Universidad Nacional Autónoma de México

FACULTAD DE CIENCIAS

"Manual de Usuario para un Sistema de Bases de Datos Relacional"

T E S I S

Que para obtener el título de Licenciado en Matemáticas

p r e s e n t a

GILBERTO FUENTES ROMERO

México, D. F. **1987**

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Contenido

INTRODUCCION

Las técnicas de las bases de datos aparecen como respuesta a la problemática existente en los sistemas de programación de las instituciones que cuentan con centros de cómputo para el desarrollo de sus múltiples funciones, como por ejemplo, el de la Tesorería del Departamento del Distrito Federal, etc.

El presente trabajo está dirigido al personal de nuevo ingreso al área de procesamiento de datos de Tesorería que se dedica al desarrollo y mantenimiento de los múltiples sistemas de la institución mencionada. Y para todos aquéllos que se interesen por el concepto de bases de datos.

El principal objetivo del trabajo consiste en proporcionar a dichos usuarios las herramientas básicas para el manejo de la información, el cual tiene que ver con los sistemas de información y con los sistemas manejadores de bases de datos (relacional) para el desarrollo de sus múltiples programas de aplicación.

Para llevar a cabo tal objetivo el trabajo se dividió en cinco capítulos.

En el primer capítulo (¿ Qué es una base de datos?) se da un breve esbozo de la evolución de las bases de datos a través de

cuatro etapas, se da la definición de la base de datos, así como a sus componentes y asociaciones.

En el capítulo dos (Arquitectura de los modelos conceptuales), se describe a los tre modelos conceptuales así como a las características que poseen. Se da un ejemplo para ilustrar el proceso de normalización y se dan algunas ventajas del modelo relacional.

En el capítulo tres (Opciones para manejar a base de datos), se dan las dos opciones para manejar a una base de datos, los sistemas de información y los sistemas manejadores de bases de datos, así como a sus principales características.

En el capítulo cuatro (Sistema manejador de bases de datos relacional DATACOM/DB), se da la descripción general de los componentes,estructura y facilidades del sistema manejador de bases de datos relacional (DATACOM/DB).

Para finalizar en el capítulo cinco (Conclusiones), se dan las experiencias e impresiones de los usuarios de DATACOM/DB.

I. ¿ Qué es una base de datos ?

I.1 Panorama histórico de las bases de datos.

Para poder responder a la pregunta ¿ Que es una base de datos ? echaremos un vistazo a la historia, la que esbozaremos mediante cuatro etapas.

I.1.1 Primera etapa (antes de 1960)

En las organizaciones más sencillas, encontramos casi siempre una colección de registros organizados para una sola aplicación determinada y en la mayoría de las bibliotecas de cintas o discos magnéticos anteriores al advenimiento de la técnica de las bases de datos, se encuentra una sorprendente cantidad de datos duplicados o redundantes. Muchos de los datos se hallan simultáneamente almacenados en varios volúmenes con distintas finalidades y también con diferentes fechas de actualización. La redundancia de datos da origen a diversos inconvenientes : en primer lugar tenemos el costo adicional del almacenamiento de copias múltiples de los mismos datos, en segundo lugar tenemos el grado de actualización de los datos; por ejemplo, para actualizar por lo menos una parte de las copias redundantes es preciso recurrir a múltiples operaciones de actualización, y por último, debido a que las distintas copias pueden hallarse en diferentes estados de actualización, el sistema tiende a proporcionar informaciones contradictorias, es decir, la información obtenida no es muy confiable. Otro de los inconvenientes en los sistemas de archivos tradicionales es que éstos no cuentan con la independencia de datos

tanto lógica como física. Por independencia lógica de datos se entiende que la modificación a la estructura general de los datos no afecta a los programas de aplicación (el cambio, desde luego, no debe eliminar ninguno de los datos que el programa necesita). Por independencia física de los datos se entiende que pueden modificarse la distribución y organización física del los datos sin afectar ni a la estructura lógica general ni a los programas de aplicación. Como los programas de aplicación en estos sistemas se ajustan a los formatos de los archivos, ante cualquier modificacion a la estructura de los datos, así como, la de los dispositivos de almacemamiento, el programador de aplicaciones se ve obligado a reescribir los programas de aplicación que fueron afectados por los cambios y repetir nuevamente los procesos de compilación y ejecución.

Las principales características de los sistemas de archivos tradicionales de esta época se listan a continuación.

- Los archivos se organizaban de modo secuencial simple.
- La estructura de los archivos lógicos de datos se presenta esencialmente igual a la estructura física.
- Del mismo archivo existen varias copias ya que se guardan las generaciones anteriores de datos.
- Los sistemas de programación (Software), se ocupan solo de las operaciones de entrada y salida.
- El programador de aplicaciones diseña la distrbución física de los datos y la incorpora a los programas de aplicación.
- Si se cambia la estructura de los datos o los dispositivos de almacenamiento, los programas de aplicación deben volver a escribirse, recompilarse y probarse.

- Los datos se diseñan y optimizan, por lo general, para una única aplicación. De ahí que los mismos datos difícilmente se intercambian entre aplicaciones.
- Alto nivel de redundancia entre los archivos de datos.

De todo lo anteriormente mencionado se deduce que los sistemas de programación de esta época dejan mucho que desear, motivo por el cual se hace inmediato pensar en un paquete de sistemas de programación que sea más eficaz para dar solución a la problemática ya descrita, si no en su totalidad, por lo menos en parte.

I.1.2 Segunda etapa (decenio 1960-1970)

Al principio de esta etapa la expresión " bases de datos " comenzó a popularizarse y un hecho relevante de esta época es la naturaleza cambiante de los archivos y de los dispositivos de almacenamiento de manera que el programador no fuese afectado por estos cambios.

Los programadores hicieron posible la modificación de la distribución física de los datos sin que por ello se alterase su estructura lógica, siempre que no se introdujeran cambios en los contenidos de los registros ni en la estructura fundamental de los archivos.

Los archivos utilizados en esta época están por lo general diseñados como los de la primera etapa, pero ya se empieza a

distinguir la independencia lógica de la física de los datos aunque de una manera muy elemental.

Las principales características de los sistemas de archivos de la segunda etapa, se resumen a continuación.

- El acceso a los archivos es secuencial o directo (al azar) a los registros (no a los campos).
- El procesamiento se hace por lotes, en línea, o en tiempo real.
- Se distingue la organización lógica de la organización física, pero las distinciones entre ellos son bastante elementales.
- Pueden cambiarse las unidades de almacenamiento sin necesidad de modificar los programas de aplicación.
- Las estructuras de datos son por lo general de los tipos secuencial, secuencial indexado, o de acceso directo simple.
- Por lo general no hay recuperación por llave (clave) múltiple.
- Se admiten ciertos recursos de seguridad, pero todavía dejan mucho que desear .
- Todavía hay tendencia al diseño y optimización de los datos principalmente para una única aplicación.
- Existe todavía mucha redundancia en los datos.
- Cuando se usan estructuras jerárquicas, el programador tiene que construir, por lo común, las relaciones de padre a hijo.
- Los sistemas de programacion proveen " Métodos de acceso ", pero no " Administracion de datos ".

Con todo lo mencionado en esta etapa, los archivos y los sistemas de

programación al igual que los de la primera etapa tienen todavía muchos inconvenientes. Motivo por el cual se fabrican sistemas de programación más eficaces para seguir resolviendo los inconvenientes existentes a lo largo de este decenio.

1.1.3 Tercera etapa (decenio 1970-1980)

Durante esta época, con la evolución del procesamiento de datos comerciales, se comprende que era conveniente independizar los programas de aplicación, de las eventuales adiciones hechas a los datos ya almacenados, tales como nuevos campos, nuevas relaciones más bién que a los cambios en los dispositivos de almacenamiento (Hardware) y de las consecuencias del aumento de volumen de los archivos. Por lo tanto la estrutura de la base de datos tendrá que ser modificada para mejorar su rendimiento o para permitir otros tipos de averiguación. Cambiaron así mismo las necesidades de los usuarios y los tipos de averiguación que les interesaban. Si se prentendía que fueran agregados nuevos elementos de datos a los registros sin que esto afectara a los programas de aplicación, era preciso que los programas se refieran a los datos a nivel de campo más bién que a nivel de registro.

De los mismos datos pueden derivarse múltiples archivos lógicos. Las aplicaciones que tienen diferentes requerimientos deben poder accesar a los mismos datos de diferentes maneras.

Como ocurre a menudo cuando un nuevo término se pone de moda, no faltaron quienes quisieron promover de categoría a sus archivos llamándolos bases de datos sin preocuparse por cambiar su naturaleza, como hubiera sido necesario para dotarlos de características de no redundancia, independencia de datos, interconectividad, protección de seguridad, y, en muchos, acceso al tiempo real.

Al crecer el tamaño de la base de datos, se hizo inevitable el cambio de la estructura lógica general. Resultó importante que esa estructura general pudiera cambiar sin forzar el cambio de los muchos programas de aplicación que la utilizaban. En muchos sistemas, los cambios de la estructura lógica general de los datos son un modo de ser : la evolución es permanente. Por esta razón, se necesitan dos niveles de independencia de datos : la independencia lógica y la independencia física de los datos.

Los sistemas de programación de este período tenían la finalidad de que los datos fueran compartidos por las diferentes aplicaciones, es decir, de las bases de datos se dedujeran múltiples archivos lógicos de acuerdo a las necesidades de los programadores en sus respectivos programas de aplicación. Ademas ya se contempla la independencia lógica de la física de los datos de una manera más compleja con respecto a la segunda etapa.

Las características predominantes de las primeras bases de datos al principio del período 1970-1980 se dan a continuación.

- De los mismos datos físicos se derivan múltiples bases de

datos lógicas.

- Se puede tener acceso a los mismos datos de diferentes maneras, según los requisitos de la aplicación.

Los sistemas de programación proveen los medios para disminuir la redundancia.

Las distintas aplicaciones comparten los mismos elementos de datos.

- La ausencia de redundancia facilita la conservación de la integridad de los datos.

- La organización del almacenamiento físico es independiente de los programas de aplicación. Puede cambiarse a menudo para mejorar el desempeño de la base de datos sin modificación de los programas de aplicación.

- Los datos son direccionables en los niveles de campo y de grupo.

- Es posible la recuperación por llaves múltiples.

- Se utilizan formas de organización de datos muy complejas sin que ello se refleje en los programas de aplicación.

Con lo mencionado en la presente etapa, notamos que los sistemas de programación de este período deben de ser más refinados, por lo que, la cuarta etapa de la evolución de las bases de datos tiene su razón de ser.

I.1.4 Cuarta etapa (requisitos actuales para los sistemas de bases de datos)

Los sistemas de programación actuales proporcionan la vista lógica global de los datos. Esta vista puede ser enteramente distinta de la que ofrece la estructura física y de la propia de los programas

de aplicación. Los sistemas de programación de las bases de datos se encargarán, en efecto, de convertir la vista que el programador de aplicaciones tiene de los datos en la vista lógica global, y transformará luego esta vista lógica global en la representación física. Además de permitir la máxima libertad para cambiar las estructuras de los datos sin tener que rehacer mucho de lo ya hecho en la base de datos.

Las principales características de los sistemas de bases de datos actuales se mencionan en seguida.

- Los sistemas de programación (Software) procuran la independencia lógica y física de los datos, permitiendo que exista una vista global independientemente de ciertos cambios en las vistas de los programas de aplicación o de la distribución física de los datos.

- Los datos pueden evolucionar sin que se incurra en costos de mantenimiento excesivos.

-Se proveen medios para que un administrador de datos actúe como controlador y custodio de los mismos y asegure que la organización de los datos sea siempre la mejor para los usuarios en general.

- Se proveen procedimientos eficaces para controlar el secreto, la seguridad y la integridad de datos.

- Se utilizan archivos invertidos en algunos sistemas para permitir una rápida exploración de la base de datos.

- Las bases de datos se diseñan de modo que puedan proveer respuestas aún a tipos de averiguación no previstas por el diseñador.

- Se facilita la migración de datos.

- Los sistemas de programación proveen un lenguaje para la descripción de datos al administrador de datos, un lenguaje de órdenes para el programador de aplicaciones, y a veces un lenguaje de interrogación para el usuario externo (" QUERY language ").

1.2 Definición de una base de datos

Una base de datos se define como una colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es la de servir a una o más aplicaciones de la mejor manera posible; los datos se almacenan de manera que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir datos nuevos y para modificar o extraer los datos almacenados.

Cuando se habla de una base de datos ésta se describe desde dos puntos de vista : el conceptual y el de implantación o instrumentación. Desde el punto de vista conceptual se describen las entidades, atributos, claves y las asociaciones que conforman a la base de datos. Desde el punto de vista implantación se describe la organización y distribución física de los datos en las unidades de almacenamiento.

Las entidades son las cosas sobre las cuales se almacena información; una entidad puede ser un objeto tangible, tal como, un alumno, una pieza o un artículo. Pero también puede ser algo intangible, por ejemplo, un suceso, un nombre de proyecto, el número

de cuenta de un estudiante, o un concepto abstracto.

Toda entidad tiene propiedades que actualmente conviene registrar, tales como, número de cuenta de un estudiante, nombre de un estudiante, dirección del estudiante, semestres cursados por el estudiante,ect. A menudo, en el procesamiento de datos, nos interesan las colecciones de entidades similares, por ejemplo, el personal académico de la UNAM, y necesitamos registrar información acerca de las mismas propiedades de cada una de ellas. A estas colecciones de entidades similares las llamamos conjuntos de entidades y a sus propiedades o características les llamamos sus atributos. Por lo común se mantiene un registro para cada entidad y agrupamos en conjuntos de registros de entidad todos los registros pertinentes a entidades similares. Los registros se refieren a atributos de las entidades y contienen los valores de estos atributos. Los atributos registrados en relación con un estudiante pueden incluir el número de cuenta, nombre del estudiante, clave de la materia que cursa, créditos de la materia, clave del profesor que imparte la materia, nombre del profesor,etc.

Se denomina clave (llave) al atributo o conjunto de atributos que la computadora utiliza para identificar un registro (o una eneada).

Clave primaria maestra es aquel atributo que identifica unívocamente un registro o una eneada, es decir, el identificador de entidad formado por uno o más atributos. Esta clave es de gran importancia porque la utiliza la computadora para localizar el registro

o la eneada por medio de un índice o un algoritmo de direccionamiento.

Clave secundaria es aquel o aquellos atributo (s) que no identifican registros únicos, sino todos aquéllos que tienen cierta propiedad y que pueden ser claves primarias para otra(s) entidad(es).

En la Fig.1 se muestra las características de la entidad ESTUDIANTES en ésta se señala los nombres de atributos, valores de atributos, claves primarias y secundarias, y una ocurrencia del registro.

Nombre de atributos	Nro-cta	Nombre	Cve-mat	Desc-mat	Cred-mat	Cve-prof	Nomb-profr
Valor del atributo	8523988-3	Martinez Radl	3051	Algebra superior I	010	0125	Ivan Flores
	8423505-1	Castro Hugo	3023	Geometria analitica	010	0072	Enrique Alvarez
Ocurrencia	8576597-9	Romero Rosa	0191	Cálculo dif.Intrg I	010	0135	Celia Montero
	8657465-3	Castro Juan	0065	Algebra Lineal I	010	0089	Mario Pineda
	8576894-4	García Javier	0065	Gemetria Moderna I	010	0089	Carlos Zamora

Fig.1 Atributos de la entidad ESTUDIANTES. En esta figura el atributo Nro-cta es la llave primaria, los atributos Cve-mat y Cve-profr son llaves secundarias.

Un dato es una representacion pequeña del mundo real, una entidad. Una base de datos representa a varias entidades de interés para una familia de aplicaciones. Estas entidades pueden ser diferentes y diversas, pero un tanto relacionadas.

La base de datos refleja estas relaciones las cuales existen dentro de la(s) entidad(es) que será(n) utilizada(s). La relación describe la menera en la cual cada uno de los datos se asocian entre sí.

Las relaciones se clasifican de acuerdo a sus características en las siguientes :

1. Relación simple (1 <-----> 1).
2. Relación de uno a muchos (1 <----- >> m).
3. Relación de muchos a muchos (m <<------>> n).

1. Relación simple (1 <-----> 1) es aquélla en que un atributo de una entidad le corresponde un solo atributo de la misma o de otra entidad y viceversa.

La Fig.2 representa los atributos de la entidad ESTUDIANTES en la cual el atributo NUMERO-CUENTA está en relación simple con el atributo NOMBRE-ESTUDIANTE y viceversa. Ya que, el número de cuenta del estudiante tiene asignado de manera única el nombre de éste y viceversa. A esta relación la representamos como : NUMERO-CUENTA <------> NOMBRE-ESTUDIANTE.

Entidad : ESTUDIANTES

NUMERO-CUENTA	NOMBRE-ESTUDIANTE	DIRECCION-ESTUDIANTE	SEXO

Fig.2 Relación simple entre los atributos NUMERO-CUENTA y NOMBRE-ESTUDIANTE de la entidad ESTUDIANTES.

2. Relación de uno a varios (1 <-----> m) es aquélla en que un atributo de una entidad le corresponden varios atributos de la misma entidad o de otra. Para mostrar esta relación consideremos a una base de datos educacional, en la que algunas de sus entidades pueder ser ESTUDIANTES, MATERIAS, PROFESORES, etc.

los atributos asociados a la entidad ESTUDIANTES puden ser : NUMERO-CUENTA, NOMBRE-ESTUDIANTE, DIRECCION-ESTUDIANTE, SEXO, etc.

Algunos de los atributos asociados con la entidad MATERIAS pueden ser : CLAVE-MATERIA, NOMBRE-MATERIA, CREDITOS-MATERIA,HORAS-MATERIA, etc.

Algunos de los atributos asociados con la entidad PROFESORES pueden ser : CLAVE-PROFESOR, NOMBRE-PROFESOR, DIRECCION-PROFESOR, HORAS-TRABAJO y SALARIO.

Lo natural en una institución educacional, es que, los alumnos inscritos en un semestre determinado cursan varias materias. Así tenemos, por ejemplo, que la relación que hay entre las entidades ESTUDIANTES y MATERIAS es de uno a muchos, la cual se ilustra en el diagrama de la Fig.3 y se representa por ESTUDIANTES <-----> MATERIAS.

Nro-cuenta	Nomb-estudiante	Dirección-estudiante	Cve-mat	Descripción-materia	Cred-mat
8615685-1	Rodríguez Carlos	16 de Septiembre México D.F	0191	Cálculo Dif. Integr I	010
			0087	Geometría Moderna I	010
			0097	Algebra Superior I	010
			0065	Algebra Lineal I	010
8514873-3	Martínez David	Av. Universidad 1258 San Angel	0244	Geom. Analítica I	010
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Fig.3 La relación que hay entre las entidades ESTUDIANTES y MATERIAS es de uno a muchos.

3. Relación de muchos a muchos (m <<------>> n) es aquélla en la que un atributo de una primera entidad le corresponden varios atributos de una segunda entidad, y recíprocamente, un atributo de la segunda entidad le corresponden varios atributos de la primera entidad. Para describir esta relacion consideraremos la entidad ESTUDIANTES y la entidad MATERIAS de nuestra base de datos educacional descrita anteriormente. Lo común en esta base de datos educacional es que un estudiante está inscrito en varias materias y cada materia tiene varios alumnos, por lo que la relación que hay entre las entidades ESTUDIANTES y MATERIAS es de muchos a muchos, a tal relación la ilustramos en la Fig.4. y se representa por ESTUDIANTES <<----->> MATERIAS.

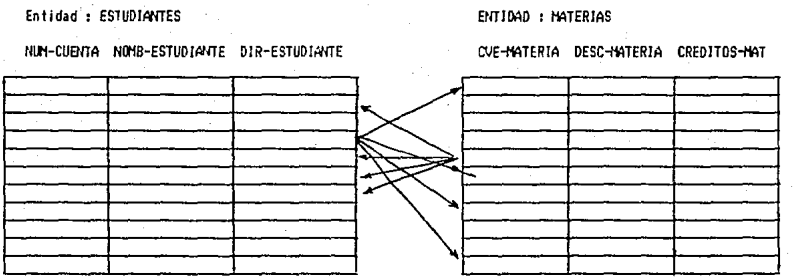


Fig.4 Relación de muchos a muchos entre las entidades ESTUDIANTES y MATERIAS.

Diferencias entre la organización lógica y la organización física de los datos.

Organización lógica	Organización física
La simplicidad es importante.	La organización compleja suele ser ventajosa. Los sistemas de programación ocultan esta complejidad a los usuarios.
Los requerimientos de los programas de aplicación se ajustan a la estructura lógica de los datos.	Los requerimientos de los programas de aplicación pueden referirse a datos de formas totalmente diferentes

	de las de almacenamiento de los datos.
La independencia de los datos es de principal importancia.	La independencia de los datos carece de importancia si se proveen medios para reestructurar los datos físicos sin alteración del esquema.
El empleo eficiente del almacén es de escasa importancia	El empleo eficiente del almacén es un factor de la mayor importancia para la selección de la técnica de representación y distribución física de los datos.
Las características de la unidad de almacenamiento no deberían afetcar la organización lógica.	Las características de la unidad de almacenamiento, tales como la lpngitud de pista y el tamaño de volumen, son factores determinantes para la organización del almacenamiento.
En los archivos lógicos	La eliminación de la

existe a menudo un alto grado de redundancia.	redundancia es uno de los objetivos de la organización física.
Las inserciones y eliminaciones no tienen por lo general efetco sobre las estructuras lógicas.	Las inserciones y eliminaciones suelen tener un gran efetco sobre la distribución física de los datos, especialmente en el caso de los archivos volátiles.
Los medios de direccionamiento de los datos no tiene mayor efetco sobre las estructuras lógicas.	Las técnicas de direccionamiento tienen un gran efetco sobre la distribución del almacenamiento físico.
Las claves secundarias no tienen mayor efetco sobre las estructuras lógicas.	Las claves secundarias y los archivos invertidos son factores de importancia para la estructura física.
La secuencia de los registros es la de las claves primarias, o las secuencias en que el programa de aplicación las usa.	La secuencia de los registros se elige a partir de otras consideraciones, tales como la abreviación de los

	tiempos de acceso o la reduccion del tamaño de los indices.
Las relaciones entre registros pueden representarse mediante simples (apuntadores) jerárquicos.	Complejos conjuntos de apuntadores son a menudo indispensables para vincular registros de datos fisicamente separados.
La protección contra fallas o pérdida de los datos no afetca por lo general a la organización lógica.	Las técnicas para reanudación y recuperación de archivos suelen plantear requisitos fisicos especiales, tales como la duplicación de parte de los datos.
La organización lógica debe ser estable, de modo que no haya necesidad de reduplicar los programas.	La distribución fisica puede ser cambiante, prevista para periódicas reorganizaciones, afinaciones y posiblemente migración de datos.

II. Arquitectura de los tres modelos conceptuales de bases de datos.

Como es sabido en el medio informatico, existen tres modelos conceptuales en los sistemas administradores de bases de datos. Los cuales, dependiendo de las estructuras utilizadas para representar los datos, se clasifican en Relacional, Jerárquico y de Redes.

II.1 Arquitectura del modelo relacional

Una relación es cualquier subconjunto del producto cartesiano de uno o varios dominios, entendiendo a un dominio simplemente como un conjunto de valores. Las componentes de la relación se llaman eneadas o renglones.

La relacion la podemos ver también como una tabla, en la que los renglones representan una ocurrencia de la relación, y las columnas sus atributos. En la Fig.5 se muestra a una relación (tabla) de profesores que pertenecen a una determinada institución educacional, así como algunos atributos (dominios), que se relacionan con ellos.

Cve-Profr	Nomb-Profr	Dirección-Profesor	Categoría-Profr	Hras-Trabajo	Salario-Profesor
8475	S. Ramírez	Congreso 143 Tlalpan D.F.	Definitivo	48	497587
7812	R. Cedillo	Copilco 123 San Angel	Definitivo	28	363872
8146	A. Solís	Miramontes 28 Contreras	Interino	15	204858
8089	D. Romero	5 de Mayo Centro	Definitivo	38	381225

Fig.5 Las columnas (atributos) de la relación PROFESORES representan los dominios de la relación en la que una eneada (renglón) es una ocurrencia de ésta.

La base de datos construída por medio de relaciones es una base de datos relacional, es decir, una base de datos es, por lo tanto, relacional cuando está construída con matrices planas de módulos de datos.

Cuando una tabla tiene n columnas, se dice que la relación es de grado n. Las relaciones de grado 2 se llaman binarias, las de grado 3, ternarias, y las de grado n, enearias.

II.1.1 Manipulación de relaciones.

Se basa sobre los conceptos de álgebra relacional y cálculo relacional (cálculo de predicados)

El álgebra relacional la definimos como un conjunto de operaciones sobre las relaciones. Cada operación toma una o más relaciones como su (s) operando (s) y produce otra relación como resultado.

El álgebra relacional se compone de dos grupos de operadores: los operadores tradicionales de la teoría de conjuntos, a saber, la unión, la intersección, la diferencia y el producto cartesiano (todos un poco modificados para operar con relaciones en lugar de conjuntos arbitrarios), y los operadores relacionales especiales de selección, proyeccion y reunión.

Para todas las operaciones tradicionales con excepción del producto cartesiano, las dos relaciones operando deben de tener el mismo grado.

En la presente sección se discuten las operaciones para manipular una o más relaciones. Estas operaciones tienen la finalidad de recuperar, alterar, y adicionar información acerca de las entidades a nuestra base de datos.

Las operaciones que discutiremos son : Unión, Intersección, Diferencia, Producto cartesiano, Seleccion, Proyección y Ligado ("Join").

Las relaciones a las cuales se les aplicará las operaciones mencionadas deben de estar contenidas en la relación universo, la cual se forma con todas las combinaciones posibles de los valores de los atributos (dominios).

La unión de las relaciones R1 y R2 contenidas en la relación universo, U, se forma con la combinación de todas las eneadas pertenecientes a cada una de las relaciones, contando solamente una vez las eneadas que están tanto en R1 como en R2. La operación unión se representa con el símbolo U.

Una definición mas formal de la operación unión es :

Si R ∈ R1 U R2 ======> R ∈ R1 V R ∈ R2.

Donde ∈ significa " ser miembro de " y V significa " o ". En otras palabras, una eneada pertenece a la unión de dos relaciones, si la eneada está en cualquiera de las dos relaciones o en ambas antes de

efectuarse la unión, como se ilustra en la Fig.6 a).

La intersección de las dos relaciones R1 y R2 que están contenidas en la relación universo U, se forma con las eneadas comunes de ambas relaciones. La operación intersección se representa con el símbolo $\cap$.

Una definición más formal de la operación intersección es la siguiente :

$R \in R1 \cap R2 \Longrightarrow R \in R1 \cap R \in R2$.

Donde $\cap$ se lee como "y". Es decir una eneada R está en la intersección de las relaciones R1 y R2, cuando la eneada R se encuentra en ambas relaciones, antes de efectuar la intersección, como se ilustra en la Fig.6 b).

La diferencia de las relaciones R1 y R2, representada por R1-R2, consiste de todas las eneadas de R1, eliminando las eneadas de R1 que se encuentran en R2. El resultado se representa simbólicamente como :

$R \in R1-R2 \Longrightarrow R \in R1$ y $R \notin R2$, tal como se ilustra en la Fig.6 c).

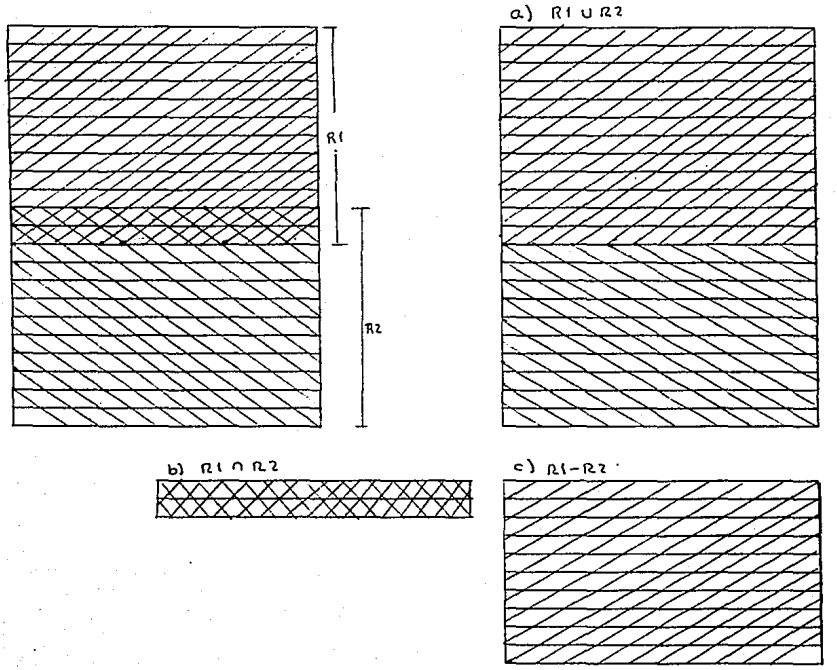


Fig.6 Operaciones de unión, intersección y diferencia de las relaciones R1 y R2.

El producto cartesiano de las dos relaciones P y Q representado por P X Q, es el conjunto de todas las eneadas r, tales que r es la concatenacion de la eneada p ∈ P y la eneada q ∈ Q. La concatenación de una eneada p = (p1,p2,p3,...,pm) y una eneada q = (qm+1,qm+2,qm+3,...,qm+n) en ese orden es la eneada r = (p1,p2,p3,...,pm,qm+1,qm+2,qm+3,...,qm+n).

La operación de selección examina todas las eneadas de una relación R y extrae solamente aquéllas en que su valor de atributo coincide con el valor del atributo que fue proporcionado por el usuario. Esta operación la podemos simbolizar como :

Select R (A=x)=Rx, esto representa todas las eneadas para las cuales el atributo A es igual a x de la relación R ; el conjunto resultante o relación se denota Rx. Para mostrar el funcionamiento de esta operación consideremos la relación de la Fig.7 en la que los atributos NUMERO-CUENTA, NOMBRE, SEXO, CALIFICACION Y CLAVE-MATERIA la conforman. La operación ejecutada sobre esta relación es seleccionar con la llave CLAVE-MATERIA 015 de la materia R.

Nro-cta	Nombre	Sexo	Calificación	Cve-materia
8612117-3	González heriberto	M	10	024
8514132-1	Zapata José	M	10	015
8412156-1	García Margarita	F	10	009
8538975-4	Cruz Leticia	F	08	015
8650466-3	Hernández Ráquel	F	10	010
8514465-2	García Armando	M	08	010
8641534-8	Sandoval Carlos	M	10	024
8417789-1	Zamora Eduardo	M	08	017
8510154-6	Cano Asunción	F	10	015
8621341-6	Romero Patricia	F	08	022

Fig.7 La relación R está formada con los atributos NRO-CTA, NOMBRE, SEXO, CALIFICACION y CVE-MATERIA.

Esta selección la representamos como sigue :

Select R(CLAVE-MATERIA=015) y la relación resultante se muestra en

la Fig.8.

Nro-cta	Nombre	Sexo	Calificación	Cve-materia
8514132-1	Zapata José	M	10	015
8538975-4	Cruz Leticia	F	08	015
8510154-6	Cano Asunción	F	10	015

Fig.8 Relación solución de la operación
Select R (Cve-materia = 015) = R 015 de la relación
que se presenta en la Fig.7.

La operación proyección simbolizada con el operador Π actúa sobre una relación dada para reducir el número de atributos que ésta contiene, eliminando las duplicadas y por lo tanto creando una nueva relación (subrelación) de menor grado.

Para mostrar un ejemplo de la operación proyección, consideremos la relación ALUMNO de la Fig.9, que también la podemos representar por la expresión ALUMNO (NUMERO-CUENTA, NOMBRE, SEXO, CLAVE-MATERIA, DESC-MATERIA Y CRED-MATERIA), la cual se proyecta mediante el enunciado ALUM = Π ALUMNO (NUMERO-CUENTA, NOMBRE, SEXO, CLAVE-MATERIA). Otra proyección de la relación ALUMNO podría ser el enunciado MATERIA = Π ALUMNO (CLAVE-MATERIA, DESC-MATERIA, CRED-MATERIA). Ambas proyecciones se ilustran el la Fig.10.

Nro-cta	Nombre	Sexo	Cve-materia	Descr-materia	Cred-materia
8645138-2	Maldonado Guilermo	M	0191	Cal. Dif. Intgr I	010
8713256-3	Garrido Ignacio	M	0244	Geom. Analitica I	010
8525835-1	Palomares Carolina	F	0087	Geom. Moderna I	010
8611136-5	Navarro Carlos	M	0065	Algebra Lineal I	010

Fig.9 Relación ALUMNO.

Nro-cta	Nombre	Sexo	Cve-materia
8645138-2	Maldonado Guilermo	M	0191
8713256-3	Garrido Ignacio	M	0244
8525835-1	Palomares Carolina	F	0087
8611136-5	Navarro Carlos	M	0065

Cve-materia	Descr-materia	Cred-materia
0191	Cal. Dif. Intgr I	010
0244	Geom. Analitica I	010
0087	Geom. Moderna I	010
0065	Algebra Lineal I	010

Fig.10 Las relaciones ALUM y MATERIA son las resultantes al proyectar la relación ALUMNO.

ALUM = TT ALUMNO (Nro-cta, Nombre, Sexo, Cve-materia).

MATERIA = TT ALUMNO (Cve-materia, Desc-materia, Cred-materia).

La operación ligado("JOIN") es la inversa de la operación proyección y tiene como finalidad agrupar simultáneamente dos o más relaciones para formar una nueva relación. Para poder agrupar a estas relaciones se necesita un hecho para correlacionar una eneada de la relación P a una o más eneadas de la relación Q. Este hecho es el atributo común que resulta cuando las relaciones se traslapan en un solo atributo, pudiendo ser el traslape en más de un atributo al cual

llamamos sub-eneada común.

La expresión R = "JOIN" P,Q (C1 rel C2) nos representa la sintaxis de la operación "join" la cual actúa sobre las relaciones P y Q para producir la nueva relacion R.

El criterio para seleccionar eneadas de cada relación se da mediante la expresión "rel" que se encuentra dentro del paréntisis y que consiste de los operadores de comparación aritmética como son : $<$, $=$, $>$, $\leq$, $\neq$, $\geq$.

Para ejemplificar el funcionamiento de la operación "JOIN" consideremos, las relaciones ALUM y MATERIA que se encuentran en la Fig.10 para las cuales se les aplica la operación "JOIN" en la que el atributo común C1 y C2 de ambas relaciones es CLAVE-MATERIA y el operador de la comparacion es la igualdad. La expresión para este ejemplo es :

R = JOIN ALUM, MATERIA (C1=C2).

Donde la relacion resultante es la relación original (ALUMNO).

El cálculo relacional fue ideado por E.F. Codd para lenguajes relacionales en los cuales, el usuario se limita a definir el resultado que desea y deja al sistema el decidir que operaciones se requieren para obtener el resultado a partir de la base de datos.

Un aspecto fundamental en la definición de los operadores que utiliza el cálculo relacional y de los lenguajes basados en él es,

la noción de variable de eneada 'libre' y 'acotada'. La variable 'libre' es análoga al de una variable global en un lenguaje de programación, esto es, una variable definida fuera del procedimiento actual. Variable 'acotada' es semejante a una variable local, es decir, una variable que se define en el procedimineto actual.

Las fórmulas del cálculo relacional son de la forma
(t/Ψ(t)), donde t es una variable de eneada de alguna longitud fija, y Ψ es una fórmula construída de átomos y una colección de operadores, los cuales se define en seguida.

los átomos de fórmulas son de tres tipos:

1. R(s), donde R es nombre de una relación y s es una eneada variable. Este átomo se coloca para asegurar que la eneada s está en la relación R.
2. s[i] θ u[i], donde s y u son eneadas variables y θ es un operador aritmético de comparación (=,≠,<,>,≤,≥). Este átomo se coloca para asegurar que la i-ésima componente de s está en la relación θ con la i-ésima componente de u.
3. s[i] θ a y a θ s[i] son como las el número 2, y a es una constante. El primero de estos átomos asegura que la i-ésima componente se s está en relación θ con la constante a, y el segundo átomo tiene un significado análogo.

Los operadores de fórmulas se construyen a partir de operadores aritméticos, operadores booleanos (AND o OR) y cuantificadores (∀,∃) de acuerdo a las reglas que se presentan a continuación.

1. Toda condición es una fórmula
2. Si f es una fórmula también lo son (f) y NOT(f)
3. Si f y g son fórmulas, también lo son (f AND g) y (f OR g)
4. Si f es una fórmula, en la cual T aparece como variable libre, entonces ∃T(f) y ∀T(f) también son fórmulas
5. Ninguna otra es fórmula

El álgebra relacional y el cálculo relacional son dos conceptos equivalentes para la manipulación de tablas (este teorema se encuentra en el libro: Principles of database systems de Jeffrey D. Ullman).

II.1.2 Normalizacion de relaciones

La normalizacion de relaciones es un proceso en el cual se clasifican relaciones, asociaciones entre objetos y así sucesivamente, en grupos, de acuerdo a las características que los grupos y las relaciones poseen. Si ciertas relaciones son satisfechas, se aplica una categoría ; si otras reglas se satisfacen, otra categoría se aplica.

Existen fundamentalmente tres propositos para la normalización de relaciones.

1. Eliminar anomalías en la actualización.
2. Simplificar la estructura de la base de datos para una mayor claridad.

3. Proporcionar una base para la estabilidad de la estructura de los datos.

Para dar cumplimiento a los propositos mencionados anteriormente, se buscan , a través de tres pasos, formas especificas para las relaciones.

11.1.2.1 Primera forma normal (1FN)

Se obtiene al eliminar todos los grupos repetitivos de la relación y colocándolos en otra nueva relación aparte debidamente nombrada. Los renglones de la nueva relación deben estar provistos de llaves candidatas (son llaves que identifican las eneadas de la relación) adecuadas para identificarlos unívocamente.

la finalidad de eliminar grupos repetitivos consiste en proteger a la base de datos diseñada inicialmente contra los futuros cambios de la misma. Si no se hiciera esto es muy posible que en una subsiguiente evolución de la base de datos se exija la eliminación de los grupos repetitivos posteriores a la redacción de los programas de aplicación que los utilizan, lo cual ocasionaría, que los programas de aplicación tendrán que ser profundamente modificados y sometidos a nuevas pruebas.

Para mostrar el proceso de normalizacón de relaciones en la primera forma normal, consideremos la relación ESTMATERIA ilustrada en la Fig.11, en la que se observa que el grupo de atributos CLAVE-MATERIA, DESC-MATERIA Y CRED-MATERIA es repetitivo. Este grupo repetitivo se

origina debido a que un alumno al inscribirse en un determinado semestre de alguna carrera impartida en alguna institición educacional, lo natural es que el alumno para ese semestre se inscriba en varias materias.

Para eliminar el grupo repetitivo de la relación ESTMATERIA mostrada en la Fig.11, se parte a ésta en las dos relaciones que se muestran en la Fig.12, con sus asociaciones y llaves para identificarlas unívocamente.

Relación : ESTMATERIA

Cve-Estudiante	Nomb-Estudiante	Dir-Estudiante	Cve-Materia	Desc-Materia	Cred-Materia

Fig.11 El grupo de atributos (Cve-Materia, Desc-Materia, Cred-Materia) de la relación ESTMATERIA es repetitivo.

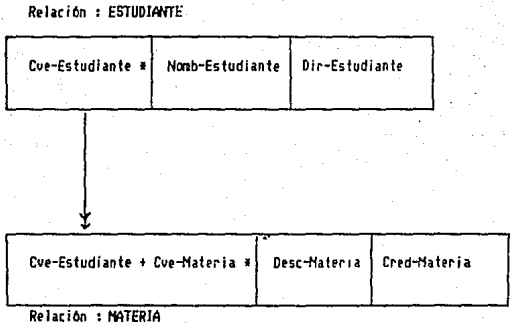


FIG.12 Tablas en la primera forma normal de la relación ESTMATERIA.

Nota : * identifica a la llave de la relación.
---)) significa asociación entre relaciones de uno a muchos.

II.1.2.2 Segunda forma normal (2FN)

Se basa en los conceptos de dependencia funcional y dependencia funcional completa entre atributos de una relación, por lo que es conveniente describir primero a estos conceptos.

II.1.2.2.1 Dependencia funcional

Al intentar dar las relaciones entre los módulos de datos, el

diseñador debe tratar de describir cuáles atributos dependen de cuáles otros.

La expresión "funcionalmente dependiente" se aclara del siguiente modo : el atributo B de la relación R es funcionalmente dependiente del atributo A de R si, en cada instante, cada valor de A está asociado con no más de un valor de B dentro de la relación R. Decir que B es funcionalmente dependiente de A es equivalente a decir que A identifica a B. En otros terminos, si en cualquier instante es conocido el valor de A, el valor de B queda determinado.

En la relación : ACTIVIDAD-PROFR (CVE-PROFR, NOMB-PROFR, SALARIO-PROFR, CVE-PROYECTO, FECH-TERMINO), las dependencias funcionales que se obtienen entre sus atributos son :

CVE-PROFR * es dependiente de NOMB-PROFR
NOMB-PROFR * es dependiente de CVE-PROFR
SALARIO-PROFR es dependiente de NOMB-PROFR o CVE-PROFR
CVE-PROYECTO es dependiente de NOMB-PROFR o CVE-PROFR
FECH-TERMINO es dependiente de NOMB-PROFR o CVE-PROFR
o CVE-PROYECTO.

Las dependencias funcionales quedan mejor comprendidas recurriendo a un diagrama, tal como el de la Fig.13 para la relación tomada como ejemplo. Si B es funcionalmente dependiente de A, dibujamos una flecha que va desde A hasta B. Los asteriscos que siguen a algunos nombres de atributos en esta figura distinguen a los atributos primos (atributos que son miembros a llaves candidatas).

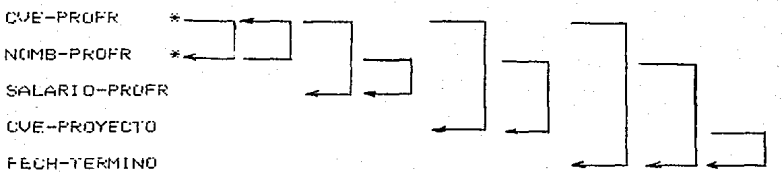


FIG.13 Dependencia funcional de la relación ACTIVIDAD-PROFR.

Es común que un atributo sea funcionalmente dependiente de un grupo de atributos, más bien que de uno solo. Tomemos por ejemplo la relación ACTIVIDAD-EST (CVE-EST, CVE-CURSO, NOMB-EST, NOMB-CURSO, GRADO-EST) mostrada en el diagrama de la Fig.14. En este diagrama se observa que el atributo GRADO-EST, de la relación ACTIVIDAD-EST es funcionalmente dependiente de la llave concatenada (CVE-EST + CVE-CURSO) o de las llaves candidatas(CVE-EST + NOMB-CURSO), (NOMB-EST + CVE-CURSO), o (NOMB-EST + NOMB-CURSO).

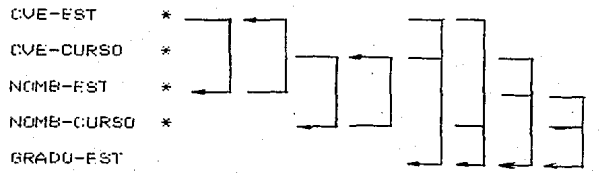


Fig.14 El atributo GRADO-EST de la tabla ACTIVIDAD-EST es funcionalmente dependiente de la clave concatenada (CVE-EST + CVE-CURSO).

II.1.2.2.2 Dependencia funcional completa

Se dice que un atributo o colección de atributos B, de una relación R es, "dependiente funcional completo" de otra colección de atributos A, de la relación R, si B es funcionalmente dependiente del total de A pero no de ningún subconjunto propio de A.

En la relación de la Fig.14, el atributo GRADO-EST es dependiente funcional completo de la llave concatenada(CVE-EST + CVE-CURSO).

Una relación R se halla en la segunda forma normal si está en la primera forma normal y cada uno de sus atributos no primos (atributos que no son candidatos a llave) es dependiente funcional completo de cada llave candidata de R. La relación de la Fig.14 se encuentra en la segunda forma normal porque su único atributo no primo, GRAD-EST, es dependiente funcional completo de cada llave candidata.

Cuando una relación no se encuentra en la segunda forma normal, el proceso para llevarla a la segunda forma normal, es mediente la separación de los atributos no primos de la relación que no son dependiente funcional completo de la llave candidata colocándolos en otra nueva relación provista de llaves adecuadas para identificarlos unívocamente.

La finalidad de este paso de normalización, es la de evitar redundancia de los atributos que dependen de un subconjunto de la llave candidata de la relación.

II.1.2.3 Tercera forma normal

Supongamos que A, B y C son tres atributos o tres colecciones de atributos de una relacion R. Si C es funcionalmente dependiente de B y B lo es de A, entonces C es funcionalmente dependiente de A. Si la correspondencia inversa no es simple, esto es, si A no es funcionalmente dependiente de B, o B no es funcionalmente dependiente de C, se dice que C es "transitivamente dependiente" de A. En el diagrama de la Fig.15 se ilustra como C es transitivamente dependiente de A.

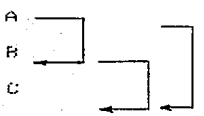


Fig.15 C es trasitivamente dependiente de A.

En la relación ACTIVIDAD-PROFR de la Fig.13 se observa que el atributo FECH-TERMINO es funcionalmente dependiente del atributo CVE-PROYECTO, y CVE-PROYECTO es funcionalmente dependiente del atributo CVE-PROFR. Por lo anterior FECH-TERMINO es transitivamente dependiente de la llave CVE-PROFR.

El proceso de convertir relaciones a la tercera forma normal es, precisamente, la eliminación de la dependencia transitiva de los atributos no primos de las llaves candidatas de la relación y colocandolos en otra nueva relación aparte provista de llaves adecuadas para identificarlos unívocamente.

Para convertir la relación de la Fig.13 a la tercera forma normal, se divide la relación en dos relaciones, como se muestra en la Fig.16.

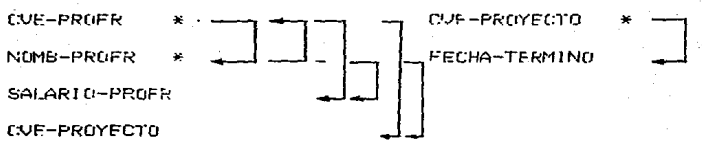


Fig.16 Relaciones que muestran la eliminación de la dependencia transitiva de la relación de la Fig.13.

La tercera forma normal la podemos definir como sigue : una relación (tabla) se encuentra en la tercera forma normal, si se halla en la segunda forma normal y cada uno de sus atributos no primos son dependientes trasitivos de cada llave candidata de la relación R.

El proceso de la normalización de relaciones, lo podemos resumir en el diagrama de la Fig.17. Es deseable que los datos estén en la tercera forma normal, porque, de lo contrario puden ocurrir anomalías en la actualización, si los atributos dependientes cambian.

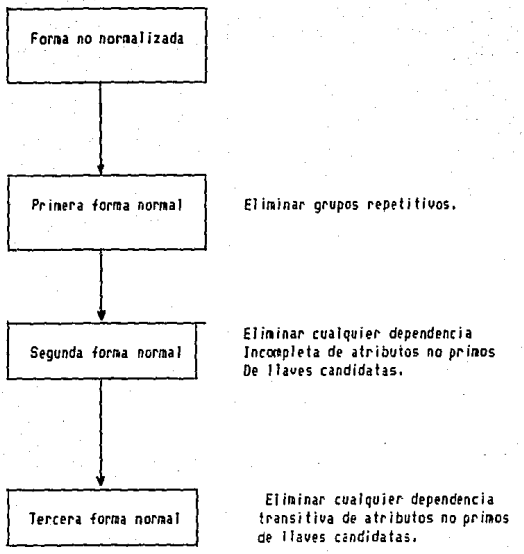


Fig.17 Pasos en el proceso de normalización de relaciones.

Cabe mencionar que ademas de las tres primeras formas normales para normalizar tablas, existen otras dos más, la cuarta forma normal y la quinta forma normal. Para nuestros propósitos con las tres primeras es suficiente, por lo que las restantes no se analizan (si hay interés en ver estas dos últimas formas normales, consultar el texto: Introducción a los sistemas de bases de datos, del autor: Date, de la editorial: Addison-Wesley Iberoamericana).

II.1.3 Ventajas al emplear tablas normalizadas.

Al emplear tablas en forma normalizada, se obtienen las siguientes ventajas :

1. Facilidad de uso. La manera más fácil de representar los datos a los usuarios que se inician en las bases de datos es a través de tablas bidimensionales.

2. Flexibilidad. Las operaciones de proyección y ligado (JOIN) permiten partir y traslapar relaciones de manera que se proporcione a los usuarios los archivos que necesitan en sus aplicaciones y en la forma en que los quieran.

3. Precisión. Las relaciones (tablas) tienen un significado preciso y pueden ser manipuladas con la matemática del álgebra o cálculo relacional.

4. Seguridad. Es más fácil instrumentar los controles de seguridad. Las autorizaciones de seguridad se refieren a relaciones. Los atributos más sensibles pueden ser trasladados a una relación aparte, con sus propios controles de autorización.

5. Relacionabilidad. Se tiene la máxima flexibilidad para relacionar atributos de diferentes conjuntos de eneadas o diferentes archivos.

6. Facilidad de instrumentación. El almacemaniento físico de los archivos planos es eventualmente menos complejo que el de los árboles o las estructuras de red (que se detallan más adelante). Los dispositivos capaces de llevar a la práctica la exploración rápida de los archivos son más factibles en el caso de los archivos que carecen de complejos sistemas de apuntadores.

7. Independencia de datos. las bases de datos están obligadas a crecer, a causa de la adición de nuevos atributos y nuevas relaciones. Se utilizarán, también, de distintas maneras. Se agregarán y eliminarán eneadas y módulos de datos. Si la base de datos es de la forma normalizada con independencia en los datos de los sistemas de programación (software), será posible estructurar los datos, y la base de datos crecerá sin llegar, en la mayoría de los casos, a volver a escribir los progrmas de aplicación.

II.1.4 Ejemplo del proceso de normalización de tablas

A continuación se da la descripción de un requerimiento correspondiente a la historia académica de un alumno en una determinada institución educacional, algunos de los datos y la relación (ilustrada el la Fig.18), que se asocian al requerimiento.

Requerimiento :

Dada una clave de estudiante producir un reporte de éste, mostrando los semestres en los cuales los cursos fueron tomados y los cursos en cada semestre.

Datos asociados al requerimiento

Nombre	Descripción
CVE-EST	Número de cuenta del estudiante
NOMB-EST	Nombre del estudiante
DIR-EST	Dirección del estudiante
NUM-SEM	Número de semestre
FECH-REG	Fecha de inscripción del estudiante
CVE-CURSO	Clave del curso

NOMB-CURSO	Nombre del curso
CVE-PROFR	Clave del profesor
NOMB-PROFR	Nombre del profesor
HRS-TRABAJO	Horas impartidas por el profesor
GRADO-EST	Grado obtenido al término del curso

Tabla asociada al requerimiento :

Tabla asociada al requerimiento

Cve-Est	Nomb-Est	Dir-Est	Num-Sem	Fech-Reg	Cve-Curso	Nomb-Curso	Cve-Profr	Nomb-Profr	Hrs-Imp	Grado-Est

Fig.18 Tabla (relación) asociada al requerimiento pedido.

Consideraciones del requerimiento

En el proceso de normalización de tablas es fundamental tener en cuenta los supuestos, ya que sin ellos no se podría normalizar a éstas. De acurdo a esto tenemos los siguientes supuestos para nuestro ejemplo de normalización: como es natural en una institución educacional, el alumno al inscribirse en una determinada carrera debe de acumular un número determinado de créditos ; para lo cual se divide la carrera en semestres y por cada semestre el alumno se inscribe en algunas materias correspondientes a aquélla hasta completar el total de créditos requeridos en la carrera.

Con respecto a lo anterior, se observa que la tabla asociada al requerimiento se encuentran campos repetitivos como lo son : NUM-SEM, FECH-REG, CVE-CURSO, NOMB-CURSO, CVE-PROFR, NOMB-PROFR, HRS-TRABAJO y GRADO-EST.

El objetivo que se busca con el requeriminto, es mostrar el proceso de normalizacion a la tabla (sin normalizar) mostrada en la Fig.18 asociada al requerimiento.

De acuerdo al proceso de normalización de tablas, nuestro primer paso es llevar la tabla (sin normalizar) a su primera forma normal, es decir, eliminar grupos repetitivos de la tabla original, como se ilustra en la Fig.19.

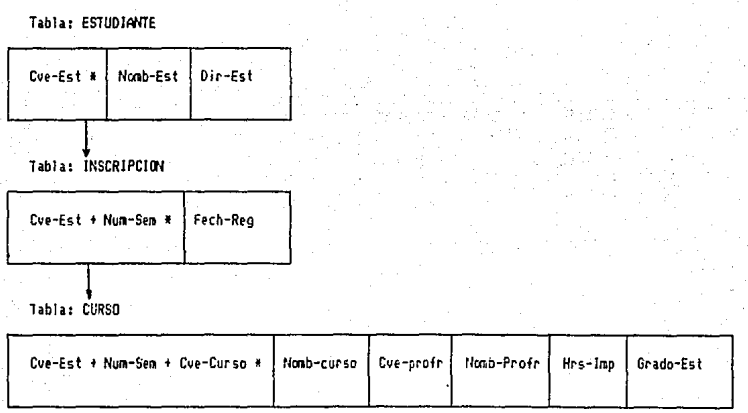


Fig.19 Tablas asociadas a la tabla de la Fig.18 en la primera forma normal.

Las tablas que se ilustran en la Fig.19, se encuentran en la primera

forma normal. Para continuar con nuestro proceso de normalización, a estas tablas las llevaremos a la segunda forma normal, es decir, remover atributos (que no son llaves y que no dependen completamente de la llave que identifica a la tabla) a otras tablas.

De las tablas mostradas en la Fig.19, observamos que el atributo no primo GRADO-EST es el único que depende completamente de la llave concatenada CVE-EST + NUM-SEM + CVE-CURSO. Por esta razón se anexa otra nueva tabla y las respectivas asociaciones con las tablas anteriores, como se ilustra en la Fig.20.

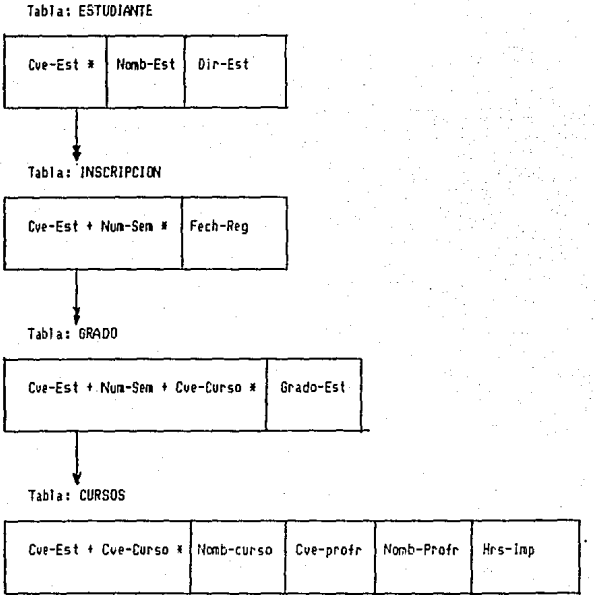


Fig.20 Tablas en la segunda forma normal asociadas a las tablas de la Fig.19.

Como en las relaciones que se presentan en la Fig.20, se observa que el atributo NOMB-PROFR, depende del atributo CVE-PROFR, y CVE-PROFR depende de CVE-EST + CVE-CURSO, es decir, hay dependencia transitiva. De acuerdo al proceso de normalización, se eliminará la dependencia transitiva aplicando la tercera forma normal, como se ilustra en la fig.21.

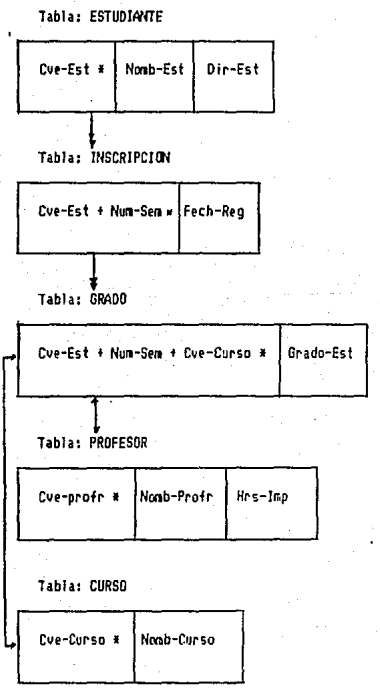


Fig.21 Relaciones en la tercera forma normal.

Como en las relaciones que se presentan en la Fig.21, no hay atributos no primos que tengan dependencia transitiva a cada una de las llaves que identifican a los renglones de las tablas, éstas se encuentran ya en la tercera forma normal.

Las tablas mostradas en la Fig.21 representan el modelo conceptual asociado a nuestro requerimiento.

II.2 Arquitectura del modelo jerárquico

Sabemos que las bases de datos jerárquicas se organizan en simples estructuras de árboles y los árboles se basan en las gráficas, por lo que primeramente recordaremos a estos conceptos.

II.2.1 Gráficas

Una gráfica está formada por un conjunto de nodos (vértices) y un conjunto de aristas. Una arista de la gráfica es simplemente el segmento de línea que determina un par de nodos.

Una sucesión de aristas forman una trayectoria. Así tenemos que en la gráfica de la Fig.22 los puntos a, b, c, d, e, f, g, h, i son los nodos de la gráfica, los segmentos de línea ab, bc, cd, de, ce, eh, ef, fg, fh y fi son las aristas de la gráfica, y la sucesión de aristas bc, ce y eh es una trayectoria de la gráfica.

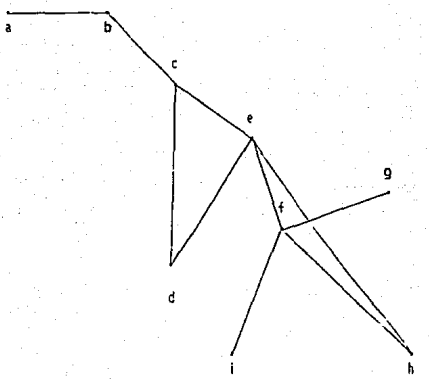


Fig.22 Los nodos y aristas son las componentes de la gráfica.

Una gráfica es conexa cuando siempre existe una trayectoria entre cualquier pareja de nodos ; en caso contrario es disconexa.

La gráfica de la Fig.22 es conexa mientras que la gráfica de la Fig.23 es disconexa ya que la trayectoria del par de nodos r y t no existe.

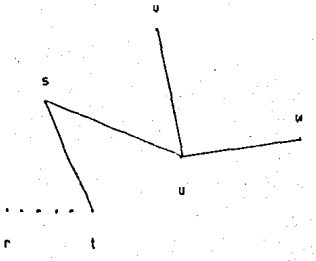


Fig.23 Gráfica disconexa.

Un ciclo en una gráfica es una sucesión de aristas que comienza y termina en el mismo nodo ; la sucesión de aristas cd, de y ec de la Fig.22 determinan un ciclo.

Una gráfica dirigida (digráfo) es aquélla en que los pares de nodos que conforman las aristas son pares ordenados. En la Fig.24 se ilustra una gráfica dirigida. Las flechas entre los nodos representan las aristas. La cabeza de cada flecha representa el segundo nodo en el par de nodos ordenados que forman una arista, mientras que la cola de la flecha representa el primer nodo en el par.

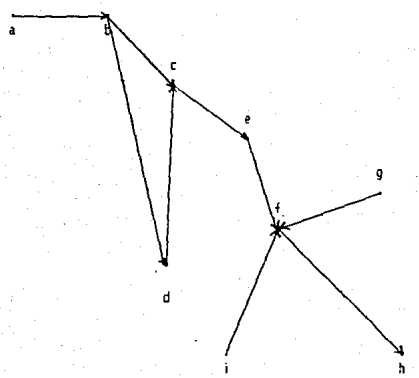


Fig.24 Una gráfica dirigida.

Una gráfica dirigida es conexa cuando existe una trayectoria dirigida entre cualesquiera dos vértices.

II.2.2 Arboles

Los árboles son gráficas o gráficas dirigidas que no contienen ciclos de tal manera que tienen un nodo principal llamado raíz, el cual no le llegan aristas ; el nodo a del árbol que se muestra en la Fig.25 es la raíz.

A los nodos que están en conexión con nodos de un nivel inmediato superior, tales como los nodos b, c, d, e, f, g, h, i, j, K, l, m, n, o y p de la misma figura son los hijos de sus padres respectivos.

Los nodos que son hijos del mismo padre, tales como los nodos e, f y g de la misma figura se llaman hermanos.

Los nodos que no les salen aristas, tales como los nodos e, f, g, c, l, m, n, o, p y K de la misma figura se llaman hojas.

Tomando como referencia el árbol de la Fig.25, el nodo a es la raíz del árbol, los nodos restantes están relacionados a la raíz, la cual está en el primer nivel. Los nodos que se pueden relacionar al nodo raíz mediante un segmento de línea (arista), son nodos de segundo nivel ; así tenemos que los nodos b, c y d del mismo árbol, son de segundo nivel. Para continuar con la misma

ternimología, un nodo de tercer nivel es aquel que pasa por dos aristas a partir de la raíz, como es el caso de los nodos e, f, g, h, i, j y K del mismo árbol.

La distancia de un nodo que no es raíz hasta el nodo raíz tiene que ver con el nivel de los nodos.

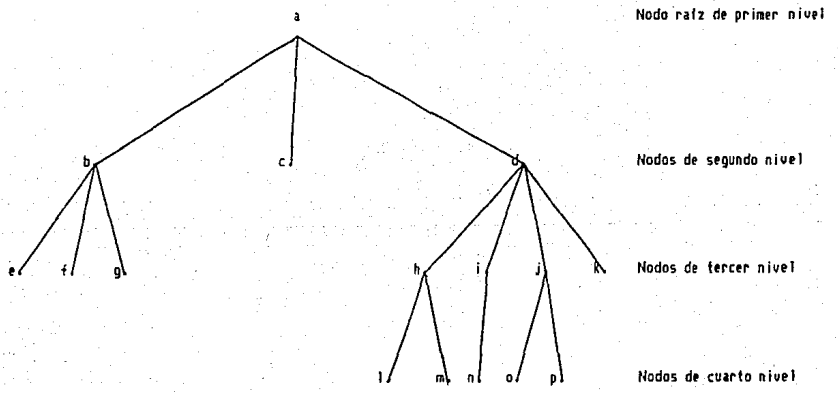


Fig.25 El nodo a es la raíz del árbol; el nodo b es el padre de los nodos e, f, g (hermanos); el nodo d es el padre de los nodos h, i, j, k; el nodo h es el padre de los nodos l, m; el nodo i es el padre del nodo n; el nodo j es el padre de los nodos o, p; los nodos e, f ,g, c, l, m, n, o, p y k son las hojas.

Un árbol puede ser dividido en subarboles, los cuales preservan

las mismas propiedades de los árboles, pero que contienen un número menor de nodos y aristas, que el original, como se muestra en la Fig.26.

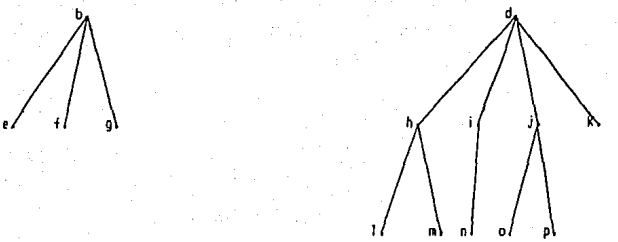


Fig.26 Dos subárboles que se obtienen al dividir al árbol de la Fig.25.

Los datos en el modelo jerárquico se organizan en simples estructuras de árboles ; una base de datos jerárquica es una colección de tales árboles, es decir, un bosque. En esta organización se requiere necesariamente que algunos nodos estén subordinados a otros. Los nodos en estas estructuras representan a tipos de registros o tipos de entidades que costituyen a la base de datos jerárquica.

Las relaciones en un árbol se obtienen mediante aristas (arcos), los cuales conectan a tipos de entidades. Así tenemos que en el árbol de la Fig.27, los nodos de segundo nivel, a la izquierda son de tipo A, al centro son los de tipo B y a la derecha son de tipo C, etc.

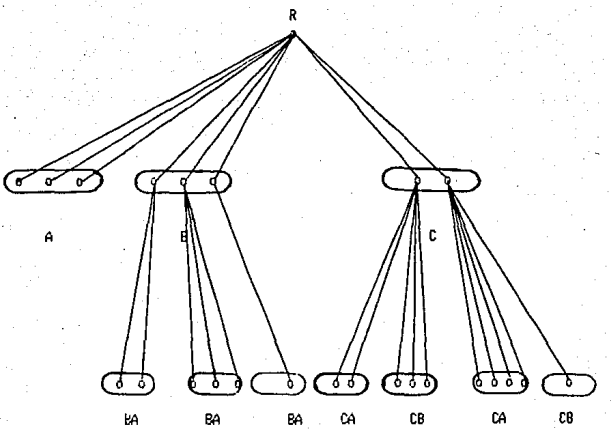


Fig.27 Tipos de nodos en la jerarquía del árbol con raíz R.

Para tener una mayor claridad de la estructura jerárquica, consideremos el diagrama de estructura que se asocia a los árboles, el cual es una manera simplificada de transferir la estructura jerárquica que prevalece y se aplica a la base de datos jerárquica. El digrama de estructura asociado al árbol ilustrado en la Fig.27 se muestra en la Fig.28.

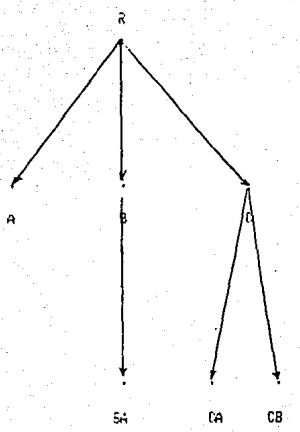


Fig.28 Diagrama de estructura asociada al árbol de la Fig.27.

Consideremos a una base de datos educacional y los tipos de entidades que se asocian a ésta ; algunos de éstos son : CURSOS, SECCIONES, PROFESORES y ESTUDIANTES. Una ocurrencia para las relaciones ASISTE e INSTRUYE se muestra en la Fig.29 y el diagrama de estructura asociado al árbol de la Fig.29 se muestra en la Fig.30.

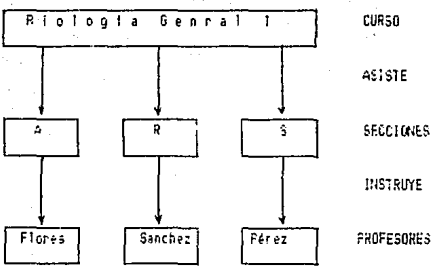


Fig.29 Una ocurrencia de la base de datos jerárquica (BDJ) para las relaciones ASISTE e INSTRUYE

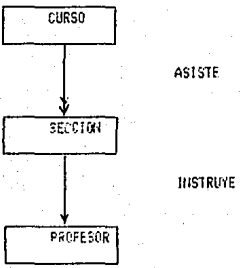


Fig. 30 Diagrama de estructura asociada a la Fig.29.

En el árbol que se ilustra en la Fig.29 se tiene que el tipo de entidad CURSO de Biología General I se divide en las secciones A, R y S. Cada sección está compuesta por un profesor que enseña a los alumnos que asisten a esa sección en los diferentes períodos (horas) que se imparte. De acuerdo a lo anterior se da origen a varios tipos de relaciones en la base de datos educacional, las cuales se mencionan en seguida : la relación ASISTE que va de la entidad CURSOS a la entidad SECCIONES es de uno a muchos, en cambio la relación INSTRUYE que va de cada sección a la entidad PROFESORES es de uno a uno y la relación que va de cada SECCION a los ESTUDIANTES que asisten a esta es de uno a muchos.

El árbol de la Fig.31 muestra la ocurrencia de la Fig.29 de una manera más detallada, así como a sus tipos de relaciones.

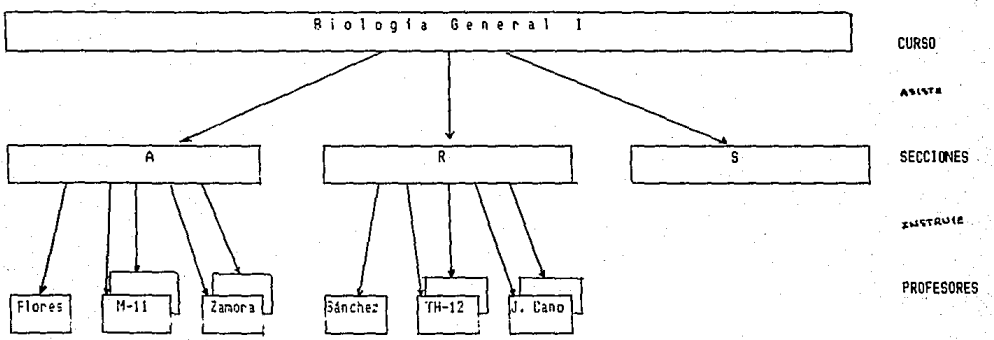


Fig.31 Las relaciones que van del tipo A a sus respectivos hijos son de uno a uno, uno a varios y de uno a varios.

Además en la misma figura notamos que existen tres tipos de apuntadores, los apuntadores izquierdos son para profesores, los apuntadores del centro son para los períodos de duración del curso, y los apuntadores derechos son para los estudiantes del curso que asisten a cada una de las secciones.

Las relaciones que hemos considerado hasta el momento en la base de datos jerárquica para nuestras aplicaciones han sido las de 1 <-----> 1 y las de 1 <----->> m, pero puede ser que las relaciones mencionadas no siempre sean las más adecuadas para nuestras aplicaciones, por lo que hemos de analizar las relaciones de m <<----->> n. Es decir, el problema consiste en representar una relación en la que para un sujeto le correspondan varios objetos e inversamente para un objeto le correspondan varios sujetos. Para lograr a una relación tal, consideremos la gráfica que aparece en la Fig.32, en la que las entidades que se encuentran a la izquierda son los sujetos (ESTUDIANTES) de la relación y los objetos que se encuentran a la derecha (MATERIAS) son los objetos de la relación.

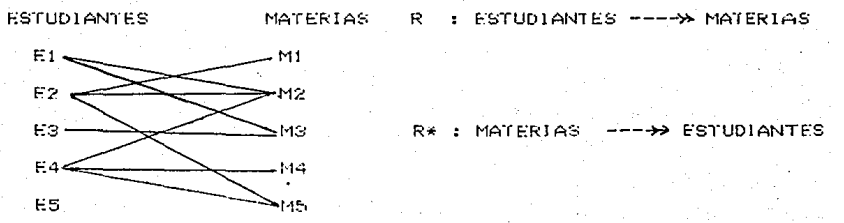


Fig.32 Gráfica de una relación simple de m <<------>> n entre izquierdos y derechos.

El diagrama de la estructura jerárquica asociada a la gráfica que se muestra en la Fig.32 se ilustra en la Fig.33.

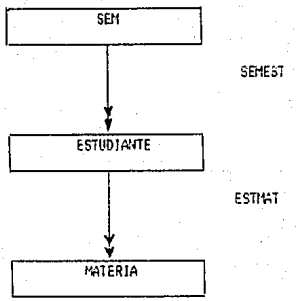


FIG.33 Diagrama de estructura jerárquica asociada a la Fig.32 para representar relaciones de m ----> n.

Otra representación de la gráfica de la Fig.32 en base de datos jerárquica y puesta en forma de árbol es la que se ilustra en la Fig.34.

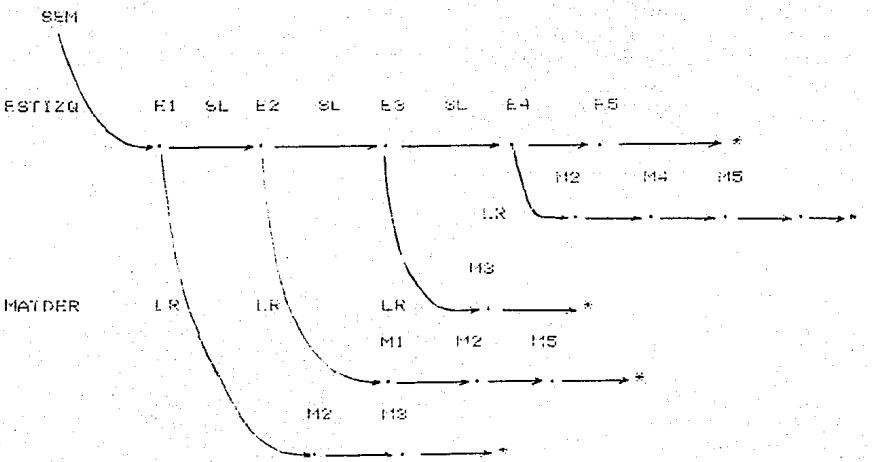


FIG.34 Una gráfica de la BDJ para representar la Fig.32.

En esta figura observamos que la raíz del árbol es SEM del cual se cuelgan dos ramas, la rama izquierda que está formada por los estudiantes E1, E2, E3, E4 y E5 y la rama derecha formada por las materias M1, M2, M3, M4 y M5. Cabe mencionar que la raíz (padre) tiene un apuntador al primer hijo E1, el cual a su vez tiene un apuntador a su hermano E2 y un apuntador a los objetos con los cuales está relacionado, en este caso M2 y M3 ; el estudiante E2 tiene un apuntador a su hermano E3 y un apuntador a los objetos con los cuales esta relacionado, en este caso M1, M2 y M5 ; siguiendo de esta manera se llega al final de la rama ESTIZQ ya que el último apuntador apunta al final de la rama ESTIZQ.

Notemos que hay tantas ocurrencias de objetos en la rama MAYDER como relaciones que se encuentran en la misma figura. Por ejemplo la materia M2 es el objeto de las tres relaciones MAYDER teniendo a los estudiantes E1, E2 y E4 como sujetos (es decir, los padres de M2). Esto último es fuente de redundancia. De ello, la forma simple de representar una relación de muchos a muchos, es haciendo que un segmento aparezca tantas veces como lo sea el objeto de alguna relación.

La Fig.35 ilustra como la gráfica de la Fig.34 es puesta en la forma de base de datos jerárquica.

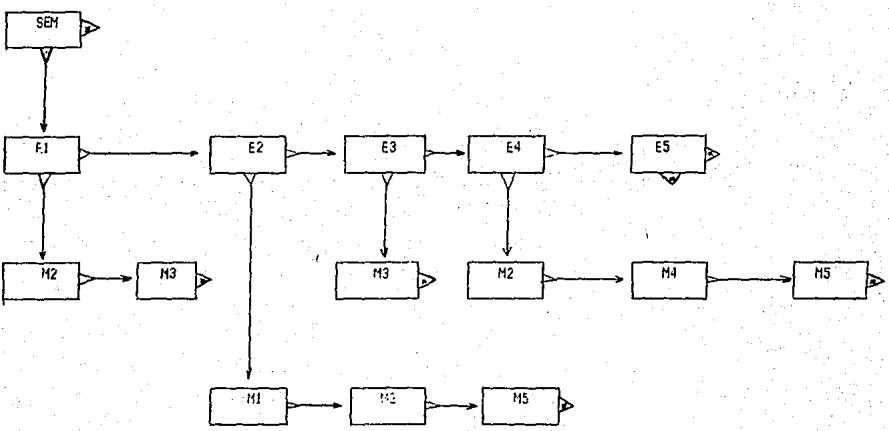


FIG.35 Una reprersentación jerárquica de la Fig.34.

Para eliminar los datos redundantes se necesita un segundo apuntador padre distinto al del apuntador padre original, tal que tenga como sujetos a elementos de entidad ESTMAT y como objetos a miembros de la entidad MATDER con los cuales están relacionados, tal como se ilustra en la Fig.36.

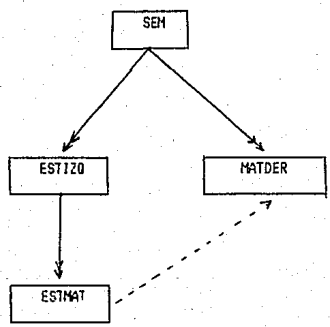


FIG.36 Un ejemplo de la BDJ para los izquierdos y derechos usando apuntadores secundarios.

Una representación gráfica para la base de datos jerárquica de la Fig.36 aplicada al ejemplo de la Fig.32 se muestra en la Fig.37.

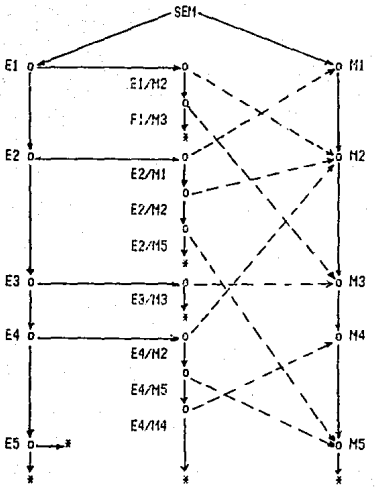


Fig.37 Una representación gráfica para la base de datos jerárquica de la Fig.36.

La base de datos jerárquica hasta aquí nos permite determinar objetos fácilmente, dado el sujeto y la relación original ; ahora el problema consiste en determinar todos los objetos izquierdos que están relacionados con algún sujeto derecho mediante la relación R*.

La forma de hacer la búsqueda de tales objetos, es similar a la ya

mencionada, es decir, la raíz de la base de datos jerárquica que se encuentra en la Fig.36 tiene dos hijos, uno para los elementos izquierdos y otro para los elementos derechos. Como antes tenemos entidades ESTIZQ en las ramas izquierdas, RI, para determinar objetos derechos : una rama para cada entidad izquierda contiene una entidad MATDER para cada entidad derecha que está relacionada con esa entidad izquierda.

Sobre la rama derecha, una rama simple consiste de entidades, cada entidad es el padre de una rama derecha, que consiste de entidades MATDER, una para cada entidad izquierda la cual se relaciona a la entidad derecha. En la Fig.38 la línea punteada DERIZQ, de una entidad MATEST a una entidad ESTIZQ es el apuntador del padre secundario. De cada entidad padre MATEST existe un apuntador a exactamente una entidad ESTIZQ.

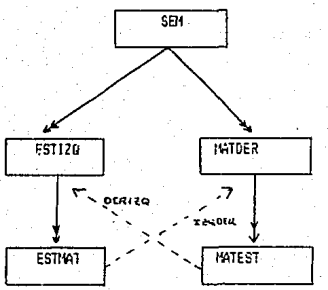


Fig.38 Una base de datos jerárquica en las direcciones izquierdos a derechos y recíprocamente.

La gráfica de la Fig.32 ha sido convertida en una gráfica completa que demuestra apareamiento físico bidireccional y aparece en la Fig.39. Notamos que las ramas simples izquierdas y derechas, ESTIZQ, y MATDER, cada una contiene entidades izquieradas y derechas.

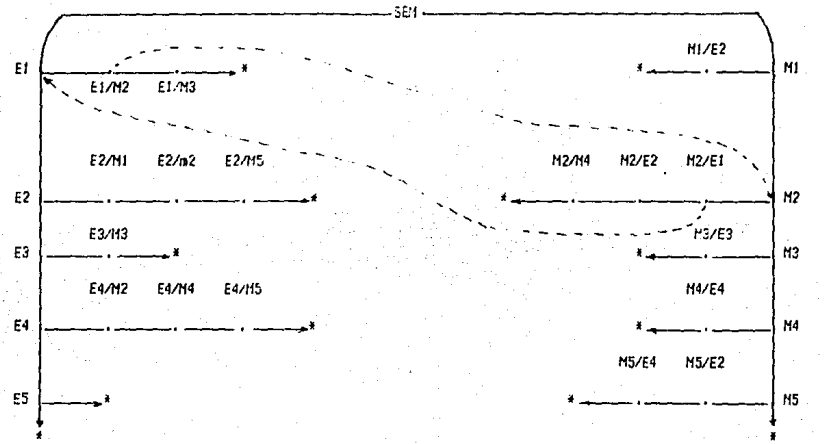


Fig.39 Los apuntadores secundarios a padres hacen posible la búsqueda bidireccional para cada segmento de la intersección.

Las operaciones que se efetcúan en una base de datos jerárquica se traduce en la eliminacion o insercion de apuntadores a la estructura de árbol.

III.3 Arquitectura del modelo red

En este modelo, la información se representa por tipos de registros interconectados por relaciones (ligas) para formar una gráfica dirigida. Los registros son agrupados en conjuntos, cada uno de los cuales consiste de un registro propietario y cero o más registro (s) miembro (s).

El conjunto(set) es el aspecto más importante del modelo red ya que representa relaciones de uno a muchos entre dos clases de entidades, las relaciones se hacen por medio de apuntadores de manera que formen un ciclo. Generalmente una "clase de entidad" significa la representación de una entidad o conjunto de entidades. El sujeto de la relación es llamado el propietario del conjunto y es el "uno" de la relación 1 <-----> m. El objeto de la relación es llamado el miembro y es los "muchos".

El ligado (set) para la base de datos red transporta una relación entre una entidad propietario como sujeto a una o más entidad (es) como objeto (s).

Para representar relaciones de muchos a muchos en la base de datos relacional se necesita incluir el concepto de "conexión" de tipos de registros. Una conexión de registros contiene la información común de los dos tipos de registros ligados por relación de muchos a muchos. Así tenemos que la Fig.40, la entidad PROFESOR es el propietario y la entidad MATERIA es el miembro de la relación IMPARTE la cual es de uno

a varios.

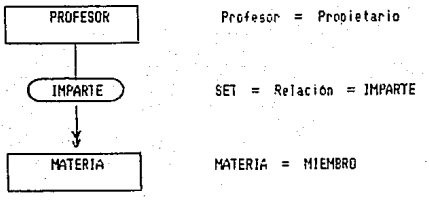


Fig.40 Diagrama de estructura de la relación (SET) IMPARTE.

Una ocurrencia de la relación IMPARTE se muestra el la Fig.41, en ésta el profesor Flores imparte tres materias que las identificamos con MAT1, MAT1 y MAT3.

Cada una de las ocurrencias comienza en el registro propietario el cual tiene un apuntador al primer miembro y cada miembro contiene un apuntador al siguiente miembro, así sucesivamente hasta llegar al último miembro el cual contiene un apuntador al propietario generando así un ciclo.

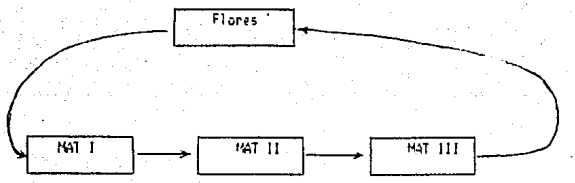


Fig.41 Una ocurrencia de la relación (Set) IMPARTE de la Fig.40.

De acuerdo al diagrama de la Fig.41, podemos decir que una ocurrencia de la relación (set) IMPARTE, es una lista circular que comienza y termina en el tipo de registro propietario PROFESOR.

Cabe mencionar que un miembro puede tener diferentes relaciones propias. Para analizar esto consideremos que un curso puede consistir de varias secciones diferentes que contienen distintos estudiantes, los cuales son instruídos por distintos profesores, tal como, se ilustra en la Fig.42. Allí se observa que los conjuntos ASIGNA, ELIGE, INSTRUYE y DURA tienen como propietario a SALON, CURSO, PROFESOR y PERIODO respectivamente, y como miembro a SECCION.

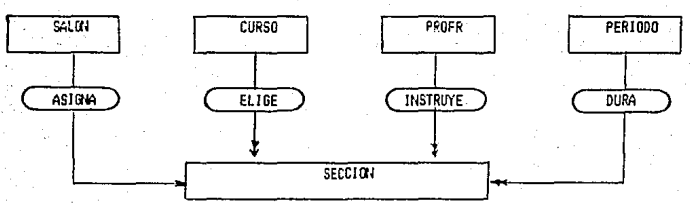


Fig.42 El miembro SECCION tiene diferentes relaciones propias.

Hasta el momento hemos considerado relaciones (sets) entre dos entidades o conjuntos de entidades, de modo que para un propietario se le asigna mediante la relación varios miembros. Ahora analizaremos la relación recíproca, es decir, dado un propietario buscar los

miembros que le corresponden mediante la relación R*, que al igual que en la base de datos jerárquica, se establece las relaciones en ambos sentidos, es decir, las de muchos a muchos.

Las relaciones de muchos a muchos en la base de datos tipo red hacen posible la utilización de "ligas", que son un tipo de registro auxiliar (conexión de tipos de registro), el cual es miembro de dos tipos de registro propietario.

Lo natural en una base de datos educacional es que las relaciones entre las entidades ESTUDIANTES y MATERIAS son de muchos a muchos. El tipo de registro auxiliar LIGA de la Fig.43 es el miembro de las relaciones ASISTE y CONTIENE, cuyo tipo de registro propietario son ESTUDIANTES y MATERIAS respectivamente.

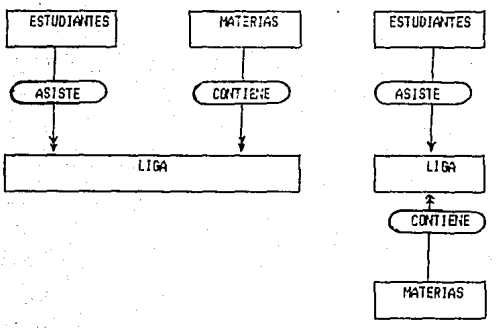


Fig.43 Una relación de muchos a muchos requiere de dos propietarios, un miembro y dos relaciones (Sets).

Para dar una ocurrencia de la relación de muchos a muchos entre

los tipos de entidades ESTUDIANTES y MATERIAS supongamos que E1, E2, E3 y E4 conforman la entidad ESTUDIANTES y M1, M2, M3, M4 y M5 conforman la entidad MATERIAS y las relaciones entre ellas se muestran en la Fig.44.

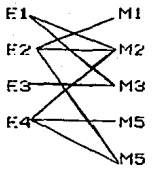


Fig.44 Relaciones entre los elementos de las entidades ESTUDIANTES y PROFESORES.

Las ligas que se utilizan para representar la ocurrencia de la Fig.44 son L1, L2, L3, L4, L5,L6, L7, L8 y L9 las cuales se ilustran en la Fig.45.

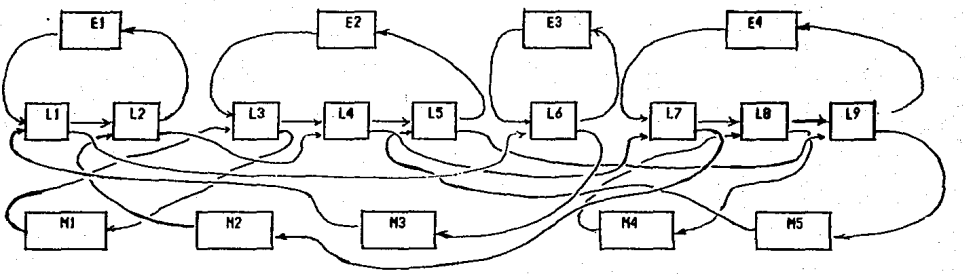


Fig.45 Ligas para la ocurrencia de la Fig.44.

Tomando como referencia el diagrama de la Fig.45, observamos que cada estudiante tiene solamente un apuntador, el cual apunta a una liga, de la cual comienza una cadena de miembros del propietario ESTUDIANTES. Cada materia tiene un apuntador, el cual apunta a una liga, de la cual comienza una cadena de miembros para el propietario MATERIAS. Se necesitan tantas ligas como relaciones haya, así tenemos que en la Fig.44, hay nueve ligas en la ocurrencia de la misma figura.

II.4 Comparaciones de los modelos

Por la facilidad de uso, no existe duda que el modelo relacional es superior. Este proporciona una construccion que el usuario entenderá facilmente. Además existen cuantiosos lenguajes de alto nivel para expresar consultas sobre los datos representados por el modelo relacional. Estos lenguajes producen sistemas basados sobre el modelo relacional disponible para personas con no demasiada habilidad para programación.

En comparación, el modelo de red requiere nuestros conocimientos de ambos tipos de registros y ligas, y sus interrelaciones. La instrumentación de las relaciones de muchos a muchos y relaciones sobre tres o más entidades no es directa, aunque con la práctica uno logra el uso de la técnica. Similarmente, el uso del modelo jerárquico requiere del conocimiento del uso de apuntadores(tipos de registro virtual) y tiene los mismos problemas que que el modelo de redes respecto a la representación de relaciones que son más complejas que las relaciones de uno a muchos entre dos entidades.

III. Opciones para manejar a una base de datos

En el presente capítulo se analizan las opciones para manejar a una base de datos, los sistemas de información y el sistema manejador de bases de datos. Además se discuten las ventajas y desventajas de cada una de ellas.

III.1 Sistemas de información

En las instituciones actuales se procesan grandes volúmenes de programas de aplicación con la finalidad de obtener información útil para la toma de decisiones de las mismas, así también como la de proporcionar información a usuarios externos que la soliciten para mejorar los enormes problemas de administración.

Para llevar a cabo tales procesos, las instituciones cuentan con sistemas de cómputo para el procesamiento de datos. Para manejar datos cuentan con las opciones de sistemas de información y sistemas manejadores de bases de datos que se detallan en seguida.

Los sistemas de información aparecen con la introducción del disco magnético (por IBM en 1960), el desarrollo de la multiprogramación y la capacidad de la utilización de terminales interactivas dando lugar al tiempo compartido a los diferentes usuarios de la institución. Los sistemas de información tuvieron gran éxito durante la década de los años setentas.

El objetivo de los sistemas de información es el de proporcionar a los distintos usuarios de las instituciones, la consulta simultánea de

la información (no redundante, almacenada en discos magnéticos), para los diferentes programas de aplicación que se encuentran escritos en lenguajes de programación de alto nivel, como por ejemplo, COBOL, PL/1, etc.

Los archivos maestros que se encuentran en los discos magnéticos cuentan con algún formato de registro específico, los cuales son utilizados por los programas de aplicación de acuerdo a las tres formas de acceso de datos, acceso secuencial, acceso secuencial indexado y acceso diretco.

Cuando los programas de aplicación se escriben en lenguaje COBOL, los archivos de entrada deben de ser acordes con respetco al tamaño y tipo del archivo maestro que será utilizado por el programa de aplicación. Cuando no se cumplen esas condiciones se obtienen errores de ejecución.

Existen dos formas para dar solución a los errores de ejecución que surgen a raíz de que el "traje no es hecho a la medida", es decir, el tamaño del registro del archivo de entrada es diferente al tamaño del registro del archivo maestro. La primera forma consiste en reformatear el archivo maestro para que la longitud de ambos registros sea igual, pero esto nos lleva a una modificación un tanto tediosa a los programas de aplicación que son afetcados por el reformateo. Una segunda forma podría ser una copia del archivo maestro en un nuevo archivo con el tamaño del registro requerido, pero esta segunda forma da origen a la redundancia de datos.

La problemática mencionada en esta sección se debe a que los sistemas de información no cuentan con un sistema manejador de bases de datos que les permita hacer uso de la independencia de datos tanto lógica como física. Es por esto que las instituciones buscan otros mecanismos para que la administración de datos en las mismas sea más eficiente ; estos mecanismos son los sistemas manejadores de bases de datos,los cuales se describen en la siguiente sección.

III.2 Sistema manejador de bases de datos (SMBD)

Es un conjunto de sistemas de programación que sirven como herramientas para hacer operativa a las bases de datos ; las herramientas tienen la capacidad de soportar y manejar a una o más bases de datos. Para que se cumplan los objetivos planteados por la base de datos , el SMBD debe de contener varios tipos de lenguajes. A estos tipos de lenguajes los clasificaremos en tres : lenguaje para la manipulación de datos, lenguaje para la descripción de datos y el lenguaje para la interrogación de datos.

III.2.1 Lenguaje para la manipulación de datos (LMD)

Este lenguaje nos sirve de interface entre los programas de aplicación y el SMBD. El LMD aparece en el programa de aplicación para solicitar entradas y salidas que requieren los usuarios para la ejecución de sus programas de aplicación. El LMD funciona por medio de comandos, para realizar entre otras, las siguientes acciones obtener (GET,FIND), modifica (MODIFY), inserta (INSERT), abre (OPEN), cierra (CLOSE), ornena (ORDER), desaparecer (REMOVE), almacena (STORE), quita

(DELETE), etc.

El SMBD que se asocia a cada una de las bases de datos tienen su propia nomenclatura y comandos.

III.2.2 Lenguajes para la descripción de datos (LDD)

Es indispensable que el programador y el administrador de datos estén capacitados para describir sus datos con precisión, especificando sus estructuras. Para ello recurren al empleo de un lenguaje para la descripción de datos. Un lenguaje para la descripción de datos es el idóneo para declarar al SMBD cuáles son las estructuras de datos que van a utilizarse.

Todo lenguaje para la descripción de datos lógica debe de cumplir las siguientes funciones :

1. Deben identificar los tipos de subdivisión de datos, tales como, módulos de datos, registros y archivos de la base de datos.

2. Debe asignar un nombre único a cada módulo de datos, tipo de registros, tipo de archivo, base de datos y otras subdivisiones de datos.

3. Debe especificar qué tipos de datos se hallan en cada uno de los módulos de datos ; tipo de registro u otra subdivisión, indicando los grupos repetitivos que existan y su secuencia.

4. Debe especificar qué tipos de módulos de datos, partes de tipos de módulos de datos o combinación de tipos de datos se utilizarán como llaves.

5. Debe especificar como se relacionan los tipos de registro para formar estructuras como las que hemos visto en capítulos anteriores.

6. Debe asignar nombres a las relaciones entre tipos de datos o tipos de registros (es decir, nombres a las líneas que conetcan bloques en los diagramas de esquema y subesquema).

7. Puede definir el tipo de codificación que el programa utiliza en los módulos de datos (binario, de caracteres, sartas de bites, etc). Esto no debe confundirse con la codificación utilizada en la representación física.

8. Puede definir la longitud de los módulos de datos.

9. Puede definir la gama de valores que se permiten asumir a un módulo de datos.

10. Puede especificar el número de módulos de datos de que está compuesto un vetcor, especificar el número de dimensiones y el tamaño de las matrices.

11. Puede especificar la secuencia de los registros en archivos de la base de datos.

12. Puede especificar la manera de verificar errores en los datos.

13. Puede especificar los códigos de seguridad para permitir la lectura o modificación no autorizada de los datos. Estos pueden operar a nivel de módulo de datos, registro, archivo o base de datos y si es necesario puede extenderse a los contenidos (valores) de los módulos de datos individuales. Por otra parte la autorización puede definirse independientemente. Probablemente deba saberlo, porque está más sujeta a cambios que las estructuras de datos y cambios en los procedimientos de autorización no deberían obligar a modificar los programas de aplicación.

14. Una descripción lógica de datos no debería especificar técnicas de direccionamiento, indicación, ni búsqueda o exploración, ni especificar la posición de los datos en las unidades de almacenamiento, porque estos tópicos se hallan en el dominio de la representación física, no en el de la organización lógica. Puede dar una indicación de cómo se utilizarán los datos, o de los requerimientos de exploración, de modo que sea posible elegir técnicas físicas óptimas, pero tales indicaciones no deben representar una limitación lógica.

La vista que tiene el programador de aplicaciones de los datos está representada por un subesquema y es a menudo diferente de la representación del esquema general (descripción general lógica de la base de datos). Mas aún, la vista representada por el esquema general es diferente de la distribución física de los datos. En

consecuencia, es preciso describir los datos de tres maneras diferentes.

1. Desde el punto de vista del programador de aplicaciones : una descripción del subesquema.

2. Desde el punto de vista lógico global : una descripción del esquema.

3. Desde el punto de vista físico : una descripción de los registros físicos y sus vinculaciones.

III.2.3 Lenguaje de interrogacion de datos (language QUERY)

Este lenguaje permite a los usuarios solicitar información a la base de datos por medio del SMBD directamente por la terminal.

Para una mayor claridad veamos los sucesos principales que se desarrollan cuando un programa de aplicación lee un registro de la base de datos por medio del SMBD.

1. El programa de aplicación declara el nombre de los datos, sus tipos y además da el valor de la llave del registro en cuestión.

2. El SMBD obtiene el subesquema (descripción de datos del programa) que utiliza el programa de aplicación y examina la descripción de los datos en cuestión.

3. El SMBD obtiene el esquema (descripción global lógica de los datos) y determina qué tipo (o tipos) lógico de datos necesita.

4. El SMBD examina la descripción física de la base de datos y determina cúal registro (o registros) físico debe leer.

5. El SMBD emite una orden al sistema operativo de la computadora,

indicando lo que debe leer del registro pedido (o los registros pedidos).

6. El sistema operativo interacciona con el almacenamiento físico en el que se encuentran los datos.

7. Los datos pedidos se transfieren desde el almacén a los almacenes intermedios("buffers") del sistema.

8. Comparando el subesquema y esquema, el SMBD deduce de los datos el registro lógico pedido por el programa de aplicación. El SMBD ejecuta todas las transformaciones que sean necesarias entre los datos tal como están declarados en el subesquema y en el esquema.

9. El SMBD transfiere los datos desde los almacenes intermedios al área de trabajo del programa de aplicación.

10. El SMBD provee una información de estado al programa de aplicación informándole sobre el resultado de su pedido, inclusive cualquier posible indicación de error.

11. El programa de aplicación puede ahora operar con los datos pedidos, los cuales se hallan en su área de trabajo.

La Fig.46 ilustra la sucesión de pasos descritos anteriormente.

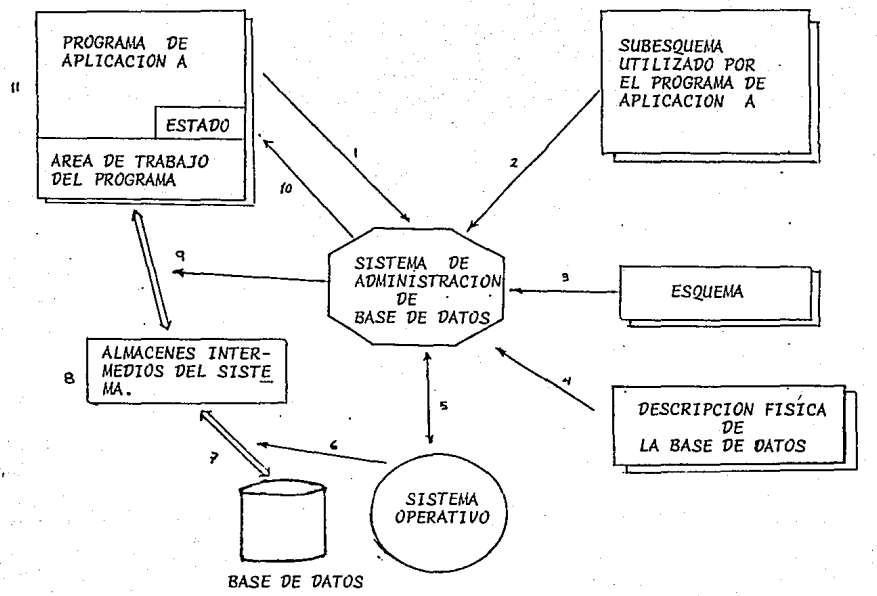


FIG. 46 Serie de pasos que se ejecutan cuando un programa de aplicación necesita un registro, usando SMBD.

III.3 Ventajas del sistema manejador de base de datos con respecto a los sistemas de información

1. La longitud de los archivos es irrelevante, pues ahora el SMBD se encarga de proporcionar los datos por medio de los subesquemas a los programadores de aplicaciones de acuerdo a la estructura en que los soliciten.

2. Se cuenta con la independencia de datos tanto lógica como

física, es decir, la protección a los programas de aplicaciones con respecto a los cambios lógicos y físicos.

3. El lenguaje de programación para los programas de aplicación reduce el consumo de recursos, como lo son : la codificación, el tiempo de respuesta y el almacenamiento de datos.

4. La actualización y mantenimiento de los archivos es mayor en los sistemas de bases de datos que en los sistemas de información, ya que cuando se utiliza el SMBD las trasacciones se hacen en línea, en cambio en los sistemas de información se hacen en batch.

5. La información es compartida por los distintos usuarios que la soliciten.

IV. Descripción del SMBD relacional (DATACOM/DB)

En el presente capítulo se describe el sistema manejador de bases de datos relacional DATACOM/DB, el cual se utiliza para la administración de datos y el desarrollo de los diferentes sistemas de aplicación en el área de procesamiento de datos de la Tesorería del Departamento del Distrito Federal. Antes de dar la descripción de DATACOM/DB, primero daremos algunas causas que dieron origen al requerimiento de tal SMBD relacional.

IV.1 Causas que originaron el requerimiento de DATACOM/DB

La Tesorería del Departamento del Distrito Federal es una institución que de acuerdo a la naturaleza de sus funciones informáticas utiliza sistemas computacionales.

En los procesos computacionales dentro de la Tesorería del Distrito Federal existían problemas diversos, algunos de ellos son :

1. La saturación de procesos de cómputo por la mayor prioridad de la operación de los sistemas de producción hacía que los tiempos de respuesta para compilaciones y pruebas fuesen mayor dando origen a una bajísima productividad del personal encargado del mantenimiento y desarrollo de sistemas de la Tesorería del Departamento del Distrito Federal.

2. Muchas áreas y procesos sustantivos de la Tesorería del Distrito Federal no estaban apoyados por servicios informáticos por no tener los recursos y se calcula que sólo en los requerimientos ya formulados para desarrollo de nuevos sistemas, existía un rezago de tres a cuatro años.

3. Todos los sistemas existentes estaban escritos en lenguaje COBOL

y cada uno de sus programas manejaban los bancos de datos definidos en los distintos archivos, es decir, no existían estándares formales para la programación ni definición de datos, y la documentación era incompleta, inconsistente y muy desactualizada.

Para dar solución a la problemática descrita anteriormente, se hizo un análisis detallado acerca del requerimiento de un administrador de bases de datos que contemplara los siguientes objetivos.

1. Administrar los datos que apoyan los procesos operativos de la institución garantizando la integración de los diferentes sistemas y eliminando inconsistencias.

2. Mejorar la productividad del personal de desarrollo de sistemas, abatiendo el rezago en la materia.

3. Disminuir el impacto de cambios en los requerimientos de los sistemas de información, por medio de una independencia lógica y física de datos y sistemas, aminorando así los esfuerzos de mantenimiento de sistemas.

De acuerdo a estos objetivos se procedió a definir los requerimientos funcionales de las herramientas de los sistemas de programación (software) que serán utilizadas para la administración de datos y desarrollo de sistemas en cuanto a:

El modelo conceptual de datos utilizados.

Las facilidades ofrecidas para el desarrollo, mantenimiento y operación de sistemas.

Las características deseables de la interacción (externa) con el usuario y la arquitectura (interna) de los sistemas de

programación.

IV.1.1 Modelo conceptual de datos

Como se sabe en el medio informático, existen tres modelos conceptuales en los sistemas administradores de datos : el relacional, el jerárquico y el de redes, dependiendo de las estructuras utilizadas para representar los datos, como ya lo hemos visto en el capítulo II.

El modelo jerárquico resulta poco flexible respecto a la implantación de nuevas aplicaciones para los datos o adición de datos nuevos requiriendose el rediseño total de la base de datos en muchos casos. Además requiere de personal altamente especializado y representa un alto consumo de recursos de cómputo.

El modelo de redes estructura los datos en conjuntos de diversos tipos, con la limitación de que un elemento no puede pertenecer a dos conjuntos del mismo tipo. Facilita las actualizaciones directas de la base de datos (debido al ligado entre datos físicamente), aunque dificulta en algo los procesos secuenciales (batch). El diseño de la base de datos requiere un entrenamiento especial, pues el modelo de redes no es de ningún modo una manera "natural" de concebir los datos. Además la programación requiere de un conocimiento detallado de la estructura física de los datos y la eventual reorganización de datos requiere un alto esfuerzo de personal y procesamiento.

El modelo relacional provee una manera simple de representar los datos en formas de tablas (relaciones) en que los renglones representan

una ocurrencia de una entidad o relación, y las columnas sus atributos. Este modelo permite de manera directa la representación de cualquier relación entre datos, es general, fácil de entender, y no requiere de la adaptación de los datos a una estructura impuesta por el sistema ; propone funciones que sin necesidad de rediseñar ni reestructurar las bases de datos permiten el acceso a ellos de distintos modos. Por su simplicidad, claridad conceptual y total independencia lógica y física, el modelo relacional es el que mejor satisface las características de un sistema administrador de bases de datos como el requerido por la Tesorería.

IV.1.2 Facilidades deseables en el sistema

Un buen sistema para la administración de las bases de datos debe contemplar :

1. Diccionario de datos. Es la base de datos que maneja información respecto a las clases de datos de la organización.

2. Núcleo manejador de bases de datos. Este es el componente central de la base de datos que intrumenta el modelo conceptual con alguna arquitectura determinada, provee los servicios de respaldo, recuperación y validación de acceso a las bases de datos y garantiza la integridad de la información en caso de fallas.

3. Generador de reportes. Debe contemplar facilidades para la obtención de reportes batch más o menos complejos permitiendo en general seleccionar datos, elaborar cálculos estadísticos, convertir y formatear datos.

4. Lenguaje de interrogación (QUERY language). Es la herramienta

que pretende satisfacer las necesidades imprevistas de información de manera directa en las pantallas.

5. Interface a lenguajes de programación. Es conveniente poder aprovechar la programación existente principalmente en lenguaje COBOL utilizando comandos para acceso a la base de datos desde lenguajes "anfitrión".

Los componentes básicos mencionados anteriormente son necesarios más no suficientes en nuestro caso, y se consideró indispensable contar también con :

6. Múltiples "vistas" de datos. La principal ventaja del modelo relacional está en contar para cada aplicación con los datos requeridos, independientemente de la forma en que estén almacenados.

7. Lenguajes de programación de mayor poder. Será imposible manejar de una manera notable la productividad del personal de desarrollo y mantenimiento de sistemas, de tener que seguir utilizando lenguajes como "COBOL" que ha pasado a ser un verdadero obstáculos en esta actividad. Al contar con definiciones de datos en un diccionario es más fácil instrumentar la generación de reportes, transacciones de consultas y actualización de las bases de datos, prácticamente sin la necesidad de una "programación" en el sentido tradicional, por lo que se llaman lenguajes de cuarta generación.

IV.1.3 Interacción con el usuario

Es de primordial importancia que los productos para administración de las bases de datos y desarrollo de sistemas sean verdaderamente fáciles de usar, sin necesidad de un muy especializado entrenamiento.

Por lo mismo es esencial la consistencia conceptual en la instrumentación del modelo relacional por el sistema. Se considera también necesaria la presencia de un ambiente interactivo para el desarrollo de sistemas, que permita la validación sintáctica, corrección, compilación, pruebas y documentación de los programas y definiciones de datos y su posible integración con los monitores de teleproceso (CICS o ROSCOE). Para ello se cuenta con ROSCOE, un monitor de teleproceso que controla la interacción entre los usuarios de terminal y el sistema operativo en el mantenimiento de programas y desarrollo de sistemas en línea. ROSCOE tiene comunicación sofisticada con los sistemas de programación (software) para asegurar un buen tiempo de respuesta y uso eficiente de recursos de cómputo. Es además un editor de pantalla que proporciona comandos que ayudan a actualizar datos de entrada y simplificar tareas de mantenimiento.

Los datos son almacenados en bibliotecas en línea, en forma mínima con el objeto de facilitar el rápido acceso, al mismo tiempo que se utiliza una cantidad mínima de espacio en disco. El lenguaje de comandos de ROSCOE permite que los datos sean editados, impresos o exportados al sistema operativo con el tratamiento del trabajo (JOB) que será ejecutado.

ROSCOE incluye un lenguaje de programación interactiva. Todos los comandos de ROSCOE y sus facilidades están disponibles para el uso de programas dentro del lenguaje de programación interactiva.

ROSCOE proporciona asistencia en línea e información para usuarios a través de la facilidad de ayuda (HELP). La facilidad de

ayuda (HELP) puede ser utilizada por usuarios nuevos o experimentados para obtener el sumario de la información acerca de la mayoría de los comandos y facilidades.

El ambiente de ROSCOE consiste del espacio de trabajo activo, la bibliotaca de ROSCOE y el usuario de terminales para introducir, editar y desplegar datos.

El espacio de trabajo es una área de trabajo temporal para nuestra sesión de trabajo . Todos los datos introducidos en la terminal son puestos en el espacio de trabajo activo, en donde, éstos pueden ser editados, almacenados en la bibliotaca de de ROSCOE, impresos, eliminados y sometidos al sistema operativo.

El sistema de control de información a usuarios (CICS/VS) es un módulo orientado a sistemas de comunicación de las base de datos. CICS también lo podemos ver como un monitor de teleproceso en línea para apoyar a bases de datos, en la interface entre el sistema operativo y los usuarios de programas de aplicación. El módulo CICS/VS (CICS/OS/VS o CICS/DOS/VS) ejecuta numerosas funciones esenciales para satisfacer la comunicación de bases de datos en tiempo real. Sus principales funciones se listan en seguida :

1. Proporciona respuesta rápida a terminales simultáneas activas en línea.
2. Controla una red de telecomunicación de dispositivos mixtos.
3. Maneja gereralmente una amplia miscelánea de transacciones para el cumplimiento de las demandas de los programas de aplicación.
4. Controla el acceso a la base de datos.

5. Maneja eficientemente los recursos del sistema, tales como, el almacenamiento dinámico para mantener el sistema en operación continua.

6. Asigna prioridades para optimizar el uso de la unidad de procesamiento central.

IV.1.4 Arquitectura interna

Si bien la eficiencia en uso de recursos de los diferentes sistemas, no es considerada el punto principal en su selección, dados los grandes volúmenes de información procesados en los sistemas de la Tesorería del Distrito Federal es necesario que la arquitectura en que se instrumente el modelo conceptual de la base de datos garantice un rendimiento aceptable. Además, el sistema debe de poder funcionar compartiendo las bases de datos desde varios procesadores simultáneamente actualizando o extrayendo información.

Es importante la separación de datos y sus descripciones e índices para facilitar la modificación de la estructura de las bases de datos sin necesidad de reorganizarlas. Para ello, es necesario un manejo eficiente de los índices en los directorios de las bases de datos.

Es difícil encontrar en el mercado reúnidos en un solo sistema de bases de datos estas características. Después de investigar en el mercado respecto a los productos existentes se llegó a la conclución que la línea DATACOM/DB es la idónea para las necesidades en materia de administración de bases de datos y desarrollo de sistemas en la Tesorería del Didtrito Federal, por satisfacer de una manera más

completa los requerimientos funcionales especificados.

IV.2 Descripción del sistema manejador de bases de datos DATACOM/DB

Este sistema pretende instrumentar en la práctica los conceptos teóricos de administración de bases de datos relacionales. Además utiliza la estructura de listas invertidas y separación de datos e índices, soportando la representación tabular del modelo relacional e instrumenta el concepto de "vistas de datos", permitiendo una imágen lógica de una tabla independiente de su almacenamiento físico.

IV.2.1 El diccionario de datos (DATADICTIONARY)

Es una base de datos con las entidades básicas de información de las bases de datos y sus vistas de usuario, aplicaciones, procesos, terminales, reportes, pantallas, personal y autorizaciones de acceso ; el usuario puede expander las relaciones y entidades que desee almacenar con el diccionario. Esta herramienta sirve de apoyo al diseño y normalización de las bases de datos relacionales al inicio de un proyecto, y controla el efecto de los cambios efectuados sobre los datos.

El diccionario permite la generación automática de código y tarjetas de control para la inclusión de definiciones de datos en programas.

IV.2.2 El núcleo manejador de bases de datos (DATACOM/DB)

Contiene las facilidades requeridas de recuperación/respaldo y protección de la integridad de la base de datos.

IV.2.3 El generador de reportes (DATAREPORTER)

Utiliza un lenguaje sencillo de formato libre para seleccionar, manipular y formatear datos. Al utilizar las definiciones del diccionario de datos a tiempo de ejecución, la generación de reportes utiliza la última versión de una definición de datos sin necesidad de recompilar los programas de (DATAREPORTER). Puede ser también utilizado para archivos convencionales.

IV.2.4 El generador de consultas en línea (DATAQUERY)

Instrumenta las funciones relacionales para la extracción de datos y por medio de un lenguaje de unos cuantos verbos permite seleccionar información parcial o totalmente. Cuenta con facilidades de formateo de reportes, cortes y totales.

IV.2.5 Interface a los lenguajes de programación.

Basado en el preprocesador METACOBOL se cuenta con un lenguaje COBOL/DL que además de permitir desde programas COBOL el acceso tanto a las definiciones de vistas de datos relacionales almacenadas en el diccionario como a las funciones del manejador de las bases de datos, costituye un lenguaje estructurado que resuelve muchas de las deficiencias de COBOL, como por ejemplo, el acceso a la base de datos, actualizaciones en línea, etc.

IV.2.6 Ambiente interactivo para desarrollo de sistemas.

Dentro del sistema DATACOM/DB se cuenta con IDEAL un sistema de sistemas de programación para el desarrolo de aplicaciones interactivas serviciales a usuarios ; diseñado para el desarrollo de aplicaciones comerciales en línea o batch, a DATACOM/DB y a procesamiento de transacciones. IDEAL provee la integración completa de las siguientes funciones : diccionario de datos, administración de bibliotecas, programación, bases de datos, comunicación de datos, reportes, edición y servicios de utileria, se cuenta con un lenguaje de "cuarta generación", lenguaje para la definición de procesamientos (LDP) ; cuenta con verbos para la manipulación de datos en las tablas relacionales en base a la descripción de dichos datos, sin necesidad de la especificación de las operaciones registro por registro. Las definiciones de formatos de reportes, pantallas y las vistas de datos son independientes de los programas de manera que una misma definición de un reporte o pantalla puede servir en muchos programas y ser modificados sin reprogramación alguna. Esta separación lleva la independencia lógica un paso más allá de la simple independencia entre programas y bases de datos.

IV.2.7 Interacción con el usuario.

Los productos de la línea DATACOM/DB presentan una integridad conceptual en su diseño y su instrumentación del modelo relacional que hace que el funcionamiento del sistema sea externamente simple y fácil de entender y usar.

IV.2.8 El ambiente interactivo para el desarrollo de sistemas

IDEAL cuenta con un uso extenso de menús para definiciones de datos y vistas de usuario, reportes, pantallas, acceso al diccionario de datos y bibliotecas de programas, además del lenguaje LDP ya mencionado, IDEAL cuenta así mismo con facilidades para depuración de programas en línea, y un editor sensible a la estructura sintáctica del lenguaje LDP, que además valida contra el diccionario de datos el acceso a las diferentes entidades de las aplicaciones al momento de ser tecleadas. Finalmente IDEAL maneja una interface con el usuario que divide la pantalla en varias "ventanas" permitiendo la interacción simultánea con varias de las facilidades del sistema.

IV.2.9 Arquitectura interna. DATACOM/DB cuenta con una arquitectura de separación de índices y datos, y mecanismos de reorganización automática y reutilización del espacio que instrumentan de hecho la estructura de árboles-b (es la forma de representar la jerarquía de apuntadores de árboles balanceados). Existe una fuerte integración entre los productos que utilizan herramientas comunes de interface con el ambiente operativo (sistema operativo, monitor de teleproceso y procesos en línea o batch) de manera que el funcionamiento de los sistemas es independiente de dicho ambiente, con la consiguiente compatibilidad de CICS y con los distintos sistemas operativos de IBM.

El consumo de recursos de IDEAL es considerable comparado con otras herramientas, lo cual consideramos aceptable pues ofrece herramientas que incrementan fuertemente la productividad de desarrollo y

mantenimiento de sistamas.

IV.3 Estructura de la base de datos para DATACOM/DB

La estructura de la base de datos la podemos describir en la Fig.47.

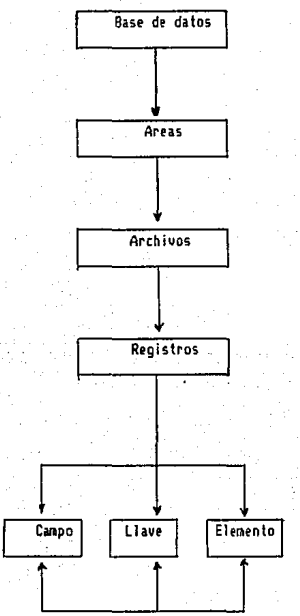


Fig.47 Estructura de DATACOM/DB.

DATACOM/DB permite hata 999 bases de datos en una base de información ; cada base de datos está formada por una área de

índices y una o más áreas de datos en las cuales los tipos de registros son almacenados. Los tipos de registros son accesados por llaves colocadas en el índice. Las llaves identifican el archivo lógico al cual pertenece el registro y los apuntadores del registro lógico dentro del área de datos. Los apuntadores establecen relaciones a través del tipo de registro en la base de datos ; de este núcleo los programas pueden ser escritos para ser sensitivos sólo a lo lógico, en lugar de lo físico en cuanto a la estructura de la información. Los cambios en la relación son manejados a través de una simple función ejecutada por el administrador de la base de datos (ABD).

Cada base de datos DATACOM/DB permite hasta 240 areas, hasta 240 archivos lógicos, tipos múltiples de llave por archivo lógico, hasta 999 tipos de llave por base de datos, hasta 180 caracteres en total para la longitud de la llave, campos múltiples concatenados por llave, cualquier número de campos contiguos en un elemento y elementos múltiples por archivo.

DATACOM/DB funciona con computadoras grandes, en un embiente de desarrollo de aplicaciones, centros de información, centros de producción y requiere de analistas, programodores, administradores de bases de datos y opradores de equipo.

IV.4 Estructura de un programa de aplicación en IDEAL

En la presente sección se dan las componentes que formarán parte en el desarrollo de un programa de aplicación así como una breve

descripción de cada uno de ellos.

Los componentes que podemos asociar a un programa de aplicación en IDEAL pueden ser :

Identificación, recursos, definición de parámetros, campos de trabajo (la sección de "working storage" de COBOL) y el lenguaje para la definición de procedimientos del programa.

IV.4.1 Identificación del programa

Es la parte que especifica la información que identifica al programa ; la proporcionan los usuarios y el propio sistema. El nombre del programa se encuentra formado de uno hasta ocho caracteres, los usuarios proporcionan el lenguaje que será utilizado y la descripción que especifica los objetivos y funciones del programa.

El sistema proporciona la fecha de creación con el formato MM/DD/YY (mes/día/año), fecha de la última modificación y compilación con el formato MM/DD/YY y la hora de la última modificación y compilación con el formato HH/MM/SS (hora/minuto/segundo).

En la Fig.48 se observa la información requerida de un programa en IDEAL y en la Fig.49 se presentan los datos prorporcionados por el usuario y sistema para la identificación del programa. Se puede observar que, gracias al editor sensible a la sintaxis, el programador sólo teclea aquellos campos que varían de un programa a otro.

```
======>
======>
======>
```

```
IDEAL : PGM IDENTIFICATION                                    FILL-IN
PROGRAM _______
DATE CREATED __/__/__  DATE LAST MODIFIED __/__/__  DATE LAST COMPILED__/__/__
                       TIME LAST MODIFIED __:__     TIME LAST COMPILED __:__
RUN STATUS _________
SHORT DESCRIPTION _____________________
LANGUAGE: IDEAL    TARGET DATE __ __ __ ACTUAL DATE __ __ __
DESCRIPTION:
  ____________________________________________________________________
  ____________________________________________________________________
  ____________________________________________________________________
  ____________________________________________________________________
  ____________________________________________________________________
```

FIG.48 INFORMACION REQUERIDA PARA LA IDENTIFICACION DE UN PROGRAMA EN IDEAL.

```
===>
===>
===>
```

```
IDEAL : PGM IDENTIFICATION                                    FILL-IN
PROGRAM CUOTA1
DATE CREATED 04/17/86  DATE LAST MODIFIED 03/16/87  DATE LAST COMPILED 08/17/87
                       TIME LAST MODIFIED 09:17     TIME LAST COMPILED 08:09
RUN STATUS NON-SHARED
SHORT DESCRIPTION ACTUALIZA LA ENT. PUEST.
LANGUAGE: IDEAL    TARGET DATE __ __ __ ACTUAL DATE __ __ __
DESCRIPTION:
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
    ____________________________________________________________
```

FIG.49 INFORMACION COMPLETA PARA LA IDENTIFICACION DE UN PROGRAMA EN IDEAL.

IV.4.2 Recursos del programa

Los recursos de un progrmama es aquello que ayuda a la construcción de los programas y pueden ser : vista lógica de datos (DATAVIEW), pantallas (PANELS), reportes (REPORTS) y subprogrmas (SUBPROGRAMS).

IV.4.2.1 Vista lógica de datos

El usuario de IDEAL percibe los datos como una colección nombrada de campos como si fuera una tabla que se encuentra almacenada en el diccionario de datos. La vista de datos es la comunicación entre el programa y la base de datos de una manera relacional. En la Fig.50 se da un ejemplo de datos, donde se describe la información asignada a los empleados de cierta institución educacional.

Nro-Clave	Nombre	Dirección	Sexo	Salario	Categoría

Fig.50 Vista lógica de datos de empleados de una determinada institución educacional.

Podemos agrupar los datos correspondientes de un empleado en la vista lógica de datos llamada EMPLEADO, como se muestra a continuación.

```
1 EMPLEADO
  2 NUMERO          U     5
  2 NOMBRE          X    24
  2 DIRECCION       X    30
  2 SEXO            B     1
  2 SALARIO         N     8
  2 CATEGORIA       X    20
```

En donde U significa campo numérico sin signo, X significa campo alfanumérico, B significa compo booleano y N significa campo numérico.

IV.4.2.2 Pantallas

En este lenguaje se nos proporciona la facilidad para tener acceso a varios tipos de pantallas a través de menús. Por ejemplo, pantalla de menús, pantalla para recibir información, pantalla para seleccionar comandos, pantalla de despliegue y pantalla de ayuda para creación y mantenimiento de pantallas. Después de que éstas son construídas, las definiciones y salidas de ellas puden ser impresas, probadas y editadas para el uso en línea.

Las funciones ofrecidas por IDEAL para la generación y mantenimiento de pantallas se muestra en la Fig.51.

```
===)
===)
===)
```

```
IDEAL: PANEL MAINTENANCE      PNL                    SYS: SAD    MENU
ENTER DESIRED OPTION NUMBER===>        THERE ARE   7  OPTIONS IN THIS MENU:
  1. EDIT/DISPLAY        - EDIT OR DISPLAY A PANEL
  2. CREATE              - CREATE A PANEL
  3. PRINT               - PRINT A PANEL
  4. DELETE              - DELETE A PANEL
  5. MARK STATUS         - MARK PANEL STATUS TO PRODUCTION OR HISTORY
  6. DUPLICATE           - DUPLICATE A PANEL TO NEXT VERSION OR NEW NAME
  7. DISPLAY INDEX       - DISPLAY INDEX OF PANEL NAMES IN SYSTEM
```

FIG.51 MANTENIMIENTO DE PANTALLAS PARA MENUS.

Los elementos utilizados en pantallas suelen ser : Identificación de pantalla, Parámetros de pantalla, Creación del diseño de pantalla, Tabla sumario de campos de pantalla, Pantalla de campo extendido y Pantalla en su forma acabada.

IV.4.2.2.1 Identificación de pantalla

La información requerida para identificación de pantallas es semejante a la información que se requiere cuando un programa se crea por vez primera, como se ilustra en la Fig.52.

```
===)
===)
===)

_____________________________________________________________________
IDEAL : PANEL IDENTIFICATION                        SYS : DOC FILL-IN
PANEL NAME _______
DATE CREATED   / /    DATE LAST MODIFIED   / /
                      TIME LAST MODIFIED   :
SHORT DESCRIPTION _____________________________
DESCRIPTION:
    ___________________________________________________________
    ___________________________________________________________
    ___________________________________________________________
    ___________________________________________________________
    ___________________________________________________________
```

FIG.52 INFORMACION REQUERIDA PARA IDENTIFICAR PANTALLAS.

IV.4.2.2.2 Parámetros de pantalla

Son los campos para asignar símbolos que tienen funciones específicas en la creación de pantallas ; por ejemplo :

1) Creación de campos de pantalla.

2) Eliminación de campos de pantalla.

3) Repetición de campos de pantalla.

4) Llenado de campos de las posiciones en desuso que traen los campos de entrada.

5) Llenado de campos que fueron definidos con espacios cuando la pantalla se despliega con la instrucción TRANSMIT o REFRESH.

6) Despliegue de caracteres que no son contemplados por IDEAL.

7) Señalamiento de los campos que tienen error de entrada.

8) Conversión de textos a mayúsculas antes de la ejecucion (ya que los textos pueden contener letras mayúsculas y/o minisculas).

9) Utilización de campos nulos.

10) Indicacion de los campos que serán suministrados y omitidos por el usuario.

11) Manejo e errores.

12) Manejo de teclas para ir al tope y fondo de la pantalla.

13) Manejo de teclas para avanzar y retroceder pantallas.

14) Manejo de la tecla para proporcionar la pantalla de ayuda.

15) Manejo de la tecla para mostrar errores.

16) Manejo de la tecla para el despliegue de la pantalla de ayuda asociada a la pantalla original.

17) Manejo de la tecla para proporcionar sufijos y prefijos a la pantalla que contiene información común a diferentes pantallas que pueden ser utilizadas por otros usuarios.

La Fig.53 se ilustra la lista completa de parámetros que se asocian a la pantalla.

```
===>
===>
===>

--------------------------------------------------------------------------------
PANEL:          CUOI            VERSION: 002 STATUS:  TEST
PANEL PARAMETERS:
START      FIELD SYMBOL ¬       NEW    FIELD SYMBOL +
END        FIELD SYMBOL ;       DELETE FIELD SYMBOL *
REPEATING GROUP SYMBOL @
INPUT  FILL CHARACTER  S        (S=SPACE, L=LOWVAL, Z=ZEROS, U=_, OTHER=ITSELF)
OUTPUT FILL CHARACTER  U        (S=SPACE, L=LOWVAL, U=_, OTHER=ITSELF)
NON-DISPLAY CHARACTER  S        (S=SPACE, OTHER=AS SPECIFIED)
ERROR FILL CHARACTER   *        (AS SPECIFIED)
CASE TRANSLATION       U        (U=UPPER, M=MIXED)
VALUE FOR NULL FIELD   L        (S=SPACE, L=LOWVAL, I= INPUT FILL CHARACTER)
REQUIRED               N        (Y=YES, N=NO)
ERROR HANDLING         B        (N=NONE, *=FILL W/ERRORFILL, H=HIGH INTENSITY,)
                                (B=BOTH: H IF ILLEGAL VALUE & * IF RQD MISSING)
PF1=HELP, PF3=CLARIFY  Y        (Y=YES, N=NO)
PF7=SCR -, PF8=SCR +   N        (Y=YES, N=NO)
PF10=SCR TOP, PF11=SCR BOT
HELP PANEL NAME  --------       VERSION ___
PREFIX PANEL NAME --------      VERSION ___
SUFFIX PANEL NAME --------      VERSION ___
```

FIG.53 LISTA COMPLETA DE PARAMETROS ASOCIADOS A PANTALLAS.

IV.4.2.2.3 Creación del diseño de pantallas

Por medio del comando LAYOUT se nos presenta una pantalla con líneas en blanco. En ellas se incrustan los campos con las posiciones exactas que serán utilizados por el programa de aplicación.

La Fig.54 presenta la pantalla en blanco, después de haber dado el comando LAYOUT y en la Fig.55 se muestran los camos incrustados en las líneas en blanco de la misma pantalla.

```
===)
===)
===)

------------------------------------------------------------------------------
IDEAL: PANEL LAYOUT                                              FILLN-IN
START     FIELD SYMBOL ¬       NEW    FIELD SYMBOL +
END       FIELD SYMBOL ;       DELETE FIELD SYMBOL *
                               REPEATING GROUP SYMBOL @
------------------------------------------------------------------------------

------------------------------------------------------------------------------
```

FIG.54 FORMATO DE CREACION DE PANTALLAS.

```
===>
===>
===>

PANEL:          CUO1            VERSION: 002 STATUS:  TEST
LAYOUT:
```

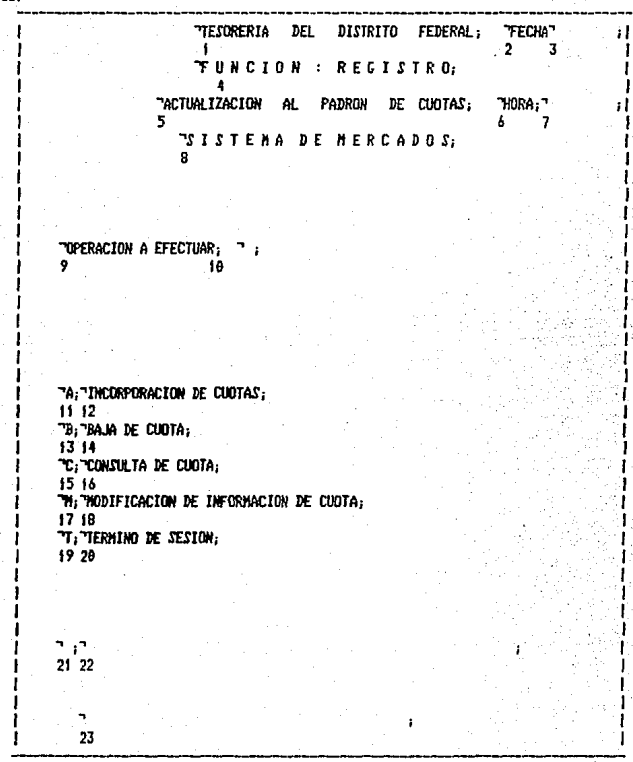


FIG.55 CAMPOS INCRUSTADOS EN LA CREACION DE PANTALLAS .

IV.4.2.2.4 Tabla sumario de campos de pantalla

Es una tabla asociada a la pantalla de creación que presenta los campos en ésta y las columnas en las cuales se les asocia la información correspondiente a cada campo ; la tabla es desplegada en la pantalla con el comando SUMMARY, la cual presenta la información básica de cada campo, como por ejemplo : nombre de campo, atributos de pantalla, tipo de campo, lugares para enteros y decimales, ocurrencias y comentarios.

La Fig.56 Presenta la tabla sumario de campos asociada a la pantalla de creación de la Fig.55.

```
===>
===>
===>
```

```
PANEL:          CUO1              VERSION: 002 STATUS:  TEST
FIELD SUMMARY TABLE:
```

SEQ	LV	FIELD NAME	ATTR	T	LEN	IN.DP	OCC	COMMENTS
1	2		PSH	X	36			TESORERIA DEL DIS
2	2		PSL	X	5			FECHA
3	2	FECHA	PAH	X	8			
4	2		PSH	X	33			F U N C I O N : R E
5	2		PSH	X	41			ACTUALIZACION AL
6	2		PSL	X	4			HORA
7	2	HORA	PSH	X	8			
8	2		PSH	X	35			S I S T E M A D E M
9	2		PSL	X	20			OPERACION A EFECTUAR
10	2	OPCION	UAH	X	1			
11	2		PAH	X	1			A
12	2		PAL	X	23			INCORPORACION DE CUOT
13	2		PAH	X	1			B
14	2		PAL	X	13			BAJA DE CUOTA
15	2		PAH	X	1			C
16	2		PAL	X	17			CONSULTA DE CUOTA
17	2		PAH	X	1			M
18	2		PAL	X	36			MODIFICACION DE INFOR
19	2		PAH	X	1			T
20	2		PAL	X	17			TERMINO DE SESION
21	2	MENSAJE1	PAH	X	1			
22	2	MENSAJE2	PAH	X	47			
23	2	MENSAJE3	PSH	X	43			

```
        LEGEND:
                SEQ=SEQUENCE NUMBER
ATTR=SCREEN ATTRIBUTES: U=UNPROT  H=HIGHLIGHT  A=327X ALPHANUMERIC
                        P=PROT    I=INVISIBLE  N=327X NUMERIC
                        S=SKIP    L=LOW-LIGHT  E=ENSURE INPUT
                                               C=CURSOR
T=FIELD TYPE: X=ALPHANUMERIC, N=NUMERIC
IN.DP=INTEGER-PLACES.DECIMAL-PLACES
OCC=NUMBER OF OCCURRENCES
```

FIG.56 TABLA SUMARIO DE CAMPOS ASOCIADA A LA PANTALLA DE LA FIG.55.

los campos presentados en la tabla de la Fig.56, con las siglas PSH

significa que los campos tienen los atributos de : protección, capacidad de salto y alta intensidad. Los campos presentados con las siglas PSL sifnifica que los campos tienen el atributo de : protección, capacidad de salto y baja intensidad. Los campos presentados con las siglas PAH tienen atributos de : protección, ser alfanuméricos y alta intensidad. Los campos presentados con las siglas UAH significa que tienen los atributos de : desprotegido, alfanumérico y con alta intensidad.

De la Fig.56 notamos que los atributos de campos, son una combinación en el ambiente de atributos de pantalla. Las combinaciones de atributos que pueden ser especificadas por los usuarios son las siguientes :

Para campos protegidos :

P [A / S*] [H / L / I] [C] [E]

Para campos desprotegidos :

[/ A /] [H / L]

U [N] [I] [C] [E]

En donde :

U es un campo desprotegido y puede ser insertado, modificado o eliminado.

P es un campo protegido que solamente es desplegado. Un campo protegido puede ser insertado, pero no puede ser modificado o eliminado.

S es un campo, el cual es saltado por el cursor ; es decir, no puede ser accesado.

H es un campo que al ser desplegado, sus caracteres tienen alta intensisdad.

I es un campo invisible, el cual los caracteres no son desplegados.

L es un campo que es desplegado con regular intensidad (baja intensisdad).

F es un campo que será tratado como si fuera insertado en la transacción actual.

A es un campo que acepta cualquier caracter.

N es un campo que acepta hasta 327 caracteres.

C es un campo que contiene el cursor, cuando se despliega la pantalla.

Toda la información listada para la tabla sumario de campos es siempre desplegada en la definición de campo extendido con el comando

```
{ FIELD  { name   }  }
{        { number }  }
{ NEXT               }
{ PREVIOUS           }
```

Esto permite que el usuario mantenga la definición de campo extendido y no trabaje con la tabla sumario de campos.

IV.4.2.2.5 Pantalla de campo extendido

La tabla sumario de campos es asociada a la definición de campo extendido y se utiliza para incluir información opcional detallada para cada campo específico, invalidar opciones en los parámetros de la pantalla y en la tabla sumario de campos, seleccionar la opción de validación y reglas de elección. Cuando la definición de campo extendido se encuentra desplegada, la línea del despliegue en la pantalla contiene al campo requerido, señalado por dos rayas horizontales (como el de la Fig.56), el cual se presenta desplegado en el tope de la región con una serie de asteriscos directamente posicionados debajo del campo, como se ilustra en la Fig.57.

```
===>
===>
===>

-----------------------------------------------------------------------
IDEAL: EXTENDED FIELD DEFN.   PNL CUD1 (002) TEST          SYS: SAD FILL-IN
                      TESORERIA   DEL   DISTRITO   FEDERAL      FECHA
                  1                                          2     ********
----------------------------------------------------------------- ....+....
 FIELD NAME       FECHA            FIELD NUMBER   3   FIELD LENGTH   8
 COMMENTS         ___________________
 TYPE             X        (X=ALPHANUMERIC, N=NUMERIC, G=GROUP)
 ATTRIBUTE        PAH      (U=UNPROT  H=HIGHLIGHT          A=327X ALPHA     )
                           (P=PROT    I=INVISIBLE          N=327X NUMERIC   )
                           (S=SKIP    L=LOWLIGHT           E=ENSURE RECEIVED)
                           (C=CURSOR                                        )
 COLOR            N        (N=NEUTRAL, B=BLUE, R=RED, P=PINK, G=GREEN,)
                           (T=TURQUOISE, Y=YELLOW, W=WHITE/BLACK      )
 EX HIGHLIGHTING  N        (N=NONE, B=BLINK, R=REVERSE VIDEO, U=UNDERSCORE)
 OUTPUT FILL CHAR U  *     (S=SPACES, L=LOWVAL, U=_, OTHER=ITSELF)
 ERROR HANDLING   B  *     (N=NONE, *=FILL *, H=HIGHLIGHT,)
                          (B=BOTH: H IF ILLEGAL VALUE & * IF REQD IS MISSING)
-----------------------------------------------------------------------
```

FIG.57 DEFINICION DE CAMPO EXTENDIDO DEL CAMPO 3 DE LA TABLA SUMARIO DE CAMPOS.

En otras palabras, en la tabla sumario de campos se selecciona el campo específico que será desplegado en la definición de campo extendido. Cualquier modificación a los atributos del campo en la definición de campo extendido se refleja en la tabla sumario de campos y viceversa.

IV.4.2.2.6 Pantalla en su forma acabada

Es la representación de la creación del diseño de la pantalla en su forma acabada que será accesada por el programa para ser probada.

IV.4.2.3 Reportes

La facilidad de la definición de reportes (FDR) proporciona medios para creación, mantenimiento y prueba de definición de reportes. La facilidad de la definición de reportes permite a los usuarios de IDEAL definir pantallas, parámetros, campos, títulos en columna para el reporte.

Los elementos para la definición de reportes pueden ser : Identificación del reporte, Parámetros del reporte, Títulos de página del reporte, Detalle del reporte y Títulos en columna para reportes.

IV.4.2.3.1 Identificación del reporte

Es la parte que especifica la información que identifica al reporte. Esta información es análoga a la requerida para un programa en IDEAL, reemplazando solamente el nombre del programa por el nombre del reporte, como se muestra en la Fig.58.

```
===>
===>
===>
_____________________________________________________________________

IDEAL : RPT  IDENTIFICATION                          SYS : DOC   FILL IN
REPORT NAME  _______
DATE CREATED __/__/__  DATE LAST MODIFIED __/__/__
SHORT DESCRIPTION _____________________________
DESCRIPTION:
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
```

FIG. 58 INFORMACION REQUERIDA PARA IDENTIFICAR REPORTES.

Las funciones ofrecidas por IDEAL para definir y mantener reportes son presentadas en la pantalla de mantenimiento de menús para reportes, por medio del comando REPORT, las cuales se muestran en la Fig.59.

```
===>
===>
===>

--------------------------------------------------------------------------------

IDEAL: REPORT MAINTENANCE     RPT                          SYS: SAD    MENU

ENTER DESIRED OPTION NUMBER ===>      THERE ARE    7  OPTIONS IN THIS MENU:

  1. EDIT/DISPLAY        - DISPLAY OR EDIT A REPORT DEFINITION
  2. CREATE              - CREATE A REPORT DEFINITION
  3. PRINT               - PRINT A REPORT DEFINITION
  4. DELETE              - DELETE A REPORT DEFINITION
  5. MARK STATUS         - MARK REPORT STATUS TO PRODUCTION OR HISTORY
  6. DUPLICATE           - DUPLICATE REPORT TO NEXT VERSION OR NEW NAME
  7. DISPLAY INDEX       - DISPLAY INDEX OF REPORT NAMES IN SYSTEM
```

FIG. 59 MANTENIMIENTO DE REPORTES PARA MENUS.

Para crear un reporte se utiliza el comando CREATE REPORT [nombre] o la opción equivalente que se muestra en la Fig.59.

IV.4.2.3.2 Parámetros del reporte

Los utilizaremos opcionalmente para seleccionar el formato en la creación de reportes en cuanto a : longitud y ancho del reporte en la página, el espaciado entre líneas y columnas, títulos en columna y la manera en que son resaltados, interrupción del control, indicación de continuación de grupo, definicion de títulos, sumario de información y la especificación de fecha de página.

El comando PARAMETERS presenta la pantalla con los parámetros que se utilizan en los reportes, los cuales se muestran en la Fig.60.

```
===>
===>
===>
--------------------------------------------------------------------------------
IDEAL : REPORT PARAMETER          RPT     MERCADO (001) TEST  SYS: DOC FILL-IN

  LINES PER PAGE ON PRINTOUT          60   (1 THRU 99)
  REPORT WIDTH                        132  (40 THRU 230)
  SPACING BETWEEN LINES               1    (1 THRU 3)
  SPACING BETWEEN COLUMNS             02   (0 THRU 66 OR A=AUTOMATIC)
  SUMMARIES ONLY                      N    (Y=YES,N=NO)
  COLUMN HEADINGS DESIRED             Y    (Y=YES,N=NO)
  COLUMN HEADINGS INDICATION          U    (U=UNDERSCORE,N=NONE,D=DASHES)
  CONTROL BREAK HEADING               Y    (Y=YES,N=NO)
  CONTROL BREAK FOOTING               Y    (Y=YES,N=NO)
  GROUP CONTINUATION AT TOP OF PAGE   Y    (Y=YES,N=NO)
  ANNOTATED COUNT IN CONTROL FOOTINGS Y    (Y=YES,N=NO)
  REPORT FINAL SUMMARY TITLE          Y    (Y=YES,N=NO)
    SPACING BEFORE SUMMARY            P    (1 THRU 9 = LINES,P=NEW PAGE)
    TITLE
  DATE
    POSITION          NO   (NO=NONE, BR=BOT.RIGHT, BL=BOT.LEFT , BC=BOT.CTR.,
                            TR=TOP RIGHT, TL=TOP LEFT, TC=TOP  CENTER)
    FORMAT            MM/DD/YY
  PAGE NUMBERS
    POSITION          TR
    FORMAT            H    (D=DIGITS ONLY, H=WITH HYPHENS, P=PAGE       NNN)
  PAGE HEADING
    HEADING
    POSITION          C    (C=CENTER, L=LEFT JUSTIFY, R=RIGHT JUSTIFY)
--------------------------------------------------------------------------------
```

FIG. 60 PARAMETROS PARA REPORTES.

IV.4.2.3.3 Títulos de página del reporte

Se utiliza opcionalmente para definir títulos de página más elaborados que los que pueden ser definidos en los parámetros del reporte. La opción o localización de títulos de página se especifica siempre en estos parámetros. Los componentes de la definición de títulos de página para reportes pueden ser : nombres

de campos, literales, funciones y expresiones aritméticas que pueden ser utilizadas como parte de un título de página. Los nombres de campos serán conocidos por la aplicación en IDEAL que produce el reporte ; es decir, serán nombres de campos en vista lógica de datos, pantallas, datos de trabajo, o parametros.

Para proporcionar la información requerida en la definición del título de página de reportes, se hace a través del comando HEADING, el cual proporciona una pantalla para incluir el formato del título que se requiere , tal como se muestra en la Fig.61.

```
===>
===>
===>

_____________________________________________________________________

IDEAL: RPT PAGE HEADING       RPT REPOESEC (001) TEST       SYS: SAD  FILL-IN

                                                 COLUMN
                                                 W
FIELD NAME, LITERAL, FUNCTION                    ID TAB  EDIT PATTERN  COMMAND
OR ARITHMETIC EXPRESSION                         TH
==================== TOP ======================= == ===  ============  =======
---------------------------------------------------- --  ---  ----------  -------
_____________________________________________________ _  ___  __________  _______
---------------------------------------------------- --  ---  ----------  -------
_____________________________________________________ _  ___  __________  _______
---------------------------------------------------- --  ---  ----------  -------
_____________________________________________________ _  ___  __________  _______
---------------------------------------------------- --  ---  ----------  -------
_____________________________________________________ _  ___  __________  _______
---------------------------------------------------- --  ---  ----------  -------
_____________________________________________________ _  ___  __________  _______
---------------------------------------------------- --  ---  ----------  -------
_____________________________________________________ _  ___  __________  _______
---------------------------------------------------- --  ---  ----------  -------
==================== BOTTOM ==================== == ===  ============  =======
```

FIG.61 INFORMACION REQUERIDA PARA TITULOS DE PAGINA PARA REPORTES.

En la pantalla de la Fig.61 observamos que se encuentra dividida en cuatro regiones, en la primera se proporciona el título de página

para el reporte, en la segunda se introduce la información relacionada con el formato del reporte, en la tercera se especifica el formato en el cual serán impresos los valores de los campos y en la cuarta región los usuarios introducen comandos de IDEAL para la edición de esta información.

IV.4.2.3.4 Detalle del reporte

El comando DETAIL nos proporciona una pantalla en blanco para introducir información respecto a las líneas de detalle ; esta información puede incluir : clasificación de campos, interrupción de control en cuanto al requerimiento de cortes y sumario de funciones. La información requerida para el detalle de un reporte se presenta en la Fig.62.

```
===>
===>
===>
____________________________________________________________________________
IDEAL: RPT DETAIL DEFN.       RPT REPOESEC (001) TEST        SYS: SAD   FILL-IN
FIELD NAME, LITERAL        SORT   BREAK  FUNCTION COLUMN
FUNCTION, OR                L A   L S I  T M M A  H W
ARITHMETIC EXPRESSION       V /   V K N  O A I V  D ID TAB EDIT PATTERN COMMAND
                            L D   L P D  T X N G  G TH
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
========== TOP ==========   = =   = = =  = = = =  = == === ============ ======
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
-------------------------   - -   - - -  - - - -  - -- --- ------------ ------
======== BOTTOM =========   = =   = = =  = = = =  = == === ============ ======
```

FIG.62 INFORMACION REQUERIDA PARA EL DETALLE DE REPORTES.

IV.4.2.3.5 Títulos de columna para reportes

El comando COLUMN presenta una pantalla en blanco para introducir títulos en columna que reemplazarán los valores por omisión (default) que fueron definidos en los parámetros del reporte. Esta pantalla la podemos ver como una extensión lógica a la pantalla de detalle para reportes, como se muestra en la Fig.63.

```
===>
===>
===>

_____________________________________________________________________________

IDEAL: RPT COLUMN HEADINGS    RPT REPOESEC (001) TEST        SYS: SAD   DISPLAY
FIELD NAME, LITERAL       COLUMN
FUNCTION OR               H U
ARITHMETIC EXPRESSION     D ID TAB HEADINGS                              COMMAND
                          G TH
__________________________ _ __ ___ ____________________________________ _______
========== TOP ========== = == === ==================================== =======
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
__________________________ _ __ ___ ____________________________________ _______
========= BOTTOM ========= = == === ==================================== ======
```

FIG.63 INFORMACION REQUERIDA PARA TITULOS DE COLUMNA.

En esta figura la sigla HDG especifica si el título de columna del reporte será o no será impresa. Y las siglas HEADINGS especifica a la cadena que será introducida en el título de columna para reportes.

IV.4.3 Definición de parámetros

El comando PARAMETER proporciona una pantalla en blanco a la que se le proporcionará los nombres y descripciones de los datos que serán registrados cuando un determinado programa invoca a un subprograma. Estos se especifican para ser llamados por el programa. La pantalla para la definición de parámetros se presenta en la Fig.64.

```
===>
===>
===>

_____________________________________________________________________

IDEAL: PARAMETER DEFINITION   PGM PGAGUA08 (001) TEST       SYS: SAD   DISPLAY
LEVEL FIELD NAME          T CHR/DGTS OCCUR  U   COMMENTS/DEP OR/COPY   COMMAND
----- ------------------- - -------- -----  -   -------------------- ------
===== ======= TOP ======= = ======== =====  =   ==================== ======
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
----- ------------------- - -------- -----  -   -------------------- ------
===== == BOTTOM ========= = ======== =====  =   ==================== ======
```

FIG. 64 DEFINICION DE LOS PARAMETROS DEL PROGRAMA.

En la pantalla de parámetros del programa se especifica la jerarquía (mediante un número) para el orden de los campos ; el nombre de campo normalmente es un nombre de campo o grupo ; el tipo del campo podría ser : alfanumérico, numérico, de condición y

booleano. Además se especifica el número de caracteres y enteros del campo, el número de ocurrencias del campo y comentarios acerca de los campos.

IV.4.4 Campos de trabajo

Por medio del comando WORK se proporciona una pantalla en blanco para introducir los nombres y descripciones de los datos que serán utilizados por un programa de aplicación. Para la definición y mantenimiento de éstos se hace uso de la pantalla que se presenta en la Fig.65.

```
===>
===>
===>

IDEAL: WORKING DATA DEFN.     PGM PGAGUA88 (001) TEST        SYS: SAD   FILL-IN
LEVEL FIELD NAME          T CHR/DGTS OCCUR VALUE/COMMENTS/DEP ON/COPY   COMMAND
===== ======= TOP ======= = ======== ===== ============================ ======
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
----- ---------------- - -------- ----- ---------------------------- ------
===== ======BOTTOM======= = ======== ===== ============================ ======
```

FIG.65 DEFINICION DE DATOS DE TRABAJO DEL PROGRAMA.

En esta pantalla se especifica la jerarquía para ordenar campos :

los nombres de los campos elementales, el nombre de campo o grupo, tipo de campo o grupo, número de caracteres o dígitos del campo, número de ocurrencias del campo o grupo, valor/comentario/dependencia de un nombre de campo (esta columna se utiliza en una de las opciones : un valor para la ocurrencia o un comentario acerca del campo o una dependencia sobre el nombre del campo).

IV.4.5 Lenguaje para la definición de procedimientos (LDP).

Es el componente de un programa de aplicación en IDEAL que forma con todas las instrucciones de LDP necesarias para definir el procesamiento lógico del programa. La secuencia en que son ejecutadas las instrucciones en el procesamiento lógico del programa es en el orden natural, es decir, de arriba hacia abajo.

En este lenguaje, como en cualquier otro de alto nivel, se encuentran instrucciones que son utilizadas para ejecutar repetidamente una serie de acciones mientras que una cierta condición o condiciones se satisfagan ; otras instrucciones se utilizan para seleccionar una serie de acciones cuando una determinada condición se satisface ; en caso de no cumplirse la condición otra serie de acciones se ejecutan ; otras intrucciones se utilizan para seleccionar una serie de acciones en caso de cumplirse una condición de todas las posibles ; otras instrucciones se utilizan para ejecutar subrutinas dentro de programas y por último instrucciones que nos permiten invocar a subprogramas. Dependiendo de la finalidad de cada una de las intrucciones de LDP las clasificaremos para su explicación en los grupos siguientes : manejo

de pantallas, manejo de impresion, modificación a la estructura de la base de datos y control de flujo del programa.

IV.4.5.1 Manejo de pantallas

Como ya mencionamos con anterioridad las pantallas constituyen uno de los recursos de un programa de aplicación en IDEAL, el formato de pantallas será utilizado por el programa con la finalidad de presentar a los usuarios las distintas opciones que pueden ser ejecutadas, éstas opciones pueden ser : consulta, actualización, modificación y fin de sesión para el uso inmediato en línea de la estructura de la base de datos en cuestión. Además otros formatos de pantallas se utilizan para insertar datos con los cuales se ejecutará el programa ; en caso de que se hallan introducido datos incorrectos se emiten mensajes con alta intensidad a los usuarios con el objeto de indicar a éstos cuales datos deberán ser corregidos para la ejecución del programa.

Dependiendo de la utilidad de las pantallas asociamos las instrucciones TRANSMIT, REFRESH, y SET ATTRIBUTE para el manejo de las mismas.

La intrucción TRANSMIT se utiliza para mostrar en una terminal la pantalla que fue previamente definida con la facilidad de la definición de pantallas (FDP) para recibir datos que serán proporcionados por los usuarios. Una vez recibidos los datos, la intrucción TRANSMIT, válida automáticamente y edita todos los datos, de acurdo a las reglas de edición especificadas en la definición de la pantalla.

La sintaxis de la instruccion TRANSMIT es la siguiente :

```
TRANSMIT    nombre-pantalla
     [REINPUT]
     [CLEAR]
     [CURSOR AT {HOME                  }]
                {identificador-campo   }].
```

Donde :

El nombre-pantalla representa a la pantalla especifica, la cual se forma de 1 a 8 caracteres.

REINPUT es una palabra reservada que especifica que todos los campos en la pantalla que tienen comúnmente el atributo RECEIVED, tendrán aún este mismo atributo después del TRANSMIT (cuando la pantalla es recibida nuevamente), sin importar que el usuario incruste nuevos valores a estos campos. REINPUT se utiliza en situaciones donde el usuario está cambiando repetidamente valores. Cuando todos los campos de entrada requeridos son introducidos y validados, entonces se procesa la pantalla completa de un solo golpe o cuando los mismos valores tendrán que ser inlcuídos a campos o para una serie de registros. REINPUT ocasiona la retención del campo con el atributo RECEIVED a lo largo de múltiples transmisiones de la misma pantalla, una vez que el usuario ha registrado un valor en el campo.

CLEAR es otra palabra reservada que se utiliza para limpiar todos los campos desprotegidos a la "salida del caracter ocupado" que fue previamente definido en (FDP) antes de la transmisión.

La cláusula CURSOR AT tiene como objeto anular la posicion por omisión (dafault) del cursor en la pantalla.

HOME es otra palabra reservada que indica la posición del cursor sobre el campo por omision (default), como se definió en la (FDP).

Identificador-campo representa el identificador de campo para el cual el cursor sera posicionado.

La ejecución de subsecuentes instrucciones TRANSMIT para la misma pantalla ocasiona que los valores de todos los campos sean modificados por el programa o por el usuario, después que se manda el último TRANSMIT.

Los campos en una pantalla pueden ser inicializados moviendo valores a éstos antes de que la instrucción TRANSMIT sea ejecutada.

La instrucción TRANSMIT ocasiona automáticamente un CHECKPOINT (ver el grupo de instrucciones para el control de flujo del programa) en cualquier vista de datos de ADR/DATACOM/DB que sea actualizada. Esto significa que la instrucción BACKOUT (ver grupo de instrucciones para el control de flujo del programa) puede restaurar el estado de respaldo de las vistas de datos, solamente a las más recientes instrucciones de CHECKPOINT o TRANSMIT. Si se ejecuta una instrucción TRANSMIT dentro del rango de una construcción FOR y posteriormente se hace una prueba para actualizar los valores de cualquiera de los campos para el mismo registro de la vista de datos, y si otra tarea diferente ha eliminado a

un registro, o ha actualizado los valores de cualquiera de los campos en ese registro, entre el TRANSMIT y las actualizaciones subsecuentes, surge el procedimiento de error (ERROR PROCEDURE).

La instrucción REFRESH se utiliza para restaurar todos los campos de la pantalla nombrada a sus valores y atributos iniciales, como fueron previamente definidos en (FDP). Además la instrucción REFRESH asegura que se ha realizado una copia fresca de la pantalla y se encuentra disponible para otra nueva acción.

La sistaxis de la instrucción REFRESH es como sigue :

```
REFRESH nombre-pantalla.
```

La instrucción SET ATTRIBUTE se utiliza para desplegar los campos en la pantalla, así como a sus características.

la sintaxis de la intrucción SET ATTRIBUTE es :

```
        ┌ATTRIBUTE  [S] ┐ ┌atributos ┐
   SET  {               } {          }
        └ATTR           ┘ └'x . . .' ┘ [TEMP]  ON
identificador-campo...
```

Donde :

Los atributos representan las características de los campos que pueden ser una o más de las siguientes palabras reservadas :

palabra reservada	caracter inicial	significado
HIGHLIGHT	H	alta intensidad
LOWLIGHT	L	intensidad normal
INVISIBLE	I	campo invisible
PROTECTED	P	campo protegido
UNPROTECTED	U	campo desprotegido
SKIP	S	campo con salto
ENSURE RECEIVED	E	campo tratado sobre la transacción actual
NUMERIC	N	campo numérico
ALPHANUMERIC	A	campo alfanumérico
CURSOR	C	campo con el cursor.

El identificador x es una literal alfanumérica encerrada entre apóstrofos que consiste de uno o más caracteres que son las letras iniciales para uno o más atributos listados anteriormente.

La palabra reservada TEMP limita el restablecimiento de los atributos para la siguiente ejecución de la instrucción TRANSMIT. El atrbuto ENSURE RECEIVED es temporalmente implícito. Todos los otros atributos serán especificados explícitamente con TEMP para la limitación del restablecimiento.

La palabra reservada ON especifica el campo o campos para recibir el nuevo atributo o atributos.

El identificador de campo identificador-campo es el campo para el

cual los atributos serán especificados.

IV.4.5.2 Manejo de impresión

Cuando los recursos que utiliza un programa en IDEAL han sido creados, el programa se encuentra disponible para su compilación. Al final del proceso de compilación, el sistema asocia un número de salida a tal proceso, mediante el cual se despliga en pantalla el listado de compilación. Si hay errores de compilación se edita el programa para hacerle las correcciones necesarias para ser compilado nuevamente, en caso de seguir existiendo errores se repite de nuevo el proceso hasta que no haya errores de compilación. Una vez que ya no existen errores de compilación el programa está disponible para ser ejecutado ; al final del proceso de ejecución, nuevamente el sistema asocia un número de salida por medio del cual se despliega en pantalla el listado de la prueba. Si queremos el listado de compilación y prueba del programa en papel se aplica un proceso batch a través del comando PRINT.

Las instrucciones asociadas al manejo de impresión de la información son las siguientes : PRODUCE, LIST, LIST ERROR y ASSIGN REPORT.

La instrucción PRODUCE se utiliza para generar un reporte. Cada iteración de la instrucción PRODUCE genera una línea de detalle para el reporte que fue previamente definido dentro de la facilidad de definición de reportes (FDR). La instrucción PRODUCE, está gereralmente contenida en una estructura FOR o LOOP.

La sintaxis de la instrucción PRODUCE es como sigue :

PRODUCE nombre-reporte.

Donde :

El nombre-reporte especifica el nombre del reporte formado de 1 a 8 caracteres que recibirá la línea de detalle.

La intrucción LIST se utiliza para transmitir datos al archivo de salida sin el uso de una definición de reporte. La instrucción LIST es útil para el despliegue de mensajes simples o para el despliegue del contenido de los campos para su examen.

La sintaxis de la instrucción LIST tiene los dos siguientes formatos:

Sintaxis 1 :

LIST especificación-de-lista

La especificació-de-lista designa los datos que serán listados con el formato siguiente :

```
 ┌'literal'              ┐  ┌       ┌'literal'              ┐ ┐
 │ campo elemental       │  │       │ campo elemental       │ │
{  grupo-alfa             }  │ [,]  {  grupo-alfa             } │ ...
 │ función altanumérica  │  │       │ función alfanumérica  │ │
 └ SKIP                  ┘  └       └ SKIP                  ┘ ┘
```

Sintaxis 2 :

LIST ERROR.

La instrucción LIST ERROR se utiliza para listar información acerca de una condición de error que ha terminado la ejecución.

La instrucción ASSIGN REPORT se utiliza para invalidar dinámicamente la instrucción RUN o cualquier destino actual, y permite salidas de reportes que serán manejados individualmente.

La sintaxis de la instrucción ASSIGN REPORT es com sigue :

```
ASSIGN REPORT  nombre   [TO  nombre-alterno ]
[                {LIBRARY                                ]]
[DESTINATION     {SYSTEM nombre-sistema   [COPIES n]}    ]]
[DISPOSITION {disp} ]
[MAXLINES  m ]
[DESCRIPTION  'cadena' ]
```

Donde :

El nombre es del reporte o la palabra RUNLIST.

El nombre-alterno es una literal numérica o el identificador de un campo alfanumérico que se utiliza para especificar un nombre alterno para el reporte , como un nombre de destino (on OS) o como SYSnnn (en DOS).

El nombre-sistema es una literal alfanumérica o el identificador

de un campo alfanumérico utilizado para especificar el nombre del destino del sistema de impresión.

La n es una literal numérica o el identificador de un campo numérico utilizado para especificar el número de copias del reporte.

La palabra disp es una literal alfanumérica o un campo alfanumérico con el valor de KEEP, HOLD, o RELEASE.

La letra m es una literal numérica o el identificador de un campo numérico utilizada para especificar el número máximo de líneas del reporte que será generado.

La palabra cadena es una literal alfanumérica o el identificador de un campo alfanumérico utilizada para suministrar una descripción del reporte y no excederá a 32 caracteres.

IV.4.5.3 Instrucciones para la modificación a la estructura de la base de datos

En este grupo se dan las instrucciones que nos permiten insertar, eliminar y modificar registros de los archivos de una base de datos mediante un programa de aplicación en IDEAL. Las instrucciones asociadas al mantenimiento de la base de datos son :
FOR NEW, DELETE, MOVE, SET, ADD y SUBTRACT.

La instruccion FOR NEW se utiliza para insertar nuevos registros en las vistas de datos(ADR/DATACOM/DB), o para agregar nuevos registros

al final de un archivo secuencial de la vista de datos.

La sintaxis de la intrucción FOR NEW es como sigue :

```
[<<Etiqueta>>]
    FOR  [THE] NEW nombre-vista-de-datos
         intrucciones
       [WHEN DUPLICATE ]
       [  instrucciones]
    ENDFOR.
```

Donde :

Etiqueta es un nombre opcional de 1 a 15 caracteres para la contrucción de la intrucción FOR NEW. El nombre también puede ser utilizado como referencia para la construcción de la instrucción QUIT.

la cláusula FOR [THE] NEW especifica la acción que será tomada para agregar o insertar cada registro nuevo. La instrucción FOR NEW ocasiona la inicialización de valores en el registro de la vista de datos. Si valores iniciales de campos fueron especificados en el diccionario de datos (ADR/DATADICTIONARY), éstos son utilizados. En caso contrario los valores de campos en la vista de datos son inicializados a ceros (para campos numéricos) y blancos(para campos alfanuméricos).

La palabra reservada THE puede ser agregada para mayor claridad de

la instruccion FOR NEW.

El nombre-vista-de-datos representa el nombre de la vista de datos para la cual será agregado un nuevo registro. La vista de datos necesariamente será actualizada como resultado de la inserción del registro.

La cláusula WHEN DUPLICATE contiene instrucciones que serán ejecutadas cuando al insertar un nuevo registro a la vista de datos se obtienen llaves duplicadas en los registros de la misma, ya que, en una base de datos datos no se permite llaves con valores duplicados. El criterio para invalidar la duplicación de llaves es definido por el administrador de la base de datos correspondiente en la definición de la misma. Si la cláusula WHEN DUPLICATE es incluída y la duplicación se permite, la cláusula WHEN DUPLICATE es ignorada.

La palabra reservada ENDFOR indica la finalización de la construcción FOR. Si se anidan construcciones FOR, éstas se aparean de adentro hacia afuera con sus correspondientes ENDFOR para la finalización de las mismas. Cualquier campo agregado en ese tiempo en el registro de la vista de datos puede ser referido después del EDNFOR a menos de que la instrucción QUIT sea utilizada.

La instrucción DELETE se utiliza para eliminar el registro actual referido por la vista de datos ; solamente puede ser utilizada en una construcción FOR FIRST o FOR EACH. El resultado de la eliminación de registros de la vista de datos ocasiona que ésta sea actualizada.

La sintaxis de la construcción DELETE es la siguiente :

DELETE nombre-vista-de-datos.

Donde :

El nombre-vista-de-datos especifica el nombre de la vista de datos de la cual el registro actual será eliminado.

Las instrucciones MOVE y SET se utilizan para trasladar datos de un campo emisor a un campo receptor, el valor del campo emisor permanece después de haber realizado el movimiento.

Las sintaxis de las instrucciones MOVE y SET son :

Sintaxis 1 (SET y MOVE) :

```
SET    campo-numérico =   { expresión-numérica      }
                          { expresión-alfanumérica  }

MOVE   { expresión-numérica      }
       { expresión-alfanumérica  }   TO campo-numérico.
```

El movimiento de datos a un campo elemental numérico deben de sujetarse a las siguientes reglas :

1) Si los campos emisor y receptor ambos son numéricos, el movimiento se efectúa de acuerdo al alineamiento del punto decimal implícito con truncamiento de dígitos decimales si es necesario.

2) Cuando el valor de un campo emisor alfanumérico se mueve a un receptor numérico, el valor del campo emisor se convierte primero a numérico aplicando la función $NUMBER a la información alfanumérica.

Sintaxis 2 (SET y MOVE) :

```
SET   campo-alfanumérico = { expresión-alfanumérica }
                           { campo-numérico         }
                           { literal-numérica       }

MOVE { expresión-alfanumérica }
     { campo-numérico         }   TO campo-alfanumerico.
     { literal-numérica       }
```

El movimiento de datos a un campo elemental alfanumérico deben de sujetarse a las siguientes reglas :

1) Si los campos emisor y receptor son alfanuméricos, el movimiento de datos se realiza como sigue :

a) Cuando la longitud del campo del valor del emisor es mayor que la longitud del campo receptor, hay truncamiento a la derecha.

b) Cuando la longitud del campo receptor es mayor a la longitud del campo emisor, el campo receptor se llena con espacios a la derecha, hasta completar la igualdad de ambas longitudes.

c) Cuando las longitudes del campo emisor y receptor son iguales se realiza una copia exacta.

2) Cuando el valor del campo emisor es numérico y se mueve a un campo receptor alfanumérico, el campo emisor primero se convierte a campo alfanumérico aplicando la función $STRING. El resultado de esta conversión puede ser que resulte un campo de longitud mayor a la del campo receptor. Entonces la longitud del volor del campo ya convertido será truncado por la izquierda de acuerdo a la diferencia de las longitudes de los campos emisor y receptor.

Sintaxis 3 (SET y MOVE) :

```
SET   grupo-1 =   grupo-2  {BY NAME     }
                           {            }
                           {BY POSITION }

MOVE  grupo-1  TO  grupo-2 {BY NAME     }
                           {BY POSITION }
```

El movimiento de datos de un grupo a otro debe de estar sujeto a las siguientes reglas :

1) Ambos emisor y receptor deberán ser grupos.

2) El movimiento de datos de cada campo emisor a los correspondientes del grupo receptor está sujeto a las reglas descritas en la sintaxis 1 y 2 de las instrucciones (SET y MOVE).

La claúsula BY NAME especifica el movimiento de datos por nombre de cada campo del grupo emisor a un campo llamado idénticamente en el grupo receptor, si existe alguno. Ocurrencias dentro de los grupos repetitivos (en cualquiera) será compatible. Solamente datos de campos que tienen los mismos nombres, subordinados a los grupos respectivos y elementales, son movidos. Redefiniciones dentro de un grupp que emite,

no son emisores elegibles ; sin embargo redefiniciones en el grupo receptor sí son elegibles. Se emite un mensaje de error en la compilación si el número de ocurrencias no concuerda. Ocurre un error de ejecución si el número de ocurrencias "que dependen de" o de parámetros en la ocurrencia de campos no es el mismo.

La claúsula BY POSITION especifica el movimiento por posición del valor del primer campo elemental en el grupo emisor al primer campo elemental del grupo receptor, sin importar su nombre ; el segundo campo del grupo emisor al segundo del grupo receptor etc.

Las estructuras de los campos serán compatibles cuando :

1) Sus estructuras tengan forzosamente el mismo número de campos elementales.

2) Las ocurrencias de valores sean idénticas y del mismo nivel relativo.

3) Cuando la correspondencia de campos elementales sea compatible ; es decir, banderas solamente pueden corresponder a banderas, campos numéricos y alfanuméricos deberán corresponder necesariamente a otros campos numericos y alfanuméricos respectivamente.

Sintaxis 4 (solamente para SET) :

SET bandera = { TRUE / FALSE }

Donde :

El campo bandera es de tipo bandera, es decir, solo se le puede asignar una de las palabras reservadas TRUE o FALSE.

Sintaxis 5 (solamente para SET) :

SET nombre-condición = TRUE.

Donde :

Nombre-condición representa en nombre de un campo de condición el cual le asigna el valor verdadero (TRUE).

Las instrucciones ADD y SUBTRACT se utilizan para incrementar o decrementar el valor de campos numéricos y pueden ser utilizadas como alternativas de la instrucción SET.

La sintaxis de las instrucciones ADD y SUBTRACT es como sigue :

$$\text{ADD} \left\{ \begin{array}{l} \text{campo-numérico} \\ \text{literal-numérica} \\ \text{campo-alfanumérico} \end{array} \right\} \text{TO campo-numérico}$$

$$\text{SUBTRACT} \left\{ \begin{array}{l} \text{campo-numérico} \\ \text{literal-numérica} \\ \text{campo-alfanumérico} \end{array} \right\} \text{FROM campo-numérico.}$$

Durante la ejecución de los campos emisor y receptor contendrán necesariamente valores numéricos, o se presentará un error de ejecución.

Los operandos de las instrucciones ADD y SUBTRACT no tendrán necesariamente el mismo número de posiciones a la derecha del punto decimal. Cuando una expresión con decimales es sumada o restada de un campo con un valor entero, los decimales no aparecen en la pantalla. Cuando una expresión con un valor entero es sumado o restado con decimales, los decimales son retenidos.

Los operandos de ADD y SUBTRACT tampoco tendrán necesariamente que estar definidos con el mismo número de posiciones a la izquierda del punto decimal.

IV.4.5.4 Instrucciones para el control de flujo del programa

En esta sección se dan las instrucciones que nos permiten seguir el control de flujo del programa en cuanto a repeticiones, búsquedas, ordenaciones, condiciones, bifurcaciones, continuación de control en caso de falla del sistema, etc.

Las instrucciones asociadas a estas acciones son : DO, LOOP, FOR EACH/FIRST, FOR NEXT, IF, SELECT, QUIT, PROCESS NEXT, CHECKPOINT y CALL.

La instrucción DO se utiliza para invocar a otro procedimiento en el mismo programa, al cual se le transfiere el control. Cuando este procedimiento termina, el control de flujo regresa y continúa la ejecución con la instrucción que sigue al DO en la invocación del procedimiento.

La sintaxis de la instrucción DO es :

```
DO nombre-procedimiento.
```

Donde :

Nombre-procedimiento es el nombre del procedimiento formado de 1 a 15 caracteres, el cual será invocado.

La instrucción LOOP se utiliza en la repetición de una o más instrucciones de acuerdo al resultado de la evaluación de una o más condiciones.

La instrucción LOOP tiene tres tipos de sintaxis, las cuales describimos en seguida :

Sintaxis 1 :

```
[<< Etiqueta >>]
    LOOP
       instrucciones
      [{WHILE}
       {UNTIL} instrucciones-condición]  . . .
    ENDLOOP.
```

Donde :

<<etiqueta>> es un nombre opcional de 1 a 15 caracteres para la construcción del LOOP. La etiqueta sobre el LOOP es el nombre del grupo de proposiciones que conforman el LOOP. Puede ser utilizado como referencia al LOOP desde otras instrucciones, tales como QUIT o PROCESS NEXT o como operando de ciertas funciones, tales como, $COUNT.

La palabra reservada WHILE se utiliza como condición de ciclo que será ejecutado repetidamente mientras que la condicion se evalúe a verdadero.

La palabra reservada UNTIL se utiliza como condición para indicar el ciclo que será ejecutado repetidamente hasta que la condición se evalúe a verdadero (mientras que la condición se evalúe a falso).

La palabra reservada ENDLOOP indica el fin de la construcción LOOP.

Sintaxis 2 :

```
[<<etiqueta>>]
    LOOP
        expresion-numérica-1 TIMES
        instrucciones
       ⎡⎧WHILE⎫                          ⎤
       ⎣⎩UNTIL⎭ instrucciones-condición ⎦. . .
    ENDLOOP.
```

Donde :

La clausula expresión-numérica-1 TIMES especifica el número de veces que se va a repetir la ejecución del grupo de proposiciones. Si el valor de la expresión-numérica-1 es menor o igual a cero, las instrucciones no son ejecutadas. Si el valor de la expresión contiene fracción decimal se redondea al siguiente entero mayor o igual.

Sintaxis 3 :

```
[<<etiqueta>>]
  LOOP
     VARYING identificador
       [FROM expresión-numérica-2]
       [BY expresion-numérica-3]
       [{UP  } THRU  expresión-numérica-4]
        {DOWN}
            instrucciones
            [{WHILE} instrucciones-condicion] ...
             {UNTIL}

  ENDLOOP.
```

Donde :

La palabra reservada VARYING especifica al iterador, que debería de ser un identificador de un campo numérico. Su valor será variado cada vez que pasa un ciclo.

El identificador es de un campo numérico cuyo valor será incrementado o decrementado cada vez que se ejecuta un ciclo. El campo es variado por el valor de la expresión-numérica-3.

La cláusula FROM expresión-numérica-2 proporciona el valor inicial a partir del cual el ciclo es ejecutado. El de omisión (default) es el uno.

La cláusula BY expresión-numerica-3 especifica el valor

utilizado para incrementar o decrementar el valor del identificador. El de omisión es el uno cuando se especifica UP ; menos uno cuando se especifica DOWN.

En la clausula { UP / DOWN } THRU expresión-numérica-4

La expresión-numérica-4 especifica el valor que será comparado con el valor del identificador. El ciclo termina cuando utilizamos UP y el valor del identificador excede al valor de la expresión, o cuando utilizamos DOWN y el valor del identificador es menor que el valor de la expresión.

La palabra reservada UP especifica el valor del identificador que será incrementado por el valor de la expresión-numérica-3. El ciclo termina cuando el valor del identificador excede al valor de la expresión-numérica-4.

La palabra reservada DOWN especifica el valor del identificador que será decrementado por el valor de la expresión-numérica-3. El ciclo termina cuando el valor del identificador es menor que el valor de la expresión-numérica-4.

La instrucción FOR EACH/FIRST se utiliza para procesar un conjunto de registros (o procesar un único registro) de una vista de datos. Todas las instrucciones en el rango del FOR se aplican al registro o registros seleccionados de la vista de datos nombrada.

La sintaxis de la instrucción FOR/FIRST es :

```
[<<etiqueta>>]
      FOR [EACH                                  ]
          [ALL                                   ]
          [[THE] FIRST [num]] nombre-vista-de-datos
          [WHERE condición-donde]
          [ORDERED BY [UNIQUE] identificador [[,] identificador]]...
            instrucciones
          [WHERE NONE                            ]
          [      instrucciones                   ]
      ENDFOR.
```

Donde :

<<etqueta>> es un nombre opcional de 1 a 15 caracteres para la construcción FOR. Este nombre también puede ser utilizado como referencia para la construcción de las instrucciones QUIT y PROCESS NEXT, y como el operando de ciertas funciones tales como $COUNT.

Las palabras reservadas EACH y ALL especifican que las instrucciones en el rango de la construccion FOR se aplican a todo registro que satisface la condición-donde. EACH y ALL pueden seleccionarse indistintamente.

La cláusula [THE] FIRST [num] especifica que las instrucciones en el rango de la construcción FOR se aplican al número de registros

especificados con [num] que satisfacen la condición-donde.

La palabra reservada WHERE es opcional y especifica que las instrucciones en el rango de la construcción FOR se aplican a aquellos registros que satisfacen la condición-donde.

La condición condición-donde tiene las siguientes características :

1) Si la condición es un nombre de condición éste estará necesariamente en la vista de datos existente a la que se hace referencia.

2) El operando izquierdo de cada expresión relacional necesariamente tendrá el identificador de campo o un grupo-alfabético que estará en la vista lógica de datos referida.

3) Si el operando izquierdo es alfanumérico, el operando derecho será necesariamente alfanumérico.

4) Si el operando izquierdo es numérico, el operando derecho, puede ser una expresión numérica, un campo alfanumérico, un grupo alfabético o una expresión alfanumérica. Notemos sin embargo, que cuando el operando derecho no es una expresión numérica, se genera una llamada de atención cuando el programa se compila y si el operando derecho no puede ser convertido a numérico, se presenta un error en el momento de ejecutar el programa.

5) El operando izquierdo de una expresión-relacional en una condición-donde no debe ser necesariamente calificado en cuanto a la vista de datos, puesto que esta referencia está en la vista de datos de la cláusula FOR. Sin embargo todas las palabras reservadas utilizadas como operandos tendrán que ser calificadas.

6) El operando derecho de una expresión-relacional puede ser cualquier expresión aritmética o alfanumérica, pero no puede ser ningún campo referido en la vista de datos nombrada en la claúsula FOR.

7) Las funciones booleanas o banderas no pueden ser condiciones simples.

8) Los subíndices utilizados en la condición-donde no tiene que estar en la vista de datos referida.

La claúsula ORDERED BY es opcional que determina el orden lógico en el cual los registros son procesados. Esta cláusula solamente se aplica a las vistas de datos(ADR/DATACOM/DB). Si esta cláusula se omite los registros de la vista de datos serán procesados en un orden óptimo.

La palabra reservada UNIQUE es opcional y especifica que solamente un registro para cada valor del identificador (es) del ORDERED BY es recuperado.

El(los) identificadores pude(n) ser campo alfanumérico o grupo-alfabético, pero no para grupos no alfanuméricos. Los identificadores pueden ser indexados, pero no por campos de la vista de datos referida.

La cláusula WHEN NONE es opcional y se coloca al final para especificar que cuando ninguno de los registros cumple con la condición-donde las instrucciones siguientes al WHEN NONE son ejecutadas.

La instrucción FOR NEXT se utiliza para especificar una serie de instrucciones que se aplican solamente al registro siguiente de un archivo secuencial en la vista de datos. Si un FOR FIRST fue ejecutado anteriormente para la misma vista de datos, el registro siguiente en secuencia es utilizado. Si el FOR FIRST no fué ejecutado anteriormente, FOR NEXT acude al primer registro. Notemos que esta construcción se aplica solamente a archivos secuenciales en vista de datos, y solamente para aplicaciones en batch (los archivos secuenciales pueden ser leídos).

La sintaxis de la instrucción FOR NEXT es :

```
[<<etiqueta>>]
    FOR [THE] NEXT  nombre-vista-de-datos
        instrucciones
       [WHEN NONE          ]
       [  instrucciones    ]
    ENDFOR.
```

Donde :

La cláusula FOR [THE] NEXT especifica que la acción que será tomada se aplica solamente al siguiente registro de un archivo secuencial en la vista de datos.

Nombre-vista-de-datos es el nombre de la vista de datos de la cual los registros serán leídos.

La cláusula WHEN NONE es opcional y especifica que cuando no

existe el registro "siguiente" (es decir, no hay más registros en la vista de datos) las instrucciones siguientes al WHEN NONE serán ejecutadas.

La instrucción IF se utiliza para elegir una de dos posibles alternativas. Cuando una condición determinada se satisface se ejecuta una serie de proposiciones, en caso de no cumplirse dicha condición otra serie de instrucciones se ejecuta.

La sintaxis de la instrucción IF es como sigue :

```
IF  condición
    [THEN]
        instrucciones
  [ELSE                ]
  [    instrucciones   ]
END1F.
```

Donde :

La condición la define el usuario. Las instrucciones que siguen inmediatamente se ejecutan solamente si la condición es verdadera.

La palabra reservada THE es opcional y se agrega para una mayor claridad de la instrucción IF.

La palabra reservada ELSE es opcional y marca el principio de un conjunto de instrucciones que serán ejecutadas solamente si la condición es falsa.

la palabra reservada ENDIF especifica el fin de la construccion IF. Cuando se anidan istrucciones IF, cada una de éstas se aparea con su ENDIF correspondiente comenzando de adentro hacia afuera para finalizar a cada una de las instrucciones IF anidadas.

La instrucción SELECT se utiliza para la ejecución de una o más acciones diversas dependiendo la ejecución de cada una de las acciones de que se cumplan una o más condiciones.

La instrucción SELECT tiene tres tipos de sintaxis, los cuales se describen a continuación :

Sintaxis 1 :

```
SELECT  selector-sujeto
⎧WHEN ⎧expresión-numérica     ⎫ ⎡  ⎧    expresión-numérica  ⎫⎤...⎤
⎨     ⎩expresión-alfanumérica ⎭ ⎣OR⎩expresión-alfanumérica  ⎭⎦   ⎥
⎩     instrucciones                                              ⎦
  ⎡        ⎧OTHER⎫          ⎤
  ⎢WHEN    ⎩NONE ⎭          ⎥
  ⎣        instrucciones    ⎦
  ⎡WHEN ANY                 ⎤
  ⎣        instrucciones    ⎦
 ⎧ENDSEL   ⎫
 ⎩ENDSELECT⎭
```

Donde :

El selector-sujeto debe de ser el identificador de un campo numérico o alfanumérico o una literal numérica o alfanumérica, cuyo valor es utilizado para determinar la acción a seleccionar.

La cláusula WHEN expresión-numérica
expresión-alfanérica especifica un posible valor (o valores). Si el valor de un "WHEN" es igual al valor actual del selector-sujeto, las instrucciones que siguen a la palabra reservada WHEN son ejecutadas.

En esta sintaxis se nota que solamente las instrucciones que siguen a la primera condición verdadera del WHEN en la prueba son ejecutadas, opcionalmente seguidas por la cláusula WHEN ANY.

La cláusula WHEN OTHER/NONE es un enunciado que debe ser el último, que especifica que cuando ninguno de los valores listados iguala al selector-sujeto, las instrucciones siguientes a la cláusula WHEN OTHER o a la cláusula WHEN NONE son ejecutadas. Las palabras reservadas OTHER Y y NONE pueden ser utilizadas indistintamente.

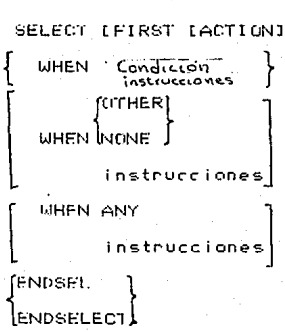
La cláusula WHEN ANY es un enunciado final que especifica que cuando cualquier valor del WHEN iguala al valor actual del selector-sujeto, las instrucciones que siguen al WHEN ANY son ejecutadas adicionalmente a las instrucciones que siguen al caso de igualdad.

Las palabras reservadas ENDSEL o ENDSELECT indican el final de la construcción SELECT. Si se anidan construcciones de SELECT, cada una

de éstas debe de emparejarse con su correspondiente ENDSEL comenzando de adentro hacia afuera.

La sintaxis 1 de la construcción SELECT permite seleccionar al conjunto de instrucciones que siguen al primer valor del WHEN igual al valor del selector-sujeto. Solamente el conjunto de instrucciones que siguen al primer valor igualado son ejecutadas.

Sintaxis 2 :



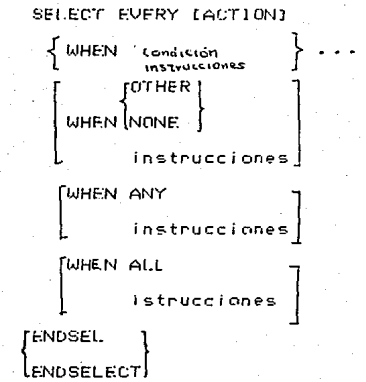
Donde :

La cláusula FIRST [ACTION] puede ser agregada para claridad de la construcción SELECT.

La sintaxis 2 de la instrucción SELECT selecciona al conjunto de instrucciones que siguen a la primera condición verdadera en una serie de condiciones diferentes. Solamente las instrucciones del primer conjunto que siguen a la primera condición verdadera son ejecutadas.

La sintaxis 3 de la instrucción SELECT hace que se ejecuten todos y cada uno de los conjuntos para los cuales la condición sea verdadera.

Sintaxis 3 :



Donde :

La palabra reservada [ACTION] es opcional y puede ser agregada para mayor claridad.

La cláusula WHEN ALL es un enunciado opcional que debe ir al final, que especifica que cuando todas y cada una de las condiciones son verdaderas, las instrucciones que siguen a la cláusula WHEN ALL son ejecutadas adicionalmente a las instrucciones que siguen a cada condición verdadera.

La instrucción QUIT se utiliza para alterar el flujo de la ejecución de una o más construcciones, procedimientos, o programas que se estén ejecutando, y opcionalmente cede el control a otro procedimiento. En seguida la ejecución continúa como si las construcciones, procedimientos o progrmas afectados hubieran terminado en forma normal.

La sintaxis de la instrucción QUIT es :

```
QUIT [ etiqueta-de-LOOP-o-FOR                                          ]
     | PROGRAM                                                         |
     | RUN                                                             |
     | etiqueta-de-procedimiento [TRANSFER [TO] procedimiento]         |
     | PROCEDURE                 [TRANSFER [TO] procedimiento]         |
     | ALL TRANSFER [TO] procedimiento                                 ]
```

Donde :

Si la proposición QUIT aparece sin operandos, termina la ejecucion de la construcción actual del LOOP o FOR o procedimiento. Las instrucciones siguientes al ENDLOOP, ENDFOR, o ENDPROC son ejecutadas. Cuando la instrucción QUIT se aplica a una construcción FOR, no se realiza la actualización al registro actual de la vista de datos.

Etiqueta-de-LOOP-o-FOR es el nombre de la construcción cuya ejecución será suspendida. La instruccion QUIT y el nombre pueden

solamente ser ejecutados en el rango de alguna construcción nombrada, o en alguna construcción ERROR PROCEDURE.

PROGRAM es una palabra reservada ; cuando es utilizada con QUIT suspende el subprograma o programa en ejecución. El procedimiento principal de un programa implica QUIT PROGRAM, por lo que no se requiere para una terminal normal.

Run es utilizada con un QUIT para terminar la ejecución presente.

Etiqueta-de-procedimiento representa el nombre de un procedimiento que será suspendido al pasar el control en la ejecución.

PROCEDURE es una palabra reservada que se utiliza con QUIT para terminar el procedimiento actual y pasar el control a otro procedimiento. Cuando se utiliza sin la cláusula TRANSFER, el control regresa al procedimiento que lo invocó.

La palabra reservada ALL cuando se utiliza con QUIT, termina el procedimiento actual, más todos los procedimientos invocados por éste y transfiere el control al procedimiento nombrado. QUIT ALL tendrá necesariamente una cláusula de transferencia.

La cláusula TRANSFER [TO] es opcional y causa que el control sea pasado al procedimiento nombrado en la cláusula. La palabra reservada TO puede ser incluída para mayor claridad.

Procedimiento es el nombre de 1 a 15 caracteres del nuevo

procedimiento que recibirá el control. En caso de QUIT ALL, este procedimiento se convertirá en el nuevo procedimiento principal del programa.

Instrucción PROCESS NEXT

Cuando se utiliza, causa que el control de flujo continúe con la iteración siguiente de una construcción repetitiva (LOOP o FOR EACH). Si la construcción actual es un FOR EACH, cualquier actualización al registro presente es abandonada, y continúa procesando el siguiente registro.

La sintaxis de la instrucción PROCESS NEXT es :

PROCESS NEXT [etiqueta].

Donde :

La cláusula PROCESS NEXT ocasiona que la iteración actual de una construcción repetitiva (LOOP o FOR EACH) sea suspendida. PROCESS NEXT sin etiqueta aparece forzosamente en el rango lexicográfico de la construcción LOOP o FOR EACH. PROCESS NEXT con etiqueta aparece necesariamente en el rango lógico de la construcción LOOP o FOR EACH.

Cuando la instrucción aparece en el rango de un FOR EACH, cualquier actualización al registro presente de la vista de datos es abandonada lo mismo para campos cuyos valores han sido "cambiados" como

resultado indirecto de esta ejecución.

La etiqueta se refiere a la construcción FOR o FOR EACH para la cual la iteración presente es suspendida. La instrucción "PROCESS-NEXT-etiqueta" estará dentro del rango lógico de la construcción identificada con la "etiqueta", esto es, PROCESS NEXT tiene que referirse a la construcción actual o de un nivel lógico más alto.

La instrucción CHEKPOINT señala a IDEAL que toda la base de datos activa para una unidad lógica de trabajo ha terminado satisfactoriamente. CHECKPOINT establece el más reciente estado de todas las vistas de datos en la base de datos, aplicable solamente a éstas.

La sintaxis de la instrucción CHECKPOINT es :

CHECKPOINT.

La instrucción BACKOUT se aplica solamente a la vista lógica de datos para restaurar todos los registros en la vista de datos de la base de datos al estado anterior a la última instrucción CHECKPOINT o TRANSMIT. Si la instrucciones CHECKPOINT o TRANSMIT no fueron previamente ejecutadas, todas las actualizaciones en la ejecución son removidas.

La sintaxis de la instrucción BACKOUT.

BACKOUT.

La instrucción RELEASE se utiliza para el control de recursos durante el procesamiento normal de una aplicación en IDEAL. Los recursos utilizados permanecen disponibles hasta que el programa termina de ejecutarse. Cuando el mismo recurso, tal como, subprograma, reporte, pantalla, es utilizado mas de una vez en la ejecución, es conveniente tener el recurso disponible para ser llamado nuevamente por el programa principal. Sin embargo en ocaciones, cuando el recurso ya no es necesario para el programa de aplicación, es deseable que no tengamos estos recursos ocupando memoria todo el tiempo ; en estos casos llamamos a una de tres intrucciones RELEASE, que son proporcionados para recursos independientes : RELEASE PROGRAM, RELEASE PANTALLA y RELEASE REPORT.

La instrucción RELEASE PROGRAM se utiliza para liberar un subprograma en IDEAL cuando éste ya no es necesario para el programa principal. Esta intrucción no afecta a la lógica de invocación de programas, solamente afecta a la ejecución. Cuando se omite esta instrucción el subprograma permanece listo esperando ser llamado nuevamente por el programa principal hasta el final de la ejecución.

La sintaxis de la instrucción RELEASE PROGRAM es :

RELEASE PROGRAM [nombre].

Donde el nombre especifica al subprograma que será liberado. Si el

nombre se omite el RELEASE se aplica al programa actual.

La instrucción RELEASE PANEL se utiliza para liberar una pantalla después de que ésta ya no es necesaria como recurso del programa.

La sintaxis de la instrucción RELEASE PANEL es :

RELEASE PANEL nombre.

El nombre especifica la pantalla que será liberada. Cuando se hace referencia a una pantalla liberada (con la instrucción TRANSMIT o REFRESH) una copia fresca es retroalimentada a la biblioteca de la pantalla.

La instrucción RELEASE REPORT se utiliza para liberar un reporte por una de dos razones. Un reporte será liberado ya que el programa ha terminado con éste, y ya no es necesario como recurso del programa ; o un reporte será liberado para que un reporte nuevo haga uso de la misma definición del reporte que será generado más adelante en el mismo programa. si el mismo reporte fué reinvocado con la instrucción PRODUCE sin haber sido liberado anteriormente, todas las líneas nuevas generadas para ese reporte serán anexadas al reporte existente, más bien que siendo generado como un reporte separado. Cuando el reporte fue liberado éste es generado como un nuevo reporte separado al anterior.

La sintaxis de la instrucción RELEASE REPORT es :

RELEASE REPORT [nombre].

Donde nombre es el reporte que será liberado.

La instrucción ASSIGN DATAVIEW se utiliza para asociar una vista lógica de datos utilizada por un programa con una base de datos identificada.

La sintaxis de la instrucción ASSIGN DATAVIEW es :

ASSIGN DATAVIEW nombre BDID identificador-de-la-base-de-datos.

Donde :

El nombre indica a la vista lógica de datos que será asociado con la base de datos.

El identificador-de-la-base-de-datos es un literal numérica o el identificador de un campo numérico o Alfanumérico utilizado para identificar a la base de datos con la cual la vista de lógica de datos será asociada. El valor será igual a tres digitos o caracteres.

La instrucción CALL se utiliza para invocar a otro programa o subprograma al programa principal. La liga entre el programa principal y el programa invocado es a través de parámetros de entrada y los correspondientes de salida al finalizar la ejecución de éste. Después de finalizar la ejecución el programa o subprograma, el control de flujo regresa al programa principal con la instrucción que

sigue al CALL.

La sintaxis de la instrucción CALL es :

```
CALL   nombre-programa [USING]
       [INPUT  parm-1,...,parm-n  ]
       [UPDATE] parm-1,...,parm-n ]...
```

Donde :

El nombre-progrma es el nombre externo del programa o subprograma que será invocado, definido por el usuario y formado de 1 a 8 caracteres.

La palabra reservada USING es opcional y puede ser agregada para mayor claridad.

La palabra reservada INPUT indica que el programa o subprograma invocado puede referenciar, pero sin modificar, los parámetros especificados. Si ambas palabras INPUT y UPDATE se omiten , UPDATE se asume mientras que INPUT no sea especificado.

Los parámetros parm-1,... parm-n son los datos que serán dados al programa invocado. los parámetros de entrada pueden ser literales o los identificadores de campos o grupos (pantallas y vista lógica de datos son vistas como grupos).

La palabra reservada UPDATE indica al programa llamado que puede

modificar el valor de los parámetros especificados. Si ésta palabra se omite, el de omisión es UPDATE a no ser que se especifique INPUT.

Los parámetros parm-1,parm-2,...,parm-n son los módulos de datos que serán pasados al programa llamado. Los parámetros actualizables serán identificadores de campos o grupos (pantallas y vista lógica de datos son vistas como grupos). Las literales no pueden ser utilizadas como parámetros actualizables.

La instrucción ERROR PROCEDURE se utiliza para especificar un conjunto de acciones que serán invocadas cada vez que ocurra un error de ejecución. Así mismo proporciona medios para el procesamiento anormal de errores explícitos para invalidar los errores por omisión.

La sintaxis de la instrucción ERROR PROCEDURE es :

```
<<ERROR>> PROCEDURE
      instrucciones
[ENDPROC       ]
[ENDPROCEDURE  ]
```

Donde :

la cláusula <<ERROR>> PROCEDURE identifica la acción que tomará efecto cuando un error de ejecución ocurra. Las instrucciones en el rango de <<ERROR>> PROCEDURE puede consistir de instrucciones de IDEAL/LDP necesarias para el procesamiento de error. La última

instrucción es comúnmente una instrucción QUIT o PROCESS NEXT.

IV.4.5.5 Funciones intrínsecas.

Además de las instrucciones anteriormente ya descritas se cuenta con un conjunto de funciones intrínsecas de la computadora que pueden ser utilizadas para completar el desarrollo de un programa de aplicación en IDEAL. Cada función intrínseca proporciona un único valor al programa en el punto de referencia. Todas las funciones intrínsecas comienzan con el símbolo "$". De acuerdo al funcionamiento de cada una de ellas. Las clasificaremos en los grupos siguientes: funciones para el procesamiento de pantallas, funciones para el manejo de errores, funciones para la manipulación de datos y funciones para identificacion del término de sesion.

IV.4.5.5.1 Funciones para el procesamiento de pantallas.

Las siguientes funciones se aplican al procesamiento de pantallas.

```
{ $PF1          }
{ $PF2          }
{   .           }
{   .           }
{   .           }
{ $PFn          }
{ $ENTER-KEY    } [(nombre-pantalla)].
```

$CURSOR (identificador-campo).

$RECEIVED (identificador-campo)

$PANEL-GROUP-OCCURS (nombre-pantalla).

Donde :

$PFn y $ENTER-KEY regresan un valor booleano de verdadero (true) o falso (false) según si la tecla señalada fue oprimida.

El nombre-pantalla se utiliza cuando existe simultáneamente más de una pantalla activa. La pantalla por omisión es la última pantalla recibida.

$CURSOR retorna un valor booleano de verdadero o falso según si el cursor está en el campo señalado. Un valor de falso es retornado si la pantalla no ha sido deplegada.

El identificador-campo es el identificador de un campo elemental que será examinado. Este campo debe de estar definido necesariamente en la pantalla.

$RECEIVED regresa un valor booleano de verdero o falso según si fue recibido del usuario un campo, para el campo especificado $RECEIVED regresa el valor de verdadero en tres casos :

1) En realidad el usuario modifica el valor del campo especificado sobre el más reciente TRANSMIT de la pantalla (incluyendo redetipificacion existente de caracteres, o ERASE-EOF).

2) El usuario modifica el valor del campo especificado sobre un previo TRANSMIT subsecuente de la pantalla.

3) El campo especificado fué definido con un atributo de "E" (ensure-received) en una de dos formas :

A) Con la instrucción SET ATTRIBUTE 'E'.

B) Definido como "ensure-received" cuando la pantalla fue especificada.

$PANEL-GROUP-OCCURS regresa el número de ocurrencias de un grupo repetitivo en una pantalla.

El nombre-pantalla representa a la pantalla a la cual el número de ocurrencias será requerido.

IV.4.5.5.2 Funciones para el manejo de errores

Las funciones asociadas al manejo de errores son :

$ERROR-CLASS, $ERROR-TYPE, $ERROR-DVW-STATUS, $ERROR-NAME, $ERROR-VALUE, $SUBSCRIPT-POSITION, $ERROR-PGM, $ERROR-PROC, $ERROR-STMT y $ERROR-DESCRIPTION que pueden ser utilizadas en la estructura de la instrucción ERROR PROCEDURE.

$ERROR-CLASS regresa un código de tres letras que identifica la categoría general del error.

$ERROR-TYPE regresa un código de tres letras que identifica el tipo específico del error, como sigue :

CLASE	TIPO	DESCRIPCION
NUM		ERROR NUMERICO
	NUM	un campo numérico contiene un valor numérico incorrecto.
ARI		ERRORES ARITMETICOS.
	OFL	una condición de overflow ha ocurrido.
	SQR	se ha intentado extraer la raíz cuadrada de un número negativo.
	UNS	se ha intentado asignar un valor negativo a un campo numérico sin signo.
	DVZ	se ha intentado dividir por cero.
	EXP	se ha violado la regla de los exponentes, es decir, no hay un valor entero numérico, o excede a 999.
REF		ERRORES DE REFERENCIA.
	REF	se ha realizado una referencia a un campo para el cual ninguna posición ha sido establecida.
	UPD	se ha realizado un intento para actualizar un campo que no es actualizable.
	PIU	se ha realizado un intento para pasar un parámetro de INPUT a un campo designado para recibir un parámetro actualizable.
	PAT	el atributo de un parámetro no iguala

al atributo correspondiente del campo para el cual será pasado.

PNO el número de parámetros excede al número permitido.

SUB ERRORES DE SUBINDICE.

SUB la forma de un subíndice es inválida, es decir, éste es menor que 1 o mayor que el número de ocurrencias.

ODO un valor de ODO excede el máximo permitido.

GRP el número de ocurrencias entre los grupos emisor y receptor no es el mismo.

SST el primer parámetro de una subcadena es menor que 1 o la longitud del parámetro es menor que 0.

DVW ERRORES DE LA VISTA DE DATOS.

DVW existe un error en la vista lógica de datos ($ERROR-DVW-STATUS).

PGM ERRORES DEL PROGRAMA.

IQR un QUIT ha sido utilizado incorrectamente.

PRO se ha realizado un intento para introducir recursivamente un proceedimiento.

PGM un intento ha sido realizado para introducir un programa activo.

SEQ ERRORES DE SECUENCIA.

FOR se ha realizado un intento para anidar un FOR en la misma vista lógica de datos.

DEL se ha especificado que un DELETE ha sido inválido, es decir, no fue emitido

		de una construcción FOR FIRST o
		FOR EACH o fue aplicada a una vista
		lógica de datos que no es actualizable.
	ADB	se ha realizado un intento para ASSING a una
		vista lógica de datos que está activa.
	ARS	se ha realizado un intento para ASSING a un
		reporte que está activo.
MIS		MISCELANEA DE ERRORES.
	SDV	se ha realizado un intento para leer una
		vista lógica de datos de un archivo
		secuencial en línea.
	DTE	un valor invalido ha sido especificado para
		la función $DATE o $TIME.
	ARP	una disposición invalida ha sido
		especificada para un reporte.
SYS		ERRORES DEL SISTEMA.
	CVR	un error del sistema ha occurrido.
	SYS	errores serios del sistema.

$ERROR-DVW-STATUS regresa un código de dos letras que identifica al estado de una vista lógica de datos solicitada (solamente cuando se utiliza $ERROR-CLASS regresa el código de "DVW") :

I1 Se ha obtenido el final de la vista de datos (SEQ).
I2 Determina errores en los datos de estos registros (DR)
I3 Problema en la integridad del registro ha occurrido (BD).
I4 Más de 16 archivos en secuencia son utilizados (SEQ).
I5 OS : instrucción DD perdida para el archivo secuencial (SEQ).

DOS : tarjeta inválida ASSGN (SEQ).

16 La longitud actual del registro de un archivo secuencial es mayor que la longitud del registro en la vista lógica de datos (SEQ).

$ERROR-NAME regresa el nombre del error del campo, subíndice, vista lógica de datos, procedimiento o programa que depende de $ERROR-TYPE.

$ERROR-VALUE retorna apropiadamente el valor de los campos erróneos.

$SUBSCRIP-POSITION regresa la posición del índice (1,2, o 3) en el error del índice, en caso contrario, regresa 0.

$ERROR-PGM regresa el nombre del programa que contiene el error.

$ERROR-PROC regresa el nombre del procedimiento que contiene el error.

$ERROR-STMT regresa el número de secuencia de la instrucción que contiene el error.

$ERROR-DESCRIPTION regresa un mensaje alfanumérico de hasta 80 caracteres que describe al error.

IV.4.5.5.3 Funciones para manipular datos

De acuerdo al valor que regresan cada una de las funciones, a

estas las clasificaremos en los grupos siguientes : funciones numéricas, funciones alfanuméricas y funciones booleanas.

IV.4.5.5.3.1 Funciones numéricas son funciones que regresan valores numéricos.

La sintaxis de las funciones numéricas es :

Nombre-función [parm-1,...,parm-n].

Donde :

El nombre de la función representa la función intrínseca proporcionada por IDEAL.

Los módulos de datos parm-1,...,parm-n son los parámetros para la función. Hay dos tipos de parámetros : parámetros simples y parámetros que serán especificados con una palabra reservada seguida por un signo de igual. El número, tipo y significado de cada parámetro sera especificado a cada función intrínseca. Algunas funciones no usan parámetros. Las finciones asociadas a la clase de funciones numéricas son : $ABS, $COUNT, $INDEX, $NUMBER, $REMAINDER, $ROUND, $SQRT y $PANEL-GROUP-OCCURS.

Función $ABS regresa el valor numérico de una expresión numérica.

La sintaxis de la función intrínseca $ABS es :

$ABS (expresión-numérica).

La función $COUNT regresa el número de iteraciones presentes en la construcción FOR EACH o LOOP referida.

La sintaxis de la instrucción intrínseca $COUNT es :

$COUNT (etiqueta).

Donde etiqueta es el nombre de la construcción FOR EACH o LOOP. Antes de la ejecución de cualquier construcción referida, el 0 es regresado al término de la construcción referida ; el número total de de iteraciones es regresado. Solamente los nombres para la instrucción FOR EACH o LOOP son operandos validos de $COUNT.

La función intrínseca $INDEX regresa la posición inicial de la primera ocurrencia de una subcadena dentro de una expresión alfanumérica en la forma de una literal numérica. $INDEX regresa 0 si la subcadena no se encuentra en la expresión.

La sintaxis de la función intrínseca $INDEX es :

$INDEX (expresión-alfanumérica,SEARCH=subcadena).

Donde :

La expresión-numérica es la cadena que será examinada para la

primera ocurrencia de la subcadena.

La subcadena es el identificador de una literal o grupo alfanumérico que será buscada en la expresión alfanumérica.

La función intrínseca $NUMBER regresa el valor numérico al efectuar la conversión de un valor alfanumérico dado.

La sintaxis de la función intrínseca $NUMBER es :

$NUMBER (x).

Donde x es el identificador de un campo o grupo alfanumérico, cuyo valor consistirá solamente de un signo opcional (+ o -) seguido de numerales con un punto decimal opcional.

La función intrínseca $REMAINDER regresa el residuo después de de haber realizado la división de una expresión numérica por otra.

La sintaxis de la función intrínseca $REMAINDER es :

$REMAINDER (m,div=n).

Donde m es el dividendo (éste será una expresión numérica) y div=n es el divisor que también será una expresión numérica.

La función intrínseca $ROUND regresa el valor de entrada, redondeado por el factor de redondeo especificado.

La sintaxis de la función intrínseca $ROUND es :

$$\$ROUND \quad (x, \begin{Bmatrix} FACTOR=f \\ ATTR=id \end{Bmatrix})$$

Donde x es la expresión numérica que será redondeada ; FACTOR indica que el valor será obtenido al redondear la expresión numérica al valor más cercano de acuerdo al factor de redondeo ; f representa el factor de redondeo ; ATTR indica que el valor es obtenido por redondeo a la unidad más cercana basada sobre los atributos del identificador especificado ; y el identificador id de un campo numérico cuyos atributos designan el factor de redondeo.

La función intrínseca $SQRT regresa la raíz cuarada de una expresion numerica.

La sintaxis de la función intrínseca $SQRT es :

$SQRT (expresión-numérica).

Donde la expresión-numérica representa al valor que se le extraerá la raíz cuadrada.

IV.4.5.5.3.2 Funciones alfanuméricas

Son aquéllas que regresan una expresión alfanumérica.

La sintaxis de las funciones alfanuméricas es :

Nombre-función [parm-1,...,parm-n].

Las funciones asociadas a la clase de las funciones alfanuméricas son :

$DATE, $EDIT, $HIGH, $LOW, $SPACES(ES), $STRING, $SUBSTR, $TIME, $TRANSLATE, y todas las funciones asociadas al manejo de errores.

La función alfanumérica $DATE regresa el valor alfanumérico con la fecha actual en el formato especificado.

La sintaxis de la función intrínseca $DATE es :

$DATE ('patrón-fecha' [,DATE=x]).

Donde patrón-fecha es la sucesión de caracteres (máximo 30) que representan las componentes y notación de la fecha, como sigue :

Componente Notación	Significado	EJEMPLO (supongamos 12 de junio de 1987)
YEAR	año completo	1987
YY	año sin siglo	87
Y	año sin década	7
MONT	nombre de mes en mayúscula	JUNIO
LCMONTH	nombre del mes en minúscula	junio
MON	nombre del mes abreviado en mayúscula	JUN

LCMON	nombre del mes abreviado en minúscula	jun
MM	número del mes con ceros	01
M	número del mes sin ceros	1
DD	día del mes con ceros	12
D	día del mes sin ceros	12

La x es una literal alfanumérica de 6 caracteres en el formato 'YYMMDD' o un campo alfanumérico de 6 caracteres que contiene un valor en ese formato. Si $DATE se omite, el día de la fecha actual es el de omisión.

La función intrínseca $EDIT regresa un valor alfanumérico al editar una literal o un campo dado de acuerdo al padrón especificado.

La sintaxis de la función intrínseca $EDIT es :

$EDIT (literal o campo, PIC='padrón-edición).

Donde la literal o campo es el modulo que será editado y el 'padrón-edición' especifica el formato deseado del campo o literal.

La función intrínseca $HIGH regresa el valor mayor en la sucesión alfanumérica examinada.

La función intrínseca $LOW regresa el valor mínimo en la sucesión alfanumérica examinada.

La función intrínseca $SPACE o $ESPACES regresa un espacio en blanco.

La función intrínseca $STRING regresa un valor alfanumérico de la sucesión de parámetros por concatenación de los valores obtenidos para cada parámetro.

La sintaxis de la función intrínseca $STRING es :

$STRING (parm-1,...,parm-n).

Donde la sucesión de parámetros parm-1,...,parm-n son los módulos que serán concatenados (en seguida se hacen conversiones a expresiones alfanuméricas, si es necesario).

Las acciones de $STRING varía con el tipo de parámetros como sigue :

1) En campos o literales alfanuméricos no hay cambio.

2) En los campos bandera, se convierten a un caracter de T o F.

3) El grupo será transformado a la concatenación del resultado

de $STRING sobre cada uno de los campos en el grupo. Si el grupo contiene un OCCURS, todas la ocurrencias son concatenadas, se ignora módulos redefinidos en el grupo.

4) El campo numérico será convertido a representación alfanumérica como sigue : si el número es negativo, un signo menos es generado, de otro modo, ningún signo aparece en la representación. La porción entera del número es representada en forma de despliegue de los numerales, en compañía de ceros de acuerdo a los atributos del campo. Pero sin la compañía de ceros, las unidades de posición correspondientes serán convertidas a blancos. Si, en este resultado intermedio se llevan blancos y un signo menos aparece, el signo menos es "flotante" a la posición adyacente al primer dígito significativo. Si el campo tiene una o más posiciones decimales, un punto decimal es generado, seguido por la parte decimal llevando ceros de acuerdo a los tributos del campo.

5) En la literal numérica cada dígito numérico será convertido a su correspondiente numeral alfanumérica, un signo más (+) convertido a un blanco, un signo menos se conserva.

6) No serán anidadas las funciones alfanuméricas utilizadas como parámetros de $STRING.

La función intrínseca $SUBSTR regresa una expresión alfanumérica que es parte (o toda) de otra expresión alfanumérica.

La sintaxis de la función intrínseca $SUBSTR es :

$SUBSTR(expresión-alfanumérica
[START=inicio][,LENGTH=longitud]).

Donde :

La expresión-alfanumérica es cualquier expresión alfanumérica. El inicio es la posición inicial de la expresión numérica que será extraída, el cual será especificado como un identificador o literal numérico. Si se asigna a un campo alfanumérico, el valor del campo que se obtiene con espacios de acuerdo a las reglas de SET o MOVE. La posición por omisión es el 1. Si el valor de inicio es menor que 1, el valor de 1 se asume.

La longitud es el número de caracteres que será extraída de la expresion alfanumérica comenzando en el inicio. La longitud será especificada como una literal o identificador numérico. Si la longitud se omite, todos los caracteres desde el principio hasta el final de la expresión alfanumérica son extraídos. Si el valor de longitud excede a la longitud restante de la expresión alfanumérica, entonces esta longitud es utilizada. Si el valor extraído es menor, se regresa el valor nulo como valor de la función.

IV.4.5.5.3.3 Funciones booleanas son aquellas que regresan un valor booleano de verdadero (TRUE) o falso (FALSE).

Las funciones intrínsecas asociadas a esta clase de funciones son : $ALPHABETIC, $NUMERIC, $PFn, $ENTER-KEY, $CURSOR y $RECEIVED. Las

cutro últimas ya se explicaron.

La función intrínseca $ALPHABETIC regresa un valor booleano de verdadero o falso según si el módulo alfanumérico dado es alfabético, es decir, consiste solamente de las letras mayúsculas o minúsculas desde A hasta Z y/o blancos.

La sintaxis de la función intrínseca $ALPHABETIC es :

$ALPHABETIC (x).

Donde x es el identificador de un campo o grupo alfanumérico.

La función intrínseca $NUMERIC regresa un valor booleano de verdadero o falso según si el valor de un campo es convertible a numérico.

La sintaxis de la función intrínseca $NUMERIC es :

$NUMERIC (x).

Donde x es el identificador de un campo numérico o alfanumérico, o de un grupo alfanumérico. Para que $NUMERIC regrese un valor verdadero para un módulo alfanumérico, el modulo consistirá solamente de un signo opcional de (+ o -) seguido por numerales con un punto decimal opcional. Se permite llevar o traer blancos, pero son ignorados en la entrada del valor alfanumérico.

IV.4.5.5.4 Funciones para identificación (clase de funciones alfanuméricas)

La función intrínseca $TERMINAL-ID regresa la identificación de la terminal mediante cuatro caracteres en la forma de un valor alfanumérico.

La sintaxis de la función intrínseca $TERMINAL-ID es :

$TERMINAL-ID.

Esta función puede ser utilizada, por ejemplo, para prueba de seguridad o para propósito de contabilidad.

La función intrínseca $USER-ID regresa la identificación del usuario mediante tres caracteres en la forma de un valor alfanumérico.

La sintaxis de la función intrínseca $USER-ID es :

$USER-ID.

Esta función pude utilizarse, por ejemplo, para prueba de seguridad o para propósitos de contabilidad.

La función intrínseca $USER-NAME regresa el nombre del usuario en 15 caracteres en la forma de un valor alfanumérico.

La sintaxis de la función intrínseca $USER-NAME es :

$USER-NAME.

Esta función puede ser utilizada, por ejemplo, para prueba se seguridad o para propósitos de contabilidad.

La función intrínseca $ACCOUNT-ID se utiliza en una instrucción SET para asignar dinámicamente la identificación de una transacción mediante 4 caracteres que será catalogado después del siguiente TRANSMIT.

La sintaxis de la función intrínseca $ACCOUNT-ID es :

$ACCOUNT-ID.

La función intrínseca $FINAL-ID se utiliza en una instrucción SET para asignar dinámicamente la identificacion de una transacción que será catalogada cuando finaliza la sesión de IDEAL explícitamente con el comando OFF, o implícitamente el comando SET RUN QUIT IDEAL es puesto a "Y".

La sintaxis de esta última función es :

$FINAL-ID.

De las instrucciones descritas utilizaremos a aquéllas que nos permitan desarrollar el proceso lógico del programa de aplicación

específico, el cual para su ejecución necesita de algunos comandos de IDEAL. (ver anexo-1).

V. Conclusiones

V.1 Experiencias y sensaciones de usuarios

De los objetivos planteados por la dirección de procesamiento de datos de la Tesorería del Distrito Federal en cuanto a: una buena administración de datos, el mejoramineto de la productividad del personal del área de desarrollo de sistemas y la aminoración del esfuerzo de mantenimiento de sistemas de esta institución, condidero que no se han logrado en su totalidad, dado que las facilidades y recursos del sistema manejador de datos (DATACOM/DB) no se están empleando de manera óptima. Esto se debe a que existieron y aún existen serios problemas atribuibles no al producto, sino a la forma en que se instrumentó y al deficiente apoyo técnico prestado por la compañía ADR/KRONOS. Entre los problemas que se pueden observar, se encuentran los siguientes: la resistencia al cambio, la instalación del sistema DATACOM/DB, el aprendizaje del mismo, el aislamiento de las bases de datos y la rotación del personal por el área de procesamiento de datos. Con respecto a la resistencia al cambio, de los sistemas de archivos tradicionales basados en sistemas de información a otros nuevos sistemas basados en DATACOM/DB, hubo personas del área de procesamiento de datos que se oponían al cambio, argumentando, que el cambio no era de ninguna manera trivial y que por lo tanto era conveniente continuar con los sistemas tradicionales, de los cuales se tenía un amplio conocimiento. Los argumentos anteriores considero que son son justificables dado que como se ha visto el producto es muy poderoso. En cuanto a la instalación del sistema DATACOM/DB, se vió afectado el departamento de soporte técnico que conjuntamente con técnicos de la compañía de ADR/KRONOS de México se encargaron de instrumantar este sistema. Debido al desconocimiento del mismo por ambas

partes no fue fácil la tarea, pero al final se logró hacerlo funcionar con asesoría de la compañía ADR/KRONOS internacional. Con respecto al aprendizaje, a mi modo de ver resultó ser un problema sustantivo. Aunque al principio se dieron cursos de capacitación para algunos miembros del personal del área de desarrollo de sistemas, la evaluación del aprendizaje fue mínimo debido a que los primeros cursos se dieron en Inglés por técnicos de la compañía ADR/KRONOS internacional, lo cual fue un serio problema, otros cursos se dieron en español pero con una pésima calidad por técnicos de la compañía ADR/KRONOS de México. Finalmente se logró que nuevamente se dieran otros cursos en español por técnicos de la compañía ADR/KRONOS internacional, lo cual mejoró un poco la situación del aprendizaje. Otros aspectos que influyeron en el aprendizaje fueron la falta de práctica y la documentación en cuanto a manuales de consulta y guías de usuario, los cuales se encuentran en Inglés. En cuanto al aislamiento de las bases de datos esto fue provocado debido a que en un principio cada uno de los departamentos del área de procesamiento de datos analizó y diseñó sus bases de datos por separado para sus aplicaciones particulares. Lo anterior se intentó solucionar posteriormente con un nuevo análisis y diseño de las bases de datos para integrarlas, tarea que actualmente se está llevando a cabo. Con respecto a la rotación de personal, puedo decir que esto afectó de una u otra manera, ya que un número considerable de personas que habían participado en el análisis, diseño de los modelos conceptuales de las bases de datos, programación de los sistemas y en la administración de los recursos dejaron la tesorería.

Con todo lo mencionado se puede pensar que el sistema manejador de

bases de datos (DATACOM/DB) no ha apoyado en la solución de la problemática de Tesorería, y que por lo tanto, este sistema no cumple con las expectativas de la Tesorería en cuanto a la administración de datos. No hay que caer en tal extremo, considero que lo que se necesita es que el personal del área de desarrollo de sistemas se capacite más para poder utilizar en forma óptima las facilidades y recursos del sistema DATACOM/DB. No se duda que dicho personal con su entrega ,dedicacion, responsabilidad y profesionalismo, como lo demuestran los sistemas, que han sido liberados (sistema integral de fiscalización, registro de cobro de obligaciones fiscales, sistema único de registro, sistema de volantes de control, sistema de control de mobiliario y equipo, sistema de información para el control de los mercados públicos del D.f, sistema de placa permanete y sistema de control de procesos jurídicos) que en un tiempo no lejano puedan dar solución a la problemática existente. Y por otro lado el personal de nuevo ingreso proporcionarle los recursos necesarios para un rapido aprendizaje. En este punto condidero que el presente trabajo apoyará en gran medida.

Anexo-1

Comandos de IDEAL

En el ambiente de IDEAL se cuenta con una gama de comandos que son utilizados por el personal de la Tesorería dedicado al desarrollo y mantenimiento de los diferentes programas de aplicación de la misma. En la presente sección no se pretende dar todos los comandos de IDEAL si no más bien aquellos comandos de uso más frecuentes, los cuales clasificaremos en los siguientes grupos : comandos asociados al manejo de líneas, comandos asociados para desplegar información, comandos asociados para modificar información, comandos asociados para compilación y ejecución de programas, comandos asociados al manejo de impresión y comandos asociados al manejo de pantallas.

Comandos asociados al manejo de líneas

Se utilizan para eliminar, para copiar, para mover, para insertar y para duplicar una o varias líneas del programa de aplicación antes o después según la posición del cursor, la cual está asociada con un número específico de la secuencia del programa. Los símbolos que utilizaremos para estos comandos son:

D borra una sola línea

DD...DD	borra un rango de líneas
C	copia una sola línea antes o despues del cursor
CC...CC	copia un rango de líneas antes o después del cursor
A	A: después del cursor
B	B: antes del cursor
M	mueve una sola línea antes o después del cursor
MM...MM	mueve un rango de líneas antes o después del cursor
A	
B	
[n]I	inserta una o varias líneas después del cursor (1 ≤ n ≤ 100)
[n]R	repite una o varias líneas después del cursor.

Comandos asociados para desplegar información

Su función es proporcionar información acerca de programas, vista lógica de datos, pantallas, reportes, entidades existentes en el estado de prueba, compilaciones y ejecuciones. La información proporcionada por estos comandos solamente es para consulta, es decir, la información no se puede modificar. Los comandos que asociamos para tales funciones son los siguientes: DISPLAY PROGRAM, DISPLAY DATAVIEW, DISPLAY PANEL, DISPLAY REPORT, DISPLAY INDEX y DISPLAY OUT.

Nota: para no escribir todos estos comandos en su forma completa utilizaremos las siguientes abreviaciones:

DISPLAY = DIS

PROGRAM = PGM

DATAVIEW = DVW

PANEL = PNL

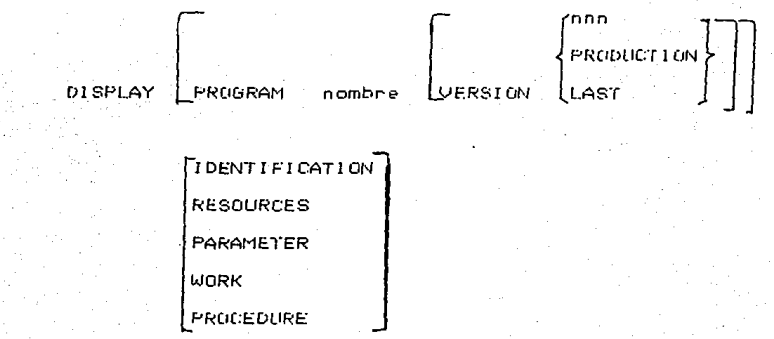
REPORT = RPT

INDEX = IND

Comando DISPLEY PROGRAM

Despliega el listado del programa en forma protegida, es decir, no se puede hacer ninguna modificación a este listado.

La sintaxis del comando DISPLEY PROGRAM es:



Comando DISPLAY DATAVIEW

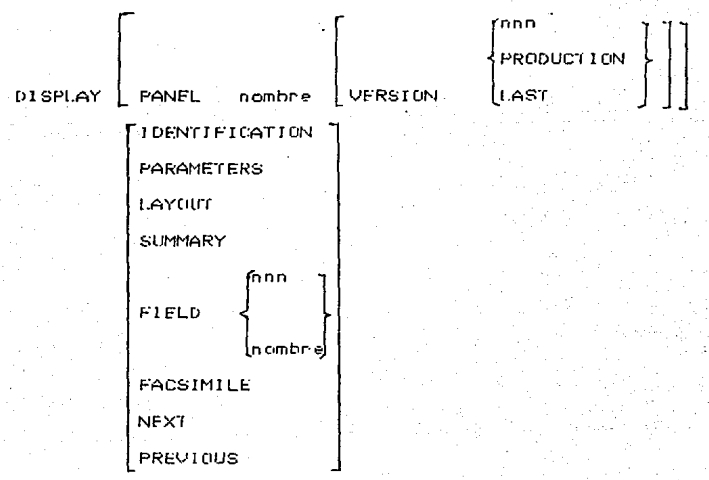
Despliega el nombre de la vista lógica de datos y los campos asociados a ésta. Nuevamente la información de la vista lógica de datos no se puede modificar.

La sintaxis del comando DISPLAY DATAVIEW es:

Comando DISPLAY PANEL

Despliega el formato de la pantalla en forma protegida, es decir, no se puede modificar el formato de la pantalla.

La sintaxis del comando DISPLAY PANEL es:



Nota: para no escribir todos estos comandos en su forma completa utilizamos las siguientes abreviaciones:

V	=	VERSION
PARM	=	PARAMETERS
LAY	=	LAYOUT
SUM	=	SUMMARY
FAC	=	FACSIMILE

Comando DISPLAY REPORT

Despliega el formato del reporte en forma protegida, es decir, este formato no es modificable.

La sintaxis del comando DISPLAY REPORT es:

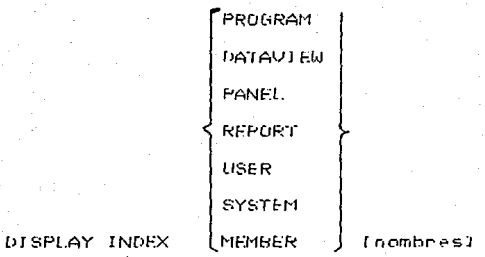
```
                                              ⎧nnn         ⎫
DISPLAY [ REPORT nombre [ VERSION  ⎨PRODUCTION ⎬ ] ]
                                              ⎩LAST        ⎭
          ⎡IDENTIFICATION⎤
          ⎢PARAMETERS    ⎥
          ⎢HEADING       ⎥
          ⎢DETAIL        ⎥
          ⎣COLUMN        ⎦
```

Comando DISPLAY INDEX

Despliega programas, vistas lógicas de datos, pantallas, reportes,

usuarios, sistemas y miembros segun la ocurrencia del tipo de entidad especificada.

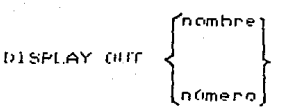
La sintaxis del comando DISPLAY INDEX es:



Comando DISPLAY OUT

Despliega la salida de compilacion y ejecución que se encuentra en la biblioteca de salida.

La sintaxis del comando DISPLAY OUT es:



comandos asociados para modificar información

Su función es proporcionar información para consulta o modificación en cuanto a programas, pantallas y reportes. Los comandos que asociamos a tales funciones son las siguientes: EDIT PROGRAM, EDIT

PANEL, CHANGE, RENUMBER, CREATE Y DELETE.

Comando EDIT PROGRAM

Se utiliza para desplegar un programa del estado de prueba, el cual sera modificado o consultado.

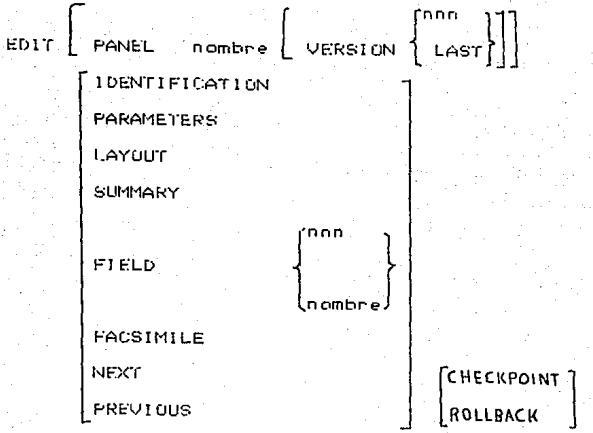
La sintaxis del comando EDIT PROGRAM es:

```
                                              {nnn }
EDIT [ PROGRAM   nombre [ VERSION  {LAST}]]
       [IDENTIFICATION]
       [RESOURCES     ]
       [PARAMETERS    ]
       [WORK          ]      [CHECKPPOINT]
       [PROCEDURE     ]      [ROLLBACK   ]
```

Comando EDIT PANEL

Se utiliza para desplegar el formato de pantalla, el cual es consultado u modificado.

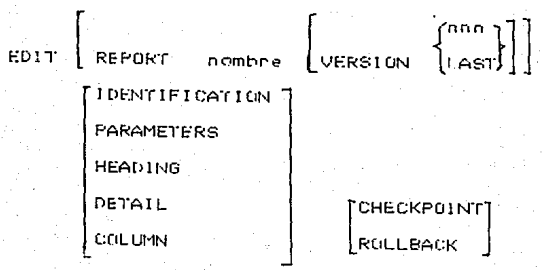
La sintaxis del comando EDIT PANEL es:



Comando EDIT REPORT

Se utiliza para desplegar el formato del reporte que será modificado u consultado.

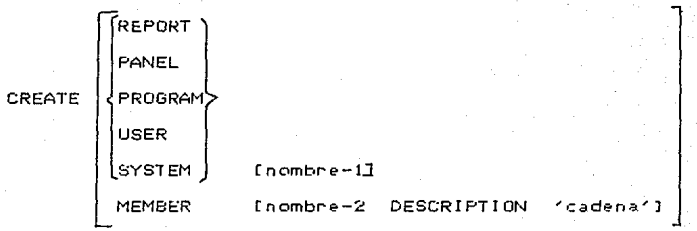
La sintaxis del comando EDIT REPORT es:



Comando CREATE

Se utiliza para la creación de pantallas, reportes, programas, usuarios, sistemas y miembros.

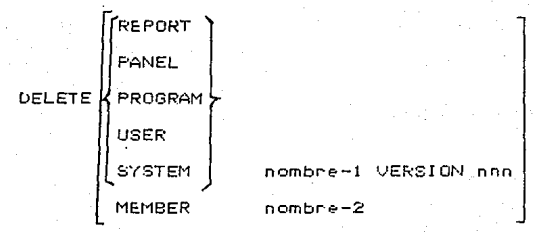
La sisntaxis de la instrucción CREATE es:



Comando DELETE

Se utiliza para borrar pantallas, reportes, programas, usuarios, sistemas y miembros.

la sintaxis del comando DELETE es:



Comando CHANGE

Se utiliza para cambiar una cadena por otra.

La sintaxis de la instrucción CHANGE es:

CHANGE / cadena-a / cadena-b /.

Comando RENUMBER

Se utiliza para renumerar los números de secuencia de la entidad presente.

La sintaxis del comando RENUMBER es:

RENUMBER [[BY] n] , n deberá ser 1, 10, 100 o 1000.

Comandos asociados al proceso de compilación

La función de estos comandos es obtener la compilación de un programa en el estado de prueba. Los comandos asociados a tal función son: COMPILE ó COM *. Cuando se compila un programa que se encuentra en la biblioteca de programas se utiliza el comando COMPILE, en caso de que el programa se encuentre desplegado en pantalla se utiliza el comando COM *.

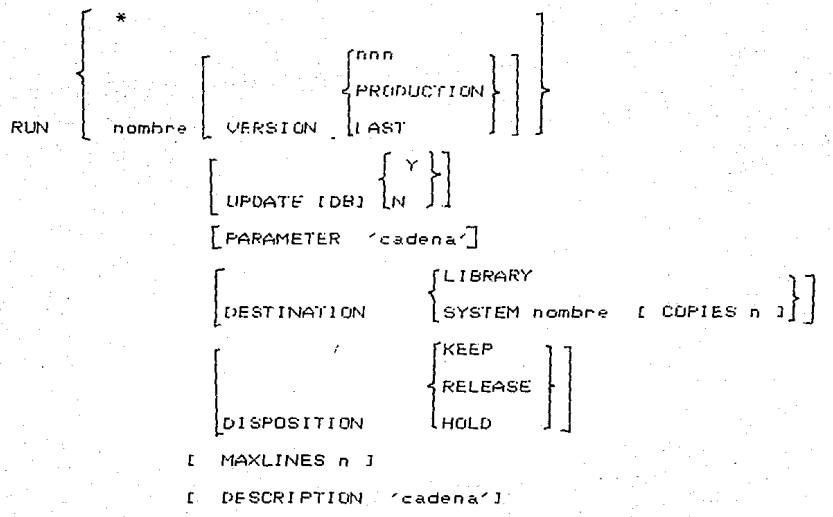
La sintaxis del comando COMPILE es:

```
COMPILE [ * | nombre [ VERSION {nnn | LAST} ] ]
        [ IDE {Y | N} ]
        [ EXD {Y | N} ]
        [ ADV {Y | N} ]
        [ MEL {Y | N} ]
        [ DESTINATION {LIBRARY | SYSTEM  nombre  [ COPIES número ]} ]
        [NAME nombre]
        [ DISPOSITION {KEEP | RELLEASE | HOLD} ]
     [DESTINATION  'cadena']
```

Comandos asociados al proceso de ejecución

La función de estos comandos consiste en iniciar la ejecución del programa. Los comandos a tal función son: RUN ó RUN *. Cuando se ejecuta un programa que se encuentra en la biblioteca de programas, se utiliza el comando RUN, en caso de que el programa se encuentre desplegado en pantalla, se utiliza el comando RUN *.

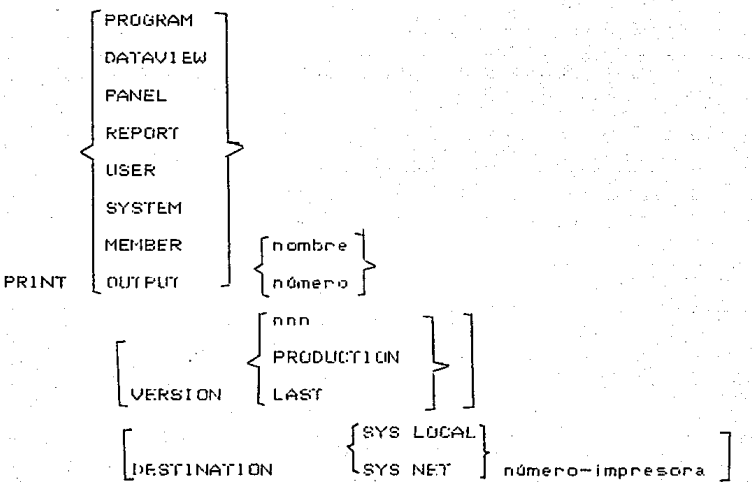
La sintaxis del comando RUN es:



Comandos asociados al manejo de impresión

La función que tienen estos comandos es la de suministrar la impresión de una ocurrencia de entidad especifica, la cual puede corresponder a: programa, vista lógica de datos, pantalla, reporte, usuario, sistema, miembro y salida de compilación o ejecución. El comando que se asocia a tal función es: PRINT.

la sintaxis del comando PRINT es:



Comandos asociados al manejo de pantallas

La función de estos comandos es adelantar, retroceder, ir al tope, ir al fin de una pantalla especifica. Los comandos utilizados para estas funciones son los siguientes: SCROLL FORWARD / BACKWARD, SCROLL TOP / BOTTOM.

Comando SCROLL FORWARD / BACKWARD

Se utiliza para avanzar o retroceder un pantalla.

BIBLIOGRAFIA

1. Aplication Development Reference Manual (ADR)
2. Data Base Design (ADR) Aplied Data Research
3. Estudio de viabilidad para la adquisición de un sistema anministrador de bases de datos (Tesorería del Distrito Federal)
4. Cics General Information
   (Customer Information Control Systems)
5. Organización de las bases de datos
   Editorial : Prentice Hall (PHH)
   Autor : James Martin
6. Data Base Architecture
   Editorial : Van Nostrand Reinhold Company
   Autor : Ivan Flores
7. Principles of Database Systems
   Editorial : Computer Science Press
   Autor : Jeffrey D. Ullman
8. Strategic Data-Planning Methodologies
   Editorial : Prentice Hall
   Autor : James Martin