en -- cualquier elemento de la Base de Datos o grupo repetitivo, es - pecificando encabezados y líneas al final de cada página física, página lógica y del reporte entero, control de impresión, in -- clusión de campos, contador de páginas, y la acumulación de sub totales o totales por medio de postulados condicionales. Al - rededor de 100 reportes pueden ser generados de una sola vista- de los archivos de índices de la Base de Datos.

LA FACILIDAD DE ARCHIVO SECUENCIAL

Para instalaciones UNIVAC, esta facilidad permite al usua- rio de SYSTEM 2000 crear y accesar Bases de Datos que residen - en cinta magnética. Ya sea la facilidad de Report Writer o -- bien el programa del lenguaje procedural pueden ser utilizados- para accesar una Base de Datos Secuencial. Una capacidad de en cadenamiento dentro de la facilidad del lenguaje procedural per mite al usuario establecer asociaciones lógicas entre Bases de- Datos separadas físicamente.

Una Base de Datos Secuencial puede así ser vista como una- extensión lógica de una Base de Datos de acceso directo.

LA FACILIDAD DEL MONITOR DE TELEPROCESO

Teleproceso 2000 (TP 2000) es un paquete generalizado, fa-

cil de usar en comunicaciones de datos, el cual puede utilizarse en conjunción con SYSTEM 2000 para proveer acceso a la Base de Datos por medio del lenguaje natural o bien por la facilidad del lenguaje procedural. Puede ser utilizada con archivos de datos OS/370 o puede utilizarse en combinación de estos últimos y Bases de Datos de SYSTEM 2000. Efectúa interface con la mayoría de las terminales de comunicación del mercado actual. SYSTEM 2000 ha sido puesto en interface con los monitores de teleproceso disponibles comercialmente tales como TCAM, ALPHA, HYPERFASTER y BEST.

LA FACILIDAD DE MULTI-USUARIOS

Esta facilidad permite la comunicación entre una o más regiones o particiones y una sola copia de SYSTEM 2000. Los comandos de ACCESO INMEDIATO del monitor de teleproceso y los requerimientos de SYSTEM 2000 de los programas en lenguaje procedural en otras particiones y regiones son procesados simultáneamente en una o más Bases de Datos. Estas son protegidas de actualizaciones simultáneas cuando se encuentran en este medio ambiente.

MODULOS COMPONENTES DE SYSTEM 2000

MODULO DE CONTROL

Este es el primero de los modulos funcionales de SYSTEM - 2000 y el usuario es asignado automaticamente a él, cuando inicia el acceso al sistema. El otro camino por medio del cual el usuario obtiene acceso a este módulo es mediante el comando: CONTROL.

Este módulo provee la habilidad para nombrar, así como -- accesar Bases de Datos, ejecuta análisis de seguridad, mantiene el control de los passwords, da de baja Bases de Datos conser - vadas en medios externos de almacenamiento, tales como cintas - magnéticas, y activa y manipula un archivo de actualización usa do para modificar registros a la Base de Datos.

MODULO DEFINE

Este módulo ha sido diseñado para asistir en la resolución de la primer interrogante que se plantea el usuario de SYSTEM - 2000, que sería ¿Como organizar ó estructurar los datos que deseo cargar en mi Base de Datos para accesarla más tarde en etapas de recuperación y actualización?.

Aprender a optimizar el poder del comando DEFINE dentro -- del medio ambiente particular en el cual es utilizado, tomará - algún tiempo de estudio, pero aprendiendo los mecanismos de lo- que usualmente ejecuta se convierte en una simple herramienta.- El primer y realmente único paso requerido es definir la Base - de Datos. ¿ Que es una Base de Datos ?. Una definición que pu diera ayudar es la siguiente: Una Base de Datos es una colec - ción organizada de datos acerca de "algo". Por ejemplo, un inventario de materiales es una Base de Datos, o los blocks de -

cuentas de una compañía puede ser una Base de Datos. En SYSTEM 2000, el usuario define tanto la naturaleza como los límites de su Base de Datos también como desee organizarla. En virtud de- que diferentes tipos de datos sugieren diferentes organizacio - nes, SYSTEM 2000 provee al usuario de flexibilidad para organi- zar o estructurar su Base de Datos en la manera más apropiada.

MODULO DE ACCESO

El módulo de acceso de SYSTEM 2000 está provisto para re - cuperar, actualizar y ejecutar operaciones de carga de datos en un medio ambiente de procesos ya sea interactivos ó Batch.

Este módulo se accesa tecleando el comando ACCESS o indi - rectamente si dentro del módulo de CONTROL incluimos el comando DATA BASE NAME IS # # .

De este punto el usuario puede emplear una o varias opera- ciones generales interactivas tales como DESCRIBE, LOAD etc. ó- puede utilizar el proceso QUEUE que es un proceso Batch, para - recuperar o actualizar un gran número de datos, esta operación- termina mediante el comando TERMINATE.

PROCESO DE CARGA . La carga de datos en la Base de Datos puede- ser ejecutada por medio de dos diferentes procesos; SYSTEM 2000 Básico ó Lenguaje Procedural (PL). Las dos opciones estan dis ponibles para el usuario, el proceso a utilizar en cualquiera - de ellas es una función de las siguientes características:

1.- Formato de Datos.

Los datos deben estar en un formato específico antes - que el SYSTEM 2000 Básico pueda ser utilizado. Si los datos en un formato tal que la máquina puede leerlos,- se encuentran disponibles entonces:

a) Un programa de conversión tendrá que ser escrito - para cambiar estos datos a los formatos llamados "valores- cadenas" para SYSTEM 2000 Básico o bien,

b) PL puede ser usado directamente.
Si los datos no estan en formato tal que la máquina - pueda leerlos, entonces deben hacerse primero dispo - nibles a esta lectura y de ahí aplicar las reglas sub secuentes.

2.- Conocimiento de Programación.
PL demanda un grado de conocimiento de programación.- SYSTEM 2000 no lo requieren.

3.- Cantidad de Datos.
Una pequeña cantidad de datos requeriría SYSTEM 2000- por un tiempo de implementación más corto.
Grandes cantidades de datos se requieren para un programa PL, porque el tiempo empleado para preparar datos de entrada se incrementa relativamente.

Una vez que la Base de Datos ha sido definida, los datos - pueden ser cargados inmediatamente en ella. En virtud de que - el módulo de CONTROL inicializa, abre y hace residentes todas - las tablas de la Base de Datos cuando una nueva Base de Datos - es definida, el proceso de carga puede estar complementando el- mismo trabajo después de definir la nueva Base de Datos, o esta puede estar accesada en un trabajo separado para cargar los valores de los datos.

El proceso de carga utiliza el proceso QUEUE en el módulo- de ACCESO para introducir nuevas entradas lógicas a la Base de- Datos.

GENERALIDADES DEL SISTEMA

OBJETIVO DEL SISTEMA

Cubrir las necesidades inmediatas de información de compra de material de una empresa, mediante la creación de una Base de Datos que:

- Contenga información necesaria de requisiciones, pedidos y proveedores.
- La información contenida en la Base de Datos esté al día y sirva como base para la toma inmediata de decisiones - sobre compra de material (activar colocación de pedidos, activar entrega de material, etc.)
- El contenido de la Base de Datos sea consultado en linea por todas las unidades foráneas de la empresa.

DEFINICION DE LA BASE DE DATOS

- Contenido de la Base de Datos

  La información contenida en cada grupo de datos es:

  DATOS DE REQUISICIONES

  Clave de requisición
  Año presupuestal de la requisición
  Fecha de entrega de la requisición al Depto. de Proveduría y Almacenes.
  Fecha de cierre de concurso.
  Fecha en que entro a autorización el pedido

  DATOS DEL PEDIDO

  Clave de pedido
  Total de lotes del pedido

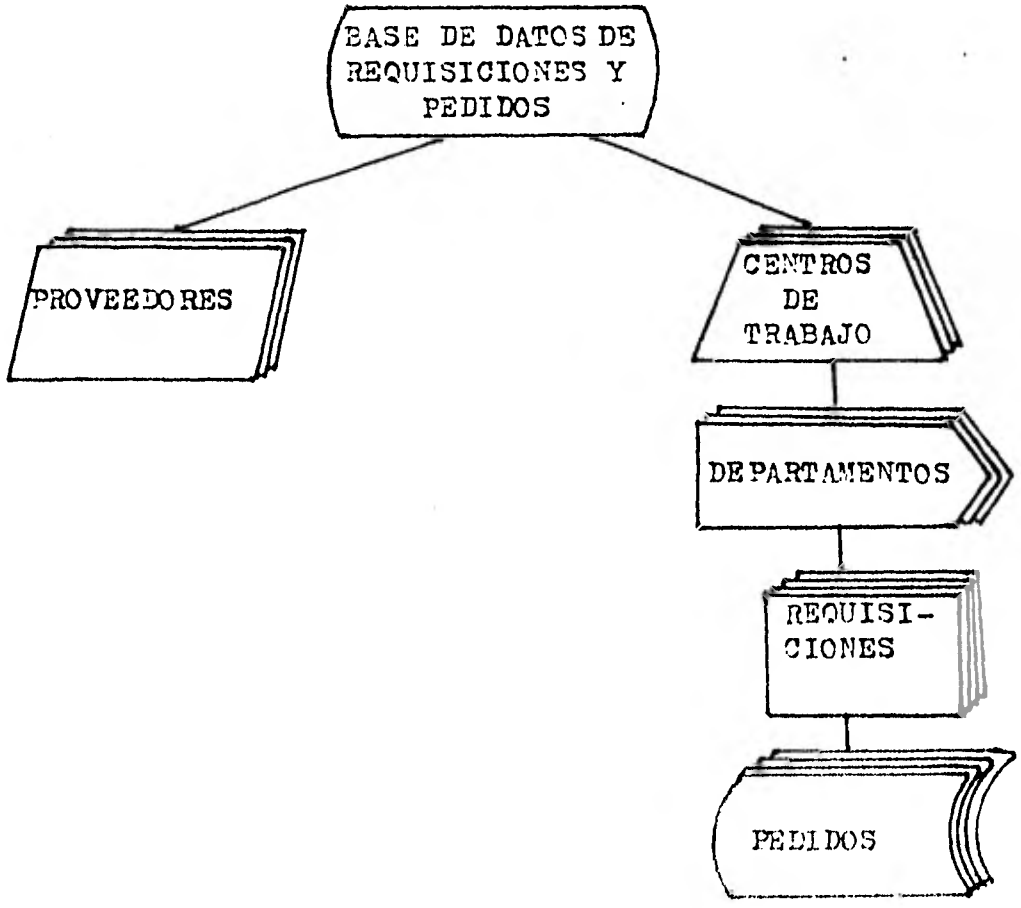
Importe del pedido
Clave del proveedor
Partida presupuestal
Fecha de entrega del pedido al proveedor
Fecha de promesa de entrega del material
Fecha de inicio de recepción del material en el almacén
Fecha de recepción completa del material en el almacén

DATOS DE PROVEEDOR

Clave de proveedor
Razón Social
Telefonos
Direcciones

La Asociación de estos datos se muestra en el esquema siguiente:

ESQUEMA DE LA BASE DE DATOS



DESCRIPCION DE LA BASE DE DATOS

```
describe;
SYSTEM RELEASE NUMBER        2.60F
DATA BASE NAME IS DAXXRP
DEFINITION NUMBER       9
DATA BASE CYCLE     18061
   1*  BANCO-REQUISICIONES-PEDIDOS (NAME X(10))
   2*  FECHA-GENERACION (NON-KEY DATE)
   3*  PROVEEDORES-REGISTRADOS (INTEGER NUMBER 9(10))
   4*  CENTROS-REGISTRADOS (INTEGER NUMBER 9(10))
   5*  DEPTOS-REGISTRADOS (INTEGER NUMBER 9(10))
   6*  REQUISICIONES-REGISTRADAS (INTEGER NUMBER 9(13))
   7*  PEDIDOS-REGISTRADOS (INTEGER NUMBER 9(15))
  10*  CENTRO (RG)
    11*  CLAVE-CENTRO (NAME XXX IN 10)
    12*  NOMBRE-CENTRO (NON-KEY NAME X(30) IN 10)
   100*  DEPARTAMENTO (RG IN 10)
     102*  CLAVE-DEPARTAMENTO (NAME XXX IN 100)
     200*  REQUISICION (RG IN 100)
       201*  CLAVE-REQUISICION (NAME X(12) IN 200)
       202*  CONSECUTIVO-REQ (INTEGER NUMBER 9999 IN 200)
       203*  ANO-PRESUPUESTAL (INTEGER NUMBER 99 IN 200)
       204*  FECHA-ENTREGA-OFA (NON-KEY DATE IN 200)
       205*  FECHA-CIERRE-CONCURSO (DATE IN 200)
       206*  FECHA-AUTORIZACION (DATE IN 200)
       300*  PEDIDO (RG IN 200)
         301*  ESTADO (NON-KEY NAME X IN 300)
         302*  TOTAL-LOTES (NON-KEY INTEGER NUMBER 9999 IN 300)
         303*  PARTIDA-PRESUPUESTAL (NON-KEY NAME X(10) IN 300)
         306*  IMPORTE (NON-KEY MONEY $9(9).99 IN 300)
         307*  PROVEEDOR (NAME X(7) IN 300)
         309*  CLAVE-PEDIDO (NAME X(13) IN 300)
         310*  FECHA-ENTREGA-PROVE (NON-KEY DATE IN 300)
         311*  FECHA-PROMESA-ENTREGA (DATE IN 300)
         312*  FECHA-INICIO-RECEPCION (DATE IN 300)
         313*  FECHA-RECEP-COMPLETA (DATE IN 300)
  50*  PROVEEDORES (RG)
    51*  CLAVE-PROVEEDOR (NAME X(7) IN 50)
    52*  RAZON-SOCIAL (NON-KEY NAME X(40) IN 50)
    53*  TELEFONO-1 (NON-KEY NAME X(7) IN 50)
    54*  TELEFONO-2 (NON-KEY NAME X(7) IN 50)
    55*  DIRECCION-P1 (NON-KEY NAME X(40) IN 50)
    56*  DIRECCION-P2 (NON-KEY NAME X(40) IN 50)
```

CONFIABILIDAD DE LA BASE DE DATOS

La confiabilidad de la Base dependerá del usuario, quien - deberá seguir al pie de la letra las instrucciones proporcionadas para actualizar la información de la Base de Datos.

SEGURIDAD DE LA BASE DE DATOS

La persona que autorice el usuario principal para consul - tar o actualizar la Base de Datos, deberán tener conocimientos elementales del manejo de terminales, la misma persona deberá - asignar diferentes tipos de autoridad para limitar tanto el tipo de operación como los datos sobre los que se pueden realizar ya que un mal manejo de la Base puede dañarla o destruirla.

TIPOS DE AUTORIDAD

"PASSWORD MAESTRO"

La persona que tenga este tipo de "password" tendrá acceso ilimitado a la Base de Datos y podrá realizar cualquier opera - ción de actualización y/o consulta.

"PASSWORD DE USUARIOS"

Las personas que tengan este tipo de "Password" tendrán - acceso limitado a la Base de Datos y solo podrán realizar alguna operación de actualización o de consulta.

USUARIOS DE LA BASE DE DATOS

Tipo de Password: Maestro
Operaciones: Cualquier operación de actualización y consulta
Responsable: Usuario principal en México

Tipo de Password: Usuarios
Operaciones: Cualquier operación de consulta y actualización de los datos:

Fecha inicio recepción de materiales.

Fecha recepción completa de materiales.

Responsables: Usuarios foráneos de las siguientes ciudades:

Reynosa

Tampico

Poza Rica

Cuenca del Papaloapan

Coatzacoalcos

Catalina

Venta de Carpio

Monterrey

INTEGRACION DE LA BASE DE DATOS

PROCESOS:

ALTA REQUISICION.

Cuando se tenga registrada la fecha de entrega de requisición al Departamento de Proveduría y Almacenes, podrá realizarse este proceso siguiendo los pasos correspondientes.

CORRECCION REQUISICION.

Cuando se tenga que modificar algun dato de la requisición, deberá ejecutarse este proceso observando las reglas establecidas.

REGISTRO DE FECHAS DE MOVIMIENTO DE REQUISICION EN D.P.A.

Para el seguimiento de los trámites de la requisición sedeberá ejecutar este proceso, una vez que se conozca la fechade cierre de concurso, deberá ejecutarse el paso correspondiente indicado en este proceso, así como cuando se conozca la fecha en que pasó a autorización el pedido.

ALTA PEDIDO

Una vez que el pedido haya sido autorizado y se conozca la fecha de entrega del pedido al proveedor, podrá ejecutarseeste proceso siguiendo los pasos indicados.

CORRECCION PEDIDO

Cuando se tenga que modificar algún dato de pedidos, podrá ejecutarse este proceso, siguiendo los pasos indicados.

BAJA PEDIDO

Cuando se tenga que dar de baja un pedido, deberá ejecutarse este proceso, siguiendo los pasos indicados.

REGISTRO DE FECHAS DE MOVIMIENTO DE PEDIDOS EN EL ALMACEN

Para el seguimiento del pedido, se deberá ejecutar este -- proceso una vez que se conozca la fecha en que se inició la recepción del material en el almacén, así como cuando se terminó- de entregar el material en el almacén.

ALTA PROVEEDOR

Una vez que se conozca el proveedor al que se le colocó el pedido, podrá ejecutarse este proceso, siguiendo los pasos indicados.

CORRECCION PROVEEDOR

Cuando se desee modificar algun dato de proveedor, podrá - ejecutarse este proceso, siguiendo los pasos indicados.

BAJA PROVEEDOR

Cuando se desee dar de baja un proveedor, deberá ejecutarse este proceso, siguiendo los pasos indicados.

ALTA CENTRO

Cuando no exista una clave de centro en la Base de Datos - deberá ejecutarse este proceso, siguiendo los pasos indicados.

CORRECCION CENTRO

Cuando se quiera corregir el nombre del centro, deberá -- ejecutarse este proceso, siguiendo los pasos indicados.

ALTA DEPARTAMENTO

Cuando no exista una clave de departamento en la Base de- Datos, deberá ejecutarse este proceso, siguiendo los pasos indicados.

INSTRUCTIVO DE OPERACION DE LOS PROCESOS

PROCESO ALTA REQUISICION

PASO 1 Verificar que la clave de requisición no exista.

PROCEDIMIENTO

Teclear *BUSCA-REQUISICION (clave requisición, clave-departamento, clave centro)

RESPUESTA

1.1

0 SELECTED DATA SETS. No existe la requisición, ir al Paso 2

1.2

CLAVE-REQUISICION* Clave requisición

FECHA-ENTREGA-DPA* Fecha entrega a DPA

Indica que la clave de requisición ya existe en la Base de Datos, el usuario debe revisar si la clave deseada fue tecleada correctamente, en este caso - procederá a realizar el análisis para determinar - la causa del error. Si la clave de requisición se tecleó equivocada, regresar al Paso 1.

PASO 2 Verificar que la clave de Departamento exista,

PROCEDIMIENTO

Teclear *BUSCA-DEPARTAMENTO (clave departamento,clave centro)

PROCEDIMIENTO

2.1

0 SELECTED DATA SETS No existe el Depto. para ese centro. Ir al Paso 3.

2.2

CLAVE-DEPARTAMENTO* Clave departamento

CLAVE-CENTRO * Clave centro

Indica que la clave de departamento ya existe en la Base de Datos. Ir al Paso 8.

PASO 3 Verificar que exista el centro

PROCEDIMIENTO

Teclear +BUSCA-CENTRO (clave centro)

RESPUESTA

3.1

0 SELECTED DATA SETS. No existe el centro de la Base de Datos. Ir al Paso 4

3.2

CLAVE-CENTRO clave centro

Indica que la clave de cento ya existe en la Base de Datos. Ir al Paso 6.

PASO 4 Proporcionar la información correspondiente al centro

PROCEDIMIENTO

Teclear * DATOS-ALTA-CENTRO (clave centro, nombre -- centro)

RESPUESTA

4.1 -1 SELECTED DATA SETS -

La adición de datos se realiza con éxito.

Ir al Paso 5.

4.2 Cualquier otro mensaje regresar al Paso 4.

PASO 5 Verificar los valores que se adicionaron al centro

PROCEDIMIENTO

Teclear * LEE-CENTRO (clave centro)

RESPUESTA

CLAVE-CENTRO* Clave centro

NOMBRE-CENTRO* Nombre centro

5.1

Si la información está incorrecta, realizar el proceso de CORRECCION-CENTRO (ver indice) e ir al -- Paso 6

5.2

Si la información está correcta el proceso termina con éxito. Ir al Paso 6.

PASO 6 Proporcionar la información correspondiente al depar - tamento.

PROCEDIMIENTO

Teclear * DATOS-ALTA-DEPARTAMENTO (clave departa -- mento, clave centro)

RESPUESTA

6.1 - 1 SELECTED DATA SETS -

La adición del departamento se realizó con exito. Ir al Paso 7

6.2 - Cualquier otro mensaje regresar al Paso 6

PASO 7 Verificar los valores que se adicionaron al Departa -- mento.

PROCEDIMIENTO

Teclear * LEE DEPARTAMENTO (clave departamento, clave centro)

RESPUESTA

CLAVE-DEPARTAMENTO* Clave departamento

7.1 -O SELECTED DATA SETS -
Verificar las claves dadas en LEE-DEPARTAMENTO, si estan incorrectas ir al Paso 7. Si están correctas, regresar al Paso 6.

7.2 Si la información está correcta, el proceso se terminó con éxito. Ir al Paso 8

PASO 8 Proporcionar la información correspondiente a la requisición.

PROCEDIMIENTO

Teclear * DATOS-ALTA-REQUISICION (Clave requisición consecutivo req., año presupuestal, fecha entrega - DPA., clave departamento, clave centro)

RESPUESTA

CLAVE REQUISICION *	Clave requisición.
CONSECUTIVO REQ *	Consecutivo requisición
AÑO PRESUPUESTAL *	Año presupuestal
FECHA-ENTREGA-DPA *	Fecha de entrega a DPA

9.1 Si la información está incorrecta, ir al proceso de CORRECCION-REQUISICION (ver índice)

9.2 Si la información está correcta, el proceso de - termino con éxito.

PROCESO REGISTRO DE FECHAS DE MOVIMIENTOS REQUISICIONES EN D.P.A.

Las fechas que se llevarán como control del estado de una requisición, serán (FECHA-CIERRE-CONCURSO) que nos indica cuando termina un concurso determinado y (FECHA-AUTORIZACION) que nos indica cuando sus pedidos pasaron a autorización.

Para registrar la primera fecha, ejecutar proceso 1.1, si se desea registrar la fecha de autorización, ejecutar proceso 1.2.

PROCESO 1.1.

PROCEDIMIENTO:

Teclear *FIN-CONCURSO (clave requisición, clave Depto. clave centro, fecha de cierre de concurso)

RESPUESTA

1.1.1. - 1 SELECTED DATA SET -
El proceso se efectuó con éxito, ir Proceso 13

1.1.2 - 0 SELECTED DATA SET -
Revisar si la clave de requisición está mal tecleada, de ser así, regresar al proceso 1.1.
Si la clave está correcta, avisar al Depto. que la requisición no está dada de alta.

1.1.3. Cualquier otro mensaje regresar al proceso 1.1.

PROCESO 1.2 Registro de la fecha en que los pedidos pasaron a autorización.

PROCEDIMIENTO:

Teclear * ENTRADA-AUTORIZACION (clave requisición, - clave departamento, - clave cento, fecha de- ingreso a autorización)

RESPUESTA

1.2.1 - 1 SELECTED DATA SETS -
El registro se efectuó con éxito, ir al Proceso 1.3

1.2.2. - 0 SELECTED DATA SETS -
Revisar si la clave de requisición fué mal te - cleada, de ser así, regresar al proceso 1.2 en- caso contrario, avisar al Depto, que la requisi ción no esta dada de alta.

1.2.3. Cualquier otro mensaje, regresar al proceso de- registro de fechas de movimientos de reqs, en - D..P.A.

PROCESO 1.3

Verificar que la(s) fecha(s) alimentadas quedaron regis- tradas correctamente.

PROCEDIMIENTO

Teclear *LEE-REQUISICION (clave requisición, clave -- departamento, clave centro)

RESPUESTA:

CLAVE REQUISICION *	Clave de requisición
CONSECUTIVO-REQ *	Consecutivo de la req.
AÑO-PRESUPUESTAL *	Año presupuestal

FECHA-ENTREGA-DPA *	Fecha entrega a D.P.A.
FECHA CIERRE-CONCURSO *	Fecha de cierre concurso
FECHA-ENTREGA-AUTORIZACION *	Fecha de entrada a autorización.

1.3.1. Si las fechas están correctas, el proceso se terminó con éxito.

1.3.2. Si las fechas están incorrectas, regresar al procedimiento inicial del proceso 1.1.

PROCESO BAJA-PEDIDO

PASO 1 - Verificar que el pedido que se desea eliminar, exista.

PROCEDIMIENTO:

Teclear * LEE-PEDIDO (clave del pedido, clave requisición)

RESPUESTA

1.1. Si la respuesta es:

ESTADO *
TOTAL-LOTES *
PARTIDA-PRESUPUESTAL *
IMPORTE *
PROVEEDOR *
CLAVE-PEDIDO *
FECHA-ENTREGA-PROVE *
FECHA-PROMESA-ENTREGA *
FECHA-INICIO-RECEPCION *
FECHA-RECEP-COMPLETA *

Revisar que se trate del pedido que se desea eliminar, - de ser así, ir al Paso 2. De otro modo, verificar claves y regresar al Paso 1.

PASO 2. Dar de Baja el pedido

Teclear *ELIMINA-PEDIDO (clave del pedido, clave de requisición, clave departamento , clave centro).

RESPUESTA

2.1 - 1 SELECTED DATA SET -

La operación se efectuó con éxito y se dió de baja el pedido. Ir al Paso 3.

2.2 Cualquier otro mensaje regresar al Paso 2.

PASO 3 Verificar que el pedido eliminado sea el indicado.

Teclear * LEE-PEDIDO (clave pedido, clave requisición.)

RESPUESTA

3.1 - 0 SELECTED DATA SET -
La eliminación fué correcta, se termina el proceso satisfactoriamente.

3.2 CLAVE-REQUISICION
ESTADO *
TOTAL-LOTES *
PARTIDA PRESUPUESTAL *
IMPORTE *
PROVEEDOR *
CLAVE-PEDIDO *
FECHA-ENTREGA-PROVE *
FECHA-PROMESA-ENTREGA *
FECHA-INICIO-RECEPCION *
FECHA-RECEP-COMPLETA *

Esta respuesta implica:

1.- Que alguna clave o ambas inclusive, fueron tecleadas equivocadamente, en tal caso, regresar al Paso 3 y teclear correctamente, o bien

2.- Que el pedido eliminado no sea el indicado; en tal caso, regresar al Paso 1 y efectuar, si es posible, -

El PROCESO ALTA-PEDIDO con el pedido removido equivocadamente.

PROCESO REGISTRO DE FECHAS DE MOVIMIENTO DEL PEDIDO EN EL ALMACEN

Las fechas que se llevan para control de entrega del pedido en el almacén, son la fecha en que se inicia la recep -- ción del pedido (FECHA-INICIO-RECEPCION) y la fecha en que el pedido se terminó de surtir (FECHA-RECEP-COMPLETA). Para registrar la primera fecha ejecutar el proceso 1.1. Si se de - sea el registro de la fecha de terminación de la entrega, ejecutar el proceso 1.2.

PROCESO 1.1. Registro de la fecha de inicio de recepción.

PROCEDIMIENTO:

Teclear * ENTRADA-ALMACEN (Clave requisición, clave - pedido fecha de primera entrada al almacen)

RESPUESTA

1.1.1. - 1 SELECTED DATA SETS -
El proceso se efectuó con éxito, ir al proceso 1.3

1.1.2. - 0 SELECTED DATA SETS -
Revisar si las claves de requisición y pedido se - teclearon equivocadas, de ser así, regresar al PROCESO 1.1. Si las claves están correctas, comunicarse al Depto. para avisar que el pedido no está dado de alta.

1.1.3. Cualquier otra respuesta, regresar al proceso 1.1

PROCESO 1.2 Registro de la fecha de terminación de entrega - en el almacén.

PROCEDIMIENTO:

Teclear *ENTREGA-TERMINADA (clave de requisición, clave de pedido, fecha de terminación de entrega)

RESPUESTA:

1.2.1. - 1 SELECTED DATA SETS -
El proceso se efectuó con éxito, ir al Proceso - 1.3.

1.2.2. - 0 SELECTED DATA SETS -
Revisar si las claves de requisición y pedido se-teclearon equivocadas, de ser así, regresar al -- Proceso 1.2.
Si las claves están correctas, comunicarse al -- Depto. para avisar que el pedido no está dado de-alta.

1.2.3. Cualquier otra respuesta, regresar al proceso de-fechas de movimiento del pedido en almacén.

PROCESO 1.3 Verificar que la(s) fecha(s) alimentadas, quedaron registradas correctamente.

PROCEDIMIENTO

Teclear * LEE-PEDIDO (clave pedido, clave requisición)

RESPUESTA

REQUISICION *	Clave de requisición
ESTADO *	Indica si el pedido esta vi - gente.
TOTAL LOTES *	Lotes del pedido
PARTIDA PRESUPUESTAL *	Partida Presupuestal
IMPORTE *	Importe
PROVEEDOR *	Clave del proveedor
CLAVE PEDIDO *	Clave de pedido
FECHA-ENTREGA-PROVE *	Fecha de entrega a Proveduria
FECHA-INICIO-RECEPCION *	Fecha en que se inicio la en- trega.
FECHA-RECEP-COMPLETA *	Fecha de terminación de entre ga.

1.3.1. Si las fechas estan correctas, el proceso se ter minó con éxito.

1.3.2. Si las fechas están incorrectas, iniciar nueva - mente este proceso.

PROCESO CORRECCION PROVEEDOR

Los datos que se pueden corregir son:

RAZON SOCIAL - Procesar Paso 1
TELEFONO - Procesar Paso 2
CALLE NUMERO - Procesar Paso 3
COLONIA CIUDAD - Procesar Paso 4
TELEFONO ADICIONAL - Procesar Paso 5

NOTA.- En caso de que la clave del proveedor este equivocada, se deben ejecutar los procesos de:

A) BAJA-PROVEEDOR y
B) ALTA-PROVEEDOR únicamente.

Será necesario ejecutar tantos pasos como datbs se deseen modificar, al terminar los pasos de corrección necesarios, ir al Paso 6.

PASO 1 CORREGIR RAZON SOCIAL.

PROCEDIMIENTO:

Teclear *CAMBIO-RAZON-SOCIAL (clave proveedor, razon social)

RESPUESTA:

1.1 - 1 SELECTED DATA SET -
La modificación se efectuó con éxito.

1.2 Cualquier otro mensaje, ir al Paso 1

PASO 2 CORREGIR TELEFONO

PROCEDIMIENTO:

Teclear * CAMBIO-TELEFONO-1 (clave proveedor,tele -

fono.

RESPUESTA:

1.1. - 1 SELECTED DATA SET -
Indica que la modificación se realizó con éxito

1.2 Cualquier otro mensaje ir al Paso 2.

PASO 3 CORREGIR CALLE NUMERO

PROCEDIMIENTO:

Teclear *CALLE-NUMERO (clave proveedor, calle y número)

RESPUESTA:

1.1 - 1 SELECTED DATA SET -
Indica que la modificación se realizó con éxito,

1.2 Cualquier otro mensaje, ir al Paso 3.

PASO 4 CORREGIR COLONIA CIUDAD

PROCEDIMIENTO:

Teclear * COLONIA-CIUDAD (clave proveedor, colonia- y ciudad)

RESPUESTA:

1.1 - 1 SELECTED DATA SET -
Indica que la modificación se realizó con éxito.

1.2 Cualquier otro mensaje ir al Paso 4.

PASO 5 CORREGIR TELEFONO ADICIONAL

PROCEDIMIENTO:

Teclear * TELEFONO-ADICIONAL (clave proveedor, telefono adicional)

RESPUESTA:

1.1 - 1 SELECTED DATA SET -
Indica que la modificación se realizó con éxito

1.2 Cualquier otro mensaje ir al Paso 5.

PASO 6 COMPROBAR CORRECCIONES PROVEEDOR.

Verificar que los datos modificados estén correctos.

PROCEDIMIENTOS:

Teclear * LEE-PROVEEDOR (clave del proveedor)

RESPUESTA:

CLAVE PROVEEDOR	- Clave proveedor
RAZON SOCIAL	- Razón Social
TELEFONO	- Teléfono
CALLE NUMERO	- Calle y número
COLONIA CIUDAD	- Colonia y Ciudad
TELEFONO ADICIONAL	- Teléfono adicional

6.1 Si los datos están correctos, fin del proceso

6.2 Si alguno está incorrecto, repetir el Paso de correcciones correspondientes y regresar al Paso 6.

PROCESO ALTA CENTRO

PASO 1 Verificar que la clave de centro no exista.

PROCEDIMIENTO

Teclear *BUSCA-CENTRO (clave centro)

RESPUESTA

1.1 0 SELECTED DATA SETS No existe el centro en la Base de Información. Ir al Paso 2

1.2 CLAVE-CENTRO* Clave centro

Indica que la clave de centro ya existe en la Base de Información. El usuario debe revisar si la clave deseada fué tecleada correctamente; en este caso proceder a realizar el análisis para determinar la causa del error. Si la clave del centro se tecleó equivocada, regresar Paso 1

PASO 2 Proporcionar la información correspondiente al centro

PROCEDIMIENTO

Teclear *DATOS-ALTA-CENTRO (clave centro, nombre centro)

RESPUESTA

2.1 - 1 SELECTED DATA SETS -

La adición de datos se realizó con éxito. Ir al paso 3

2.2 Cualquier otro mensaje regresar al Paso 2.

PASO 3 Verificar los valores que se adicionan al centro.

PROCEDIMIENTO

Teclear * [illegible] (clave centro)

RESPUESTA

CLAVE-CENTRO *	Clave centro
NOMBRE-CENTRO *	Nombre del centro

3.1 Si la información está incorrecta, realizar el proceso de corrección-centro (ver índice)

3.2 Si la información está correcta, el proceso terminó con éxito.

PROCESO CORRECCION CENTRO

El dato que se puede corregir es:

NOMBRE CENTRO - Procesar el Paso 1

NOTA: Si la clave de centro está equivocada, procesar:
BAJA CENTRO (avisar el error a las Ofnas. Centrales)
ALTA CENTRO únicamente

PASO 1 CORRIGE NOMBRE CENTRO

PROCEDIMIENTO:

Teclear *CAMBIO-NOMBRE-CENTRO (clave centro, nombre centro)

en donde:

Nombre Centro es el valor que se desea sustituir por el

Nombre Centro Anterior

RESPUESTA:

1.1 - 1 SELECTED DATA SET -
Indica que la modificación se efectuó con éxito.
ir al Paso 2

1.3 Cualquier otro mensaje ir al Paso 1

PASO 2 COMPROBAR CORRECCION NOMBRE CENTRO

Verificar si el dato modificado está correcto.

PROCEDIMIENTO:

Teclear * LEE-CENTRO (clave centro)

RESPUESTA:

CLAVE CENTRO * Clave centro

NOMBRE CENTRO * Nombre centro

2.1 Si los datos están correctos, el proceso se efectuó con éxito.

2.2 Si el dato está incorrecto, ir al Paso 1.

PROCESO ALTA DEPARTAMENTO

PASO 1 Verificar que la clave de departamento no exista en el centro dado.

PROCEDIMIENTO

Teclear * BUSCA-DEPARTAMENTO (clave departamento - clave centro)

RESPUESTA

1.1 0 SELECTED DATA SETS -
Indica que la clave del departamento no existe para ese centro. Continuar al paso 2.

1.2 CLAVE-DEPARTAMENTO * Clave departamento
CLAVE-CENTRO * Clave centro

Indica que la clave de departamento ya existe en la Base de Información. El usuario debe revisar si la clave deseada fué tecleada correctamente, en este caso procederá a realizar el análisis para determinar la causa del error. Si las claves de departamento ó centro se teclearon equivocadas, regresar al Paso 1.

PASO 2 Verificar que exista el centro.

PROCEDIMIENTO:

Teclear * BUSCA-CENTRO (clave centro)

RESPUESTA

2.1 0 SELECTED DATA SET. Revisar si la clave del centro fué tecleada erroneamente, en tal caso regresar al Paso 2 si la clave se tecleó correctamente indica que el centro no existe en la Base de Información, en tal caso ir al PROCESO ALTA CENTRO y una vez efectuado este, ir al Paso 3.

2.2 CLAVE-CENTRO * Clave centro
NOMBRE-CENTRO * Nombre del centro

Indica que el centro ya existe en la Base de Información. Ir al Paso 3

PASO 3 Proporcionar la información correspondiente al Departamento.

PROCEDIMIENTO

Teclear * DATOS-ALTA-DEPARTAMENTO (clave departamento clave centro)

RESPUESTA

3.1 - 1 SELECTED DATA SETS -
111 La adición del departamento se realizó.

DESCRIPCION DE LAS CADENAS Y FUNCIONES QUE PERMITEN LA OPERACION DE LOS PROCESOS UTILIZADOS EN LA INTEGRACION DE LA BASE DE DATOS.

```
describe strings;
 500* BUSCA-PROVEEDOR (STRING (PRINT/NAME/ C51,C52 WHERE C51 EQ *1*;)
      )
 501* DATOS-PROVEEDOR (STRING (INSERT TREE C50 * 0 * 1 EQ 51* *1* *52
      *          *2* * 53* *3* * END*;))
 502* LEE-PROVEEDOR (STRING (PRINT/NAME/ C50 WHERE C51 EQ *1*;))
1000* BUSCA-DEPARTAMENTO (STRING (PRINT/NAME/ C102,C11 WHERE C102 EQ
      *1*        AND C11 EQ *2*;))
1001* DATOS-ALTA-DEPARTAMENTO (STRING (INSERT TREE C100 * 0 EQ 102* *
      1*       * END* WHERE C11 EQ *2*;))
1002* LEE-DEPARTAMENTO (STRING (PRINT/NAME/ C11,C102 WHERE C102 EQ *1
      * AND C11  EQ *2*;))
2000* BUSCA-REQUISICION (STRING (PRINT/NAME/ C201,C204 WHERE C201 EQ
      *1* AND    C102 EQ *2* AND C11 EQ *3*;))
2001* DATOS-ALTA-REQUISICION (STRING (INSERT TREE C200 * 1 EQ 201* *1
      * *        202* *2* * 203* *3* * 204* *4* * END * WHERE C102 EQ
       *5* AND C11 EQ *6*;))
2002* LEE-REQUISICION (STRING (PRINT/NAME/ C201,C202,C203,C204,C205,C
      206        WHERE C201 EQ *1* AND C102 EQ *2* AND C11 EQ *3*;))
3000* BUSCA-PEDIDO (STRING (PRINT/NAME/ C309,C201,C307 WHERE C309 EQ
      *1*        AND C201 EQ *2*;))
3002* LEE-PEDIDO (STRING (PRINT/NAME/ C201,C300 WHERE C309 EQ *1* AND
       C201 EQ   *2*;))
4000* BUSCA-CENTRO (STRING (PRINT/NAME/ C11,C12 WHERE C11 EQ *1*;))
4001* DATOS-ALTA-CENTRO (STRING (INSERT TREE C10 * 0 * 1 EQ 11* *1* *
           12* *2* * END *;))
4002* LEE-CENTRO (STRING (PRINT/NAME/ C11,C12 WHERE C11 EQ *1*;))
 601* TITULOS-1 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS       PAN
      TALLA      1,L(13)F E D I D O,L(10)F-PROMESA,L(19)   I M P O R
      T E))
 602* TITULOS-2 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS       PAN
      TALLA      2))
 630* SUMA-REQ (STRING (PRINT/NAME/ *C802* WHERE C11))
 631* SIN-PEDIDO (STRING (PRINT/NAME/ *C803* WHERE C201 EXISTS AND C3
      09         FAILS AND C11))
 632* CON-PEDIDO (STRING (PRINT/NAME/ *C804* WHERE C201 EXISTS AND C3
      09         EXISTS AND C11))
 633* SUMA-IMPORTES-PEDIDOS-POR REQUISICION (STRING (PRINT/NAME/ *C80
      1*          WHERE C11))
 701* PANTALLA-1 (STRING (*C601*,L(9)PROVEEDOR/ C309,C311,C306,C307 W
      HERE        C201 EQ *1*;LIST/TITLE L(17)NUMERO DE PEDIDOS,L(17)
      SUMA DE IMPORTES,L(12)REQUISICION/ *C800*,*C801*,C201 WHERE SAM
      E;))
```

```
/02* PANTALLA-2 (STRING (*C602*,L(9)PROVEEDOR,L(13)REQUISICION,L(15)
     EQUISICIONES/ C1..C11,C201 WHERE C11 EQ *1*; *C603* EQ *1*; *C
     631* EQ *11; *C632*          EQ *14; *C633* EQ *1*;))
604* TITULO3-4 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS      PANTA
     LLA          4,L(3)DEPARTAMENTO,L(2)/DEPARTAMENTO,L(2)REQUISICION
/04* PANTALLA-4 (STRING (*C604*,L(13)    AND*FALSE*,DESTAL,L(10)ENTRE
     GA A* O.P. A./ C11,C102,C201,C203,C204 WHERE C201 EXISTS AND C1
     09 FAILS AND C102            EQ *2* AND C11 EQ *1*; *C631* EQ *
     1* AND C102 EQ *2*;))
605* TITULOS-5 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS      PANTA
     LLA          5,L(13)REQUISICION,L(12)F E D I D O,L(10)F-PROMESA,
     L(12)IMPORTE,L(9)PROVEEDOR/ C201,C309,C311,C303,C307;))
705* PANTALLA-5 (STRING (*C605* WHERE C312 FAILS AND C311 LT *3* AND
      C102       EQ *2* AND C11 EQ *1*:LIST/TITLE L(7)CENTRO,L(13)DEP
     ARTAMENTO,L(28)NUMERO DE PEDIDOS VENCIDOS/ C11,C102,*C800* WHE
     RE SAME;))
503* CALLE-NUMERO (STRING (ASSIGN DIRECCION-P1 EQ *2* * WHERE C51 EQ
      *1*;))
504* COLONIA-CIUDAD (STRING (ASSIGN DIRECCION-P2 EQ *2* * WHERE C51
     EQ           *1*;))
505* TELEFONO-ADICIONAL (STRING (ASSIGN TELEFONO-2 EQ *2* * WHERE C5
     1 EQ         *1*;))
506* CAMBIO-RAZON-SOCIAL (STRING (ASSIGN C52 EQ *2* * WHERE C51 EQ *
     1*;))
507* CAMBIO-TELEFONO-1 (STRING (ASSIGN C53 EQ *2* * WHERE C51 EQ *1*
     ;))
606* TITULOS-6 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS      PANTA
     LLA          6))
706* PANTALLA-6 (STRING (*C606*,L(9)PROVEEDOR,L(13)REQUISICION,L(15)
     P E D I D O,L(10)F-PROMESA/ C307,C201,C309,C311 WHERE C312 FAIL
     S AND C311 LT *2*            AND C307 EQ *1*:LIST/TITLE L(24)TO
     TAL PEDIDOS VENCIDOS/ *C800* WHERE SAME;))
708* PANTALLA-8 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS      PANT
     ALLA         8,L(29)DATOS GENERALES DEL PROVEEDOR/ C51 WHERE C51
      EQ *1*;PRINT/NAME/          C50 WHERE C51 EQ *1*;))
709* PANTALLA-9 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS      PANTA
     LLA          9,L(29)DATOS GENERALES DEL PEDIDO/ C309 WHERE C309
     EQ *1*;PRINT/NAME/ C201,     C300 WHERE C309 EQ *1*;))
610* TITULOS-10 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS      PANT
     ALLA         10,L(13)REQUISICION,L(13)F E D I D O,L(10)F-PROMESA
     ,L(9)PROVEEDOR))
611* TITULOS 11 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS      PANT
     ALLA         11,
/10* PANTALLA-10 (STRING (*L610*,L(10) INICIO+ENTREGA/ C201,C309,C31
     1,C307      C312 WHERE C312 EXISTS AND C311 FAILS AND C102 EQ *2
     * AND C11 EQ *1*:LIST/TITLE L(30)PEDIDOS PARCIALMENTE SURTIDOS
     / *L8001 WHERE SAME;)
611* PANTALLA-11 (STRING (*L611*,L(5 DEPTO.,L(13)REQUISICION,L(13)F E
      D I D O,L(10)INICIO+ENTREGA/ C102,C20
     3,C30 *,C31*2,C3 )                  WHERE C 3 EXI  C 201 C  EQ *1*;
     ID  *                                                    *C800* SAM
     E;)
612* TITULOS 12 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS      PANTA
     LLA          12,
```

```
712*  PANTALLA-12 (STRING (*C612*,L(10)PROVEEDOR,L(13)REQUISICION,L(1
      5)F E D 1 D 0,L(10)F-PROMESA/ C307,C201,C309,C311 WHERE C307 EQ
       *1*;LIST/TITLE L(26)NUMERO  TOTAL DE PEDIDOS/ *C800* WHERE SAN
      E;))
613*  TITULOS-13 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS    REQ P
      ENDIENTES))
713*  PENDIENTES (STRING (*C613*,L(13)REQUISICION,L(10)ENTREGA A+ G.P
      .A.,L(10)CIERRE+CONCURSO/ C201,C204,C205 WHERE C205 FAILS;))
614*  TITULOS-14 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS    REQ E
      N CONCURSO))
714*  EN-CONCURSO (STRING (*C614*,L(13)REQUISICION,L(10)ENTREGA A+ G.
      P.A.,L(10)CIERRE+CONCURSO/ C201,C204,C205 WHERE C205 GT *1*;))
615*  TITULOS-15 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS    REQ E
      N ELABORACION DE PEDIDOS))
715*  EN-ELABORACION (STRING (*C615*,L(13)REQUISICION,L(10)ENTREGA A+
       G.P.A.,   L(10)CIERRE+CONCURSO/ C201,C204,C205 WHERE C206 FAIL
      S AND C205 LT *1*;))
616*  TITULOS-16 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS    REQ E
      N AUTORIZACION))
716*  EN-AUTORIZACION (STRING (*C616*,L(13)REQUISICION,L(10)ENTREGA A
      + G.P.A.,  L(10)CIERRE+CONCURSO,L(10)REQ EN AUT+ORIZACION/ C201
      ,C204,C205,C206 WHERE        C206 EXISTS AND C309 FAILS;))
3003* PEDIDO-COLOCADO (STRING (ADD C300 EQ 307* *3* *311* *4* *306* *
      5*          *END* WHERE C201 EQ *1* AND C309 EQ *2*;))
3004* ENTRADA-ALMACEN (STRING (ASSIGN C312 EQ *3* * WHERE C201 EQ *1*
       AND        C309 EQ *2*;))
3005* ENTREGA-TERMINADA (STRING (ASSIGN C313 EQ *3* * WHERE C201 EQ *
      1*          AND C309 EQ *2*;))
3006* CANCELACION-PEDIDO (STRING (ASSIGN C301 EQ *3* * WHERE C201 EQ
      *1*         AND C309 EQ *2*;))
2003* CAMBIO-REQUISICION (STRING (ASSIGN C200 EQ 201* *1* *202* *2* *
      203*        *3* *204* *4* *END* WHERE C201 EQ *1* AND C102 EQ *5
      * AND C11 EQ *6*;))
2004* FIN-CONCURSO (STRING (ASSIGN C205 EQ *4* * WHERE C201 EQ *1* AN
      D C102 EQ  *2* AND C11 EQ *3*;))
2005* ENTRADA-AUTORIZACION (STRING (ASSIGN C206 EQ *4* * WHERE C201 E
      Q *1* AND  C102 EQ *2* AND C11 EQ *3*;))
3007* CAMBIO-TOTAL-LOTES (STRING (ASSIGN C302 EQ *5* * WHERE C309 EQ
      *1* AND C201 EQ *2* AND C102 EQ *3* AND C11 EQ *4*;))
3008* CAMBIO-PARTIDA-PRESUPUESTAL (STRING (ASSIGN C303 EQ *5* * WHERE
       C309 EQ *1* AND C201 EQ *2* AND C102 EQ *3* AND C11 EQ *4*;))
3009* CAMBIO-IMPORTE (STRING (ASSIGN C306 EQ *5* * WHERE C309 EQ *1*
      AND C201 EQ *2* AND C102 EQ *3* AND C11 EQ *4*;))
3010* CAMBIO-PROVEEDOR (STRING (ASSIGN C307 EQ *5* * WHERE C309 EQ *1
      * AND C201  EQ *2* AND C102 EQ *3* AND C11 EQ *4*;))
3011* CAMBIO-FECHA-ENTREGA-PROVE (STRING (ASSIGN C310 EQ *5* * WHERE
      C309 EQ *1* AND C201 EQ *2* AND C102 EQ *3* AND C11 EQ *4*;))
3012* CAMBIO-FECHA-PROMESA-ENTREGA (STRING (ASSIGN C311 EQ *5* * WHER
      E C309 EQ   *1* AND C201 EQ *2* AND C102 EQ *3* AND C11 EQ *4*;
      ))
4003* CAMBIO-NOMBRE-CENTRO (STRING (ASSIGN C12 EQ *2* * WHERE C11 EQ
      *1*;))
 900* ELIMINA-PROVEEDOR (STRING (PRINT/NAME/ C50 WHERE C51 EQ *1*;REM
      OVE C50     WHERE C51 EQ *1*;))
```

```
 900*  ELIMINA-C1 (STRING (*C901* WHERE C11 EQ *1* AND C102 EQ *2*
       AND C201   EQ *3* AND C102 EQ *3* AND C11 EQ *4*;REMOVE (*C
       900 WHERE C307 EQ *1* AND C201 EQ *2* AND C102 EQ *3* AND C11 E
       Q *4*;))
 901*  AL-FE (STRING (*C901* WHERE C300 = 0 OR C11 EQ *1* OR C102 EQ *2*
       *2* *3100*  */* *END* WHERE C201 EQ *1* AND C102 EQ *3* AND C11
       EQ *4*;))
 693*  TITULO-3A (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS
       PANTALLA  3))
 795*  SUMA-REQUI (STRING (PRINT/NAME/*C802* WHERE C11 EQ *1*))
 603*  TITULO-3 (STRING (LIST/TITLE D(3)PETROLEOS MEXICANOS
                         PANTALLA    3,L(6)CENTRO,L(12)DEPARTAMENTO,L
       (12)REQUISICION.L(13)F E B I B O,L(9)PROVEEDOR/ C11,C102.C201,C
       307,C307))
 795*  PANTALLA-3 (STRING (*C603* WHERE C11 EQ *1* AND C102 EQ *2*; *C
       630* EQ *1* AND C102 EQ *2*; *C632* EQ *1* AND C102 EQ *2*;))
 794*  PANTALLA-3B (STRING (*C603* WHERE C102 EQ *1*; PRINT/NAME/ *C80
       1* WHERE C102 EQ *1*;))
 793*  PANTALLA-3C (STRING (*C603* WHERE C11 EQ *1*;))
---
\
```

FUNCIONES

```
describe functions;
 6003  [illegible]
 [illegible]
 [illegible]
 [illegible]
 [illegible]
---
\
```

DIRECTORIO DE PANTALLAS

1- PANTALLA 1

Dada una requisición, la pantalla 1 despliega la siguiente información:

PEDIDO	F-PROMESA	IMPORTE	PROVEEDOR
Clave Pedido	Fecha promesa	Importe	Clave Provee

Correspondientes a la requisición dada, y además obtiene los siguientes totales:

NUMERO DE PEDIDOS	SUMA DE IMPORTES	REQUISICION
XXX	$$$$	Clave-Requisic.

Para accesarla el usuario deberá teclear:

* PANTALLA-1 (Clave-requisición)

Obteniendo lo antes descrito

2- PANTALLA 2

Dado un centro, la pantalla 2 despliega todas las requisiciones pertenecientes a ese centro de trabajo, en la siguiente forma:

CENTRO	DEPARTAMENTOS	REQUISICIONES
Clave-centro	Clave-departamento	Clave-Requisic.

y además obtiene los siguientes totales:

TOTAL REQUISICIONES	XXXX
REQUISICIONES - SIN - PEDIDO	YYYY
REQUISICIONES - CON - PEDIDO	XXXX
IMPORTE-REQUISICIONES	$$$$

El usuario deberá teclear

* PANTALLA 2 (clave-centro)

3- PANTALLA 3

Dado un centro-departamento, la pantalla 3 despliega todas las requisiciones pertenecientes a éste, en la siguiente forma:

CENTRO	DEPARTAMENTO	REQUISICIONES
Clave-centro	Clave-Departamento	Clave-Requisicion

Obteniendo además los siguientes totales:

TOTAL-REQUISICIONES	XXXX
REQUISICIONES-SIN-PEDIDO	YYYY
REQUISICIONES-CON-PEDIDO	XXXX
IMPORTE-REQUISICIONES	$$$$

Para accesarla el usuario deberá teclear.

* PANTALLA 3 (clave-centro, clave-departamento)

4- PANTALLA 4

Dado un centro-departamento, la pantalla 4 despliega las requisiciones sin pedido perteneciente a éste.

La información es la siguiente:

CENTRO	DEPARTAMENTO	REQUISICION	AÑO PRESUPUESTAL	ENTREGA A: D.P.A.
Clave-ctro.	Clave-Dento.	Clave-Req.	Año-Presupuest.	Fecha - entrega D.P.A.

Además obtiene el total de estas requisiciones, despliega:

REQUISICIONES - SIN PEDIDO [illegible]

Para accesarla el usuario deberá teclear:

*PANTALLA 4 (Clave-centro, clave-Departamento)

NOTA: Es posible que en el desplegado de - los totales de las pantallas 2, 3, 4- aparezca el siguiente mensaje
O SELECTED DATA SETS
Esto implica que no existen elementos en la Base de Información que satisfacen la condición requerida.

Tal desplegado omitirá el nombre del total, el cual es - fácil de consultar en este instructivo.

5- PANTALLA 5

Dado un centro-departamento, informa de aquellos pedi - dos cuya fecha de promesa de entrega esté vencida y cuya fecha de inicio de recepción, esté sin registrar, desple gando la información en la forma siguiente:

REQUISICION	PEDIDO	F-PROMESA	IMPORTE	PROVEEDOR
Clave Requisición	Clave Pedido	Fecha Promesa Entrega	Importe	Proveedor

Además despliega:

CENTRO	DEPARTAMENTO	NUMERO DE PEDIDOS
Clave-centro	Clave-departamento	Total de pedidos

Para accesarla el usuario deberá teclear:

* PANTALLA 5 (Clave-centro, clave-departamento, fecha deseada de vencimiento)

NOTA: La fecha de vencimiento deseado deberá teclearse - en la siguiente forma:

MM/DD/AA Donde MM es el mes
DD es el día
AA es el año

6- PANTALLA 6

Dado un proveedor la pantalla 6 despliega sus pedidos -- con fecha de promesa de entrega vencida y con fecha de - inicio de recepción ausente, en la siguiente forma:

PROVEEDOR	REQUISICION	PEDIDO	F-PROMESA
Clave-Proveed.	Clave-Req.	Clave-pedido	Fecha-Promesa de-Entrega.

Además, despliega el total de estos pedidos en la siguiente forma:

TOTAL PEDIDOS VENCIDOS

El usuario deberá teclear para accesarla:

* PANTALLA 6 (clave-proveedor, fecha del día de acceso)

NOTA: Para la fecha utilizar el mismo formato descrito anteriormente en (PANTALLA 5-(NOTA))

7- PANTALLA 7

Esta pantalla se encarga de informar para toda la empresa, el número total de requisiciones, cuantas tienen pedidos, cuantas no tienen pedidos y el importe de estos.

8- PANTALLA 8

Dado un proveedor, esta pantalla despliega sus datos generales.

Teclear *PANTALLA 8 (clave-proveedor)

9- PANTALLA 9

Dado un pedido, esta pantalla despliega sus datos generales, el usuario deberá teclear:

* PANTALLA 9 (clave-pedido)

10- PANTALLA 10

Dado un centro-departamento la pantalla 10 informa de los pedidos con fecha de inicio-de recepción presente y fecha de recepción-completa ausente, ésto es, de aquellos pedidos que ya empezaron a surtirse, pero que no han sido cubiertos totalmente.

El usuario deberá teclear:

* PANTALLA 10 (clave-centro, clave-departamento)

11- PANTALLA 11

Dado un centro, la pantalla 11 despliega los pedidos con fecha de inicio de recepción presente y fecha de recepción completa presente también, esto es, aquellos pedidos que fueron cubiertos totalmente.

El usuario deberá teclear:

* PANTALLA 11 (clave-centro)

12- PANTALLA 12

Dado un proveedor, la pantalla 12 nos informa del número total de pedidos de dicho proveedor.

El usuario deberá teclear:

* PANTALLA 12 (clave del proveedor)

13- PANTALLA 13

Esta pantalla se encarga de informar por centro, del número total de requisición, de estas cuántas son con pedido, cuantas sin pedido y el importe de estos.
Ejecutar el proceso como se indica en el Instructivo de - procesos de consulta.

14- PANTALLA 14

Por centro-departamento esta pantalla nos informa de:
Número total de requisiciones
Número total de requisiciones con pedido
Número total de requisiciones sin pedido
Importe total de pedidos.
Ejecutar el proceso como se indica en el Instructivo de - procesos de consulta.

15- PANTALLA 15

Por departamento, la pantalla 15 nos despliega:
Número total de requisiciones
Número total de requisiciones con pedido
Número total de requisiciones sin pedido
Importe total de los pedidos
Ejecutar el proceso como se indica en el Instructivo de - procesos de consulta.

16- PANTALLAS DE "SEGUIMIENTO DE REQUISICIONES"

A) Pendientes Informar las requisiciones que existen en- la Base que no han sido enviadas a concur- so.

Para acceder a el usuario debe teclear:

* PENDIENTES *

B) EN CONCURSO: Informa las requisiciones que existen en la Base que están en la etapa de Concurso.

Para accesarla teclear:

* EN-CONCURSO (fecha del día de acceso)

C) EN ELABORACION: Informa las requisiciones que ya cerraron su concurso y están en la etapa de elaboración de pedidos.

Para accesarla teclear:

* EN-ELABORACION (fecha del día de acceso)

D) EN AUTORIZACION: Informa las requisiciones que ya fueron atendidas y cuyos pedidos están en la etapa de autorización.

Para accesarla teclear:

* EN-AUTORIZACION *

INSTRUCTIVO DE OPERACION PARA PROCESOS DE EJECUCION DE CONSULTA

GENERALIDADES

Las consultas programadas para el sistema, que se muestran en el directorio de consultas, están separadas en dos -- grupos, que determinan el proceso para efectuar la propia -- consulta.

Estos procesos y sus consultas son:

Por el lenguaje natural

Pantalla 1
" 2
" 3
" 4
" 5
" 6
" 8
" 9
" 10
" 11
" 12

Pantallas de seguimiento de requisición

Pendientes
En-concurso
En-elaboración
En-autorización

Por Reporte Writer

Pantalla 7

Pantalla 13
" 14
" 15

PROCESO

Ejecución de consulta através de lenguaje natural.

PROCEDIMIENTOS

1- Conexión lógica de la terminal
2- Acceso al manejador SYSTEM 2000
3- Acceso a la Base de Datos de requisición y pedidos
4- Opcional. Asignación de reporte a un archivo de impresión.
5- Ejecución de la consulta requerida
6- Terminación de sesión de consulta o actualización.
7- Opcional. Impresión del archivo de resultado.

Los procedimientos: 1, 2, 3 y 6 se describen en el instructivo de operación en unidades foráneas.
Los procedimientos: 4, 5 y 7 se detallan a continuación.

4- Asignación del reporte a un archivo de impresión.

OBJETIVO- (Procedimiento exclusivo para la Unidad Administradora de Materiales en México)
OPCIONAL
Este procedimiento se utilizará cuando se desee imprimir en el computador central los resultados, si se usa, los resultados no pueden -

ser observados en la pantalla.

P A S O S

A) Una vez accesada la Base de Datos, teclear:
Report File is Reporte:

B) Si la respuesta es:

El procedimiento se terminó, cualquier otra respuesta, ir al paso A

5- Ejecución de la consulta requerida:

OBJETIVO- Mandar a ejecución el procedimiento de consulta deseado.

PASOS - A) Una vez seleccionada la pantalla de consulta en el Directorio de Pantallas, teclear lo que se indica en el mismo directorio.
Ejemplo: Consulta de Pantalla 3
Teclear: *PANTALLA-3 (800,320)
Donde: 800 corresponde a la clave del centro deseada
320 a la clave de departamento.

Los valores deben ser proporcionados encerrados en paréntesis, en el orden indicado en el directorio de pantallas y separados, un dato de otro, por una sola coma.

El procedimiento se ha terminado.

NOTA IMPORTANTE: Si la información que se solicitó rebasa la capacidad de la pantalla, se debe oprimir rapidamente la - -

tecla de espacio para detener la salida, mientras se copia la información que se requiera, el despliegue se vuelve a activar oprimiendo la tecla de RETURN ( )

7- Impresión del archivo de resultados

OBJETIVO- (Procedimiento exclusivo para la Unidad Administradora de Materiales en México.

OPCIONAL- Si se optó por la generación de un archivo de impresión (Procedimiento 4)
Con este procedimiento se envian a imprimir al computador central los resultados generados en el procedimiento de consulta.

PASOS: A) Una véz terminada la sesión con SYSTEM 2000 através del EXIT, y estando nuevamente bajo el modo COMAND, teclear:

FILES

En la respuesta de la terminal aparece en LOCAL FILES el archivo REPORTE, de no ser así, regresar al Procedimiento 4.

B) Enviar el archivo REPORTE a impresión
Teclear REWIND,REPORTE
Respuesta COMMAND-
Teclear BATCH, REPORTE, PRINT, UAME

Procedimiento se ha terminado.

NOTA: Los listados deben solicitarse en la sala de máquinas de la Gerencia de Informática con el nombre IUAME.

PROCESO Ejecución de consulta através de [illegible]

Procedimientos:

1 - Conexión lógica de la Terminal

2 - Acceso al archivo requerido de consulta

3 - Ejecución de la consulta requerida

El procedimiento 1, se describe en el instructivo de opera - ción en Unidades Foráneas.

Los procedimientos 2 y 3 se detallan a continuación:

2- Acceso al archivo requerido de consulta

OBJETIVO- Tener en disponibilidad inmediata el archivo - de consulta.

PASOS -

1- Llamado al archivo permanente que contiene la consulta.

Teclear:

ATTACH, CONSULT, PANTALLA XX, ID=UAMCENT, CY=1

Si el computador responde: COMMAND

El procedimiento se ejecutó con éxito.

Si la respuesta es FILE NOT CATALOGUED volver al Paso 1 de persistir esta respuesta el operador central debe - ejecutar el procedimiento

RESTAURACION DE ARCHIVOS PERMANENTES

y regresar al Paso 1.

NOTA IMPORTANTE: XX toma los valores, 13, 14 ó 15 dependiendo cuál consulta desee el usuario accesar.

La función de cada consulta está ampliamente explicada en el-

Directorio de Pantallas.

3- Ejecución de la consulta requerida.

OBJETIVO - Mandar a ejecución la consulta requerida.

Pasos - Una vez seleccionada la consulta en el Directorio de Pantallas.

Teclear:

BATCH, CONSULT, INPUT

NOTA: Los listados deben solicitarse en la sala de máquinas de la Gerencia de Informática, con los siguientes nombres:

Si la consulta ejecutada fué:

CONSULTA	LISTADO
PANTALLA 7	FUAIS07 nn
PANTALLA 13	FUAIS13 nn
PANTALLA 14	FUAIS14 nn
PANTALLA 15	FUAIS15 nn

Donde nn son 2 caracteres asignados por el computador.

EJEMPLOS DE ALGUNAS DE LAS PANTALLAS ANTES DESCRITAS

*pantalla-1 (202325000014)

```
 PETROLEOS MEXICANOS        PANTALLA      1
                  82/03/11

  P E D I D O     F-PROMESA       I M P O R T E        PROVEEDOR
***
* 80011942050     04/25/1979              $16344.00   1011169
  NUMERO DE PEDIDOS    SUMA DE IMPORTES    REQUISICION
***
*                  1           $16344.00   202325000014
---
\
```

*pantalla-3(202,320)

```
 PETROLEOS MEXICANOS                                  PANTALLA     3
                                  82/03/11

  CENTRO    DEPARTAMENTO   REQUISICION     P E D I D O      PROVEEDOR
***
* 202        320           202320070040    80011046090      1011005
* 202        320           202320070040    80011046396      1010177
* 202        320           202320070091    80011922686      1011005
TOTAL--EQUIPO,MUEBLES .
(PRIMARIO-FED,MUEBLES $42,[illegible]
---
```

*pantalla-2(202)

PETROLEOS MEXICANOS PANTALLA 2
82/03/11

CENTRO	DEPARTAMENTOS	REQUISICIONES
***		
* 202	320	202320090040
* 202	320	202320190091
* 202	321	202321600792
* 202	321	202321600793
* 202	321	202321600794
* 202	321	202321600795
* 202	321	202321600796
* 202	321	202321600798
* 202	321	202321690067
* 202	321	202321690069
* 202	321	202321690079
* 202	321	202321690080
* 202	322	202322090157
* 202	324	202324000555
* 202	324	202324000598
* 202	324	202324000612
* 202	324	202324000659
* 202	324	202324000660
* 202	324	202324000670
* 202	324	202324000674
* 202	324	202324000689
* 202	324	202324000766
* 202	324	202324100570
* 202	324	202324100790
* 202	324	202324100813
* 202	324	202324100826
* 202	324	202324190002
* 202	324	202324190035
* 202	324	202324195002
* 202	324	202324200823
* 202	324	202324200827
* 202	324	202324200833
* 202	325	202325000014
* 202	325	202325000311
* 202	325	202325000360
* 202	325	202325000361
* 202	325	202325000362
* 202	325	202325000363
* 202	325	202325000403
* 202	325	202325000404
* 202	325	202325000405

*pantalla-5(202320090040)

PETROLEOS MEXICANOS PANTALLA 5

82/03/11

REQUISICION	P E D I D O	F-PROMESA	IMPORTE	PROVEEDOR
***				
* 202320090040	80011946396	02/24/1980	$13/09.00	101C177
* 202320190091	80011922086	12/24/1979	$421170.00	101C003

CENTRO	DEPARTAMENTO	NUMERO DE PE DIDOS VENCIDOS
***		
* 202	320	2

\

*pantalla-6(101a380.05/09/80)

PETROLEOS MEXICANOS PANTALLA 6

82/03/11

PROVEEDOR	REQUISICION	P E D I D O	F-PROMESA
***			
* 101A380	20132/200059	80011821314	03/12/1979
* 101A380	20132/690017	80011920029	03/16/1979
* 101A380	22132/351617	80011821310	03/12/1979
* 101A380	22132/690152	80011923402	11/26/1979
* 101A380	24132/366561	80011920278	10/25/1979
* 101A38	24132/623660	80011924325	03/16/1980
* 101A38	24132/690112	80011923907	03/24/1980
* 101A380	24132/690113	80011923041	04/22/1980
* 101A380	24132/690120	80011922448	02/24/1980
* 101A382	24132/690124	80011922467	02/26/1980
* 101A382	[illegible]	[illegible]	[illegible]
* 101A382	[illegible]	[illegible]	[illegible]
* 101A382	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]

***

\

```
*pantalla-8(101a001)

  PETROLEOS MEXICANOS     PANTALLA        8
                  82/03/11

  DATOS GENERALES DEL PROVEEDOR
***
* 101A001

  CLAVE-PROVEEDOR* 101A001
  RAZON-SOCIAL* AMERICAN DENTAL PRODUCTS, S. A.
  TELEFONO-1* 5850905
  TELEFONO-2* 5758113
  DIRECCION-P1* MADERO NO. 43-106
  DIRECCION-P2* MEXICO, 1, D. F.
---
\
```

```
*pantalla-9(80013922129)

  PETROLEOS MEXICANOS     PANTALLA        9
                  82/03/11

  DATOS GENERALES DEL PEDIDO
***
* 80013922129
      CLAVE-REQUISICION* 201320000040

        ESTADO* V
        TOTAL-LOTES*     1
        PARTIDA-PRESUPUESTAL* 2012013206
        IMPORTE*       $110460.00
        PROVEEDOR* 1011175
        CLAVE-PEDIDO* 80013922129
        FECHA-ENTREGA-PROVE* 09/28/1979
        FECHA-PROMESA-ENTREGA* 12/28/1979
---
\
```

**pantalla-12(

PETROLEOS MEXICANOS PANTALLA 12
82/03/11

PROVEEDOR	REQUISICION	P E D I D O	F-PROMESA
***			
* 1011169	201327090025	80011942548	05/26/1980
* 1011169	201327090060	80011942336	01/29/1980
* 1011169	201327090816	80011940473	08/24/1979
* 1011169	202321600792	80011942398	12/23/1979
* 1011169	202325000014	80011942050	04/25/1979
* 1011169	202325090084	80011942277	12/12/1979
* 1011169	202327090117	80011942626	07/20/1980
* 1011169	202327200840	80011946040	09/14/1979
* 1011169	202329000785	80011942072	08/24/1979
* 1011169	202329090023	80011940632	12/19/1979
* 1011169	203328201151	80011942180	10/23/1979
* 1011169	221325001825	80011942158	10/20/1979
* 1011169	221327090243	80011942480	01/04/1980
* 1011169	221327090373	80011942600	04/23/1980
* 1011169	221328001754	80011942087	07/19/1979
* 1011169	221328001754	80011944037	05/04/1979
* 1011169	221328090152	80011942267	09/27/1979
* 1011169	233320090217	80013921014	08/19/1979
* 1011169	233320090230	80013921032	08/17/1979
* 1011169	233320090233	80013920968	10/09/1979
* 1011169	233320090233	80013960123	08/17/1979
* 1011169	233320090237	80013920971	09/19/1979
* 1011169	233320090239	80013921092	08/10/1979
* 1011169	241324103693	80011942014	03/21/1979
* 1011169	241324103834	80011942482	01/13/1980
* 1011169	241325003752	80011946000	03/21/1979
* 1011169	241327090160	80011942613	03/27/1980
* 1011169	241327090183	80011942506	02/21/1980
* 1011169	241327203618	80011942186	09/23/1979
* 1011169	241327290007	80011942277	12/26/1979
* 1011169	241327290007	80011942277A	
* 1011169	241327290008	80011942645	04/21/1980
* 1011169	241327300476	80011942264	10/22/1980
* 1011169	241327300477	80011942267	10/22/1980
* 1011169	241327300478	80011942662	11/19/1980
* 1011169	241328000402	80011942268	09/19/1980
* 1011169	241328000511	80011942269	09/23/1980
* 1011169	241328000527	80011942023	09/23/1979
* 1011169	241328000528	80011942024	09/23/1979
* 1011169	241328000529	80011942025	09/23/1979
* 1011169	241328000530	80011942026	09/23/1979
* 1011169	241328000531	80011942027	09/23/1979
* 1011169	241328000532	80011942028	09/23/1979
* 1011169	241328000533	80011942029	09/23/1979
* 1011169	241328000534	80011942030	09/23/1979
* 1011169	241328000535	80011942031	09/23/1979
* 1011169	241328000536	80011942032	09/23/1979

PROCEDIMIENTO PARA OPERAR EL SISTEMA

Operación en Unidades Foráneas

La operación normal del sistema se realiza através de la ejecución cronológica de los siguientes procedimientos:

1 - Conexión lógica de la terminal
2 - Acceso al manejador SYSTEM 2000
3 - Acceso a la Base de Datos de requisiciones y pedidos.
4 - Ejecución de procesos requeridos de actualización y consulta.
5 - Terminación de sesión de actualización y/o consulta.
6 - Apagado de la terminal.

1 - CONEXION LOGICA DE LA TERMINAL

OBJETIVO: Establecer comunicación entre la Unidad procesadora de datos y el computador central.

PASOS.

A) Oprimir la tecla de POWER en la terminal

B) Cuando aparezca el mensaje "PLEASE LOGIN" teclear la instrucción LOGIN y enviarla, oprimiendo la tecla de RETURN (A)

LOGIN (A)

C) Al ser solicitado el ENTER USERNAME, teclear la clave de usuario que se le haya asignado previamente, y enviarla.

D) Al ser solicitado el ENTER PASSWORD, teclear la - clave asignada previamente, y enviarla.

YYYYYY

E) Si las operaciones anteriores se realizaron de manera correcta, el usuario se encontrará bajo el - modo COMMAND, de haber cometido algún error re -- gresar al Paso C.

F) Aumentar el tiempo de procesador disponible al - usuario. Ejecutar:

ETL, 1000

2 - ACCESO A SYSTEM 2000

OBJETIVO: Tener disponible para su utilización através de- la terminal, al manejador de Base de SYSTEM 2000.

Pasos - Llamado a SYSTEM 2000, bajo el postulado COMMAND Teclear:

ATTACH, S2K260, ID=MRI, MR=1

Si la respuesta de la terminal es FILE NOT -- CATALOGUED avisar telefónicamente al responsable central del sistema.
Si la respuesta es: COMMAND
El SYSTEM 2000 está disponible.

Cualquier otra respuesta, regresar al Paso A.

3 - ACCESO AL BANCO DE DATOS DE REQUISICIONES Y PEDIDOS

OBJETIVO: Tener disponible para consulta o actualización - la información correspondiente a la Base de re - quisiciones, pedidos y proveedores de nuestro sis- tema.

Pasos A) Teclear

S2K260

a lo que el sistema responde:

AÑO/MES/DIA HORA BEGIN SYSTEM 2000 VERSION NNNN

Cualquier otra respuesta regresar al Paso A.

B) Proporcionar los identificadores de nuestra Base.

Teclear:

USER, Password asignado- como usuario del sistema.

Respuesta

Teclear

ID IS UAXXREQ

Respuesta

DBN IS UAXXRP (ver nota al calce)

Si las claves fueron tecleadas correctamente la respuesta es:

-556- ASSIGNED UAXXRP CYCLE nnnn FECHA HORA

La Base de Datos se encuentra disponible.

Si la respuesta es:

DATA BASE DOES NOT EXIST, O DATA BASE DAMAGED

Probablemente uno de los valores fué tecleado equivocadamente, regresar al Paso B.

De persistir esta respuesta avisar telefonicamente al responsable central del sistema, en la UAM México.

NOTA:

Si las operaciones que se van a realizar con la Base de Datos, son únicamente de consulta, para permitir el acceso de otros usuarios simultaneamente, teclear en lugar de

la instrucción marcada:

SHARED DBN IS UAXXRP

4 - EJECUCION DE PROCESOS REQUERIDOS DE CONSULTA Y ACTUALIZACION

Seleccionar en el diagrama de flujo de información, los procesos requeridos para las necesidades del usuario, y ejecutarlos como se indica en el instructivo de procesos.

5 - TERMINACION DE SESION DE ACTUALIZACION y/o CONSULTA

OBJETIVO - Dar por terminado la sesión, despidiendo la Base de Datos.

Pasos - A) Teclear

CONTROL;

Respuesta /

Teclear

Exit;

Respuesta

- 506 - CLOSED UAXXRP nn Fecha HORA

END SYSTEM 2000 VERSION 2,60 F

COMMAND -

Fin del procedimiento,

Cualquier otra respuesta, regresar al Paso A

6 - APAGADO DE LA TERMINAL

OBJETIVO - Desconectar la terminal del computador central

Paso - A) Teclear

LOGOUT

Respuesta
CPA nn,nnn SEG, nn.nnn ADJ
SYSTEM n HRS: nn min,
Fecha LOGGED OUT AT nn,nn,nn,

Se realizó el corte correctamente.

Cualquier otra respuesta ir al Paso A.

B) Oprimir la tecla de POWER en la terminal.

OPERACION EN UNIDAD CENTRAL

Las actividades diarias que debe ejecutar el operador central del Sistema, son:

AL_INICIO_DE_LA_JORNADA

1 - Conexión lógica de la terminal
2 - Investigar si la Base de Datos del sistema existe en - en el computador central
Ejecutar los procedimientos:

- Acceso al Manejador SYSTEM 2000
- Acceso a la Base de Datos de requisiciones y pedidos.

Si la respuesta es:
556 -UAXXRP ASSIGNED CYLE NN FECHA HORA

Verificar que el número nn corresponda al ciclo que se - muestra en el reporte de la última utilización del procedimiento "OBTENCION DE RESGUARDO DE LA BASE DE DATOS" (ver ejemplo), de ser así, continuar con la actividad 3, en caso de que el número de ciclo sea diferente, realizar los procedimientos:

- Borrar Base de Datos
- Restauración de la Base de Datos

y pasar a la actividad 3 -

Si la respuesta es:
DATA BASE IS DAMAGED ONLY ACCESS IS PERMITTED

Ejecutar los procedimientos:

- Borrar Base de Datos
- Restauración de la Base de Datos

y pasar a la actividad 3.

En cualquiera de las dos respuestas siguientes:

DATA BASE DOES NOT EXIT 6
STATUS OF DATA BASE IS INCONSISTENT

Proceder a ejecutar el procedimiento
RESTAURACION DE LA BASE DE DATOS
e ir a la actividad 3.

3 - Investigar si los archivos permanentes requeridos por el - sistema están catalogados.
Ejecutar el procedimiento.
Lectura de archivos permanentes

En caso de terminación anormal de este procedimiento, ejecutar el de:

- RESTAURACION DE ARCHIVOS PERMANENTES.

EN EL TRANSCURSO DE LA JORNADA

Si el operador recibe aviso de alguna unidad foránea de -- que la Base de Datos no puede ser accesada, realizar los si - - guientes procedimientos.

- Rescatar operaciones del día
- Restauración de la Base de Datos, a partir del último - resguardo y de las operaciones del día.

AL CIERRE DE LA JORNADA

4 - Obtener 2 copias en cinta magnética de la Base de Datos -- actualizada.
Ejecutar 2 veces el procedimiento.
OBTENCION RESGUARDO DE LA BASE DE DATOS

Todos los procedimientos que sean ejecutados deben ser revisados por medio del procedimiento REVISION DE RESULTADOS DE - CORRIDA, y deberán conservarse listados o impresiones de todos- los procedimientos ejecutados durante los últimos 5 días, en - orden cronológico.

Los procedimientos que a continuación se detallan son:

7 - Restauración de la Base de Datos
8 - Borrador de Base de Datos
9 - Lectura archivos permanentes
10- Restauración de archivos permanentes
11- Obtención resguardo de la Base de Datos
12- Rescatar operaciones del día
13- Restauración de la Base de Datos a partir del último res -- guardo y de las operaciones del día anexo.
14-.Control de cintas magnéticas
15- Asignación de número de cinta por editor
16- Revisión de resultados de corrida de procedimientos.

Los correspondientes a "Conexión lógica de la terminal" - "Acceso al manejador -SYSTEM 2000" y "Acceso a la Base de Datos"

se representan en la sección del instructivo de operación, en - unidades foráneas.

7 - RESTAURACION DE LA BASE DE DATOS

OBJETIVO - Alimentar al computador la información de la Base de Datos del sistema que reside en la última cinta de resguardo, obtenida en el procedimiento "obtención resguardo de la Base de Datos"

Pasos 1 - Llamado al archivo permanente que contiene el - programa.

Teclear:

ATTACH,RESTAUR,DECKRESTAURA,ID=UAMCENT,CY=1

Si el computador responde:

COMMAND-

Continuar el paso 2.

Si la respuesta es FILE NOT CATALOGUED, volver al Paso 1, de persistir esta respuesta el operador central debe ejecutar el procedimiento "RESTAURACION DE ARCHIVOS PERMANENTES" y regresar al paso 1.

2 - El archivo permanente de este procedimiento queda identificado como RESTAUR, seleccionar en la hoja de control de cintas la cinta correspondiente al último procedimiento de obtención de resguardo de la Base de Datos; ejecutar el procedimiento de asignación de número de cinta por editor.

3 - Mandar ejecutar el trabajo al procesador central

Teclear:

BATCH,DECK,INPUT,VARS

si la respuesta es:

COMMAND -

Pasar al punto 4.

Cualquier otra respuesta teclear ^A y regresar al punto 3.

4-Consultar si el trabajo ya fué procesado.

Teclear

FILES

Si aparece en la respuesta en el área de REMOTE OUTPUT-JOB el nombre LAUREXX continuar al paso 5: de otra manera esperar un minuto y volver al paso 4.

5-Ejecutar el procedimiento REVISION DE RESULTADOS DE CORRIDA:

Si éste tuvo terminación anormal, teclear FILES, a lo-que el sistema responde el nombre de los archivos permanente disponibles. Si aparece entre ellos el archivo RESTAURAR regresar al paso 2, si no aparece regresar al paso 1.

Si tiene terminación normal, el procedimiento ha sido-terminado.

8 - BORRADOR DE BASE DE DATOS

OBJETIVO

Como su nombre lo indica, consiste en borrar la Base de Datos del computador central.

PASOS

1) Llamado al archivo permanente que contiene el programa.

Teclear:

ATTACH,PURGA,DECKPURGA,ID=UAMCENT,CY=1

Si el computador responde

COMMAND -

continuar al paso 2

Si la respuesta es FILE NOT CATALOGUED, volver al paso 1, de persistir esta respuesta el operador central debe ejecutar el procedimiento RESTAURACION DE-ARCHIVOS PERMANENTES y regresar al paso 1.

2) El archivo permanente de este procedimiento queda identificado como PURGA, mandar ejecutar el trabajo-al procesador central.

Teclear:

BATCH, PURGA, INPUT, HERE

Si la respuesta es:

COMMAND -

Pasar al punto 3.

Cualquier otra respuesta teclear BA y regresar al punto 2.

3) Consultar si el trabajo ya fué procesado.

Teclear:

FILES

Si aparece en la respuesta el nombre LUAPxx en el área de REMOTE OUTPUT FILES continuar al paso 4; de otra manera esperar un minuto y volver al paso 3.

4) Ejecutar el procedimiento REVISION DE RESULTADOS DE - CORRIDA. Si éste tuvo terminación anormal, teclear -

FILES

a lo que el sistema responde el nombre de los archivos permanentes disponibles.

Si aparece entre ellos el archivo PURGA, regresar al - paso 2; si no aparece, regresar al paso 1.

Si tiene terminación normal, el procedimiento ha sido- terminado.

9 - LECTURA ARCHIVOS PERMANENTES

OBJETIVO

Es una protección de los archivos permanentes para evitar su pérdida.

PASOS

1) Llamado al archivo permanente que contiene el programa.

Teclear:

ATTACH,LECTURA,DECKLECTURA,ID=UAMCENT,CY=1

si el computador responde

COMMAND -

continuar al paso 2.

si la respuesta es FILE NOT CATALOGUED, volver al paso 1; de persistir esta respuesta el operador central debe ejecutar el procedimiento RESTAURACION DE ARCHIVOS PERMANENTES y regresar al paso 1.

cualquier otra respuesta, regresar al paso 1.

2) El archivo permanente de este procedimiento queda identificado como LECTURA, mandar ejecutar el trabajo al procesador central.

Teclear

BATCH, LECTURA, INPUT, HERE

si la respuesta es

COMMAND -

pasar al punto 3.

cualquier otra respuesta teclear ÁA y regresar al punto 2.

3) Consultar si el trabajo ya fué procesado, teclear -

FILES

si aparece en la respuesta el nombre LUAATxx, en el-área de REMOTE OUTPUT FILES, continuar al paso 4,de-otra manera esperar un minuto y volver al paso 3.

4) Ejecutar el procedimiento REVISION DE RESULTADOS DE-CORRIDA.

Si este tuvo terminación normal

Teclear

FILES

a lo que el sistema responde el nombre de los archi-vos permanentes disponibles.

Si aparece entre ellos el archivo LECTURA, regresar-al paso 2.

Si no aparece, regresar al paso 1.

Si tiene terminación normal, el procedimiento ha si-do terminado.

10 - RESTAURACION DE ARCHIVOS PERMANENTES

OBJETIVO

Alimentar al computador el (o los) archivos permanentes que se hayan borrado, los cuales residen en la última cinta de resguardo de los mismos.

PASOS

1) Llamado al archivo permanente que contiene el pro - grama.

Teclear

ATTACH,RESTAU,LECKRESTAURAPERM,ID=UAMCENT,CY=1

si el computador responde

COMMAND-

continuar al paso 3.

Si la respuesta es FILE NOT CATALOGUED, volver al - paso 1; de persistir esta respuesta ejecutar el siguiente procedimiento:

Teclear

EDITOR

a lo que el computador responde:

''

Teclear

C,S

La respuesta debe ser

ENTER LINES

Teclear

LUAPF,T100,P30,TL,NOMBRE DEL USUARIO

REQUEST,[illegible]TAPE,P=,S,T,No.CINTA EN QUE RESIDEN LOS-

ARCH.PER..

LOAD,P=[illegible],I,CL.

UNLOAD,DUMTAPE.
PF=DECKPURGA,ID=UAMCENT,CY=1
PF=DECKRESCATA,ID=UAMCENT,CY=1
PF=DECKRESTAURA,ID=UAMCENT,CY=1
PF=DECKLECTURA,ID,CY=1
PF=PANTALLA7,ID=AUMCENT,CY=1
PF=PANTALLA13,ID=UAMCENT,CY=1
PF=PANTALLA14,ID=UAMCENT,CY=1
PF=PANTALLA15,ID=UAMCENT,CY=1
PF=DECKRESGUARDO,ID=UAMCENT,CY=1
PF=DECKREGENERA,ID=UAMCENT,CY=1
=

LA RESPUESTA SERA

''

2) EJECUTAR EL PROCEDIMIENTO DE ASIGNACION DE NUMERO DE CINTA POR EDITOR.

Rescatar el procedimiento, Teclear:

S,RESTAU,N

A lo que el sistema responde

''

Catalogar el procedimiento teclear:

CATALOG,RESTAU,DECKRESTAURAPERM,ID=UAMCENT,CY=1

Salir de editor, teclear: B

la respuesta es

COMMAND -

continuar al paso 3

3) Mandar ejecutar el trabajo al procesador central. Teclear.

BATCH,RESTAU,INPUT,HERE

Si la respuesta es

COMMAND -

Pasar al punto 4

Cualquier otra respuesta teclear ^A y regresar al punto 3.

4) Consultar si el trabajo ya fue procesado

Teclear

FILES

Si aparece en la respuesta el nombre LAUPFxx en el área de REMOTE OUTPUT FILES continuar al paso 5 de otra manera esperar un minuto y volver al paso 4.

5) Ejecutar el procedimiento REVISION DE RESULTADOS DE CORRIDA.

Si éste tuvo terminación anormal, Teclear

FILES

A lo que el sistema responde el nombre de los archivos permanentes disponibles. Si aparece entre ellos el archivo RESTAU regresar al paso 2. Si no aparece regresar al paso 1.

Si tiene terminación normal, el procedimiento ha sido terminado.

11 - OBTENCION RESGUARDO DE LA BASE DE DATOS

OBJETIVO

La protección de la información de la Base de Datos -- del sistema resguardándola en una cinta magnética.

PASOS

1) Llamado al archivo permanente que contiene el pro - grama.

   Teclear

   ATTACH,RESGUAR,DECKRESGUARDO,ID=UAMCENT,CY=1

   si el computador responde

   COMMAND -

   continuar al paso 2

   Si la respuesta es FILE NOT CATALOGUED, volver al - paso 1, de persistir esta respuesta el operador central debe ejecutar el procedimiento RESTAURACION DE ARCHIVOS PERMANENTES y regresar al paso 1.

   Cualquier otra respuesta, regresar al paso 1.

2) El archivo permanente de este procedimiento queda - identificado como RESGUAR, seleccionar en la HOJA - DE CONTROL DE CINTAS una de ellas, de acuerdo al -- procedimiento CONTROL DE CINTAS MAGNETICAS.

   Ejecutar el procedimiento de ASIGNACION DE NUMERO - DE CINTA POR EDITOR;

3) Mandar ejecutar el trabajo al procesador central.

   Teclear:

   BATCH, DECK, INPUT, HERE

   si la respuesta es

   COMMAND -

   Pasar al punto 4.

Cualquier otra respuesta teclear BA y regresar al punto 3.

4) Consultar si el trabajo ya fué procesado.
Teclear

FILES

si aparece en la respuesta el nombre LUADxx en el área de REMOTE OUTPUT FILES continuar al paso 5, de otra manera esperar un minuto y volver al paso 4.

5) Ejecutar el procedimiento REVISION DE RESULTADOS DE CORRIDA.
Si éste tuvo terminación anormal, Teclear

FILES

A lo que el sistema responde el nombre de los archivos permanentes disponibles. Si aparece entre ellos el archivo RESGUAR, regresar al paso 2, si no aparece regresar al paso 1.
Si tiene terminación normal, el procedimiento ha sido terminado.

12 - RESCATAR OPERACIONES DEL DIA

OBJETIVO

Es una protección de las operaciones o movimientos del día, copiándolos en una cinta magnética.

PASOS

1) Llamando al archivo permanente que contiene el programa:
   Teclear:
   ATTACH,RESCATA,DECKRESCATA,ID=UAMCENT,CY=1
   si el computador responde
   COMMAND -
   continuar al paso 2.

   Si la respuesta es FILE NOT CATALOGUED, volver al paso 1, de persistir esta respuesta el operador central debe ejecutar el procedimiento RESTAURACION DE ARCHIVOS PERMANENTES y regresar al paso 1.
   Cualquier otra respuesta regresar al paso 1.
2) El archivo permanente de este procedimiento queda identificado como RESCATA, seleccionar en la HOJA DE CONTROL DE CINTAS una de ellas, de acuerdo al procedimiento CONTROL DE CINTAS MAGNETICAS;
   Ejecutar el procedimiento de ASIGNACION DE NUMERO DE CINTAS POR EDITOR.

3) Mandar ejecutar el trabajo al procesador central.
   Teclear
   BATCH,DECK,INPUT,HERE

   Si la respuesta es
   COMMAND -
   Pasar al punto 4.

Cualquier otra respuesta teclear ^A y regresar al -- punto 3.

4) Consultar si el trabajo ya fué procesado
Teclear

FILES

si aparece en la respuesta el nombre LUALCxx en el - área de REMOTE OUTPUT FILES, continuar al paso 5, de otra manera esperar un minuto y volver al paso 4.

5) Ejecutar el procedimiento REVISION DE RESULTADOS DE-CORRIDA.
Si éste tuvo terminación anormal, teclear

FILES

a lo que el sistema responde el nombre de los archivos permanentes disponibles.
Si aparece entre ellos el archivo RESCATA, regresar- al paso 2. Si no aparece regresar al paso 1.
Si tiene terminación normal, el procedimiento ha sido terminado.

13 - RESTAURACION DE LA BASE DE DATOS A PARTIR DEL ULTIMO RES - GUARDO Y DE LAS OPERACIONES DEL DIA ANEXO.

OBJETIVOS

Alimentar al computador la información de la Base de - Datos del Sistema que reside en la última cinta de resguardo, obtenida en el procedimiento "Obtención Res -- guardo de la Base de Datos" más la información obtenida en la cinta de salida del procedimiento "Rescatar operaciones del Día".

PASOS

1) Llamado al archivo permanente que contiene el programa.

Teclear:

ATTACH,REGENER,DECKREGENERA,ID=UAMCENT,CY=1

si el computador responde,

COMMAND -

continuar al paso 2

si la respuesta es FILE NOT CATALOGUED, volver al - paso 1: de persistir esta respuesta el operador central debe ejecutar el procedimiento:

RESTAURACION DE ARCHIVOS PERMANENTES

y regresar al paso 1

2) El archivo permanente de este procedimiento queda - identificado como

REGENER

Seleccionar en la hoja de control de cintas, la cinta correspondiente al último procedimiento de "Ob - tención de Resguardo de la Base de Datos", así como-

la cinta generada por el último procedimiento de - "Rescatar Operaciones del Día".

Ejecutar el procedimiento de "Asignación de Número- de Cintas por Editor".

3) Mandar ejecutar el trabajo al procesador central.

Teclear:

BATCH, DECK, INPUT, HERE

Si la respuesta es

COMMAND -

pasar al punto 4.

Cualquier otra respuesta, teclear

%A y regresar al paso 3.

4) Consultar si el trabajo ya fué procesado.

Teclear:

FILES

si aparece en la respuesta LUALRxx en el área de - REMOTE OUTPUT FILES, continuar al paso 5, de otra - manera, esperar un minuto y volver al paso 4.

5) Ejecutar el procedimiento REVISION DE RESULTADOS DE CORRIDA.

Si éste tuvo terminación normal, Teclear

FILES

A lo que el sistema responde el nombre de los archivos permanentes disponibles.

Si aparece entre ellos el archivo RETRIEVER regresar- al paso 2. Si no aparece, regresar al paso 1.

Si tiene terminación normal el procedimiento ha sido terminado.

14 - CONTROL DE CINTAS MAGNETICAS

OBJETIVO

Llevar un control operativo de las cintas magnéticas - que se tienen asignadas.

GENERALIDADES

A) Se cuenta con 10 cintas magnéticas, asignadas y el identificador de cada cinta es su número.

B) Se deben rolar las cintas magnéticas para su uso, tomando la más antigua para el proceso en turno.

C) Actualizar en el control operativo de cintas magnéticas cada uno de los eventos que se vayan presentando, en la forma anexa.

D) La descripción del llenado de la forma y la secuencia - de los eventos, es la siguiente:

Núm. de Cinta - Se anotará el número con el que se IDENTIFICA EXTERNAMENTE LA CINTA EN LA GERENCIA DE INFORMATICA.

Fecha de asignación de cinta - Se indicará con la fecha en el momento en que se vaya a usar una determinada cinta- en un proceso.

Fecha Baja - Cuando una cinta se vaya a usar en otro pro - ceso, se anotará su fecha de baja.

Programa que la Genera - Nombre del proceso que va a generar el archivo en dicha cinta.

Nombre del Archivo - Nombre del archivo generado por el - proceso en la cinta.

Confirmación de

Resultados - Confirmar que efectivamente se generó correctamente el archivo en la cinta usada.

NOTA: Al generar un nuevo archivo, en la cinta correspondiente, anotar la fecha de baja en fecha-baja del proceso anterior y la nueva fecha de asignación actualizando en el "control operativo de cintas magnéticas" con el nuevo proceso.

E) Los procedimientos que generan cinta son:

LUALD - Obtención resguardo de la Base de Datos.
LUALO - Rescatar operaciones del día.

15 - ASIGNACION DE NUMERO DE CINTA POR EDITOR

OBJETIVO

Asignar el número de cinta en el deck de corrida que - se va a ejecutar, por medio de editor.

PASOS

A) Teclear el comando

EDITOR

B) Cuando aparezcan dos puntos . . indica que ya está bajo editor.

C) Secuenciar el archivo correspondiente al procedimiento primario bajo editor.

Teclear

E,NOMBRE DEL ARCHIVO PERMANENTE,S,

Contesta:

' '

D) Listar el archivo editado

Teclear

L,A

Contestando:

Se listará en forma secuencial cada instrucción, localizar el número de instrucción correspondiente a la línea de REQUEST.

E) Cambiar el número de la cinta en la línea correspon - diente al "REQUEST"

Ejemplo:

Teclear /Numero Cinta/= /D-777/, V, 140

D-777- Número de cinta asignada al proceso

V- Es para que aparezca la línea con la nueva cinta.

140 - Línea correspondiente al "REQUEST"

F) Si al aparecer la línea, tiene el número correcto de - la cinta, Teclear:

Y

Contestación:

1 CHANGE (S) y pasar al paso G

de lo contrario, Teclear:

N

Contestación:

0 CHANGE (S)

Volver al paso E.

G) Salvar el nuevo archivo con número de cinta

Teclear:

S,DECK,O,N

Contestación:

' '

Teclear

B

H) Volver al procedimiento primario que solicitó la asig nación de cinta.

16 - REVISION DE RESULTADOS DE CORRIDA (DE PROCEDIMIENTOS)

OBJETIVO

Verificar los resultados obtenidos en el proceso llevado a cabo, por medio del comando PAGE

PASOS

A) Teclear . . FILES
y si en la contestación el archivo se encuentra en - - OUTPUT, o sea:
REMOTE OUTPUT FILES - nombre de nuestro proceso, ir al paso B. En caso contrario, regresar a A.

B) Volver local el archivo que se encuentra en OUTPUT
Teclear - BATCH, NOMBRE DEL ARCHIVO, LOCAL
Contestación: COMMAND

C) Una vez que se tiene local el archivo conectarse con - PAGE para revisar resultados.
Teclear: PAGE, NOMBRE DEL ARCHIVO
Contestación: READY . .

D) Ya estando dentro de PAGE, REVISAR EL DAY-FILE para - conocer los resultados.
Teclear $_{+,+,}$ - H
Contestación - Listará una parte desde su inicio del - DAY-FILE hasta que aparezca LINE 99
- Como referencia la primera línea que aparece contiene el nombre del proceso.

D) Ir revisando cada bloque o parte del DAY-FILE, mien - tras no aparezca el mensaje de terminación anormal;

Teclear: + cada vez que aparezca
LINE número hasta el
"END OF INFORMATION"

F) Una vez revisados los resultados de la corrida salirse- de PAGE

Teclear: E
Contestación: COMMAND -

G) Si la corrida tuvo salida o terminación normal mandar - a imprimir dicha corrida.

Teclear: BATCH, NOMBRE DEL ARCHIVO,PRINT,NOMBRE ARCHIVO P/IMPRESION
(4 DIGITOS)

H) El listado de la corrida saldrá a impresión con los 4,- dígitos más una "I" inicial

Ejemplo:

BATCH,FUACOH2,PRINT,UACO

Salida a impresión: como

IUA CO xx

I) Si apareció el mensaje entre asteriscos (+++) de terminación anormal, no se manda a imprimir.

CONCLUSIONES

El ritmo acelerado que prevalece en todos los ambitos vitales de este mundo, ha originado que día a día se desarrollen nuevas técnicas, herramientas, etc. cada vez mas eficientes y precisas que permitan un control adecuado de todos los problemas que este rítmo acarrea.

La informática como uno de estos ámbitos vitales no podía quedar atrás, y la gente especializada en esta corriente ha dado pasos gigantescos através de los años para solucionar la organización, recuperación, almacenamiento y actualización de la información. Si pensamos en la información como una base fundamental para analizar, evaluar, estudiar, preveer en fin todo lo necesario para poder efectuar una etapa de planeación, que permita un mejor funcionamiento de las instituciones y resuelva con solvencia los muchos problemas de la actualidad, entonces la información es un órgano vital en nuestro revolucionado mundo, y debemos cuidarla.

Sin embargo, como se sigue de la exposición de este trabajo, podemos afirmar que vamos por buen camino pero no hemos llegado aún a la meta ideal del control de información. No existe aún una computadora capaz de soportar un ambiente de Base de Datos y las organizaciones con mayor potencial industrial y económico no corren el riesgo de una conversión a sus sistemas actuales.

El tiempo es el factor que nos permitirá ver cristalizar los esfuerzos de los informáticos en su lucha por lograr esta meta ideal, hablar de ideal siempre es peligroso pero seamos optimistas.

Como parte integral del ambiente de Base de Datos están - valga la redundancia las Bases de Datos, y como una herramienta poderosa que facilita su operación están los GDBMS.

SYSTEM 2000 es uno de estos modernos paquetes y fué el elegido para el desarrollo de nuestra Base de Datos. Hablar de -- ventajas y desventajas de este manejador sería tema de un tra - bajo de investigación que consumiría un buen número de horas, - para nosotros fué el mas viable en primer lugar porque soporta- estructuras jerárquicas de Base de Datos, clase a la cual pertenece nuestra Base y en segundo lugar porque está diseñado para- operar con las computadoras de la serie CYBER de CDC y en la -- instalación en la que desarrollamos este trabajo se cuenta con- una computadora CYBER-170.

BIBLIOGRAFIA

- BACHMAN, C W, y WILLIAMS, S.B. A general-purpose programing system for random access memories. Ed. Spartan Books. 1964- Washington, D.C.
- CLIMENSON, W. D. File organization and search techniques. - Ed. Wiley. New York, 1966.
- Introduction to IBM/360 direct access storage deviees and - organization methods. Ed. IBM Corp. New York,1966.
- PETERSON, W.W. Addressing for random-acces storage. Ed. IBM 1957.
- SCHORR, H. y WAITE, W.M. An efficient machine-independent- procedure for garbage collection in various list structures. 1967.
- MURRAY,J.T. Programming in R P G II. Ed. McGraw-Hill. 1971.
- Mark IV Systems, Techmical System Description. Calif. 1974.
- Mark IV, Data Pro Research Corporation, New Jersey, 1975.
- ASI-ST, An advanced approach to Data Management, 1974.
- IDS Reference Manual, General Electric information systems division, Phoenix, Arizona, 1969.
- BLEIR, R.F. Treating Hierarchical Data Structures in the - SDC Timeshared Data Management System (TDMS), 1967.
- Information Management System Virtual Storage (IMS/VS) -- General Information Manual. Ed. IBM Corporation.
- CODASYL, Data Base Task Group (DBTG). Ed. Association for - Computing Machinery, New York, 1971.
- Data Base Management System Requirements. Ed. Data Base -- Requirements Group. 1971.

- TOTAL Reference Manual, Ed. Cincom Systems Corp., Cincinnati, Ohio.

- System 2000 Reference Manual, Ed. MRI Systems Corp., Austin, Texas, 1973.

- FLYNN, R.L., A Brief History of Data Base Management. 1974.

- CODASYL Stored Data Description and Translation Task Group, - Stored Data Description and Data Translation: A Model and - Language.

- SENKO, M.E., ALTMAN, E.B., ASTRAHAN, M.M. y P.L. FEHDER, Data Structures and Accessing in Data Base Systems. Ed. I.B.M. - Systems Journal. 1973.

- ASTRAHAN, M.M. Concepts of a Data Independent Accessing Model Ed. IBM Research Report, San Jose California, 1972.

- ALTMAN, E.B., Specifications in a Data Independent Accessing-Model. Ed. IBM Research Report. San Jose, California, 1973.