SISTEMAS DE INFORMACION GERENCIAL

Fecha	Duración	Tema	Profesor
Agosto 4	17 a 19 h	Sistemas: Concepto de Sistemas y Aspectos de Entropía y Retroalimentación	Lic. Fernando Arenas Estrada
	19 a 21 h	Ciclo de vida de un sistema Análisis	Lic. Jesús Carranza Castellano Lic. Fernando Arenas Estrada
Agosto 5	9 a 11 a.m.	Diseño	
	11 a 13 h	Programación	Lic. Jesús Carranza Castellano
	13 a 14 h	Comida	
	14 a 15 h	Instalaciones	Lic. Fernando Arenas Estrada
	15 a 16 h	Operación	
	16 a 17 h	Obsolescencia	Lic. Jesús Carranza Castellano
Agosto 11	17 a 19 h y de 19 a 21 h	Auditoria Técnico-Económico de Sistemas de Información	Lic. Jesús Carranza Castellanos Lic. Fernando Arenas Estrada
Agosto 12	9 a 11 a.m. 11 a 13 h	El Impacto de los Sistemas de Información en las aréas de la organización empresarial.	Lic. Fernando Arenas Estrada Lic. Jesús Carranza Castellano
	13 a 14 h	Comida	
	14 a 16 h 16 a 17 h	El Impacto de los Sistemas de Información en las áreas de la organización empresarial	Lic. Fernando Arenas Estrada Lic. Jesús Carranza Castellano
Agosto 18	17 a 19 h 19 a 21 h	Técnicas de los Sistemas de Información	Ac. Ernesto Archundía Dr. Víctor Guerra Ortiz
Agosto 19	9 a 11 a.m. 11 a 13 h	" " " " " " " " " "	Act. Ernesto Archundía Dr. Víctor Guerra Ortiz
	13 a 14 h	Comida	
	14 a 16 h 16 a 17 h	Técnicas de los Sistemas de Información " " " " "	Act. Ernesto Archundía Dr. Víctor Guerra Ortiz

SISTEMAS DE INFORMACION GERENCIAL

Fecha	Duración	Tema	Profesor
Agosto 25	17 a 19 h	Técnicas de los Sistemas de Información	Dr. Víctor Guerra Ortiz
	19 a 21 h	" " " "	Act. Ernesto Archundía
Agosto 26	9 a 11 a.m.	Laboratorio de Sistemas (La sesión de laboratorio se llevará a cabo en el Instituto de Investigaciones en matemáticas Aplicadas y en Sistemas, C.U.)	Dr. Víctor Guerra Ortiz
	11 a 13 h		Act. Ernesto Archundía
	13 a 14 h	Comida	
	14 a 16 h	Laboratorio de Sistemas	Dr. Víctor Guerra Ortiz
	16 a 17 h		Act. Ernesto Archundía

'edcs. 11, VIII, 78.

- INDICE -

SISTEMAS DE INFORMACION GERENCIAL

SISTEMAS: CONCEPTO DE SISTEMA Y ASPECTOS DE ENTROPIA Y RETROALIMENTACION

I

SISTEMAS: CONCEPTO DE SISTEMA Y ASPECTOS DE ENTROPIA Y RETROALIMENTACION

Durante las últimas tres décadas la ciencia ha visto surgir y fortalecerse una rama específica del conocimiento denominada "CIBERNETICA" que fue introducida a la jerga científica por Norbert Wienner, con el afán de conjugar en el vocablo todo lo relacionado con "el estudio analítico del isomorfismo de la estructura de las comunicaciones en los mecanismos, los organismos y las sociedades" (1). y desde un principio tuvo una fuerte connotación filosófica, estableciendo abiertamente un paralelismo entre los "sistemas" cualesquiera que fuese su naturaleza (sociales, biológicas, mecánicas).

Las reflexiones de Wienner sobre la psicología y el sistema nervioso, así como la lucha entre el progreso y el caos constituyeron a mitad de este siglo los ingredientes de una polémica todavía inconclusa que se vió recrudecida por sus posteriores referencias al "aprendizaje" de las máquinas (2) y la exaltada ponderación de la idea del universo contingente atribuida a W. Gibbs.

Aún bajo tales tormentas, la Cibernética tomo carta de naturalización en el mundo de la investigación y con ella su objeto de abstracción: el sistema.

La voz "sistema" tiene en el lenguaje desde siempre la connotación de agrupamiento de partes ordenadas entre sí y su acepción común la relacionaba a la Biología, como conjunto de órganos que intervienen en alguna función vegetativa. (3).

Sin embargo, la Cibernética le ha asignado un sentido específico más concreto; a pesar del hecho de que cada autor que haya tratado el tema define y describe de manera distinta tal concepto un acercamiento al significado que aquí nos interesa puede encontrarse en las palabras de Duhalt Krauss describiendo las:

"Notas esenciales del concepto de Sistema"

a) Un conjunto de cosas o partes;
b) Integradas e interdependientes;
c) Cuyas relaciones entre sí y con sus atributos las hacen formar un todo unitario y organizado;
d) Que cumple determinado propósito o realiza determinada función;
e) Y que puede mantener cierto grado de estabilidad, aunque la materia y la energía que lo compongan estén sujetas a cambios constantes.

Esta definición conviene a cualquier clase de sistema. (4).

Así pues, quedan establecidos los componentes del concepto de sistema como: organización, interacción e interdependencia; integración; funcionalidad y estabilidad.

Queda claro también, que el sistema circulatorio de un ser vivo -en cuanto a sistema- es idéntico a un sistema de gobierno o a un sistema administrativo, quedando todos ellos inscritos en nuestra definición arriba señalada.

Aún así, siendo el objeto de estudio de la Cibernética los sistemas, y dentro de ellos los procesos de comunicación y control, caen fuera del sentido implícito del vocablo sistema (en tanto parte de la Cibernética) algunos sistemas naturales como los orográficos, astronómicos y otros que, no siendo ni máquinas, ni organismos vivos, ni organismos sociales, no desarrollan ningún proceso de comunicación y/o control. (5).

En otro orden de ideas, los sistemas se presentan en el universo bajo una estricta jerarquización, subdividiéndose a su vez en subsistemas, o bien agrupándose en algo que podemos describir como un supra-sistema; de hecho puede pensarse que cada sistema no es sino un "sistema de sistemas", siendo posiblemente el átomo el más pequeño sistema analizado por el hombre.

"De esta manera, un organismo es concebido como un todo integrado, susceptible de ser considerado como dividido en subsistemas asociados en la operación total; su estructura está formada por varios subsistemas ordenados jerárquicamente, o acoplados, en donde la salida de uno de ellos se convierte en la entrada de otro". (6)

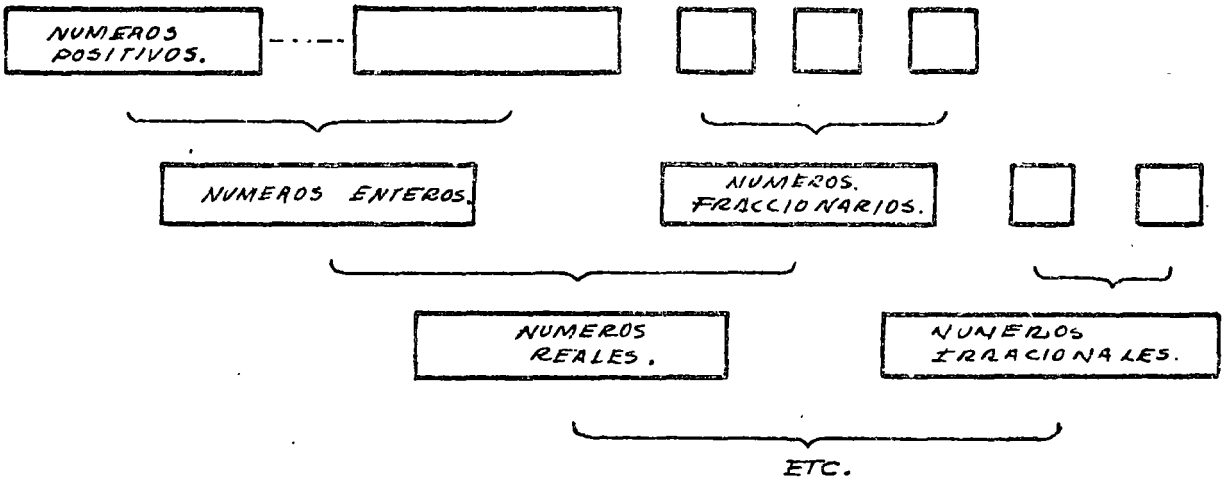
La analogía (y más que eso, aplicación del enfoque de sistemas) con la teoría de conjuntos no puede ser más feliz: donde leemos, por ejemplo, que el conjunto de los números enteros está contenido en el conjunto de los números fraccionados y que este último se inscribe en el conjunto de los números reales, podríase escribir que el sistema de reclutamiento está contenido en el sistema de administración de recursos humanos, mismo que se incluye en el sistema de administración de recursos de todo tipo, en la administración general de cualquier empresa; al menos este símil nos parece fructuoso, si observamos la figura 1.

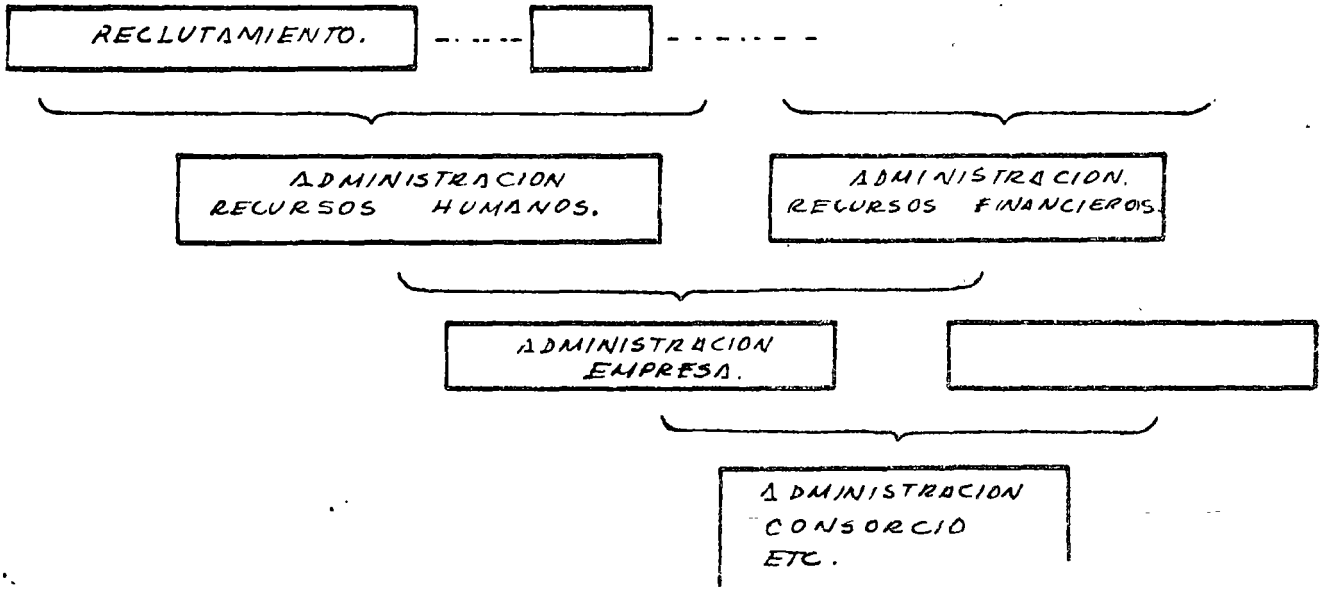
Alrededor de todo esto, se encuentra actualmente en desarrollo una "teoría general de sistemas" impulsada principalmente por la "sociedad para el desarrollo de una teoría general de sistemas" que encabeza el sociólogo G. Von Berthalanffy (7) y que fue fundada en 1957.

Asimismo se habla de una "teoría general de los sistemas de información" que incluye numerosas teorías subordinadas, (8) por tal razón, es frecuente la confusión de ambas expresiones; conventualmente podemos distinguirlas basadas en la universalidad del concepto "sistema" frente a la particularidad relativa (dado un adjetivo) del concepto "sistema de información", con lo que se configura una relación entre ambos de género próximo y diferencia específica, de acuerdo a la concepción aristotélica de definición.

Para los fines de este curso resulta evidente que nuestra atención había de centrarse en los sistemas de información, lo que nos permite hacer a un lado la encarnizada polémica que acerca de la virtual existencia de una teoría general de sistemas, continúa en debate. (9)

FIGURA 1


NUMEROS POSITIVOS.
NUMEROS ENTEROS.
NUMEROS. FRACCIONARIOS.
NUMEROS REALES.
NUMEROS IRRACIONALES.
ETC.


RECLUTAMIENTO.
ADMINISTRACION RECURSOS HUMANOS.
ADMINISTRACION. RECURSOS FINANCIEROS.
ADMINISTRACION EMPRESA.
ADMINISTRACION CONSORCIO ETC.

Bástenos con saber que en sistemas relativamente aislados es dable expresar axiomáticamente teorías menos generales y más elaboradas empíricamente, y que podemos entender una teoría general de sistemas como una serie de postulados que describa y explique el comportamiento más probable de cualquier tipo de sistema, permitiéndonos predecir con alto grado de certeza su comportamiento futuro. (10)

Con las anteriores salvedades, podemos intentar un acercamiento a los axiomas o postulados que integrarían la teoría general de sistemas, con la seguridad de que nos brindarán elementos de juicio suficientes para cimentar nuestra comprensión del tema:

- Unidad: Cualquier sistema es un todo indisoluble cuyas partes están interrelacionadas, son interactuantes y dependen entre sí, y el todo se conduce unitariamente, por complejo que sea.
- Subordinación: El todo (su propósito) determina a las partes(sus funciones particulares) de tal forma que estas últimas derivan su naturaleza de su posición dentro del todo.
- Estabilidad: La identidad del todo se preserva, pero las partes se modifican.
- Organización: El todo es más que la suma de las partes. La organización brinda al sistema diferentes características de los componentes individuales.
- Jerarquía: Las partes de un todo pueden a su vez subdividirse en partes (subsistemas de un sistema).

A partir de tales axiomas que hemos concentrado groseramente, surge la metodología "sistémica" que consisten en enfocar todo fenómeno observable como un sistema susceptible de descomposición en partes que pueden sujetarse a un estudio por separado cada una (análisis) sin dejar de entenderlo -en todo momento- como un universo.

El vocablo inglés "sistemic" no tiene hasta hoy ningún correspondiente en español; a diferencia de "sistematic" (sistemático) no significa aquello que "se ajusta a un sistema" sino más bien denota "referente a los sistemas". Hábida cuenta de que la diferencia es conceptual y no semántica hemos de adoptar el anglicismo (de ingrata fonética tal vez) de "sistémico" amparados en la circunstancia de que algunos traductores y autores saltaron sobre el idioma antes que nosotros.

Así pues, un "enfoque sistemático de un fenómeno sería el que siguiera cualquier sistema de conocimiento (estructuralista, funcionalista, etc.), mientras que un enfoque sistémico es el que concibe al fenómeno de acuerdo con la teoría de los sistemas, esto es, como parte de un sistema, como sistema o como conjunto de sistemas". (11)

Con tales herramientas, podemos ahora aplicar el enfoque sistémico a cualesquiera fenómeno observable en los organismos administrativos, vistos como sistemas sociales, verbigracia una empresa de negocios que como todo sistema atendible por la Cibernética reciben influencias externas e influyen en el exterior.

De hecho todo sistema cuenta con una entrada (insumo) desarrolla un proceso de tal entrada y arroja una salida (producto) y en una administración no es difícil concebir los insumos materiales, energéticos (entradas físicas) o informativos (entrada de comunicación, mensaje); tampoco escapa la identificación del proceso (físico o del mensaje) ni de los productos de dicho proceso (salidas materiales y de in-

formación); con la misma facilidad puede extenderse el símil a una dependencia gubernamental: "Un negocio es un conjunto de personas y recursos organizado en un todo complejo, con el propósito de alcanzar una serie específica de objetivos o metas. Por ello, un negocio puede ser todo o parte de una compañía, un departamento, una oficina u órgano gubernamental"; (12) seguimos pues validando nuestra metodología sistémica. Ver figura 2.

En este contexto, quedaría fuera de lugar afirmar que el universo, pudiendo ser estudiado como un sistema, tiende al caos y que en él, el desorden es más probable que la organización.

Aventurar una afirmación como la anterior obliga a su justificación pero hemos de adelantar que la intuición de un "universo contingente", esto es, una realidad aleatoria, no predecible, cambiante y no sujeta a plan, deberá hacernos arribar al concepto de ENTROPIA, y con él, al de RETROALIMENTACION.

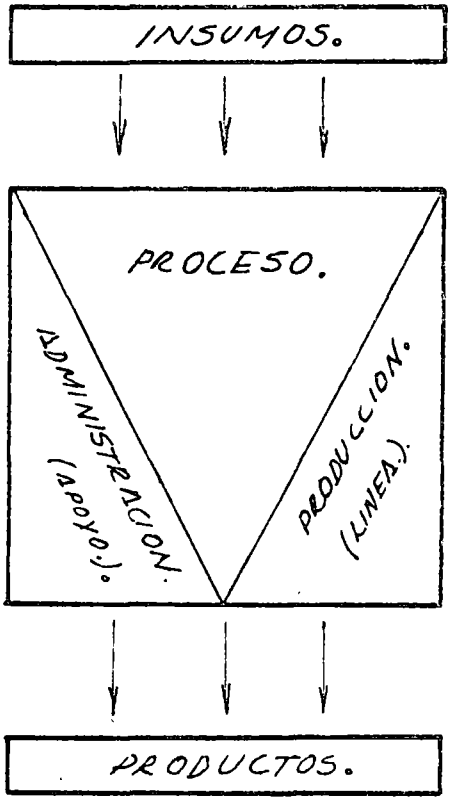
Existen aseveraciones tales como "la verdad de hoy será la mentira de mañana", y "lo único que no cambia es que todo cambia", cuyo uso generalizado no invalida su certeza dialéctica; de hecho, la ciencia avanza demostrando que los conocimientos anteriores al avance no eran exactos y en ocasiones ni siquiera parcialmente ciertos.

La construcción y destrucción de paradigmas científicas son en cierto modo inherentes a la investigación. Un paradigma conforma un "andamiaje intelectual" desde el que el investigador intenta construir nuevas teorías y cuando tal andamiaje se desvanece tiene lugar una estrepitosa revolución científica que obliga al investigador a modificar sus enfoques. Ejemplo:

El paradigma del medioevo de que la tierra era plana no impuso obstáculos mayores a la AGRIMESURA; antes bien, permitió en esa área un desarrollo suficiente; la desaparición de ese concepto permitió más tarde un acelerado progreso de la navegación, la astrología y otras ramas de la ciencia, pero ciertamente todos recordamos la oposición de algunas importantes institucipnes sociales a la difusión del nuevo paradigma.

Otros paradigmas importantes han sido: La indivisibilidad del átomo (en la física), la creación de demanda autónoma por todo incremento de oferta (en economía) y el condicionamiento Hegeliano de la existencia por la conciencia (en filosofía). Todos ellos han sido desmentidos por el avance científico.

Desde el siglo XVII y hasta finales del XIX, el paradigma de la física Newtoriana describía un universo, un cosmos, en el que absolutamente todo discurría (debía discurrir) con estricto apego a alguna ley, en el cual la totalidad del futuro se condicionaba al pasado; bajo tal paradigma la física (y la ciencia toda) se veía obligada a enunciar y formular esa ciencia como si estuviese sometida a leyes que pudiesen justificarse irremisiblemente hasta la última cifra decimal; actualmente, y gracias a la introducción de métodos estadísticos crecientemente sofisticados en la investigación, puede observarse una actitud diferente: la física no se ocupa de lo que ocurrirá siempre sino más bien de lo que pasará con una probabilidad muy grande (13); lo fundamental de tal actitud consiste en considerar no un universo sino todos aquellos universos que son posibles respuestas a un limitado conjunto de preguntas; resulta así indispensable discernir en que medida son probables (en un conjunto mayor de universos) las respuestas que pueden darse a ciertas preguntas para algunos de ellos: lo cierto para A y B, será probablemente cierto para C, D, E y F.


INSUMOS.
PROCESO.
ADMINISTRACION. (APOYO.).
PRODUCCION. (LINEA.).
PRODUCTOS.

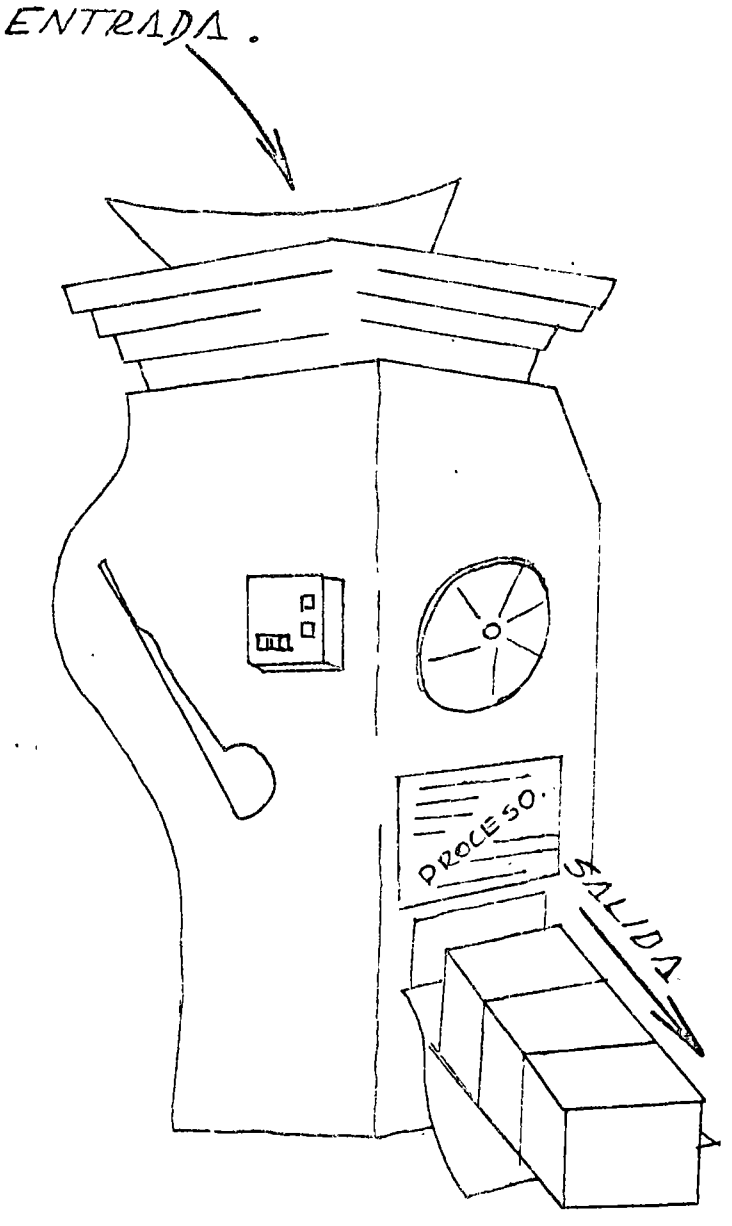

ENTRADA.
PROCESO.
SALIDA

figura 2

Se llama ENTROPIA al logaritmo de esa probabilidad cuya principal característica es la de ser siempre creciente; en el cosmos derivado de esta concepción, el caos es más probable, el orden menos probable.

Dada la ENTROPIA como una medida de la tendencia a la desorganización, la información que suministra un conjunto de mensajes es una medida de organización.

La idea de CAUSALIDAD versus CASUALIDAD ha venido regenerándose en CAUSALIDAD y CASUALIDAD integrando la contingencia como un efecto cuyas causas no pueden aprehenderse (todas ellas) en el estudio del fenómeno, pero que (en tanto efecto de esas causas desconocidas) deberá asimilado como POSIBILIDAD CONTINGENTE en la teoría.

Y de ahí la retroalimentación: En un universo contingente, cualquier sistema (social, orgánico o administrativo) deberá informarse no sólo acerca de la actividad que debe realizarse, sino también de la acción VERDADERAMENTE EJECUTADA, y con tal información rectificar en su caso sus procesos.

Al utilizar una cámara de la compresión en un submarino, no basta ordenar la apertura de los conductos que la llenarán de agua, sino que el mecanismo para abrirlos deberán ratificar que las compuertas de acceso al resto del aparato estén herméticamente cerradas, de lo contrario (esto es, sin la información acerca de las propias acciones inconclusas del mecanismo) algo pudo haber impedido el cierre del acceso, y el agua invadiría incontroladamente a todo el submarino.

Esta regulación de un aparato (sistema) respecto a su funcionamiento real (y no "deseado") se llama Retroalimentación y permite frenar la tendencia (mecánica en este ejemplo) hacia la desorganización "contingente", invirtiendo la dirección espontánea de la ENTROPIA.

Ejemplo de retroalimentación en un ser vivo es el sentido cenestésico que poseen todos los animales (incluso el hombre) y que le permite saber la posición y tensión de todos y cada uno de sus músculos, así como rectificar o ratificar el impulso inicial dado a sus miembros al ejecutar un movimiento.

Podemos concluir este capítulo afirmando que una parte importante de todo sistema lo conforma el proceso de retroalimentación, sin el cual el todo tendería inevitablemente al caos.

EL SIGNIFICADO DE LA TEORIA GENERAL DE LOS SISTEMAS

EN POS DE UNA TEORIA GENERAL DE LOS SISTEMAS.

Al repasar la evolución de la ciencia moderna topamos con un fenómeno sorprendente:
Se han presentado problemas y concepciones similares en campos muy distintos, independientemente; por lo cual ha surgido una nueva disciplina llamada Teoría General de los Sistemas. Su tema es la formulación y derivación de aquellos principios que son válidos para los "sistemas" en general. El sentido de esta disciplina puede ser circunscrito como sigue.

Conceptos, modelos y leyes parecidos surgen una y otra vez en campos muy diversos, independientemente y fundándose en hechos del todo distintos. En muchas ocasiones fueron descubiertos principios idénticos, porque quienes trabajan en un territorio no se percataban de que la estructura teórica requerida estaba ya muy adelantada en algún otro campo. La teoría general de los sistemas contará mucho en el afán de evitar esa inútil repetición de esfuerzos.

También aparecen isomorfismos de sistemas en problemas recalcitrantes al análisis cuantitativo pero, con todo, de gran interés intrínseco.

Se diría, entonces, que una teoría general de los sistemas sería un instrumento útil al dar, por una parte, modelos utilizables y trasferibles entre diferentes campos, y evitar, por otra, vagas analogías que a menudo han perjudicado el progreso en dichos campos.

Otro aspecto aún más importante de la teoría general de los sistemas es el problema fundamental de la complejidad organizada. Conceptos como los de organización, totalidad, directividad, teleología y diferenciación son ajenos a la física habitual. De esta manera, un problema fundamental planteado a la ciencia moderna es el de una teoría general de la organización. La teoría general de los sistemas es capaz en principio de dar definiciones exactas de semejantes conceptos y, en casos apropiados, de someterlos a análisis cuantitativo.

Con lo anterior, se ha indicado brevemente el sentido de la teoría general de los sistemas, A continuación se señalarán las objeciones a las que ha sido objeto.

Se ha objetado que la teoría de los sistemas no quiere decir nada más que el hecho trivial de que matemáticas de alguna clase son aplicables a diferentes clases de problemas.

Otra objeción hace hincapié en el peligro de que la teoría general de los sistemas desemboque en analogías sin sentido. Este riesgo existe, en efecto. Así, es una idea difundida considerar el Estado o la nación como organismo en un nivel superordinado.

Una objeción más pretende que la teoría de los sistemas carece de valor explicativo.

METAS DE LA TEORIA GENERAL DE LOS SISTEMAS.

Tales consideraciones se resumen así:
En varias disciplinas de la ciencia moderna han ido surgiendo concepciones y puntos de vista generales semejantes.
En la ciencia contemporánea aparecen actitudes que se ocupan de lo que un tanto vagamente se llama "totalidad", es decir, problemas de organización, fenómenos no descomponibles en acontecimientos locales, interacciones dinámicas manifiestas en la diferencia de conducta de partes aisladas o en una configuración superior, etc. Concepciones y problemas de tal naturaleza han aparecido en todas las ramas de la ciencias.

No sólo se parecen aspectos y puntos de vista generales en diferentes ciencias; con frecuencia hallamos leyes formalmente idénticas o isomorfas en diferentes campos.

Estas consideraciones conducen a proponer una nueva disciplina científica, que llamamos teoría general de los sistemas. Su tema es la formulación de principios válidos para "sistemas" en general, sea cual fuere la naturaleza de sus elementos componentes y las relaciones o "fuerzas" reinantes entre ellos.

De esta suerte, la teoría general de los sistemas es una ciencia general de la "totalidad", concepto tenido hasta hace poco por vago, nebuloso y semimetafísico. En forma elaborada sería una disciplina lógico-matemática, puramente formal en sí misma pero aplicable a las varias ciencias empíricas. Para las ciencias que se ocupan de "todos organizados", tendría significación análoga a la que disfrutó la teoría de la probabilidad para ciencias que se las ven con "acontecimientos aleatorios"; la probabilidad es también una disciplina matemática formal aplicable a campos de lo más diverso, como la termodinámica, la experimentación biológica y médica, la genética, las estadísticas para seguros de vida, etc.

Esto pone de manifiesto las metas principales de la teoría general de los sistemas:

1) Hay una tendencia general hacia la integración en las varias ciencias, naturales y sociales.
2) Tal integración parece girar en torno a una teoría general de los sistemas.
3) Tal teoría pudiera ser un recurso importante para buscar una teoría exacta en los campos no físicos de la ciencia.
4) Al elaborar principios unificadores que corren "verticalmente" por el universo de las ciencias, esta teoría nos acerca a la meta de la unidad de la ciencia.
5) Esto puede conducir a una integración, que hace mucha falta, en la instrucción científica.

El enfoque matemático adoptado en la teoría general de los sistemas no es el único posible ni el más general. Hay otra serie de enfoques modernos afines, tales como la teoría de la información, la cibernética, las teorías de los juegos, la decisión y las redes.

SISTEMAS CERRADOS Y ABIERTOS: LIMITACIONES DE LA FISICA ORDINARIA.

El primer ejemplo será el de los sistemas cerrados y abiertos. La física ordinaria sólo se ocupa de sistemas cerrados, de sistemas que se consideran aislados del medio circundante. Así la fisicoquímica habla de las reacciones, de sus velocidades y de los equilibrios químicos que acaban por establecerse en un recipiente cerrado donde se mezclan cierto número de sustancias reaccionantes.

La termodinámica declara expresamente que sus leyes sólo se aplican a sistemas cerrados. En particular,el segundo principio afirma que en un sistema cerrado cierta magnitud, la entropía, debe aumentar hasta el máximo, y el proceso acabará por detenerse en un estado de equilibrio. O sea, que la tendencia hacia la máxima entropía o la distribución más probable es la tendencia al máximo desorden.

Encontramos sistemas que, por su misma naturaleza y definición, no son sistemas cerrados. Todo organismo viviente es ante todo un sistema abierto. Se mantiene en continua incorporación y eliminación de materia, constituyendo y demoliendo componentes, sin alcanzar, mientras la vida dure, un estado de equilibrio químico y termodinámico, sino manteniéndose en un estado llamado uniforme (steady) que difiere de aquél. Tal es la esencia misma de ese fenómeno fundamental de la vida llamado "metabolismo", los procesos químicos dentro de las células vivas.

No ha sido hasta años recientes cuando hemos presenciado una expansión de la física orientada a la inclusión de sistemas abiertos. Esta teoría ha aclarado muchos fenómenos oscuros en física y biología, y ha conducido asimismo a importantes conclusiones generales, de las cuales sólo se mencionarán dos.

La primera es el principio de equifinalidad. En cualquier sistema cerrado, el estado final está inequívocamente determinado por las condiciones iniciales. Si se alteran las condiciones iniciales o el proceso, el estado final cambiará también. No ocurre lo mismo en los sistemas abiertos. En ellos puede alcanzarse el mismo estado final partiendo de diferentes condiciones iniciales y por diferentes caminos. Es lo que se llama equifinalidad, y tiene significación para los fenómenos de la regulación biológica.

La base de la teoría de los sistemas abiertos, la aparente contradicción entre entropía y evolución desaparece. En todos los procesos irreversibles la entropía debe aumentar. Por tanto, el cambio de entropía en sistemas cerrados es siempre positivo; hay continua destrucción de orden. En los sistemas abiertos, sin embargo, no sólo tenemos producción de entropía debida a procesos irreversibles, sino también entrada de entropía que bien puede ser negativa.

A partir de estos ejemplos es de imaginarse el alcance de la teoría de los sistemas abiertos. Entre otras cosas, muestra que muchas supuestas violaciones de leyes físicas en la naturaleza no existen o, mejor dicho, que no se presentan al generalizar la teoría física. El concepto de sistemas abiertos puede ser aplicado a niveles no físicos. Son ejemplos su uso en ecología, y la evolución hacia la formación de clímax; en psicología, donde los "sistemas neurológicos" se han considerado "estructuras dinámicas abiertas"; en filosofía, donde la tendencia hacia puntos de vista "trans-accionales" opuestos a los "auto-accionales" e "inter-accionales" corresponde de cerca al modelo de sistema abierto. (Bentley).

INFORMACION Y ENTROPIA.

Otra vía que está vinculada de cerca a la teoría de los sistemas es la moderna teoría de la comunicación.

La noción general en teoría de la comunicación es la de información. En muchos casos la corriente de información corresponde a una corriente de energía.

Otra manera de medir la información, a saber: en término de decisiones. Esta medida de la información resulta ser similar a la de la entropía, o más a la de la entropía negativa, puesto que la entropía es definida como logaritmo de la probabilidad. Pero la entropía, como ya sabemos, es una medida del desorden; de ahí que la entropía negativa o información sea una medida del orden o de la organización, ya que la última, en comparación con la distribución al azar, es un estado improbable.

Otro concepto céntrico de la teoría de la comunicación y el control es el de retroalimentación.

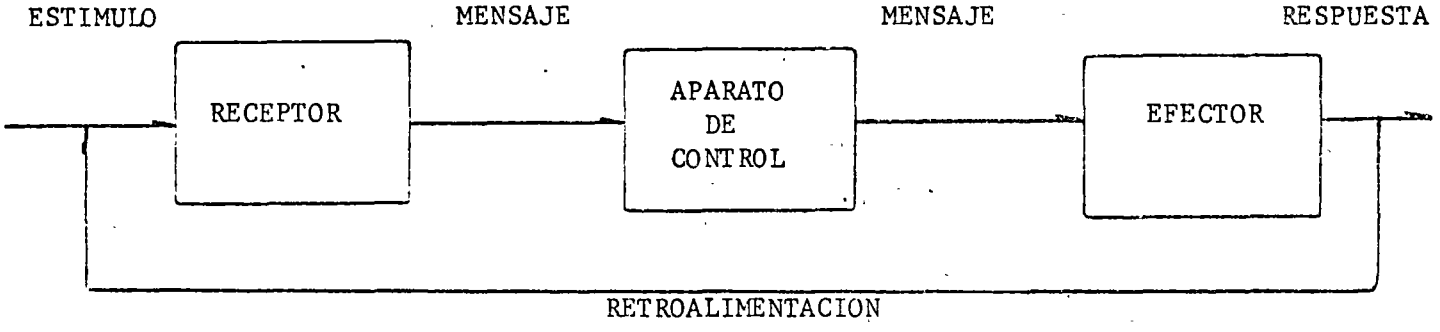


Fig. .1. Esquema sencillo de retroalimentación.

Los dispositivos de retroalimentación se emplean mucho en la tecnología moderna para estabilizar determinada acción, como en los termostatos o los receptores de radio, o la dirección de acciones hacia determinada meta: Las desviaciones se retroalimentan, como información, hasta que se alcanza la meta o el blanco.

Además, en el organismo humano y animal existen sistemas de retroalimentación comparables a los servomecanismos de la tecnología, que se encargan de la regulación de acciones. Si queremos alcanzar un lápiz, se envía al sistema nervioso central un informe acerca de la distancia que nos impidió llegar al lápiz en el primer intento; esta información es retroalimentada al sistema nervioso central para que el movimiento sea controlado hasta que se logre la meta.

Gran variedad de sistemas tecnológicos y de la naturaleza viviente siguen, pues, el esquema de retroalimentación.

Hay que tener presente, sin embargo, que el esquema de retroalimentación es de naturaleza bastante especial. Presupone disposiciones estructurales del tipo mencionado. Pero hay muchas regulaciones en el organismo vivo que tienen la naturaleza del todo distinta, a saber, aquellos en que se alcanza el orden por interacción dinámica de procesos.

CASUALIDAD Y TELEOLOGIA.

Otro punto que se menciona es el cambio en la imagen científica del mundo durante las últimas décadas.

En el punto de vista llamado mecanicista, nacido de la física clásica del siglo XIX, el juego sin concierto de los átomos, regidos por las leyes inexorables de la causalidad, generaba todos los fenómenos del mundo, inanimado, viviente y mental. No quedaba lugar para ninguna direccionalidad, orden o telos.

Las nociones de teleología y directividad parecían caer fuera del alcance de la ciencia y ser escenario de misteriosos agentes subrenaturales o antropomorfos, o bien, tratarse de un seudoproblema, intrínsecamente ajeno a la ciencia, mera proyección mal puesta de la mente del observador en una naturaleza gobernada por leyes sin propósito. Con todo, tales aspectos existen, y no puede concebirse un organismo vivo sin tener en cuenta lo que, variada y bastante vagamente, se llama adaptabilidad, intencionalidad, persecución de metas y cosas semejantes.

Característico del presente punto de vista es que estos aspectos sean tomados en serio, como problemas legítimos para la ciencia; y también estamos en condiciones de procurar modelos que simulen tal comportamiento.

Ya han sido mencionados dos de ellos. Uno es la equifinalidad, la tendencia a un estado final característico a partir de diferentes estados iniciales y por diferentes caminos, fundada en interacción dinámica en un sistema abierto que alcanza un estado uniforme; otro, la retroalimentación, el mantenimiento homeostático de un estado característico o la búsqueda de una meta, basada en cadenas causales circulares y en mecanismos que devuelven información acerca de desviaciones con respecto al estado por mantener o la meta por alcanzar.

El comportamiento teleológico dirigido hacia un estado final o meta característicos no sea algo que esté más allá de los lindes de la ciencia natural, ni una errada concepción antropomorfa de procesos que, en sí mismos, no tienen dirección y son accidentales. Más bien es una forma de comportamiento definible en términos científicos y cuyas condiciones necesarias y mecanismos posibles pueden ser indicados.

¿QUE ES ORGANIZACION?

Consideraciones análogas son aplicables al concepto de organización. Un átomo, un cristal, una molécula, son organizaciones. En biología, los organismos son, por definición, cosas organizadas. Pero aunque dispongamos de una enorme cantidad de datos sobre la organización biológica, de la bioquímica y la citología a la histología y la anatomía; carecemos de una teoría de la organización biológica, de un modelo conceptual que permita explicar los hechos empíricos.

Características de la organización, trátese de un organismo vivo o de una sociedad, son nociones como las de totalidad, crecimiento, diferenciación, orden jerárquico, dominancia, control, competencia, etc.

Hay muchos aspectos de organizaciones que no se prestan con facilidad a interpretación cuantitativa.

Tenemos así que conformarnos con una "explicación en principio", argumentación cualitativa que, con todo, no deja de conducir a consecuencias interesantes.

Como ejemplo de la aplicación de la teoría general de los sistemas a la sociedad humana mencionaremos un libro de Boulding intitulado The Organizational Revolution. Parte de un modelo general de la organización y enuncia las que llama leyes férreas, válidas para cualquier organización. Entre ellas están, por ejemplo: la ley malthusiana de que el incremento de población supera por regla general al de los recursos; está asimismo, la ley de las dimensiones óptimas de las organizaciones: mientras más crece una organización, más se alarga el camino para la comunicación, lo cual -y según la naturaleza de la organización- actúa como factor limitante y no permite a la organización crecer más allá de ciertas dimensiones críticas. La teoría de Volterra, la llamada primera ley de Volterra revela ciclos periódicos en poblaciones de dos especies, una de las cuales se alimenta de la otra. La importante ley del oligopolio afirma que, si hay organizaciones en competencia, la inestabilidad de sus relaciones, y con ello el peligro de fricción y conflictos, aumenta al disminuir el número de dichas organizaciones.

TEORIA GENERAL DE LOS SISTEMAS Y UNIDAD DE LA CIENCIA.

Concluyen las observaciones con unas palabras acerca de las implicaciones generales de la teoría interdisciplinaria.

Quizá pueda resumirse como sigue la función integradora de la teoría general de los sistemas. Hablando según lo que se ha llamado el modo "formal" -es decir, contemplando las construcciones conceptuales de la ciencia-, esto significa uniformidades estructurales en los esquemas que estamos aplicando. En lenguaje "material", significa que el mundo, o sea la totalidad de los acontecimientos observables, exhibe uniformidades estructurales que se manifiestan por rastros isomorfos de orden en los diferentes niveles o ámbitos.

Llegamos con ello a una concepción que, en contraste con el reduccionismo, podemos denominar perspectivismo. El principio unificador es que encontramos organización en todos los niveles.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

CICLO DE VIDA DE UN SISTEMA

AGOSTO, 1978.

Palacio de Minería Calle de Tacuba 5, primer piso. México 1, D. F. Tels: 521-40-23 521-73-35 5123-123

I I

CICLO DE VIDA DE UN SISTEMA

II.1 ANALISIS

El análisis constituye la primera etapa de lo que hemos denominado "ciclio de vida de un sistema" y se enfoca totalmente al intento de seccionar un posible problema en las partes que lo conforman para intentar comprenderlo, así como para desarrollar soluciones de carácter general que fuesen aplicables.

El problema-objeto de análisis puede significarse en un conjunto de procedimientos y métodos administrativos que han sido rebasados por la dinámica del área administrativa en que se inscriben y cuyo nivel de contingencia es muy elevado; eventualmente el problema a resolver (o por analizar) puede significarse por la ausencia total de métodos y procedimientos, o bien por estructuras organizativas inadecuadas, o por inexistencia de organización, o bien por cambios en los requerimientos administrativos externos, o más comúnmente, por una mezcla de todas o algunas de las causas arriba enumeradas (1).

Un análisis efectivo es una etapa crítica, verdaderamente importante para el desarrollo de futuras etapas en el ciclo del sistema, pero frecuentemente se destina a ello un volumen de recursos reducido ya que la distribución de recursos (humanos, materiales, financieros, etc.) tiende hacia labores cuyos resultados sean visibles e inmediatamente mesurables.

Alrededor del objeto de análisis se encuentran siempre las personas que requieren su solución, o que se ven afectados (directa o colateralmente) por su existencia; el conocimiento de ellas, de sus actitudes y sus puntos de vista acerca del problema es requisito a cubrirse a la brevedad posible, sin perder nunca la posición (el rol) que debe cubrir el analista de sistemas frente/junto al grupo usuario.

Convendría dividir, con criterio pedagógico solamente, la etapa de análisis en tres fases o pasos cronológicamente dispuestas, a saber: identificación, definición, y solución del problema; las tareas que implica cada fase y los documentos en que concluyen, constituyen ahora nuestro objeto de estudio.

En todas las etapas del ciclo de vida de un sistema, y particularmente en el análisis, la comunicación formal y disciplinada, así como el prurito de registro sistemático de todo cuanto pudiera servir para el sistema, pudiendo parecer actividades superflúas, resultan premisa indiscutible. El computador es una herramienta fabulosa, compleja, cara, incansable, eficiente, flexible y continuamente atacada. Todo descuido en el nacimiento de un sistema redundará más tarde en fallas que habían de distraer esfuerzos mayores para subsanarlas, que los necesarios para no propiciarlas.

La Idenficación del Problema:

La inevitable pregunta del médico al paciente de ¿qué le pasa mi amigo? siempre tendrá un efecto molesto, por cuanto la respuesta frecuente es "¡no lo sé, por eso vengo a verle!" y suele propiciar reacciones enojosas cuando el diagnóstico resulta en "usted está sano, retorne a sus labores..."; sin embargo, el primer paso que da un analista es precisamente ese.

Cuando una petición es hecha al área de sistemas referente a la solución de un problema se carece por completo -en dicha área- de elementos de juicio para evaluarla; por ende, se requiere de la identificación (generalmente confirmación) de que un problema realmente existe,y la realización de esta fase ofrece señalamientos muy importantes de la esencia del problema revelando al analista conocimientos de las vertientes del problema que al principio permanecen ocultas.

La manera en que un problema surge a la luz es importante: cuando más de una función administrativa está involucrada, a menudo significa que variados intereses y enfoques -no pocas veces en conflicto- inciden en él y representan diferentes niveles técnicos (o habilidades administrativas) que deberán ser acoplados en intento de desarrollar una solución aceptable para todas las partes involucradas.

Otras dimensiones que influyen en esta fase son el cómo y el porqué del problema; el surgimiento pudo ser gradual como cuando se presentan cambios lentos en los objetivos de la administración que dejan obsoletos poco a poco, a los procesos y métodos en uso, o como cuando el personal dedicado a la tarea reacciona con poca prestancia y oportunidad (o desinterés) al detectar incongruencias; en el otro extremo, el súbito surgimiento de un problema se debe generalmente a cambios también súbitos en los requerimientos (que suelen seguir a cambios drásticos en la organización) o bien a que el personal tiene muy poca "consistencia" administrativa y ha efectuado continuas variaciones en métodos y procedimientos que acabaron por desquiciarlas.

Naturalmente son muchas más, y muy variadas, las circunstancias 'periféricas' a la gestación - desarrollo de un problema que pueden 'dispararlo' en un momento, sorpresivamente y la mejor forma de considerarlas (un primer acercamiento) es ponerlas en contexto construyendo arreglos matriciales con los "promotores" del problema -primera dimensión-, esto es las personas y funciones inmensas en el y -segunda dimensión- los puntos de vista acerca del problema que externen los"promotores" (ver figura 3).

Este ejemplo (los datos contenidos en la matriz de la fig. 1) puede parecer exagerado pero en los casos en que los promotores difieren diametralmente en sus apreciaciones no resultan desudados ni mucho menos. Aquí hay una confrontación entre la "comunicación" del problema y la "administración" del problema que impedirá un exitoso diseño de un sistema, sin antes resolver tal situación, generalmente no técnica.

Un arreglo similar al anterior deberá en tales casos construirse con los "promotores" identificando con mayor concreción los orígenes del problema:

No existe un sistema --- porque --- no se ha autorizado ninguno.
no es indispensable.

El sistema es catastrófico -- porque -- hay negligencia de . . . .,
la organización es inestable, etc.

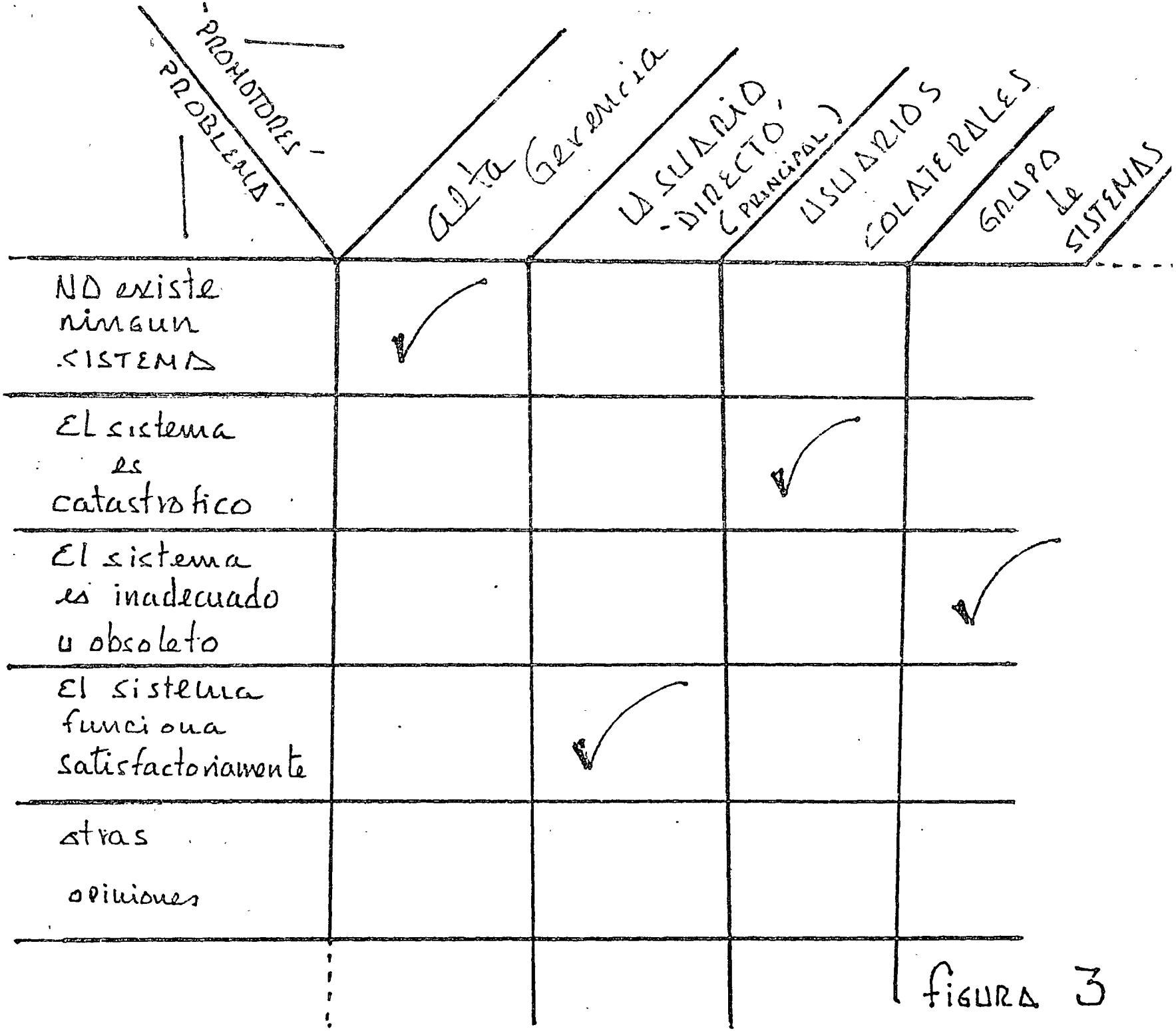

- PROMOTORES -
- PROBLEMAS -
Alta Gerencia
Usuario - Directo (Principal)
Usuarios colaterales
Grupo de sistemas
NO existe ningun SISTEMA
El sistema es catastrofico
El sistema es inadecuado u obsoleto
El sistema funciona satisfactoriamente
otras opiniones

FIGURA 3

Esto podrá mostrar posibles diferencias de opinión gestadas en la consideración de elementos diferentes en el problema, permitiendo salvar obstáculos al nuevo sistema antes de que sea diseñado; el ambiente (en torno) en el que el sistema habrá de existir, deberá ser armónico, incluyendo a los promotores, el personal operativo y los niveles de supervisión que incidan o puedan incidir en él. (2)

La culminación de esta fase estará señalada por la redacción de un documento que podemos denominar simplemente "solicitud fundamentada de un sistema" de la que el futuro usuario es principalmente responsable; el deberá relacionar claramente sus necesidades y justificarlas en términos cuantitativos, sin embargo, el analista asume un rol importante en dicha redacción, por 3 razones:
a) En operaciones corrientes con procedimientos de gran complejidad e interacción, el usuario puede omitir puntos importantes que requieran solventarse.
b) El usuario tiene tan solo un limitado conocimiento acerca de la(s) forma(s) en que el grupo de sistemas puede auxiliarle,
c) porque la mayoría de los sistemas de información pueden necesitar un marco de referencia multi-departamental para que su implantación sea viable.

Una"solicitud fundamentada de un sistema" debe incluir:
a) Propósito de la solicitud:
   Resúmenes de los problemas a resolver y los beneficios a alcanzar.
b) Perfil del sistema actual, indicando las áreas de conflicto:
   b.1) Objetivos.
   b.2) Funciones involucradas.
   b.3) Organigramas, plantilla de personal, distribución (Layouts) de áreas.
   b.4) Operaciones - descripción semi-detallada de los procedimientos en uso complementada con ejemplos de formas utilizadas, extimaciones de volúmenes y frecuencias, etc., así como un somero análisis de los costos de operación actuales.
c) Perfil del sistema futuro (solicitado) incluyendo donde sea posible los incisos del punto anterior, con enfoque específico sobre posibles cambios respecto al sistema presente.

La Definición del Problema:

Esta segunda fase del análisis responde a una pregunta más compleja; ya hemos encontrado en la fase anterior que si existe un problema y se intenta saber cual es el problema, su magnitud, sus causas, etc.

Desde luego, tras establecer quienes se ven envueltos en el problema que actitudes tienen ante él, tras descubrir variadas facetas y circunstancias que colateralmente conforma o acentúan el problema, debe necesariamente tenerse ya un amplio panorama y una razonada opinión acerca de lo que camina y lo que no camina. Sin embargo, en esta fase deberá el analista pasar de lo que fue una primera aproximación -superficial por necesidad- al estudio meticuloso y documentado.

La primera tarea dentro de esta fase se inicia con la recopilación de toda la información disponible acerca de las operaciones corrientes del sistema vigente (podemos llamar sistema a cualesquiera procedimiento en uso, por rupestre que fuese) y desde todas las fuentes posibles.

Esto involucra necesariamente juicios subjetivos acerca de la calidad y veracidad de lo recopilado; aún así, el resultado de esta labor ofrecerá un reflejo de la realidad cada vez más preciso, y no exactamente coincidente con el reflejo primario que ofrecía la fase de identificación y tampoco igual a la concepción que tienen del problema sus mismos actores considerados individualmente.

La recopilación deberá observar dos facetas: los métodos y procedimientos en uso, y el volumen de información que realmente se maneja, ambas deben confluir en un estudio del flujo de información que consiste en un detallado registro de toda la información, del camino por el que fluye de una función a otra, incluyendo aquellos intercambios de información que, siendo relevantes, no se plasman en documento alguno, por ejemplo las consultas telefónicas (y sus repeticiones en cadena) que se efectúan previamente a la toma de decisión.

Un correcto estudio del flujo de información describe sin ambiguedades toda forma documental o pieza de papel utilizada, que contiene; quien recaba los datos, cuando se llena, a donde va, cuando es archivada, destruida o actualizada; también registra toda comunicación verbal que incida en los procedimientos, y (como ya se apuntó) estimaciones periódicas (anuales, mensuales, semanales, etc...) de las cantidades en que se presentan las formas; punto muy importante es el señalamiento de "picos" en los volúmenes que afectan los procedimientos y que por lo regular se relacionan cronológicamente.

Físicamente, el estudio suele estar plasmado en matrices cuya dimensión horizontal relaciona a las funciones (o puestos) que participan en los procedimientos, y en la otra se numeran los eventos que se describen secuencialmente; igualmente suelen presentarse unidas por flechas las actividades anotadas en el cuerpo del estudio, en la medida en que unas deben o pueden preceder a otras. Siendo lo menos importante su forma de presentación, estos estudios del flujo de información suelen ser extensos, con muchos ejemplos y diagramas explicativos y altamente detallados.

Este trabajo se efectúa en estrecha colaboración con personal de nivel operativo del área que tiene el problema quienes deberán ser inducidos por el analista a integrar un verdadero equipo de trabajo; evidentemente el estudio será suplementado y ratificado mediante entrevistas directas con el nivel ejecutivo, pero ellas no pueden sustituirlo.

En nuestro medio, éste ha sido un vicio generalizado que se fundamenta por un lado en el celo directivo del usuario por ser él precisamente la fuente primaria -cuando no única- de información para el analista, considerándose como el más directamente conocedor del problema a todos los niveles, y por otro en la tendencia del propio analista a dar todo por comprendido, tras el deseo de iniciar cuanto antes etapas avanzadas en el ciclo de vida del sistema.

Al concluirse el estudio, la clara comprensión de las actividades que se realizan y cual información fluye en realidad, el analista se encuentra en posición de determinar si la intervención del recurso 'computador' es factible, innecesaria, indispensable, conveniente, etc.

Sin embargo, la interpretación del estudio suele llevar a diferentes conclusiones al grupo de sistemas y a los usuarios; una y otra vez los usuarios parecen requerir tan sólo que el computador "cubra" las actividades de desarrollo tedioso, repetitivo y voluminoso; una y otra vez ante sugerencias directas de cambios en la organización, en métodos y procedimientos, así como en la construcción de "profesiogramas" para la selección de personal, la respuesta es poco positiva sobre todo cuando las sugerencias afectan áreas fuera de la responsabilidad concreta del analista o del grupo de sistemas.

En tales casos, el estudio del flujo de información acompañado de una bien estructurada presentación gráfica y discursiva parece ser la llave de avenimiento; basados en el principio de "puedes mostrarselo pero no debes decirselo" el analista llega a inducir cambios que faculten un superior funcionamiento del proceso de diseño y del propio sistema, a futuro.

Gran número de fallas en los sistemas de información debidas a una deficiente organización son atribuidas -a posteriori- al analista que diseñó tamaña tontería; un buen recurso para inducir cambios es el contar con aliados a todos los niveles, ganados por el convencimiento y que antes de oponerse a ellos, seguramente habrán de promoverlos.

Recapitulando esta fase puede decirse que el analista, en tanto miembro del grupo de apoyo denominado aquí "gente de sistemas", se adentra en el problema y lo define usando métodos tales como:

A) Investigando la evolución del sistema vigente. ¿Cómo y por qué se realizan las actividades y funciones actualmente? ¿Cómo y por qué se hacían antes, cuando parecía no haber problema? ¿Qué ha cambiado?
B) Diagramando las actividades que conforman el sistema vigente.
C) Confirmando los objetivos frente al sistema vigente: entrevistas a nivel ejecutivo y a nivel operativo, revisión de documentación existente.
D) Evaluando la premisa de "el hombre adecuado, en el puesto adecuado", involucrando: calidad y cantidad de personal, procedimientos y equipo.

Hemos visto como el instrumento principal de esta fase es el estudio del flujo de información, sin embargo, no es éste el documento que lo concluye, sino que su terminación implica la elaboración de otro que aquí llamaremos "memoranda de comprensión".

En él comparten responsabilidades el analista y el usuario aún cuando la parte de construcción y elaboración corresponden al primero, el usuario habrá de destinar cuando menos una persona de tiempo completo, para la preparación del memoranda, y deberá documentar su aprobación y apoyo; finalmente el "memoranda de comprensión" debe incluir:

a) El resumen del sistema vigente.
b) Una relación de los objetivos de un nuevo sistema (o de modificaciones al actual) lo más breve posible.
c) Un inventario de sistemas alternativos que pudieran resultar en la solución del problema.
d) Una relación de recursos humanos que deban asignarse a la fase de solución del problema.
e) La aprobación del usuario, y del nivel ejecutivo del grupo de sistemas.

La Solución del Problema:

Es la tercera y última fase del análisis y es en ella que pueden iniciarse las actividades verdaderamente creativas, fundiendo los resultados de tareas precedentes podemos arribar a la síntesis de la etapa.

Ahora pasana primer plano las habilidades y experiencia del analista: dispone ya de ilustrativas conclusiones que sitúan al problema, dibujan su contorno e iluminan su contenido; todos los elementos de juicio están bajo su dominio y debe ahora pasar a la ofensiva, manejar sus recursos con habilidad, elegir -de entre varias- sus armas para este desafío.

Las fases previas han arrojado luz y con ella, se prefiguran ideas acerca de las soluciones alternativas de algunos procedimientos, sin embargo, ninguna aceveración habrá de hacerse a la ligera, sin un concienzudo esfuerzo autocrítico de revisión, dentro del contexto del proyecto. completo.

Si el sistema propuesto resulta de gran tamaño y complejidad, mucha gente laborará en él y toda actividad tendiente a asegurar una amplia comunicación entre el analista, el diseñador, los programadores y los usuarios será poca, dada la necesidad de que cada pieza no sólo tenga una adecuada conjunción (interface) con el resto, sino que el todo (sistema) conserve su armonía: un buen líder dentro del grupo de trabajo es esencial por esto.

La solución que se presenta bajo la forma de una propuesta de sistema contendrá recomendaciones alternativas y la indispensable argumentación que decidió la elección.

Este"diseño preliminar" (también llamado "especificaciones preliminares") del sistema, se complementa con relaciones de costos, recursos y tiempos estimados para todo el proyecto. Resulta sugestivo el que una presentación formal se lleve a cabo con la presencia de todas las partes involucradas en torno la problema,dando lugar a que los posibles impugnadores comprometan al analista a satisfacer dudas y objeciones, cruzando preguntas (y respuestas) antes de la aprobación del sistema propuesto: muy pocas veces la intervención de un "abogado del diablo" debe ser tan bien recibida, ni puede resultar de tanta utilidad.

Los documentos que sustentan esta fase son básicamente dos:
a) las alternativas del sistema
b) el sistema propuesto

En orden cronológico, se presentan como un solo documento, sin embargo, deben ser discutidos tal como fueron mencionados; el analista (y en general, la gente de sistemas) tiene ahora la oportunidad de formular y analizar varias posibilidades de solución que, ilustradas con costos y tiempos,deben ser presentados al usuario quien será el "gran elector" de la más conveniente, en términos de ofrecer al analista su punto de vista; también puede darse el caso de que la función de contraloría o control presupuestal aporte elementos de juicio, pero será en última instancia la "gente de sistemas" la que defina, entre las alternativas, cuál tiene mayor viabilidad operativa: Cuidado, los mecanismos de aprobación varían de institución en institució y no son a ellos a los que nos referimos, de hecho, caen fuera de nuestros objetivos en el curso; son los mecanismos de elección en cuanto a la relación de costos-beneficios, y las disponibilidades técnicas lo que nos interesa, y nada más.

Las alternativas más frecuentes tienen su punto de referencia en uno, varios o todos de los que a continuación se enlistan:

a) El sistema actual (vigente).
b) El sistema actual modificado substancialmente.
c) Un nuevo sistema destinado a cubrir sólo la capacidad mínima expresada en la "solicitud de un sistema".
d) Un nuevo sistema, más flexible, que ofrezca aproximaciones (necesariamente vagas) al desarrollo de otras funciones no mencionadas en la "solicitud del sistema".
e) Utilización de "paquetes" desarrollados externamente a la institución y que pueden adquirirse (y adaptarse) con facilidad.

Para todas las alternativas posibles debe presentarse un perfil que detalle:

a) Diagramas del sistema (entradas, salidas y procesos).
b) Formatos de entradas y salidas.
c) Contenido de los archivos y origen de la información (sobre todo en la utilización de bases de datos).
d) Descripción de los pasos (procesos) principales.
e) Estimaciones de volumenes (proceso y almacenamiento).
f) Estimaciones de tiempos y costos (de desarrollo y de operación).
g) Estimaciones de beneficios (económicos e intangibles).
h) Análisis de riesgos inherentes a las estimaciones anteriores.
i) Relaciones con (y efectos sobre) otros sistemas.

Finalmente, la propuesta de sistema debe ser desarrollada y presentada por el grupo de sistemas, ofreciendo una solución calificada como "óptima" a las necesidades del usuario, y definiendo en detalle la capacidad de la propuesta y las características operativas de la misma.

Será este documento el que gobierne las etapas posteriores de diseño, programación e implementación del sistema, y constituye la prueba final de que el usuario ha comunicado al analistas todas sus necesidades claramente, y este, comprendiéndolas, las compendia en su propuesta de solución (propuesta de sistema), que contendrá:

a) Resumen del problema.
b) Breve descripción del sistema vigente, señalando sus inadecuaciones.
c) Descripción del sistema propuesto.
   c.1) Objetivos específicos.
   c.2) Alcances (cambios en la organización, en las funciones invlucradas, en otros sistemas).
   c.3) Logística
        a) Diagrama general.
        b) Fuentes (y formatos de entrada) de la información.
        c) Configuración de los archivos.
        d) Funciones del proceso.
        e) Formatos de salidas.
        f) Tiempos de respuesta.
        g) Volúmenes
        h) Requisitos de control.
        i) Requerimientos de equipo.

d) Resumen del plan de desarrollo:
   d.1) Descripción de tareas mayores y sus puntos de control (mediante métodos de control de proyectos, por ejemplo la ruta crítica).
   d.2) Requerimientos de recursos.
   d.3) Distribución de responsabilidades.
   d.4) Costos.

II.2 DISEÑO

Como segunda etapa del ciclo de vida de un sistema de información, el diseño recoge los productos finales del análisis y fundado en ellos "traduce" los lineamientos que de ahí emanan en postulados de carácter técnico que sirvan de guía a las etapas posteriores, sin perder nunca el contexto determinado por el sistema aprobado en la "propuesta".

Diseño es pues, la colección de actividades necesarias para conducir (ampliando y adaptado los postulados de la propuesta) la gestación del sistema hasta el punto en que las instrucciones de cada programa pueden ser codificadas para el computador (4).

Para fines de este curso, y en obvia aclaración de nuestra afirmación anterior, explicitaremos que en adelante y desde ahora nos estamos refiriendo al diseño-programación-etc... de sistemas de información cimentados en el uso de un computador.

Los objetivos del diseño pueden dividirse en tres grandes grupos:
a) Inmediatos
b) A corto plazo
c) A largo plazo

Los objetivos inmediatos son aquellos que se persiguen antes de iniciar propiamente el esfuerzo de diseño, tales como la prefiguración de los canales de retroalimentación que conducen a consultar al usuario, los puntos y formas de (chequeo) validación de esfuerzo, la asingación REAL (no proyectada y/o aprobada) de recursos, etc.

Objetivos de corto plazo son aquellos a lograrse mediante la aplicación eficaz y eficiente de los recuros de programación (no la programación en si) tales como la modulairdad del diseño acorde al número de programadores, la complejidad de los módulos de acuerdo al talento de los mismos, la elección de técnicas de almacenamiento y de programación, la difusión de estandares, la planeación de interfaces para el rastreo (DEBUG) de cadenas de programas, etc.

Objetivos de largo plazo son los referidos a la planeación del posterior crecimiento y mantenimiento del sistema ya en actividad, tales como las holguras en capacidad de almacenamiento y en el diseño de registros lógico-físicos, la modularidad del sistema referida a la modularidad de las funciones que abarca, la definición de mecanismos de control del proceso, etc.

La etapa de diseño de un sistema de información está técnicamente orientada a responder la pregunta de ¿cómo hacerlo? una vez que la etapa de análisis nos ha dicho que hay que hacer; recurrentemente el diseñador utilizará metodológicamente las preguntas ¿Qué pasaría en caso de - - -? o bien ¿Por qué no intentar- - -?

Debemos recordar que un axioma del diseño de sistemas es que "la excepción se hace regla" y por ende el diseño debe observar las posibles contingencias del sistema.

Las fases en las que podemos subdividir esta etapa serían las siguientes:

Fase a) Diseño General del Sistema, hasta su aprobación.

Fase b) Diseño Particular o Detallado del Sistema.

En la primera fase, el objetivo subyacente es el desarrollar un modelo conceptual mediante "acercamientos" sucesivos, esto es, partiendo de lo general a lo particular. Un primer intento de modelo consiste básicamente en un diagrama que identifica los datos de entrada, describe vagamente (y brevemente) los procesos internos y enumeran las salidas (ver figura 4).

A partir de tan exiguo principio, y por desgloses cada vez más amplios debe llegarse a un diagrama general del sistema cuyos anexos mostrarán los subsistemas que lo componen, y en ellos, las cadenas de programas necesarios así como los archivos y/o bases de datos con los que trabajan, la información fuente que captan y los reportes que producen, etc.

Aún cuando el mayor detalle corresponde a la segunda fase, algunas cosas deberán quedar claras desde ésta, tales como:

- Identificación del producto principal del sistema (de acuerdo a los objetivos señalados en la propuesta) enumerando los datos que incluye y de donde proceden y que tratamiento sufren.
- Identificación de los productos secundarios.
- Identificación de los recursos de almacenamiento (bases de datos, etc.)
- Identificación de las cantidades susceptibles de brindar cifras de control.

En adición a lo anterior podemos agregar algunos principios básicos de diseño aplicables a esta fase (5):

- Los datos deben ser capturados una y sólo una vez por el sistema, sin importar sus múltiples procesos y consultas.

- La claridad y exactitud de los documentos fuente (pre-codificados de ser posible) determinará el número y la complejidad de los pasos necesarios para hacerlos parte del sistema.

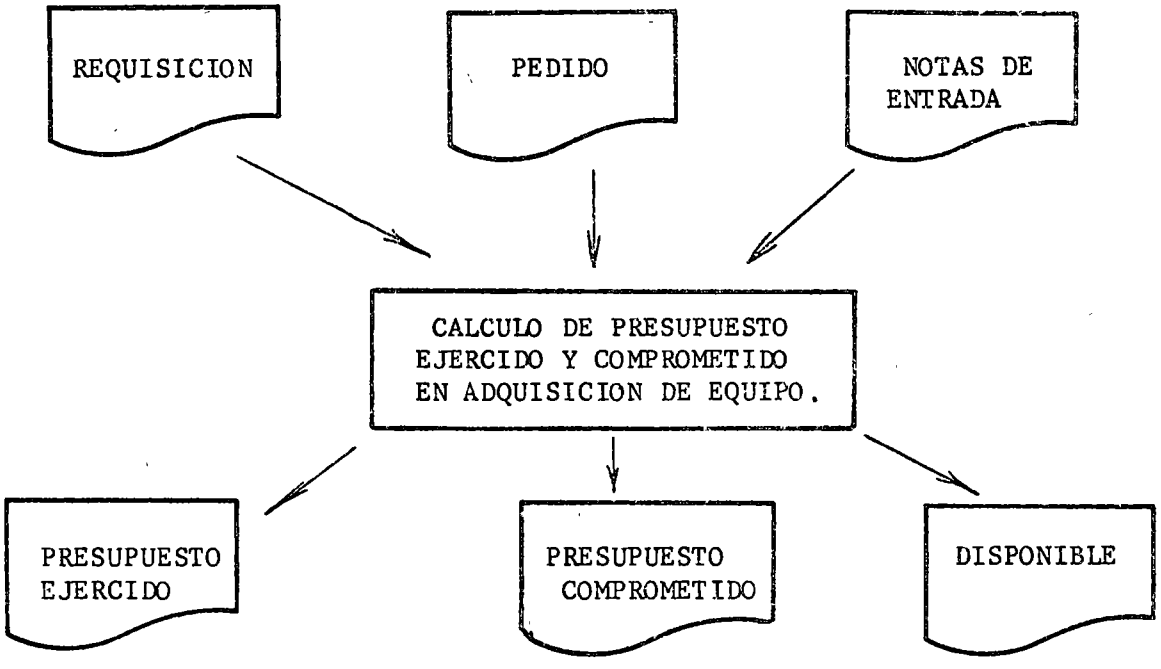
- La frecuencia de actualización debe ser al menos igual a la frecuencia de consulta.

- Usualmente sólo se justifica la captura directa e inmediata (on-line) de datos, si existe también consulta directa e inmediata; en general es poco recomendable dado que la validación de la información que entra al sistema debe efectuarse de inmediato, lo que dificulta el proceso de corrección; evidentemente los procesos de validación-corrección no deberán duplicarse.

- Las cifras de control serán actualizadas y validadas antes y después de toda actualización y/o resguardo (Backup) de manera automática (lógica de programa) previamente a su validación manual.

- Los datos serán almacenados en un sólo lugar en la base de datos (o archivos tradicionales) a excepción de aquellos que funjan como "apuntadores" y "llaves".

FIGURA 4


REQUISICION
PEDIDO
NOTAS DE ENTRADA
CALCULO DE PRESUPUESTO EJERCIDO Y COMPROMETIDO EN ADQUISICION DE EQUIPO.
PRESUPUESTO EJERCIDO
PRESUPUESTO COMPROMETIDO
DISPONIBLE

- Todo archivo deberá ser susceptible de impresión formateada y editada para fines de auditoría.

- Los archivos temporales (de transacciones por ejemplo) deberán ser retenidas cuando menos en un ciclo, si actualizan a los maestros o a la base datos.

- Todo archivo contará con procedimientos de resguardo, y de ser "no secuencial", contará con procedimientos de reorganización.

- En lo posible, las características y formatos de registros "parámetros" deberán ser semejantes.

La aprobación del diseño general, que culmina la primera fase, adquiere tan relevante importancia que algunos autores la consideran como una fase intermedia de la etapa de diseño (6).

Resumidamente, precede a tal aprobación, la comparación punto por punto de la "propuesta" del sistema contra el "diseño general" y dicho procedimiento se hace reiterativo en la medida en que la comparación arroje discrepancias, lo que implica una revisión autocrítica tanto de la propuesta como del diseño general; la participación del usuario en la aprobación resulta indispensable.

La fase de diseño detallado del sistema corresponde virtualmente a la construcción de las especificaciones de cada programa, lo que viene a ser el puente entre la idea general del diseñador, y el equipo de programadores; sin embargo, la fase también observa otros puntos de interés como diseño de formas y reportes (al detalle), establecimiento de controles de producción-distribución, etc.; aún así, las especificaciones de programas aportan el mayor peso específico en el diseño detallado del sistema, aportando algunos beneficios importantes como el ofrecer al diseñador una última oportunidad de revisar la lógica del sistema como un todo, proporcionar una comunicación expedita y documentada con los programadores, etc.

Una completa especificación de un programa debe incluir:

a) Hoja de control del programa:
Detallando el nombre y código del programa, el nombre y código del sistema (y subsistema) al que pertenece, nombres del diseñador y del programador, estimación de tiempos, fecha, etc. Configura efectivamente la "presentación" de las especificaciones y puede incluir el lenguaje en que será programado.

b) Diagrama de "bloque" del programa:
Mostrando los archivos de entrada y/o salida, con unabreve descripción del propósito del programa y de las rutinas que deba efectuar (asi mismo, los archivos deberán identificarse con nombre y código).

c) Descripción de archivos:
Incluyendo características acerca de los datos que contiene (tamaño, nombre, formato, etc.), de su longitud, sus registros, su organización, su método de acceso, su bloqueaje, etc.

d) Tablas de decisión:
Relacionando códigos (y combinaciones de ellos) que implican decisiones acerca de los procesos a efectuar; generalmente el diseñador prefiere utilizar un método narrativo para exponer el proceso lógico que se asigna al programa.

e) Tablas de referencia:
Relacionando códigos (argumentos) con las más variadas funciones (cantidades, leyendas, otros códigos, etc.) y que serán usados por el programa; deberá adjuntarse la declaración de si dichas tablas serán adicionadas al programa, o serán asimiladas por él, al ejecutarse (tablas externas).

f) Parámetros y opciones:
Son referencias que se harán al programa en el momento de ejecución, para limitar su funcionamiento a determinados códigos, o informarlo acerca de cuales reportes se desean generar, etc., esto es, para gobernar su proceso.

g) Datos de prueba:
Información artificial (creada por el diseñador) que deberá alimentarse al programa en el momento de probar su funcionamiento.

De la clara y concisa información que reciba el programador mediante las especificaciones de los programas, dependerá gradualmente su rendimiento, y el de todo el equipo; gran parte de las especificaciones conformarán la documentación del sistema, pero esto será tratadocon detalle en el capítulo correspondiente; por ahora estamos listos para encarar la siguiente etapa en el cilco de vida de un sistema.

II.3 PROGRAMACION

La escritura de programas para el computador es normalmente la más grande actividad individual en el desarrollo de un sistema, ya esta actividad puede definirse como la preparación y codificación de instrucciones para ser ejecutados por el computador.

La tareas inherentes a la etapa de programación pueden identificarse como sigue. (7):

1) Identificación de los propósitos del programa y con ella, la primera aproximación lógica a las rutinas que el computador debe efectuar y a los límites (alcances) del programa. Esto implica una revisión de las especificaciones.

2) Definición de la secuencia lógica en que los datos de entrada deberán ser procesadas y la disposición (también secuenciada lógicamente) de las rutinas; variadas técnicas están disponibles al respecto (diagramas de lógica, tablas de decisiones, etc..)

3) Traducción del planteamiento lógico del programa a códigos ejecutables por el procesador especial llamado compilador o ensamblador, mediante la codificación de instrucciones en un lenguaje de programación; es este el momento de reanalizar la decisión tomada por el diseñador (si la hay) acerca del lenguaje a utilizar.

4) Ensamblar o compilar las instrucciones mediante el uso de procesadores (software) específicos, a fin de "depurar" los posibles errores de sintaxis y/o de lógica cometidas al codificar; el proceso de compilación "traduce" las instrucciones codificadas en un lenguaje de programación, "intermedio", a un lenguaje de "máquina" comprensibles para el computador.

5) Prueba del programa; esta tarea implica suministrar al programa datos de entrada "artificiales" para observar su ejecución, comfrontando paso por paso las especificaciones (sobre todo las tablas de decisiones) previas, con los resultados de la ejecución. Los datos de prueba debieron ser diseñados con el obvio propósito de abarcar todos y cada uno de los objetivos y circunstancias posibles a enfrentar por el programa; al proceso de "rastreo" y corrección de errores en esta etapa se le denomina comúnmente con el vocablo Inglés de "DEBUG" que parece no tener traducción exacta al español.

6) Codificación de instrucciones para el control de la ejecución del programa en un lenguaje accesible a otro procesador (lenguaje de control de trabajo). Prácticamente el programador prepara instrucciones de este tipo en los momentos de compilación y prueba de programas, sin embargo es en el momento de ejecución real cuando estas instrucciones adquieren relevancia.

7) Descripción de actividades a ser realizadas por el personal que maneje operativamente el programa (ejemplo: el operador de la computadora).

Los lenguajes de programación más frecuentemente usados son, en ese orden, los que se enlistan a continuación:

a) COBOL: Orientado principalmente a la resolución de aplicaciones "mundanas" (comerciales, de negocios).

b) RPG: Orientado a la manufactura de reportes, a partir de información previamente procesada en otro lenguaje; debe su extendida utilización a los escasos requerimientos de Hardware que exige.

c) FORTRAN: Orientado principalmente a la resolución de problemas científicos y de una alta complejidad técnica.

d) ENSAMBLADOR: Orientado principalmente a la resolución de problemas del Software del propio equipo de computación (ejemplo: rutinas para el manejo de canales hacia una unidad de discos o cintas magnéticas).

Con el surgimiento de conceptos de proceso tales como el "tiempo real", "Tiempo compartido", "bases de datos", etc..., que ponen a disposición del usuario su información, en todo momento, de manera que pueden "comunicarse" con el computador directamente mediante un teletipo, o una pantalla de video, han surgido y vienen desarrollándose a grandes pasos lenguajes llamados "conversacionales" como el CMS (Conversational Monitor System) cuya principal características es la interacción usuario-computador, mediante un grupo muy general (y aún muy limitado) de instrucciones de gran potencia y manejo elemental.

Ahora bien: conforme la innovación tecnológica proporciona más flexibles herramientas para la programación y el "debugging" de programas han hecho su aparición técnicas más sofisticadas en esta fase; algunas de ellas acabamos de mencionarlas, como la programación conversacional y la programación interactiva, pero nos interesa hacer incapié en una denominada "programación estructurada" que tiene su base de sustentación en la afirmación de que la codificación de un programa debe ser comprensible no tanto para el computador o el procesador que la maneja (compilador) sino para el(los) programador(es) que deba(n) revisarla y, en su caso, modificarla.

Posiblemente el impulsor más destacado de esta técnica sea Harlan D. Mills (8) que ha realizado numerosas investigaciones al respecto.

La programación estructurada consiste básicamente en observar al programa como un todo (sistema) compuesto de módulos o rutinas de proceso (subsistemas) cuya ejecución está determinada por un módulo principal cuya jerarquía lógica en la codificación va cediendo el control de la ejecución a los distintos módulos secundarios; la más difundida característica de esta técnica es quizás la remoción de instrucciones de transferencia incondicional (GO TO) sustituidos por instrucciones de transferencia condicionada hacia rutinas "cerradas" (Perform, Do, Until...) sin embargo, podemos afirmar que tal característica no es ni con mucho esencial.

Cuando es posible, suele utilizarse aunada a la programación estructurada, la técnica de "segmentación" de programas que consiste en la definición de algunos de los módulos secundarios que componen el cuerpo del programa, como "segmentos transientes", o sea, como susceptibles de ser retirados de la memoria principal del computador hacia memorias auxiliares (unidades de discos, tambores, etc.) en tanto no sean requeridos para su ejecución.

Desde luego estas técnicas no pueden ser aplicadas correctamente en todos los lenguajes de programación, por ejemplo: la estructura del RPG impide que el ciclo de LECTURA - PROCESO - ESCRITURA sea roto en la práctica y su codificación basada en indicadores (swithes) invalida la metodología de la programación estructurada.

No está de más mencionar aquí algunas otras técnicas que elevan la productividad del programador, aún cuando las señalemos con brevedad:

Estandares de programación.-
Permiten, como su nombre lo indica, homogenizar la codificación de todos los programadores de la instalación, imponiendo métodos de trabajo comunes de probada eficiencia; ejemplos:
- Uso de definiciones de archivos comunes, generalmente catalogados en una 'biblioteca' (residente en un disco) de acceso generalizado, lo que obliga a utilizar siempre idénticos nombres para los mismos datos.
- Uso de prefijos (en los nombres) adecuados a las funciones y no al arbitrio de los programadores; siempre es confuso (aunque gracioso) encontrar datos referentes al total de rentas, denominados "MARIA".
- Uso de rutinas comunes catalogadas: todavía es frecuente el encontrar muchas rutinas diferentes para calcular el mismo dígito verificador de las cuentas de cheques.

Utilización generalizada de bibliotecas para programas "fuente" y para códigos "ejecutables"; la noción de que un programa es un conjunto de tarjetas resulta ya obsoleta y ha sido siempre insegura.

Utilización de "pruebas de escritorio" sobre el diagrama de lógica antes de iniciar la codificación del programa y sobre la codificación misma antes del período de prueba.

Finalizaremos esta etapa con la afirmación de que el proceso de programación de computadoras está más cercano a la creatividad que a aplicación de métodos esquemáticos; desde luego, cualesquiera persona puede,en un lapso relativamente corto, aprender la forma y sintaxis de un lenguaje, pero habrá de requerir cierta dosis de talento (y del preciado 'sentido común') para desarrollar correctamente un diagrama de lógica.

II.4 DOCUMENTACION

La documentación es un método de comunicación, y se refiere a un registro escrito de una fase o fases de cierto proyecto. Describe cierto pasos de un sistema, y establece criterios de diseño y de actuación que habrá que satisfacer en otras etapas futuras del proyecto. Esto permite obtener un mejor control del proyecto.

Significativamente, la documentación tiene por objeto:

1) Aminorar la deformación o la ambiguedad con respecto a los elementos incluidos en varias fases de un proyecto actual.

2) Evitar la pérdida de información clave, en caso de que el miembro del personal auxiliar encargado del proyecto decida abandonar la organización.

3) Valorar los progresos hechos en un proyecto y dar la oportunidad de estudiar algunas inconsistencias entre las fechas planeadas para la ejecución de las metas, y las fechas reales, lo que constituye una fuente de referencia para la modificación del sistema de una organización, o para esos datos históricos para el desarrollo de trabajos semejantes al que ya se ha llevado a cabo.

4) La comunicación entre los especialistas del procesamiento de datos y los no especialistas, por ejemplo, los usuarios. A menudo hay que darles instrucciones sobre la aplicación apropiada de sus sistemas. De ese modo se mantienen buenas relaciones entre el personal del procesamiento electrónico de datos y los usuarios.

5) Por lo tanto, son indispensables los proyectos debidamente documentados, para que un sistema eficiente pueda actualizarse y funcionar como una importante fuente de referencia para el desarrollo de los sistemas futuros.

CATEGORIA DE DOCUMENTACION.

El objetivo de dividir la documentación en categorías e proporcionar la documentación necesaria para soportar la operación y mantenimiento del Sistema. La documentación se divide en cuatro tipos o categorías:

1.- DOCUMENTACION GENERAL DEL SISTEMA.

La documentación general de un sistema generalmente contendrá la información de los siguientes 6 puntos:

a) Objetivos del sistema.
b) Límites o limitaciones del sistema.
c) Descripción de los principales módulos del sistema.
d) Diagramas de flujo del sistema.
e) Estructura, descripción de elementos e interrelacion de archivos para la base de datos.
f) Descripción de resultados o salidas del sistema.

2.- DOCUMENTACION DE PROGRAMAS.

Cuatro áreas principales deben documentarse para los programas:

a) Diagramas de bloque.
b) Instrucciones de proceso (codificación, etc.)
c) Procedimientos especiales.
d) Definición de las entradas.
e) Definición de las salidas.

3.- DOCUMENTACION DE OPERACION.

Para la operación deben documentarse los siguientes aspectos:

a) Diagramas de flujo de información.
b) Instrucciones del proceso antes de la computadora.
c) Instrucciones para el proceso en la computadora.
d) Instrucciones de proceso después de la computadora.
e) Procedimientos especiales.

4.- DOCUMENTACION DEL USUARIO.

La categoría de usuario deben documentarse los siguientes siete puntos:

a) Descripción general del sistema.
b) Forma que deben prepararse los datos.
c) Procedimientos para corregir errores.
d) Procedimientos de control de calidad para la información de los archivos.
e) Procedimientos especiales.
f) Cuadro de responsabilidades.
g) Glosario de términos.

TECNICAS DE PRESENTACION.

La documentación del sistema debe presentarse de tal manera que permita:
1) Minimizar el tiempo requerido para su comprensión.
2) Aumentar la velocidad de su preparación.
3) Que su presentación sea breve y concisa.

Para lograr lo anterior, la documentación suele presentarse en alguna de las tres formas siguientes:
1) Narrativa (Fig. II.4.1)
2) Diagramas (Fig. II.4.2)
3) Formas preimpresas (Fig. II.4.3)

ENTREVISTA SOBRE EL PROCEDIMIENTO

POR:_____
FECHA:_____

ASUNTO: Hoja de proceso de manufactura - Forma 387

PROPOSITO: Detallar la información respecto al método de fabricación de cada una de las partes.

LUGAR EN DONDE SE ORIGINA: Departamento de herramientas y métodos.

FUENTE DE INFORMACION: Datos sobre procedimientos y herramientas.

INFORMACION ANOTADA: Número de operación, operación, departamento, máquina, alimentación, velocidades, corte y herramientas según la sección que efectúa el proceso. Los números de herramientas y datos sobre las necesidades de material, suministrados por la sección de herramientas. Las horas-hombre y horas-máquina, proporcionadas por la sección de normas de tiempos.

NUMERO DE COPIAS: Cinco o más, según se requiera.

FRECUENCIA: En cada parte pasada por ingeniería y revisada según se requiera.

DISTRIBUCION:

Copia 1 Conservada por la sección de procesado en el expediente de partes, como un registro histórico.

Copia 2 Conservada por la sección de procesado en el expediente de partes, como una ayuda al preparar revisiones. Revisado el procesado, la copia 2 se destruirá.

Copias 3 y 4 A control de producción para fines de planeación. Se destruirán cuando no se necesiten más.

Copia 5 y todas las que se requieran. A control de producción cuando éste las solicite, para ser distribuidas a los departamentos involucrados, inclusive contabilidad de costos, de acuerdo con la fecha efectiva y serán destruidas cuando sobresean.

FIGURA II.4.1

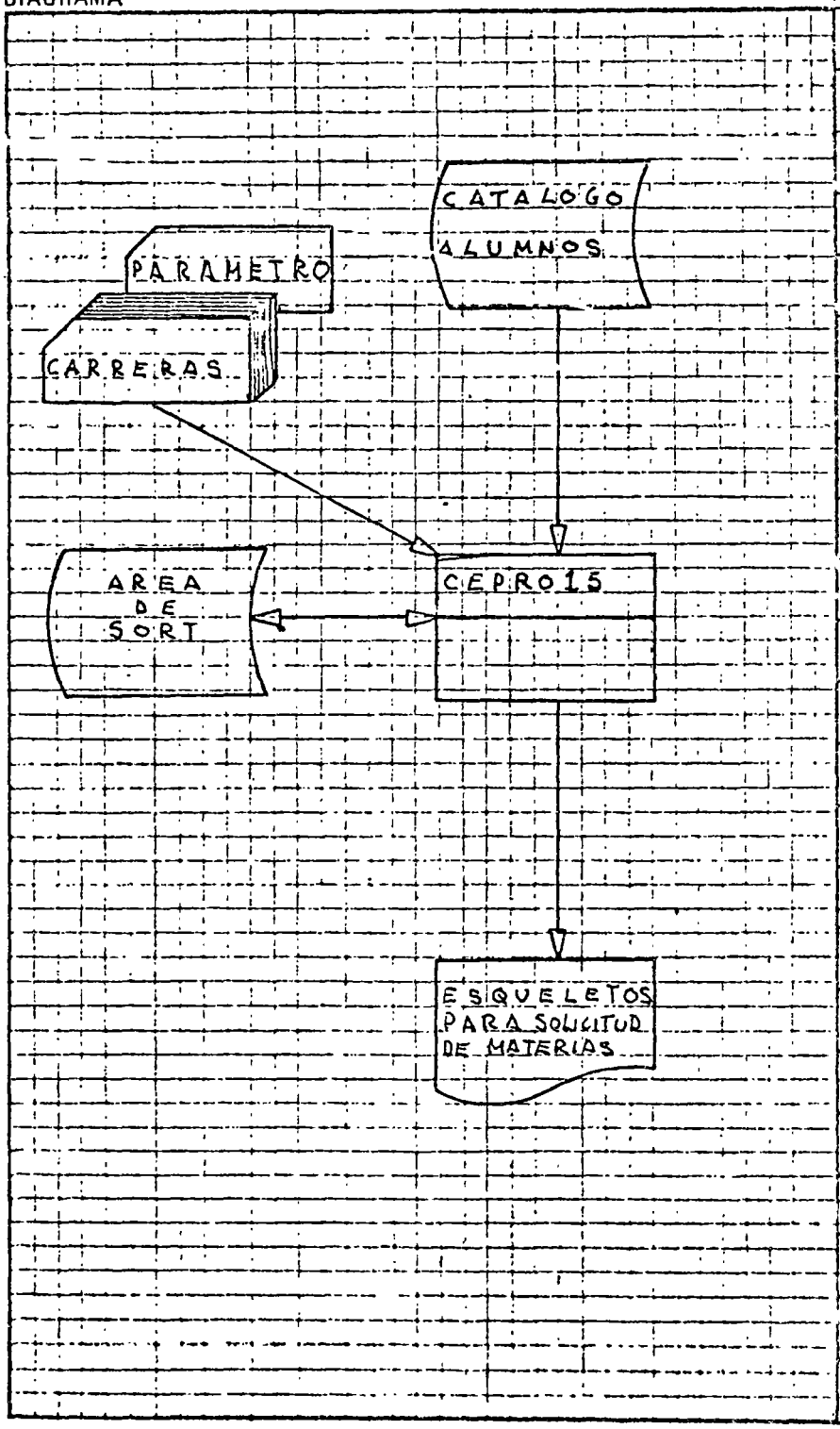
REQUERIMIENTOS DEL PROGRAMA

HOJA NO._____

Sistema _____ Identificación _____

Programa _____ Identificación _____

DIAGRAMA



PASO No.	DESCRIPCION
1	ACEPTA TARJETAS DE CARRERAS
	SE HACE UN SORT POR CARRERA
	Y NUMERO DE CUENTA, EN INPUT
	PROCEDURE SELECCIONA A
	LOS ALUMNOS QUE SEA DE
	LICENCIATURA Y QUE ESTE
	INSCRITO PARA SORTEARLOS,
	LA SORTEADOS ACEPTA LA
	TARJETA PARAMETRO QUE
	CONTIENE EL TIPO DE PERIODO
	PARA ENCABEZADOS. A CONTINUA-
	CION SE IMPRIMEN LOS
	ESQUELETOS DE SOLICITUD
	DE MATERIAS.

FIGURA II.4.2

Elaborado por _____ Revisó _____ Fecha _____

CARACTERISTICAS DEL ARCHIVO

Sistema _____ Identificación _____

Archivo _____ Identificación _____

ORGANIZACION: 1 Secuencial 2 Secuencial Indexado 3 Random ☐ TIPO DE ARCHIVO: 1 Temporal 2 Permanente ☐

Características de retención _____ Volumenes. Inicial _____ Crecimiento estimado _____

LLAVE Campo _____ Tamaño _____ Tipo _____

Factor de Block _____ Long. de Registro _____ Etiqueta _____

Densidad _____ Modo de grabación _____ Frecuencia de utilización _____

OBSERVACIONES _____

NO. CAMPO	DESCRIPCION DEL CAMPO	NOMBRE DEL CAMPO	POSICION DE	A	LONG.	TIPO	OBSERVACIONES

II.5 INSTALACION DEL SISTEMA

La instalación del sistema se refiere primordialmente a las consideraciones de "SOFTWARE" del sistema y debe llevarse a cabo con posterioridad a la prueba del sistema.

Es difícil de precisar el punto inicial de la etapa de instalación del sistema, principalmente porque algunos de los programas que lo componen, no están completamente operacionales y documentados, cuando empieza la instalación del sistema; así mismo no es fácil determinar cuando finaliza la instalación y se inicia la operación.

Para la instalación del sistema deben de realizarse normalmente algunas actividades tales como:

- Entrenamiento del usuario:

El entrenamiento de las personas que toman parte en las operaciones diarias con el sistema, es de vital importancia para facilitar y hacer bien el trabajo.

Algunos tópicos a tratar serían:

1) A personal nivel oficinista:
   - Colección de la información.
   - Sistemas de codificación.
   - Preparación de los documentos.
   - Operación de terminales.
   - Corrección de archivos.
   - Cifras de control.
   - Otros.

2) A personal de un nivel medio o administrativo:
   - Sistemas de codificación.
   - Formato y contenido de los reportes.
   - Necesidades de retención de información.
   - Posibilidades de almacenamiento de datos.
   - Establecimiento de standares.
   - Otros.

- Chequeo del sistema.

Prueba de cada programa del sistema, etc.

- Conversión del sistema:

La conversión del sistema es una actividad sumamente delicada y debe de realizarse con el mayor esfuerzo posible para tratar de obtener una conversión correcta.

El analista, junto con el usuario, deben determinar lo siguiente:

a) Método de conversión:
   - Conversión en paralelo.
   - Conversión inmediata.
   - Conversión gradual.
   - Conversión piloto.

El analista debe identificar lo siguiente:

b) Controles necesarios en la conversión:
   - Puntos de chequeo en la operación.-
     Puntos que permite verificar el proceso para ver la validez de los datos.
   - Controles contables.-
     Se refiere a ciertas cifras de control sobre campos que en conjunto se usan para verificar el manejo de datos.
   - Procedimientos de reiniciación.

c) Programa de trabajo para la conversión:
   - Identificación de los archivos a convertir.
   - Tiempo esperado de conversión.
   - Verificación del plan de conversión.-
     Checar cargas de trabajo.

- Documentación Final del Sistema:

Se debe incorporar a la documentación preliminar del sistema, la documentación de prueba y conversión del mismo para ir formando la documentación total de él. Se deberá actualizar la documentación del sistema y terminar los manuales de operación.

- Transferencia del Sistema a Control de Producción:

Durante el período de instalación del sistema, bajo control hasta ahora de analistas y programadores, debe pasar al grupo de operación. El éxito de la transferencia depende de las instrucciones adecuadas a producción. Efectivos manuales de operación.

- Orientación Gerencial:

Una vez que los gerentes han sido debidamente informados sobre las técnicas de proceso de datos, se le debe instruir sobre el sistema que va a tener a su cargo como usual y aclara concretamente la participación de su departamento en el sistema.

La ejecución adecuada de esas actividades allana muchos de los problemas de instalación.

PROBLEMAS MAS COMUNES EN EL PERIODO DE INSTALACION.

- Falta de conocimiento en el manejo de datos.-
  Para las personas ajenas a la planeación del sistema.
- Falta de planeación de la instalación del sistema.
- Conocimiento parcial del sistema.
- Sistemas parcialmente depurados presentados como listos para su operación.
- Mal entrenamiento al personal encargado de preparar la entrada al sistema.
- Aspectos psicológicos.-
  a) Resistencia al cambio.
  b) Temor al computador.
  c) Miedo a perder "control".
- Falta de instrucciones de operación o instrucciones incorrectas.

II.6 OPERACION DEL SISTEMA

La etapa de operación del sistema, es aquella en la cual el sistema instalado opera totalmente bajo control de producción. En esta etapa, por lo tanto, el sistema ha pasado totalmente de la fase de desarrollo a su fase de operación y será necesario evaluarlo para comprobar sus resultados.

En esta etapa, normalmente el control total del sistema se transfiere del grupo de sistemas, al grupo de operación.

La operación del sistema impone algunos requisitos de interés para el grupo de análisis y programación, tales como:

1.- Conformación del sistema a "standards" de instalación y operación.-

Durante la primera etapa de operación del sistema el analista debe supervisar que la operación se lleve a cabo tal y como la planeó, o sea que vigilará que desde el punto de vista operación se obtenga la calidad esperada, o sea que se reemplace los standares establecidos. Por ejemplo: que se obtenga cierta velocidad de las perforistas, o que el tanto por ciento de errores sea normal.

2.- Utilidad del uso de manuales de operación.-

El analista debe vigilar que los manuales de operación sean realmente útiles y estén en un lenguaje claro, tanto para el personal de proceso de datos como para otro tipo de personal, también involucrado en el sistema.
Aunque los operadores descubren eficiencia en los manuales difícilmente toman la iniciativa de corregirlos.

3.- Evaluación de eficiencia de las corridas.-

El analista debe revisar y evaluar la eficiencia de las corridas del computador para determinar los errores y su causa. Por ejemplo:
Se detecta que con frecuencia montan un archivo por otro y el analista descubre que la causa es el sistema empleado para identificar esos archivos, ya que nomenclaturas tales como "ARCHIVO HRTH1" contra "ARCHIVO HRHT1" la nomenclatura empleada es confusa.

4.- Mantenimiento de programas.-

No importa que también hayan sido desarrollados y probados los programas, la operación normal revela la necesidad de hacer cambios, ahora bien, si la documentación de los programas es correcta el problema de modificarlos es sencillo. La forma correcta de dar mantenimiento a los programas es la siguiente:

a) Hacer un cuidadoso análisis del problema.
b) Planear y realizar el cambio.
c) Documentar el cambio.
d) Registrar el cambio en la bitácora de mantenimiento.
e) Checar que el cambio opera bien.
f) Terminar con la documentación del cambio.

5.- Obtención de mejoras en la operación.-

Generalmente los programadores o analistas pueden hacer mejoras en la operación del sistema, ya que son el enlace entre personal de operación y el usuario. Después de observar la operación se pueden descubrir caminos que mejoran la eficiencia aunque debe esperarse cierta resistencia al cambio, por parte de los operadores.

REVISION DEL SISTEMA.

Después de que un sistema ha sido operado por un tiempo, es una buena política el revisar y evaluar su ejecución. Esta revisión deberá revelar el grado en el que el sistema esta cumpliendo sus objetivos.

La revisión deberá dirigirse principalmente a los siguientes puntos:

1.- Satisfacción del usuario con las salidas del sistema?

El analista deberá investigar directamente con el usuario la calidad de los resultados recibidos para lo cual podrá preguntar lo siguiente:
a) Oportunidad con la que recibe los reportes.
b) Exactitud de los datos.
c) Si contiene toda la información que requiere.
d) Si la información entra atrasada.
e) Si le gustaría que las cosas siguieran igual o algo debe modificarse.

2.- Eficiencia con el actual sistema respecto a métodos de entrada y colección de datos?

El analista deberá evaluar el trabajo de aquellas personas que recaban información o preparan documentos fuentes a fin de determinar si la forma en que lo hacen es eficiente, el nivel del personal es el adecuado y el tiempo en que se realiza es el correcto. En muchas ocasiones observando el método el analista podrá sugerir cambios.

3.- Contribución directa del sistema a los objetivos de la organización?

Este punto es muy subjetivo, o sea difícil de evaluar, sin embargo, el analista podrá investigar si se cumplen los objetivos establecidos, por ejemplo:
Si hay ahorro en costos, si se mejora la eficiencia de las personas, o si permitió tomar decisiones a los ejecutivos.

4.- Contribución indirecta del sistema a los objetivos de la organización?

Este punto es aún más subjetivo, sin embargo, se podrá investigar que es lo que ocurre en aquellos departamentos ajenos al sistema.

5.- Descontento con el presente sistema?

Como resultado de los puntos 3 y 4 se puede llegar a determinar que el sistema no cumple con los objetivos y en ese caso el analista junto con el usuario deben tomar la decisión de descontinuar el sistema, o bien, contribuir con el peso introduciendo modificaciones en este último caso lo que procede hacer es dar mantenimiento al sistema.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

AUDITORIA TECNICO-ADMINISTRATIVA DE SISTEMAS DE INFORMACION: EVALUACION DE RESULTADOS

AGOSTO, 1978.

Palacio de Minería Calle de Tacuba 5, primer piso. México 1, D. F. Tels: 521-40-23 521-73-35 5123-123

I I I

AUDITORIA TECNICO-ECONOMICA DE SISTEMAS DE INFORMACION

Como resultado de la etapa de obsolescencia del ciclo de vida de un sistema, surge la necesidad de establecer periódicas auditorías de los sistemas en operación con el fin de determinar el grado de eficacia y eficiencia que tienen y -en su caso- declarar un posible rediseño del sistema; tales auditorías contemplan dos vertientes del sistema, a saber:

a) Evaluación técnica del proceso del sistema.
b) Evaluación de los resultados del proceso del sistema

En la primera de ellas, el enfoque está determinado por el deseo de constatar y cuantificar el uso adecuado de los recursos de cómputo destinados al sistema, en tanto que en la segunda, la finalidad estriba en constatar y cuantificar el uso adecuado de los productos del sistema.

En otras palabras, la auditoría del proceso está referida al grupo de sistemas, en tanto que la auditoría de los resultados del proceso está referida a los grupos usuarios del sistema.

De hecho, algunos puntos sobresalientes que corresponderían a este capítulo, han sido mencionados en el capítulo anterior; habremos ahora de profundizar un poco en su descripción.

EVALUACION DEL PROCESO DEL SISTEMA.

El método más utilizado para esta fase consisten en establecer una comparación entre la "Documentación del Sistema", esto es el estado real del sistema (actual), y el "Diseño Particular del Sistema" tal como fue aprobado; esto brinda, desde luego, un basto panorama del grado de obsolescencia del sistema determinado por el número de cambios efectuados al dar mantenimiento al sistema, que deben haber sido registrados en la "documentación", cada vez que se efectuaron; tales cambios implicaron en su momento, modificaciones a los archivos (o adiciones a la base de datos) o modificaciones a los formatos de reportes, las consecuentes modificaciones a los programas, o modificaciones a los documentos fuente, o una combinación de ellos, o todos ellos.

Si los cambios de fondo se perciben en alteraciones a los objetivos del sistema mencionados en la etapa de diseño, cobrará mayor importancia la auditoría de los resultados del proceso; si los cambios parecen indicar adecuaciones del "como hacerlo", entonces la auditoría del proceso es la más importante.

Básicamente una auditoría del proceso deba abarcar:

- Que porcentaje de las "holguras" en los archivos se ha utilizado (pudiendo llegar a ser más del 100%: ampliación del archivo) o bien
- Que nuevas partidas de información (en términos de ocupación de áreas) se adicionaron a la base de datos;

- Que porcentaje de la información inicial (captada por el sistema) es información "muerta", esto es, sin uso actual;
- Que porcentaje de reportes ha sido removido (o sustituído), cambiado y adicionado al sistema;
- Que porcentaje de programas han sufrido cambios, así como
- Que porcentaje de programas han desaparecido del sistema y cual ha sido el adicionado;
- Que porcentaje representa el crecimiento de los tiempos de proceso y/o de respuesta, respecto a los delineados en la prueba del sistema, y
- En que porcentaje ha variado el volumen de información total almacenada.

La cuantificación de algunos de estos puntos es bastante subjetiva, sobretodo al estimar si las modificaciones a los programas han sido complejas o no; sin embargo este problema puede solventarse al incluir ponderadores al respecto en los estandares de documentación, utilizables al momento de realizar los cambios y registrarlos.

EVALUACION DE LOS RESULTADOS DEL PROCESO.

A diferencia de la fase anterior, cuyos beneficios recaen en el grupo de operación y se reducen -cuando más- al grupo de sistemas en general, los beneficios de esta segunda fase interesan fundamentalmente a la alta gerencia de la institución, a los usuarios del sistema y -significativamente- a la dirección del grupo de sistemas.

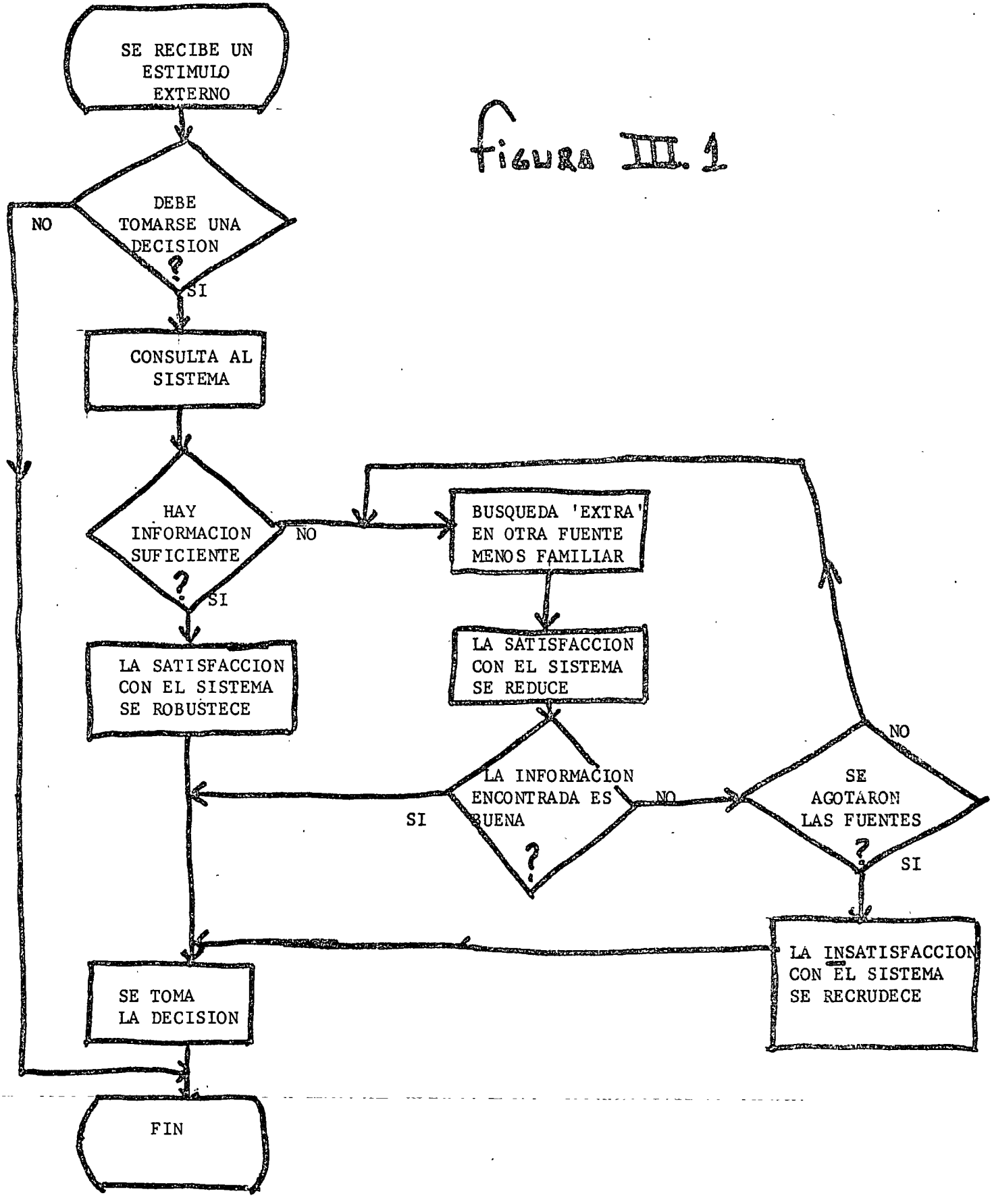
En esta fase quedan inmersos los objetivos mismos del sistema, y no sólo la eficiencia del sistema internamente; recientemente se han desarrollado métodos de evaluación para esta fase derivados de la concepción "conductista" (behavioral) de la psicología, utilizando la categoría de "satisfacción" para medir los resultados del proceso; la frase que parece definir este enfoque es: "búsquedas extras de información causan insatisfacciones al ejecutivo" (1) y revelan una escasa contribución del sistema de información, en la toma de decisiones; para ilustrar esta aceveración, véase la figura III.1

De hecho, un sistema de información es cotidianamente evaluado por los grupos usuarios al tomar decisiones; en tanto más frecuentemente se ve obligado un usuario a consultar otras fuentes de información para resolver problemas, una mayor frustración se genera en él; un alto grado de frustración conduce al usuario a una expresión de su insatisfacción con el sistema que se refleja a menudo en la paulatina pérdida de confianza en el sistema y en los sistemas de información, generalizando.

Operativamente, un sistema de información puede calificarse como efectivo en la medida en que requiera relativamente pocas "búsquedas extras" de información por parte de los grupos usuarios.

Dos conceptos son relevantes al aplicar la definición de "satisfacción" del usuario:

Figura III.1


SE RECIBE UN ESTIMULO EXTERNO
DEBE TOMARSE UNA DECISION ?
NO
SI
CONSULTA AL SISTEMA
HAY INFORMACION SUFICIENTE ?
NO
SI
BUSQUEDA 'EXTRA' EN OTRA FUENTE MENOS FAMILIAR
LA SATISFACCION CON EL SISTEMA SE ROBUSTECE
LA SATISFACCION CON EL SISTEMA SE REDUCE
LA INFORMACION ENCONTRADA ES BUENA ?
SI
NO
SE AGOTARON LAS FUENTES ?
NO
SI
LA INSATISFACCION CON EL SISTEMA SE RECRUDECE
SE TOMA LA DECISION
FIN

Primero, un sistema de información formal es diseñado a proveer eficientemente, la información requerida por un conjunto (préviamente) dado de problemas; para problemas muy específicos -principalmente tomas de decisión en la alta gerencia- relativamente importantes, pero poco frecuentes sería poco eficiente diseñar un sistema de información formal; en tales casos un subsistema AD HOC deberá construirse para cada problema, si se justificase.

Segundo, un sistema de información implica siempre un alcance pre-definido en los objetivos del mismo; las omisiones de información (dentro del universo colectado) en los reportes deberá siempre determinarse vía la experiencia acumulada por el usuario respecto a su relativa intrascendencia en el momento de diseñar el sistema.

Ahora bien: ¿Cómo medir la relación satisfacción-insatisfacción de los grupos usuarios respecto al sistema?; diversas técnicas se encuentran disponibles, algunas muy sofisticadas, pero en todas ellas es premisa fundamental la aplicación de cuestionarios a los grupos usuarios.

En la formulación y aplicación de tales cuestionarios deben dijarse explícitas las categorías que se desea medir, esto es, las dimensiones para las cuales el sistema arroja insatisfacción tales como el nivel de detalle en los reportes, el grado de actualización en ellos, su formato, su distribución, etc. También habrá de clasificarse las funciones administrativas a los que prioritariamente se dirige la información del sistema, por lo que los cuestionarios deberán invlucrar a tales usuarios antes que a otros.

Finalmente, el estudio de los cuestionarios una vez aplicados, deberá dejar claro cual es la parte (componente) del sistema que causa insatisfacción, y hacía que áreas.

Resulta obvio señalar que todas estas metas deberán estar presentes al momento de construir las preguntas del cuestionario (2).

Los resultados de esta fase de la auditoría son desde luego, para uso del grupo de sistemas en primera instancia; sin embargo, es conveniente enterar de ellos tanto a los usuarios como a la alta gerencia quienes serán factores importantes a considerar para -eventualmente- rediseñar el sistema de información auditado.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

TECNICAS DE LOS SISTEMAS DE INFORMACION

AGOSTO, 1978.

Palacio de Minería Calle de Tacuba 5, primer piso. México 1, D. F. Tels: 521-40-23 521-73-35 5123-123

IV. TECNICAS DE LOS SITEMAS DE INFORMACION.

IV.1

Sistema en Batch y Sistema en Línea.

Los sistemas de información varían de tamaño de acuerdo a la velocidad de aplicaciones que tengan y dependen en menor grado con el tamaño de la organización, para la cual esten diseñados. La tabla 1 resume las características de cuatro diferentes centros de cómputo. Como se ve en los centros de menor tamaño no es necesario tener el sistema en línea dado que se tienen pocas aplicaciones y practicamente el sistema de cómputo está dedicado a uno o dos usuarios. Sin embargo en un centro grande existen muchos usuarios y la interacción tiene que ser lo más rápido posible, por esto se crean sistemas en línea que pueden ser consultados por varias decenas de usuarios practicamente al mismo tiempo. Esto no podría lograrse si cada uno de ellos tuviera que perforar tarjetas, llevarlas a un recepcionista y esperar a que se cargaran y ejecutaran en el sistema, y todavía más, tendría que esperar a que se imprimieran los resultados y se le entregaran. Mientras que en el sistema en línea cada usuario, desde su propia terminal cargaría o leería su información en un tiempo muy corto.

TABLA 1.

INSTALACION.	CHICA	MEDIANA	GRANDE	GIGANTE.
# de personas	1-10	5-25	50-100	100-1000
Equipo	360/20 a Sistema 3	360/40 a 370/145	360/50 a 370/158	varios procesadores. 360/65 a 370/168.
Ambiente	Propósitos Especiales.	Uso general, Servicios Administratativos.	Uso general, Telecomunicación, cálculos sofisticados.	Uso general Muchas telecomunicaciones, cálculos sofisticados.
Costo de operación en Dólares.	60,000	100,000	300,000	1,500,000

IV.2 ORGANIZACION Y CONTROL DE LA FUNCION DE ADQUISICION DE DATOS

Tanto desde el punto de vista de los usuarios de una instalación de computo, como desde el punto de vista de quienes trabajan directamente en el procesamiento de datos, es de extrema importancia el aspecto relacionado con la adquisición de la información a ser procesada. En innumerables ocasiones se ha presentado el caso de que programas de aplicación y sistemas de procesamiento de datos presenten un correcto diseño, una programación impecable y suficiente documentación, pero la obtención de resultados contiene errores y produce conclusiones equivocadas y a veces peligrosas para la institución o empresa, por el simple hecho de que la adquisición de los datos fue equivocada o tuvo cierto margen de incertidumbre o de errores, en su captación, o en su transcripción al medio legible por la computadora. Por tales motivos y dada la extrema importancia de la función de adquisición de datos, se discute en las presentes notas, de una manera amplia. Con el fin de referirse directamente a una fuente que aboque del tema extensamente, se incluye en el apéndice C una copia de un artículo tomado de "The Information Systems Handbook", citado en la bibliografía.

IV.3

GENERALIDADES SOBRE BASES DE DATOS

Introducción

En los últimos años,la cantidad de datos que se ha visto obligada a manejar una empresa,se ha incrementado en grandes proporciones debido a distintos factores.

El incremento de datos no controlado dentro de una empresa provoca situaciones caóticas en ella.Suele suceder que un mismo archivo de datos se encuentra en 2 o 3 departamentos distintos y el contenido de esos archivos no concuerda en muchos aspectos.

Debido a esto,se ha visto la necesidad de contar con algún medio donde centralizar esos datos.

Este medio deberá de estar al alcanze de cualquier tipo de usuario y además debera de proveer datos concisos, exactos y reales.También debe de ser un medio de fácil acceso.

Las Bases de Datos son esos medios de almacenamiento que cumplen con las condiciones antes mencionadas.

Una Base de Datos es una colección de datos operacionales almacenados y listos para ser usados por sistemas de aplicación de alguna empresa en particular.

Por empresa en particular podemos entender todas

aquellas entidades que necesitan de datos para su funcionamiento.. Por ejemplo: La Universidad.

Por sistemas de aplicación entendemos todos aquellos sistemas que útilizan esos datos para el logro de sus objetivos.Por ejemplo: el control de estudiantes en la UNAM

Datos operacionales son aquellos datos que son vitales para el funcionamiento de una empresa.

Generalmente se identifican por identidades,donde cada identidad tendra varios atributos.Por ejemplo:

estudiantes: nombre,no.de cuenta.....

cursos: nombre,horas.....

Una característica importante en las identidades de una Base de Datos,es que se pueden establecer relaciónes entre ellas.

Todas estas relaciónes son bidireccionales y sus elementos pueden estar relaciónados en tres formas distintas:

- de uno a uno. estudiantes,nos. de cuentas
- de uno a muchos. calificaciónes por curso,estudiantes
- de muchos a muchos. estudiantes y cursos

Las Bases de Datos nacen de la necesidad de centralización de las identidades.Algunas ventajas de esta centralización son:

-la cantidad de redundancia en los datos almacenados puede ser reducida.

-los problemas de consistencia en los datos almacenados

son eliminados.

-los datos pueden ser compartidos.

-medidas de seguridad sobre los datos pueden ser aplicadas.

Un objetivo de cualquier Base de Datos debe ser la independencia de sus datos con respecto a las distintas aplicaciones que hacen uso de ellos.

Independencia de Datos.-Cuando los sistemas de aplicación hacen uso de los datos sin tener conocimiento de la forma como están organizados, decimos que los datos son independientes.

Si los sistemas de aplicación necesitan conocer la estructura de los datos para hacer uso de ellos o un sistema de aplicación influyo en la organización de estos, decimos que esos datos son dependientes de la aplicación.

Las ventajas de la independencia de datos son obvias. Una Base de Datos puede cambiar su estructura por completo y no afectar en lo más mínimo a ningún sistema de aplicación que haga uso de esa base.

Configuración de un Sistema de Base de Datos.-El siguiente diagrama da una muestra de lo que podría ser una configuración de un sistema. Esta configuración es idealizada.

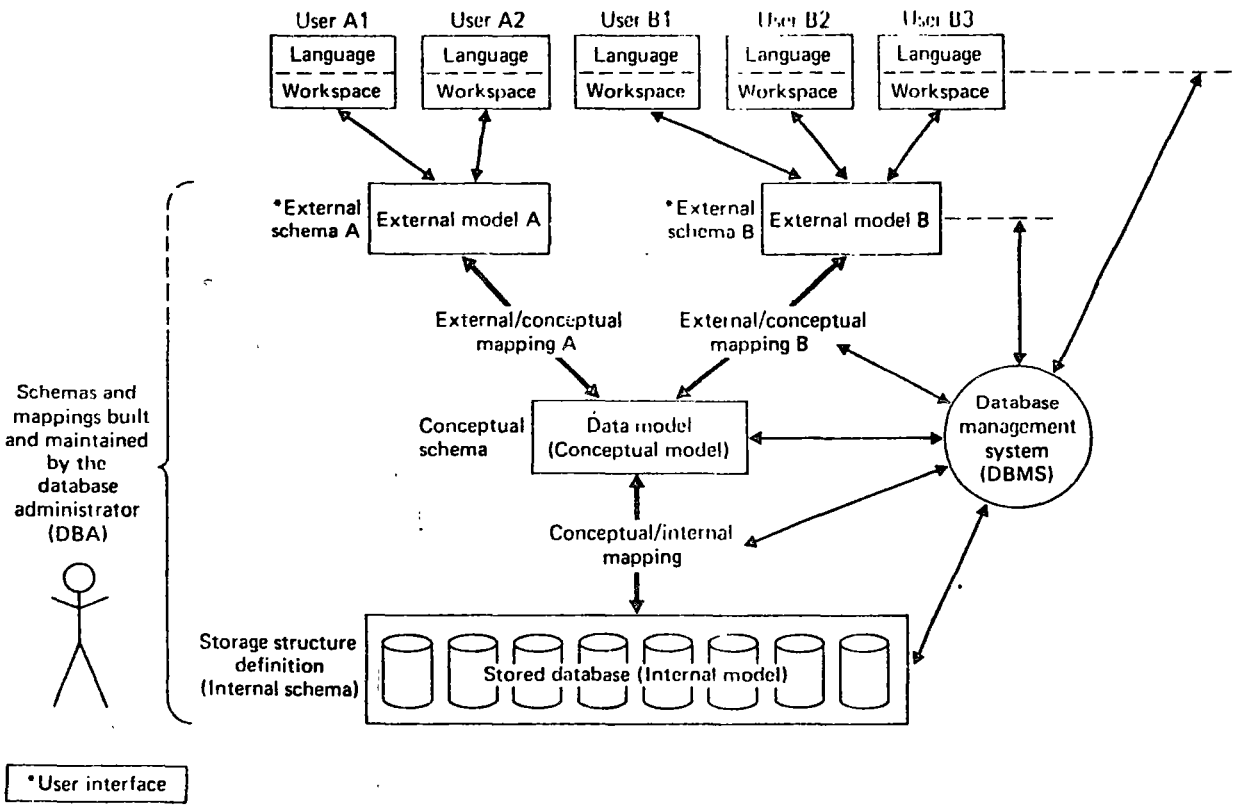


Fig. 1.3 An architecture for a database system

Algunas observaciones sobre el diagrama:

-usuarios pueden ser tanto programadores como operadores de terminales remotas

-área de trabajo es el área donde se reciben y transmiten todos los datos entre el usuario y la base.

-el modelo externo es la información como es vista por un usuario en particular.

-el esquema externo define cada uno de los distintos registros que se encuentra en el modelo externo.

-el modelo conceptual es una representación del total de información contenida en la base.

-el esquema conceptual define cada uno de los distintos registros que se encuentran en el modelo conceptual.

-el modelo interno son todos aquellos registros distintos que se encuentran almacenados físicamente.

-el esquema interno define estos registros.

-el sistema de administración de la Base de Datos es el software que maneja todos los accesos a la base.

-el administrador de la base es la persona encargada de su funcionamiento y de el contenido y estructura de la base.

Para establecer la comunicación entre el usuario y la Base de Datos existen lenguajes asociados a las distintas estructuras de el modelo conceptual.Estos lenguajes se conocen como sub-lenguajes de la base.(DSL).

Existen también lenguajes para dar de alta una Base de Datos:

-Existen lenguajes para definir los esquemas y modelos conceptuales: DDL

-Existen lenguajes para definir los esquemas y modelos externos: Sublenguajes de DDL.

Los siguientes capítulos hablaran de:

-la organización física de los datos.

-la organización de los datos en los modelos conceptual-modelo de datos- y el modelo externo.

II.- ORGANIZACION FISICA

Imaginemos que nos encontramos dentro de una empresa o una institución, digamos la Universidad.

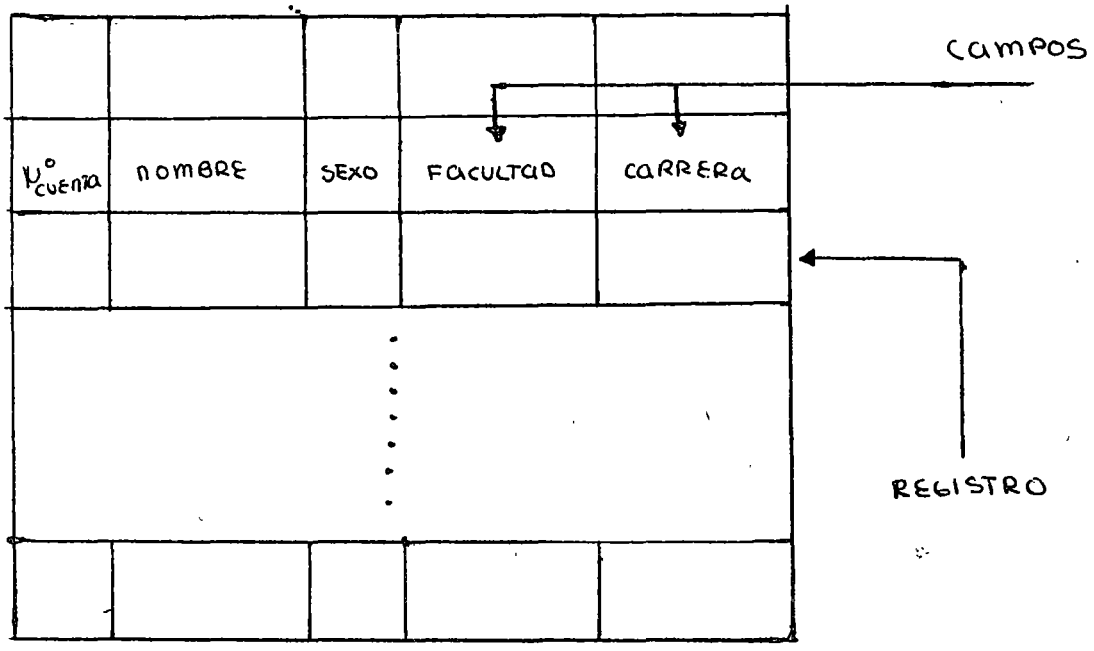
Ente algunos de los datos operacionales de la Universidad se encuentran los datos referentes a los estudiantes. Por cada estudiante tendremos su número de cuenta, nombre, sexo, Facultad a la que está inscrito y carrera que cursa.

No. CUENTA	NOMBRE	SEXO	FACULTAD	CARRERA

Al conjunto de datos referentes a un sólo estudiante le llamaremos un registro. Por lo tanto tendremos tantos registros como estudiantes inscritos tenga la Universidad.

A los datos que forman un registro les llamaremos campos. Así, un registro de un estudiante tendrá los siguientes campos:
No. Cuenta, Nombre, Sexo, Facultad y Carrera.

Y al conjunto de todos los registros de estudiantes le llamaremos el archivo de estudiantes.



Archivo de estudiantes

De estas ideas podremos concluir lo siguiente:

Archivo de datos = conjunto de registros de datos.

Registro de datos = conjunto de campos a los cuales se les asocia uno o varios datos en particular.

Diferencia entre registro lógico y registro físico.-

Si cualquier persona estuviera interesada en conocer el archivo de estudiantes, la imagen que tendría de éste, sería igual a la que hemos diseñado anteriormente. Es decir, esa persona pensaría que los datos están almacenados de la misma forma como los está viendo. Sin embargo, ésto no debe ocurrir necesariamente.

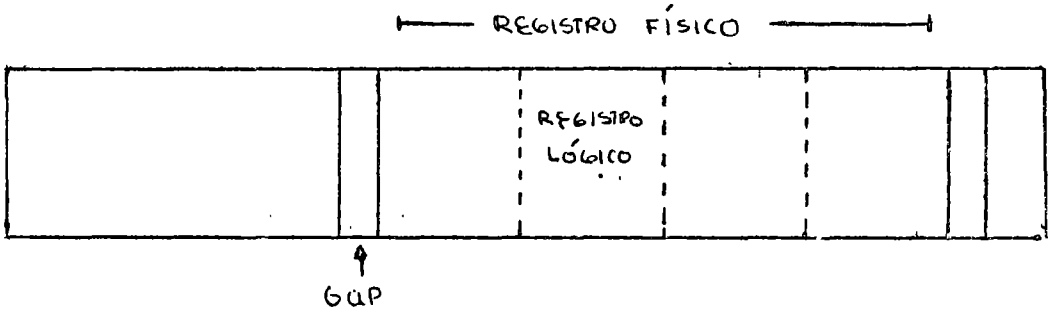
La estructura que el archivo de estudiantes tiene en disco, cinta, etc, puede ser completamente distinta a la estructura que tiene en memoria.

Al archivo, tal como lo ve el usuario es a lo que llamamos un archivo lógico. Por lo tanto, se deduce que los archivos lógicos estarán formados por registros lógicos.

Al almacenar este archivo lógico, en una cinta por ejemplo, su estructura necesariamente cambia.

Sabemos que una cinta está organizada por bloques físicos.

Y un bloque físico podría tener uno o varios registros lógicos, formando así un registro físico.



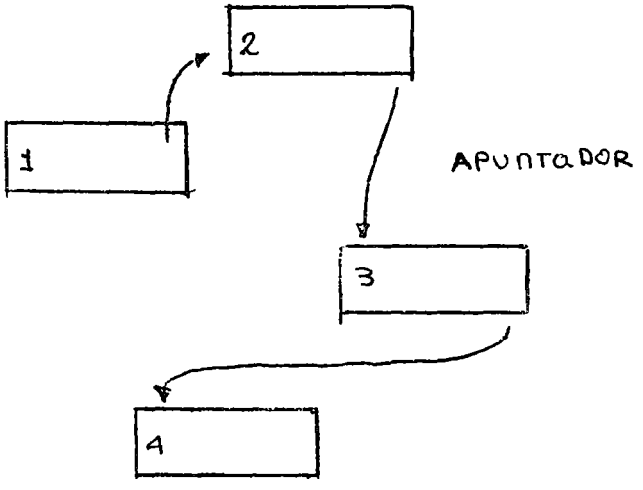
Ya que las lecturas o escrituras a cinta se hacen por medio de bloques físicos, sería conveniente tener la mayor cantidad posible de registros lógicos dentro de un bloque o registro físico.

El archivo físico es el conjunto de todos los registros físicos.-

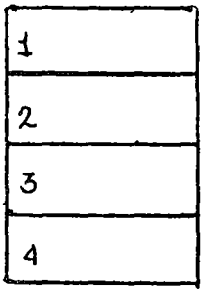
Dentro de este capítulo discutiremos las posibles técnicas para organizar nuestros registros físicos. Dado que una cinta tiene una estructura secuencial, la discusión se hará tomando a un disco como nuestro medio de

almacenamiento.

Apuntadores.- Un apuntador es un campo dentro de un registro que indica donde se localiza el siguiente registro de ese archivo.



ORGANIZACIÓN FÍSICA



ORGANIZACIÓN LÓGICA

Existen 3 tipos de apuntadores.

1.- Los apuntadores que consisten en la dirección física. Estos apuntadores son los más efectivos si se desea hacer una lectura del siguiente registro lo más rápidamente posible. Pero tienen la desventaja de ser dependientes del dispositivo que se está utilizando.

2.- Los apuntadores que consisten en una dirección relativa de el registro.- Si nuestro archivo se encuentra ordenado, el apuntador consiste en el número de registro de ese archivo. Este apuntador no es dependiente del dispositivo, pero si nosotros cambiamos

un registro de lugar dentro de un archivo, tendremos que corregir todos los apuntadores que vayan a él.

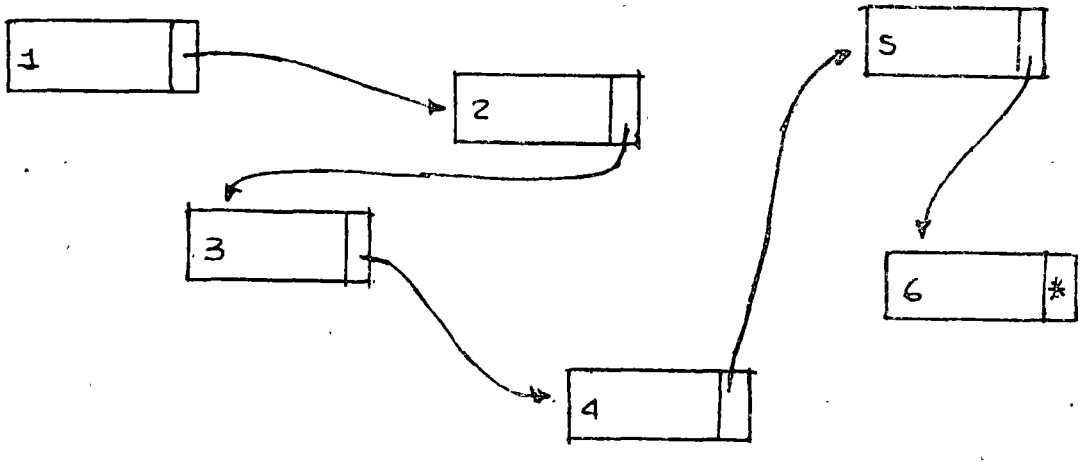
3.- Los identificadores de registros.- Estos apuntadores también se llaman simbólicos. Consisten en un campo que identifica a todos los registros. Por ejemplo: No. de cuenta de todos los alumnos. No es dependiente de el dispositivo ni dependiente de el lugar que ocupe el registro en el archivo.

Para recuperar y almacenar un registro, dado un apuntador de este tipo, se utilizan algunas técnicas de direccionamiento.

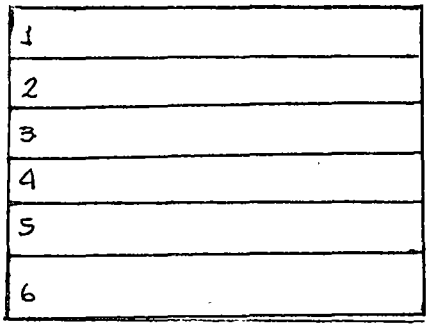
El tercer tipo de apuntadores, a pesar de no darnos la misma rapidez de recuperación como los apuntadores de dirección física, son los más usados gracias a la fexibilidad que presentan.

Cadenas y anillos.- Una cadena es un conjunto de registros de uno o varios archivos, unidos por una sucesión de apuntadores.

Organización Física



Organización Lógica

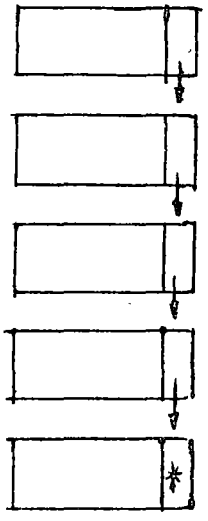


A las estructuras de cadenas también se les conoce con el nombre de listas.

Toda lista deberá de contar con una cabeza (lugar donde se inicia la lista) y una marca de fin de lista.

Pueden existir listas secuenciales y no secuenciales.

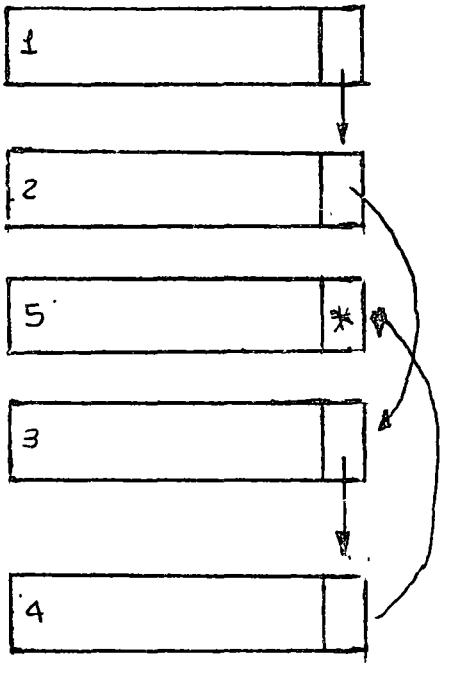
Las listas secuenciales son aquellas donde los registros están unidos bajo un cierto orden.



Lista secuencial

Si quisiéramos insertar un registro en este tipo de listas deberíamos de recorrer toda la lista hasta el lugar que le corresponde y ahí insertaría.

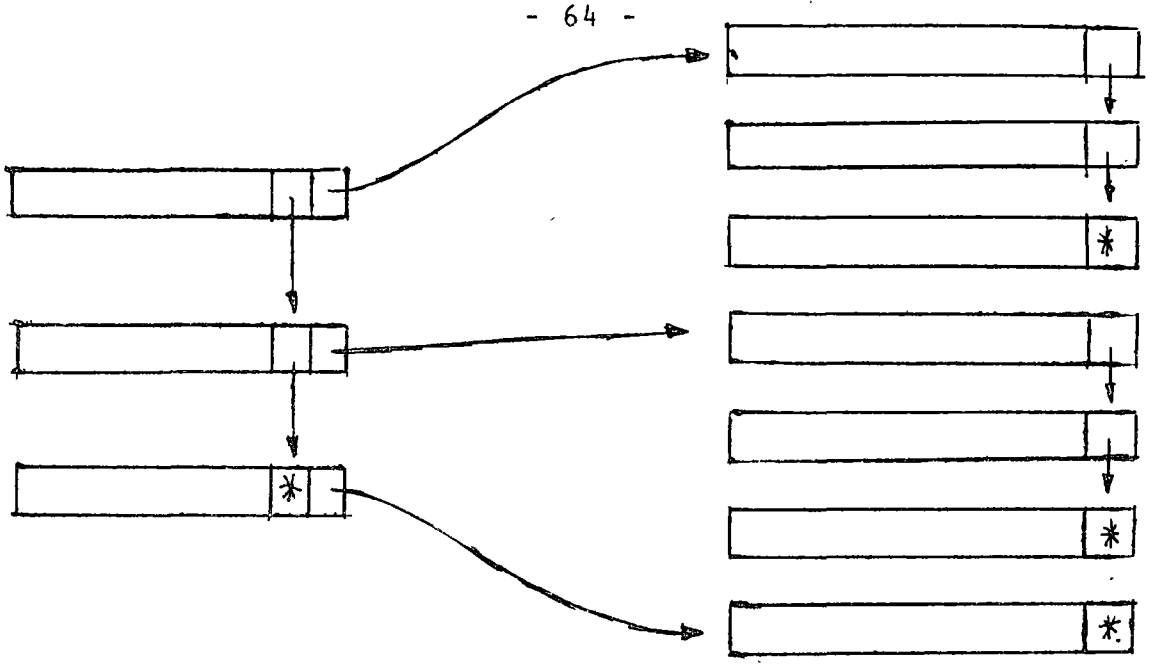
Si las listas no son secuenciales podríamos insertar un nuevo registro hasta el final de la lista.



Lista no secuencial

En todas las listas aparece el problema tanto del borrado como el del recorrido de toda una lista.

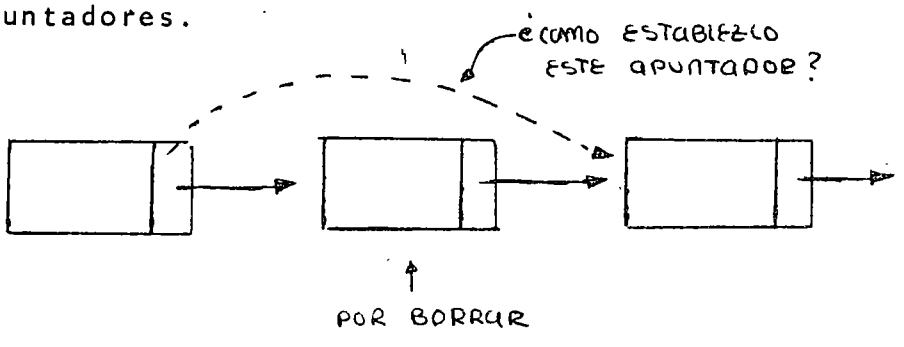
Para solucionar el segundo problema se utilizan 2 listas. Una sirve como directorio de nuestra lista principal



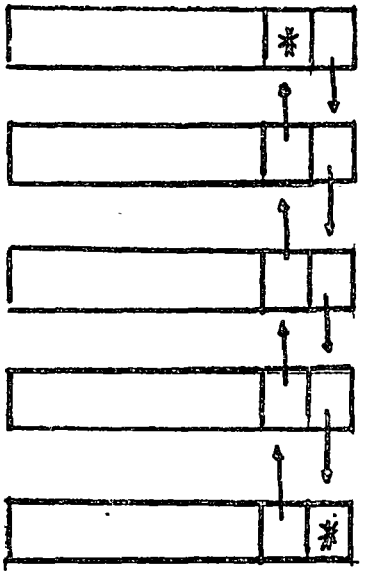
Lista Directorio

A este tipo de organizaciones se les llama multi-listas.

El problema de borrado aparece en la actualización de los apuntadores.

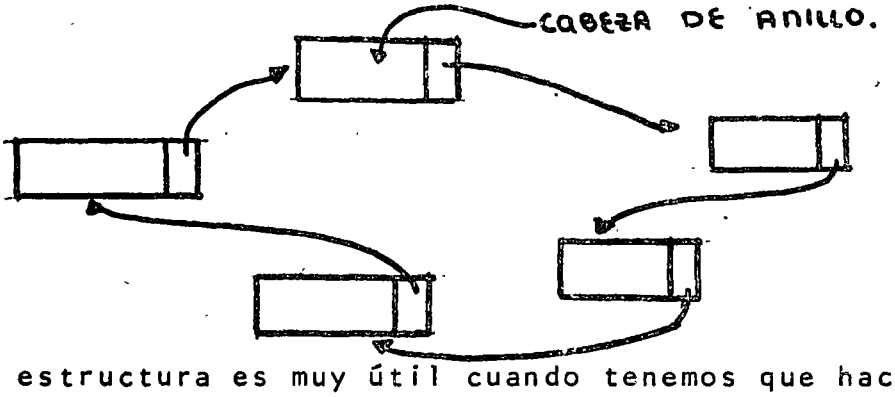


La pregunta es: ¿cómo localizamos al registro que apunta al registro que queremos borrar?. Para su solución se utilizan los dobles apuntadores. En este caso, un registro tiene un apuntador tanto al siguiente registro como al anterior.



Pueden existir también listas que se intersectan en algún registro, etc.

Una estructura de anillo es aquella donde el último registro de la lista apunta a la cabeza de éste.



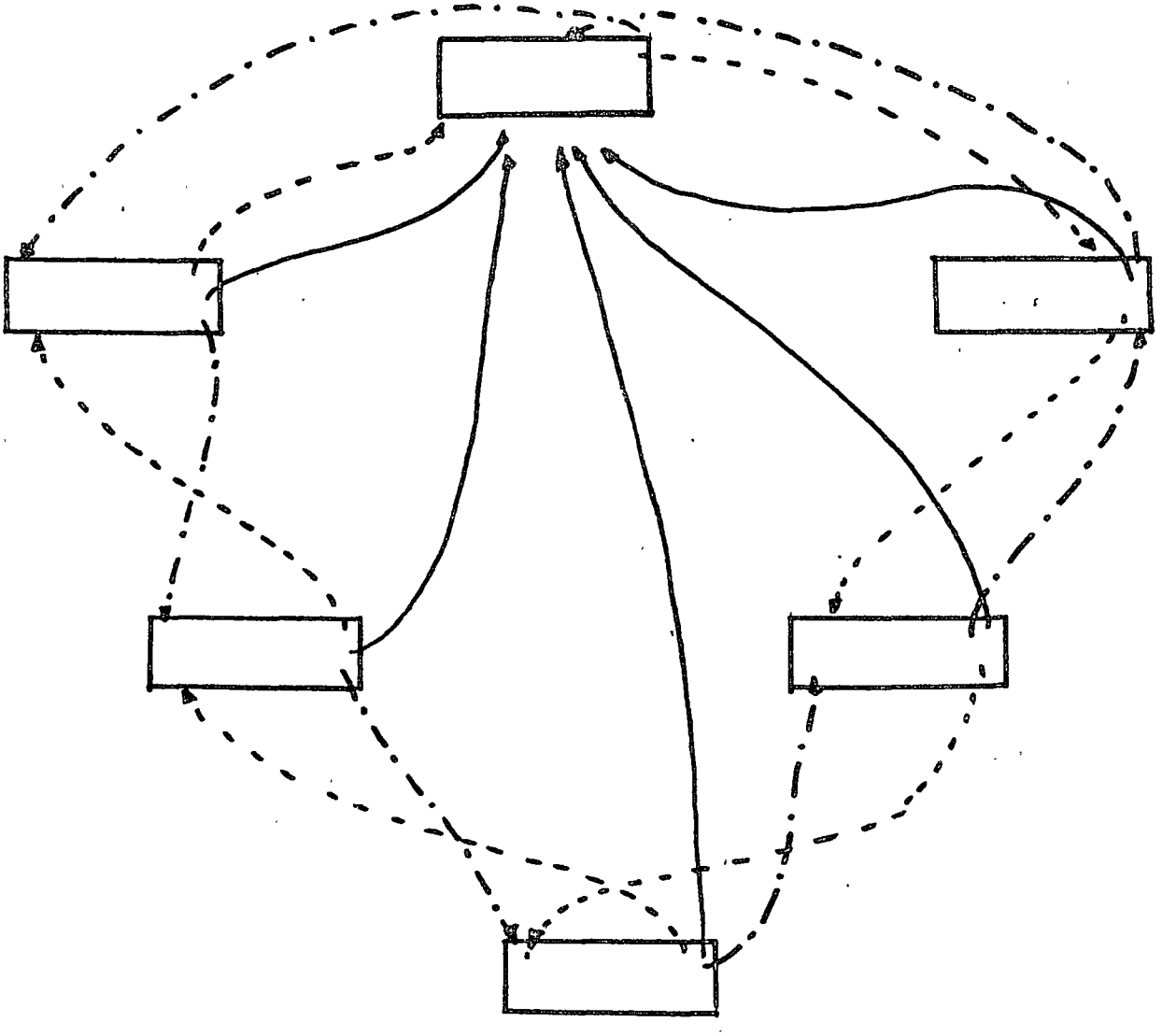
Esta estructura es muy útil cuando tenemos que hacer varios recorridos dentro de una lista. Una estructura de este tipo nos evita detectar el fin de la lista y buscar su cabeza en cada recorrido.

Aquí también son aplicables las ideas de dobles apuntadores, intersecciones, etc.

Una estructura también adolece de los mismos defectos de una lista (borrado, act, inserción), etc.

En este tipo de estructuras surge también la idea de un tercer apuntador.

Si nosotros localizamos un registro dentro de un anillo, por medio de un apuntador podemos evitar el terminar de recorrer todo el anillo para llegar a la cabeza. Esto nos ahorra tiempo en búsquedas.



Técnicas de direccionamiento.-

Los registros lógicos se identifican dentro de un archivo con una llave. Por ejemplo: el No. de cuenta de los estudiantes, el número de pieza en una fábrica, etc

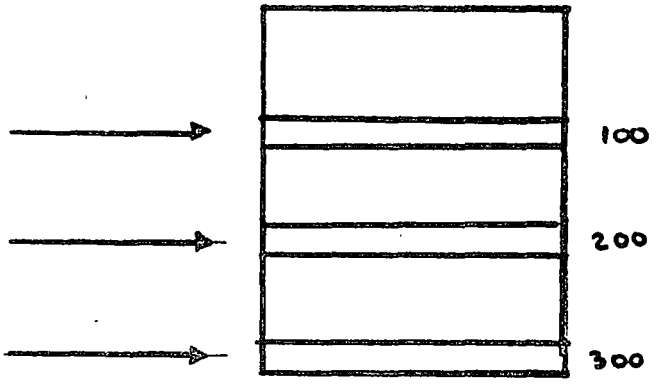
Esta llave debe de ser única. Es decir, debe de corresponder a un solo registro. Pero, un registro puede tener varias llaves. A esta llave también se le conoce con el nombre de identificador.

Las técnicas de direccionamiento son las formas como podemos recuperar o almacenar un registro lógico en un archivo físico por medio de su llave.

Las técnicas son las siguientes:

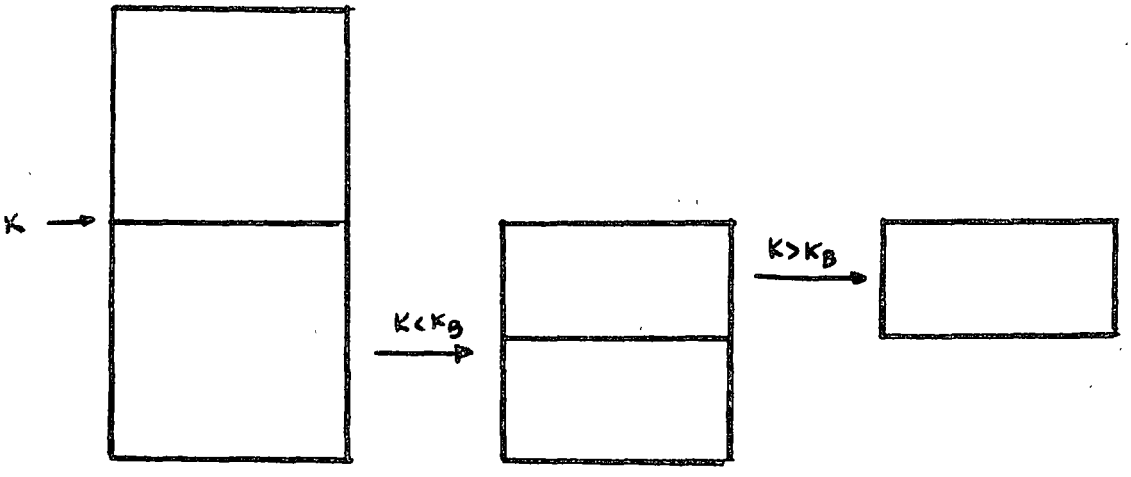
a) Recorriendo todo el archivo.- En este método vamos a recorrer todo el archivo inspeccionando llave por llave hasta encontrar el registro.

b) Búsqueda por bloques.- Si los registros de un archivo están ordenados bajo esta llave podemos inspeccionar cada registro múltiplo de 100 hasta encontrar una llave que sobrepase la llave que estamos buscando. En el bloque correspondiente podemos hacer un recorrido total de éste.



c) Búsqueda Binaria.- En esta técnica buscamos el registro que está a la mitad de el archivo ordenado obteniendo 2 bloques. Al inspeccionar la llave de ese registro comparamos:

- Si la llave buscada es menor tomamos el bloque inferior.
- Si la llave es mayor tomamos el bloque superior.



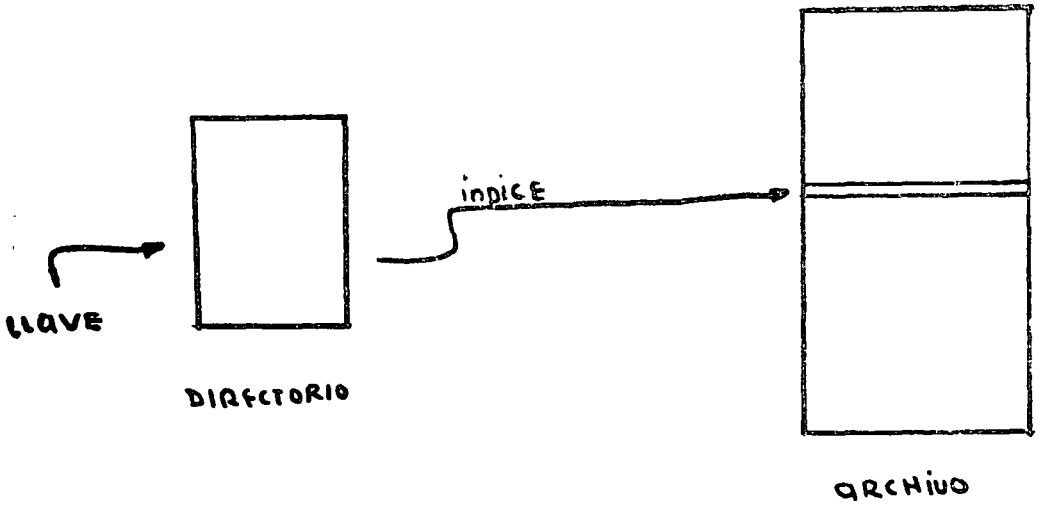
Este método no es bueno para búsquedas en dispositivos de acceso directo ya que consume mucho tiempo en la búsqueda del registro medio.

d) archivos secuenciales indicados.

Cuando los registros están organizados bajo una secuencia se utiliza un índice para accesarlos.

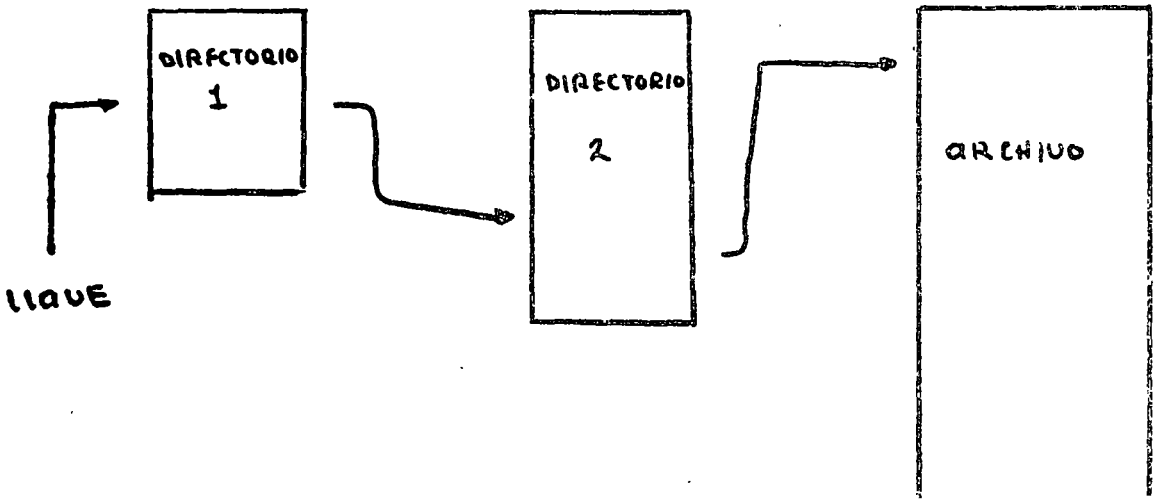
Este índice puede ser la dirección física o la dirección relativa de un bloque de registros.

Todos los índices se encuentran en una tabla (directorio) y cuando queremos recuperar un registro, la llave sirve como dato de entrada a nuestra tabla y el dato de salida será la dirección de el registro.



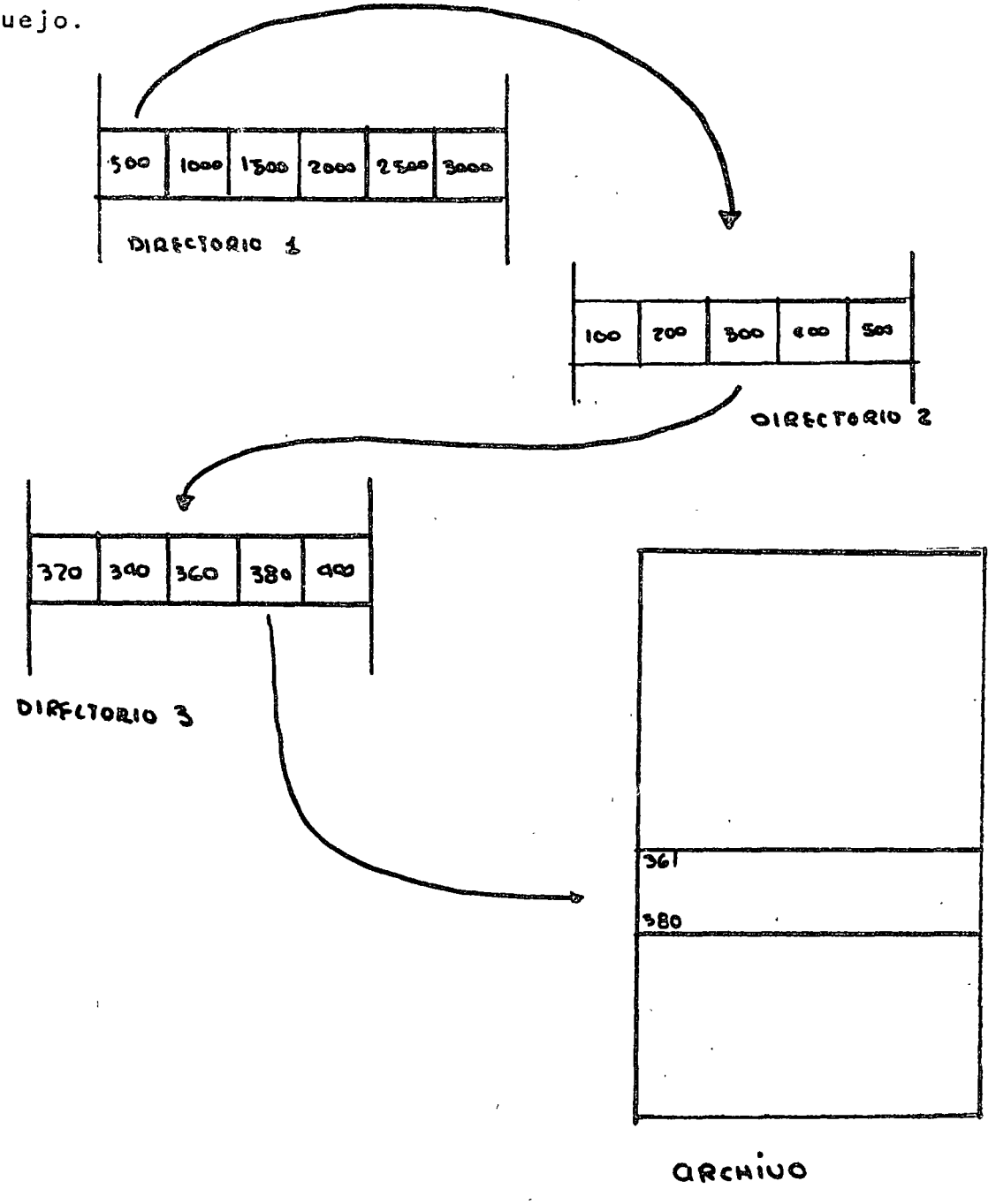
La ventaja primordial de esta estructura es que evitamos el hacer búsquedas en todo el archivo ya que éstas se hacen en nuestro directorio y en nuestros bloques, que son de tamaño menor.

Cuando solamente existe un directorio, los índices de éste se llaman primarios, sin embargo pueden a su vez existir directorios de los directorios.



En este caso la llave se utiliza para hacer nuestra búsqueda en el ler. directorio, el resultado se utiliza como llave para el 2° directorio el cual nos da la dirección de el registro.

Puede ser posible que el índice en lugar de indicarnos el lugar exacto de el registro nos de el bloque donde se encuentra y en ese bloque tendríamos que hacer un bosquejo.



e) Archivos no secuenciales indicados.

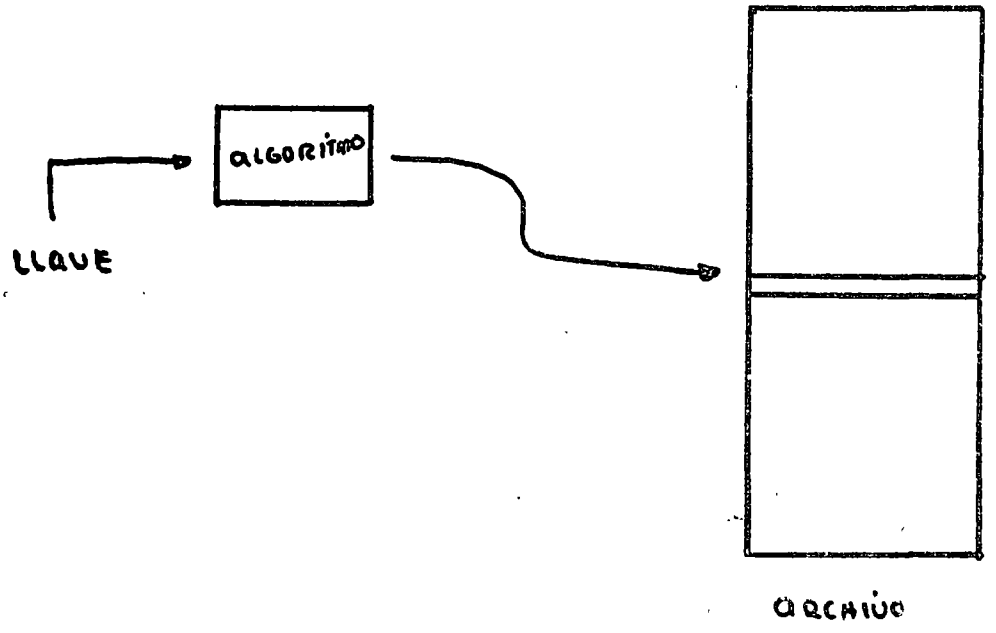
Este tipo de archivos funciona con los mismos métodos que un archivo secuencial indicado. Solamente que en este tipo de archivos el índice debe ser necesariamente la dirección del registro y no la dirección de un bloque. Esto puede hacer crecer enormemente nuestra tabla de índices.

La ventaja de un archivo no secuencial indicado es que puede ser accesado con mucha facilidad por dos directorios distintos, es decir, puede ser accesado por varias llaves.

Otra ventaja es la facilidad con la que se pueden hacer inserciones, actualizaciones y borrados.

f) Llaves = dirección.

Bajo este método se prentende convertir la llave directamente a la dirección de el registro utilizando alguna función o técnica.



Cuando es posible aplicar este método, nos provee de la forma más rápida de recuperar la información.

En esta técnica, el caso más simple es en donde la llave es igual a la dirección de el registro, sin embargo este método es casi imposible.

Dentro de las técnicas más usuales para el cálculo de las direcciones se encuentra el método aleatorio.

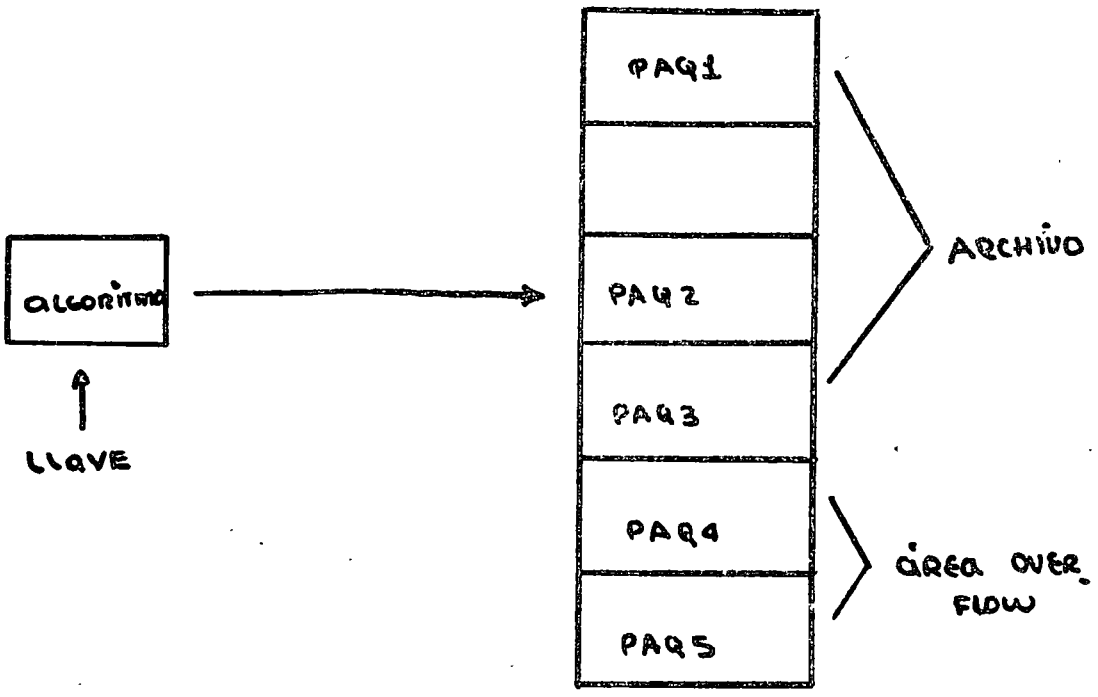
g) Método Aleatorio (Hashing).

Bajo este método, la llave es convertida a una dirección que puede ser la del registro o la del paquete de registros.

Dado el caracter aleatorio de este método, existirán direcciones de paquetes que nunca obtendremos. Esto ocasionará huecos dentro de nuestro archivo.

También puede pasar que dos llaves distintas nos den la misma dirección de el paquete.

En este caso tendremos que manejar áreas overflow donde almacenaremos aquellos paquetes cuya dirección original ya estaba ocupada.



Para el manejo de áreas overflow existen dos métodos: El overflow encadenado y el overflow distribuído.

En el 1° se reserva un área especial como área overflow.

En el 2° todos los huecos de el archivo son aprovechados como áreas para overflow.

Existen varios métodos para la conversión de llaves pero el más útil es el de la división.

En este método el No. es dividido por el número primo más cercano a el total de registros de el archivo. El residuo es tomado como la dirección de el registro.

II.- ORGANIZACION LOGICA

Como se ha discutido en el capítulo anterior, existe una imagen que el usuario tiene de la información.

Es así como hemos hablado de archivos y registros lógicos. Esta visualización tiene que estructurarse de alguna manera para facilitar la recuperación de registros lógicos. Es a esta estructura a la que llamamos estructura u organización lógica.

En el medio ambiente de bases de datos, a esta estructura también se le conoce con el nombre de modelo de datos ó modelo conceptual.

Existen 3 enfoques distintos de estructurar nuestra información lógica, es decir, existen 3 enfoques distintos de modelos conceptuales:

A) Modelo relacional

B) Modelo jerárquico

C) Modelo reticular (de redes)

Asociados a estos modelos, existen lenguajes que sirven para establecer la comunicación entre el usuario y la base.

Por cada modelo de datos existe un lenguaje asociado a este.

A continuación, se discutirá a detalle cada uno de los modelos y los lenguajes asociados a estos.

II.A. MODELO RELACIONAL

Este modelo esta estructurado bajo el concepto de relaciones. DEF: dada una colección de conjuntos D_1, $D_2 \ldots D_n$ (no necesariamente distintos) R es una relación de estos n conjuntos si es un conjunto de n-adas ordenadas ($d_1, d_2, \ldots d_n$) tal que d_1 per-

BASE DE DATOS

VENDEDORES - PARTES.

S

S#	NOMBRE	STAT	CIUDAD
S1	PEREZ	20	LONDRES
S2	GOMEZ	10	PARIS
S3	GONZALEZ	30	PARIS
S4	GUERRA	20	LONDRES
S5	BARRERA	30	ATENAS

VENDEDORES

P

P#	NOMBRE	COLOR	PESO	CIUDAD
P1	TUERCA	ROSA	12	LONDRES
P2	TORNILLO	VERDE	17	PARIS
P3	ARMELLA	AZUL	17	ROMA
P4	ARMELLA	ROJO	14	LONDRES
P5	RONDANA	AZUL	12	PARIS
P6	CLAVO	ROJO	19	LONDRES

PARTES

SP

S#	P#	CANT
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

VENDEDORES
PARTES.

tenece a $D_1 \ldots d_n \quad D_n$.

$D_1 \ldots D_n$ son los dominios de R, n es el grado.

Veamos un ejemplo de una relación:

Partes	N_0	nombre	color	peso	
	P_1	tornillo	.	.	.
	P_2	tuerca	.	.	.
	P_3		.	.	.

Generalmente una relación se representa como una tabla. En este caso tenemos la relación 'partes'.

Esta relación esta formada por 5 atributos y es de grado 5.

Sobre esta relación podemos hacer las siguientes observaciones, validas para cualquier relación:

- El orden de las n-adas o renglones es insignificante, aunque la mayoría de las veces conviene tenerlos ordenados, esto no es necesario.
- El orden de los atributos si es significante. Esto quiere decir que en una n-ada no podemos alterar el orden de los dominios con respecto a las demas n-adas.

Los dominios son los lugares de donde toman sus valores los atributos.

En este caso, existen 5 dominios, de igual nombre que los atributos.

Pero veamos el siguiente ejemplo:

Componentes

P MAYOR	P MENOR	Cantidad
P1	P2	2
P1	P4	4
P5	P3	1

Los atributos en este caso son 3, pero los dominios distintos son 2, No. de partes y cantidades vendidas.

No todas las relaciones son validas en un modelos relacional. Veamos el siguiente ejemplo:

R1

S#	PQ	
	P#	cant
S1	P1	300
	P2	200
	P3	400
S2	P1	200
	P2	100
S3	P2	100

R2

S#	P#	cant
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	200
S2	P2	100
S3	P2	100

La 1a. relación no es válida ya que una n-ada puede tener un conjunto de valores en un solo atributo. Una relación es válida si cada valor de cada atributo en una n-ada es única, es decir es atómico. La 2a. relación si es válida.

Es fácil de observar que toda relación puede convertirse -si no lo es- en una relación válida.

A las relaciones válidas se les llama normalizadas.

A el proceso de convertir una tabla no válida a una tabla válida (normalizada) se le llama normalización.

En el modelo relacional se aplica el concepto de llave ya antes discutido, recordando que la llave será el valor de un atributo que identifica totalmente a la n-ada o renglón.

Las llaves deben ser únicas, y un renglón puede utilizar la combinación de varios atributos para seleccionar su llave.

Si pensamos en términos de archivos, podemos relacionar una n-ada a un registro, un campo a un atributo y una relación a un archivo.

El modelo relacional tiene muchas ventajas:

1.- Su facilidad de acceso.

2.- La facilidad que ofrece para mantener la consistencia de la información, es decir, cuando nosotros actualizamos un atributo -color de pieza- solamente lo hacemos en ese archivo, ya que es el unico lugar donde se encuentra esa información.

En otros métodos tendriamos que buscar en varios lugares esa información, corriendo el peligro de olvidar alguno.

3.- La facilidad para la inserción o borrado de registros - podemos borrar un registro sin afectar los demas lenguajes asociados a el modelo ralacional.

El objetivo de cualquier usuario al trabajar con una base de datos es el de recuperar información.

Tradicionalmente, el tipo de recuperación que se hacía era:

- Por registro, para lo cual deberíamos de saber la llave de éste.

- Por archivos, y nosotros tendríamos que hacer las búsquedas que localizarán un registro.

Además, era bien difícil obtener resultados que fueran uniones o intersecciones de registros de distintos archivos. Generalmente deberíamos de realizar nosotros todas estas operaciones.

Veamos los siguientes ejemplos:

¿Quiero todas las ciudades del vendedor 1?

S#

S#	NOMBRE	STAT	CIUDAD
S1	PEREZ	20	LONDRES
S2	GOMEZ	10	PARIS

→

CIUDAD
LONDRES

¿Dame el nombre de todas las partes vendidas por el vendedor 1?

SP

S#	P#
S1	P1
S1	P2

P

P#	NOMBRE
P1	TUERCA
P2	TORNILLO

→

NOMBRE
TUERCA
TORNILLO

Como podemos observar, los resultados que obtenemos son relaciones. Estas inclusive pueden ser unarias - un solo renglón, una sola columna -.

Esta conclusión era de esperarse si estamos tratando con relaciones, el resultado de operar con ellas debe ser una relación.

Los lenguajes asociados al modelo relacional nos dan formas de definir las relaciones que queremos como resultados y las características que deben de tener sus elementos.

Algunos lenguajes lo hacen con operaciones basadas en el álgebra relacional: IS/1, etc.

Algunos otros se basan en el cálculo relacional: ALPHA, QUEL, otros lo hacen por medio de mapeos: SQUARE AND SEQUEL, y otros lo hacen por medio de un registro. Ejemplo: CUPID, QUERY BY EXAMPLE.

- Lenguajes basado en las operaciones de el álgebra relacional.

Existen las operaciones que todos conocemos:

UNION.- Hace la unión de 2 relaciones.

INTERSECCION.- Hace la intersección de 2 relaciones bajo una columna en específico.

DIFERENCIA.- La diferencia de 2 relaciones, A y B son aquellos registros de A que no tienen ninguna posible conexión con B.

Estas operaciones tienen como únicos requisitos que las relaciones deben ser de el mismo grado y deben de contar con un atributo común.

Existen otro tipo de operaciones que merecen más explicación.

SELECCIONA,- El operador SELECT extrae de una relación todos aquellos registros que cumplan con una condición dada. Por ejemplo:

SELECT S WHERE CIUDAD = 'LONDRES'

En este caso en el archivo s toma todos los vendedores que venden en Londres.

S#	SNOMBRE	STAT	CIUDAD
S1	PEREZ	20	LONDRES
S4	GUERRA	20	LONDRES

Es de notarse que un registro puede cumplir varias condiciones.

SELECT SP WHERE S#= 'SI' AND P#= 'PI'

- Del archivo SP quiero todos los registro que cumplan con que el No. del vendedor es S1 y el número de la pieza es P1.

S#	P#	CANT.
S1	P1	300

PROYECTA.- Esta operación selecciona de un archivo los atributos que se quieran. Es útil para cambiar el orden de los atributos en una relación.

PROYECTA S SOBRE NOMBRE, CIUDAD, STATUS.

- Dame de el archivo S dame todos los registros con nombre, ciudad, status.

NOMBRE	CIUDAD	STATUS
PEREZ	LONDRES	20
GOMEZ	PARIS	10
GONZALEZ	PARIS	30

UNE.- Este comando une 2 relaciones bajo una llave común. En el resultado de nuestra unión aparecerá repetida la llave.

DIVIDE.- Este comando obtiene de una relación binaria todos aquellos registros que tengan todos los elementos de otra relación unaria.

El atributo de la relación unaria debe ser uno de los 2 atributos de la relación binaria.

Estos son los comandos que utiliza el álgebra relacional.

Para demostrar el uso de cualquier lenguaje que aqui se discuta, analizaremos el tipo de preguntas para

a) recuperar datos

b) insertar nuevos registros

c) borrar registros

La actualización podemos verla como una combinación de borrado e inserción.

a) Recuperación.

- Quiero todos los números de los vendedores que venden la parte P2.

Selecciona SP donde P#= "P2" llámale TEMP proyecta TEMP sobre S# llámale resultados.

Como podremos ver,para efectuar esta pregunta hemos hecho 2 oraciones.

Primero hemos creado una relación con todos los vendedores que venden P2 y después hemos tomado de esa relación los números de los vendedores.

Ya que los resultados son siempre resultados podríamos unir estas oraciones en una sola.

Proyecta (selecciona SP donde P#= 'P2') sobre S# llámale resultados.

- Quiero los nombres de los vendedores que venden al menos una parte roja.

- Selecciona P donde color= "ROJO" llámale TEMP1 une TEMP1 y SP sobre P# llámale TEMP2 proyecta TEMP2 sobre S# llámale resultado.

- Proyecta (une (selecciona P donde color= 'ROJO') y SP sobre P#) sobre S# llámale resultado.

b) Inserción.- Para hacer inserciones utilizamos la operación unión.

P UNION (P7, 'LAVADOR', 'GRIS, 2, ATENAS) Llámale P.

Da de alta el registro entre corchetes en el archivo P.

B) BORRADO.- Para hacer borrados hacemos uso de la operación menos.

S Menos ('SI,?,?,?) llámale SP

Borra de el archivo SP todos los registros que tiene SI como valor en el atributo S#.

Las ventajas de este lenguaje son:

1.- Simplicidad.

2.- Completez

3.- Fácil de extenderse.

4.- Fácil de ser llamado por cualquier lenguaje de alto nivel.

5.- En ningún momento se indica la forma o método para encontrar la respuesta.

Lenguajes basados en el cálculo relacional.

Este lenguaje se basa en expresiones que definen una relación nueva en términos de las que conocemos. Esta expresión se divide en 2 partes.

La 1a. parte tendría la especificación de los atributos de la nueva relación, y la 2a. parte es un predicado que define las cualidades de esa relación.

- Por cada parte, dame su número y las ciudades donde se vende.

(SP.P#, S.CIUDAD): SP.S#=S.S#

Aquí hemos definido los elementos de mi nueva relación y de donde proceden.

El predicado indica la condición que deberán cumplir los elementos de mi nueva relación.

Existen 2 comandos en este tipo de lenguajes

GET X .- Se recupera una relación identificada con X

PUT X .- Se manda una relación identificada con X

Nuevamente, recurriremos a las operaciones de recuperación, inserción y borrado para ejemplificar el uso de este lenguaje.

a) Recuperación.

a1) Simples.

- Dame los números de todas las partes que se venden

GET W (SP.P#)

- Dame todos los detalles de todos los vendedores

GET W(S)

a2) Calificativas.

- Dame todos los vendedores de Paris con status mayor a 20

GET W(S.S#): S.CIUDAD= 'PARIS' AND S.STATUS GTR 20

- Nótese que podemos usar cualquier operación aritmética ó booleano.

a3) Con ordenamiento.

- Dame de los vendedores de Paris, su número y status. Ordénalos en orden descendiente de acuerdo al status

```
GET W (S.S#, S.STATUS): S.CIUDAD= 'PARIS' DOWN
                                          S.STATUS
```

Down indica que los resultados se darán en orden descendente de acuerdo al status.

También existe UP para orden ascendente.

a4) Recuperación usando variables de rango.

Dame el No. de vendedores que venden la parte 2

```
GET W(SP.S#): SP.P#= 'P2'
```

También podemos escribirla así:

```
RANGE SP, X
GET  W(X.S#): X.P#= 'P2'
```

Una variable de rango corre sobre el archivo en la que ha sido definida sus valores posibles son registros de ese archivo.

La principal razón de usar estas variables de rango es que en algunas preguntas nos referimos a relaciones cuya definición dependa de la existencia de algún tipo de n-ada en un archivo. En ese caso se utilizan cuantificadores (,) y para simplificar la expresión, se utilizan variables de rango con estos cuantificadores.

a5) Recuperación usando un cuantificador existencial.

- Dame los nombres de los vendedores que venden la parte 2

```
RANGE SP,X
GET W (S.NOMBRE)  : X(X.S#=S.S# AND X.P#='P2')
```

Explicación: Como todos los nombres del archivo S tal que exista una n-ada en el archivo SP que cumpla con que su número de vendedor sea igual al del archivo S y su número de pieza sea igual al P2.

- Dame los números de los vendedores que venden al menos una parte roja

```
RANGE P PX
GET W(SP.S#):  PX(PX.P#=SP.P#  AND
                    PX.COLOR= 'ROJO')
```

- Dame los nombres de los vendedores que venden al menos una parte roja.

```
RANGE P PX
RANGE SP SPX
GET W (S.SNAME):  SPX(SPX=S.S#      AND
                    PX(PX=P#=SPX.P#  PX.COLOR='COLOR))
```

Aquí ejemplificamos el uso de dos cuantificadores existenciales anidados.

Generalmente, un predicado puede escribirse de varias formas, todas equivalentes.

Lo ideal será que todos los predicados los expresemos en su forma prenex normal.

```
GET W(S.SNAME):
 SPX  PX(SPX.S#=S.S#∧SPX.P#=PX.P#∧ PX.COLOR= 'ROJO')
```

a6) Relaciones que se forman de 2 ó mas relaciones distintas.

- Por cada parte dame su número y las ciudades donde se venden

```
GET W(SP.P#, S.CIUDAD): SP.S#= S.S#
```

- Dame los números de todos los vendedores y partes en una misma ciudad

```
GET W(S.S#, P.P#): S.CITY= P.CITY
```

a7) Expresiones con cuantificadores universales.

- Dame los nombres de todos los vendedores que no venden la parte 1

```
RANGE SP SPX
GET W (S.NOMBRE):  SPX(SPX.S#≠S.S# ∧ SPX.P#≠P1)
```

Esto puede ser traducido a: quiero los nombres de todos los vendedores que no coinciden en número en los archivos SP y S o que no vendan la parte 1.

a8) Recuperación usando ambos tipos de cuantificadores.

- Dame los números de todos los vendedores que venden todas las partes

```
RANGE P PX
RANGE SP SPX
GET W (S.S#):
     PX  SPX(SPX.S#= S.S# ∧SPX.P#= PX.P#)
```

a9) Recuperación usando implicación

- Dame los números de todos los vendedores que al menos venden las partes vendidas por el vendedor?

```
RANGE P PX
RANGE SP SPX
RANGE SP SPM
GET W(S.S#):  PX(  SPX(SPX.S#='S2'∧ SPX.P#= PX.P#)
                   SPY(SPY.S#=S.S#∧SPY.P#= PX.P#)
```

"La existencia de una n-ada SPX implique la existencia de una n-ada SPY."

A1) ACTUALIZACION

- Cambia el color de la parte 2 a amarillo.

HOLD W (P,P#,P.COLOR): P.P#='P2'

W.COLOR = 'AMARILLO' ←—— EN EL LENGUAJE HUESPED

UPDATE W

HOLD funciona igual que GET, pero previene al sistema de que se intenta hacer una actualización. UPDATE se encarga de escribir la relación. Debe notarse que la llave de la nueva relación coincide con la llave de nuestra relación a actualizar.

Si un usuario después de aplicar un HOLD decide no actualizar cuenta con el operando RELEASE

RELEASE W

B) INSERCIONES

- Agrega el registro a la relación P.

P7, LAVADOR, GRIS,?,ANTENAS

W.P#=P7

W.NOMBRE=LAVADOR

←—— ESTO EN EL LENGUAJE HUESPED

W.COLOR=GRIS

W.PESO=2

W. CIUDAD=ANTENAS

PUT W(P)

Si queremos hacer inserciones con ordenamiento contamos con los operadores UP, DOWN.

PUT W(P)UP P.P#

C) BORRADO

Borro al vendedor S1

```
HOLD W(S): S.S#='SI'
DELET W
```

DELET borra el registro en la relación S, pero lo respeta en la relación W. Esto nos proteje de borrados accidentales.

¿Cómo actualizar una llave?

- Cambia el número de la pieza 2 a 8

```
HOLD W(P):P.P#='P2'
DELETE W
W.P#='P8'
PUT W(P)
```

El lenguaje ALPHA, estructurado bajo esta forma cuenta con algunas funciones de biblioteca para resolver algunos tipos especiales de preguntas. Ejemplos

- Dame el número total de vendedores

```
GET W (COUNT(S.S#)); W=S
```

COUNT.- Una función que cuenta en No. de elementos de la relación.

- Dame la cantidad total de partes 2 vendidas

```
GET W(TOTAL(SP.QTY)): SP.P#='P2'
```

TOTAL:= Suma los valores de un atributo de una relación.

VENTAJAS.- Las ventajas de este lenguaje son de hecho las mismas que en el de algebra relacional

- Simple
- Completo
- Fácil de extender
- Fácil de implementar en lenguajes de alto nivel.
- No se indica como se deben de obtener los resultados.

Existen varios lenguajes parcialmente implementados, entre ellos se encuentra SEQUEL y QUERY por ejemplo.

SEQUEL es un lenguaje que funciona de forma similar a uno basado en el álgebra relacional.

Sólamente cambia en los comandos que lo forman.

Veámos unos ejemplos:

- Dame todos los números y status de los vendedores en París.

```
SELECT S#, STATUS
FROM S
WHERE CIUDAD='PARIS'
```

Aquí podemos notar que se hace uso del comando SELECT= selecciona.

Con FROM indicamos de que archivo y con WHERE las condiciones que se deben de cumplir.

Pueden existir varias condiciones para lo cual se utilizan los conectivos AND o OR

```
SELECT S#
FROM S
WHERE CIUDAD='PARIS'
      AND STATUS>20
```

Bajo este lenguaje los resultados pueden darse duplicados. Existe un comando UNIQUE que elimina todo posible duplicado.

```
SELECT UNIQUE P#
FROM SP
```

Nótese que la respuesta daría muchos P# repetidos, UNIQUE elimina esas repeticiones.

Dentro de este lenguaje también es posible hacer anidaciones en una expresión:

```
SELECT SNOMBRE
FROM S
WHERE S# IS IN
        FROM SP
        WHERE P#='P2'
```

Nótese que primero obtendremos todos los números de los vendedores que venden P2 y después, de esos números obtendremos los nombres de los vendedores.

SEQUEL hace uso de los comandos IS IN ó IN y IS NOT IN ó NOT IN para anidar expresiones.

También podemos obtener relaciones de varias tablas:

```
SELECT UNIQUE P#,CIUDAD
FROM SP,S
WHERE SP.S#=S.S#
```

También aquí existen preguntas de cuantificaciones

- Dame los nombres de los vendedores que venden todas las partes

```
SELECT SNOMBRE
FROM S
WHERE (SELECT P#
       FROM SP
       WHERE S#.S.S#)
       =
       (SELECT P#
        FROM P)
```

Aquí tomamos todas las piezas que vende l vendedor (ler mapeo) pero con el OPERADOR= sólo nos quedamos con aquellos en que sus piezas coincidan con todas las del archivo P.

También existen variables de rango:

```
SELECT UNIQUE P#
FROM SP SPX
WHERE P# IN
   SELECT P#
   FROM SP
   WHERE S# ≠ SPX.S#
```

La relación de donde tomamos la condición es aquel formado por P# tal que el S# de el registro donde tomamos P# no coincida con el S# de cualquier otro registro.

Analicemos la siguiente pregunta

```
SELECT UNIQUE S#
FROM SP,SPX
WHERE (SELECT P#
       FROM SP
       WHERE S#=SPX.S#)
       CONTAINS
       (SELECT P#
        FROM SP
        WHERE S#='S2')
```

En esta pregunta hacemos contenciones.

Nuestro primer mapeo debe de contener al menos todos los elementos del 2° mapeo.

La pregunta corresponde a:

Dame los números de los vendedores que al menos venden las partes vendidas por S2.

Las actualizaciones se hacen de una manera muy fácil.

```
UPDATE P
SET COLOR='AMARILLO'
WHERE P#='P2'
```

Cambiamos el color de la pieza P2 a amarillo.

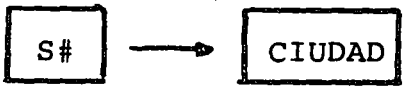
Las inserciones también se hacen de manera bastante fácil:

```
INSERT INTO P:
     P7,LAVADOR,GRIS,2,ANTENAS
```

NORMALIZACION.- Hemos hablado de que un sistema relacional solamente debe de trabajar con tablas normalizadas : tablas formadas por valores atómicos.

En esta sección se discuitrá el porque de las tablas normalizadas.

Si analizamos nuestra tabla de vendedores, podemos observar que el atributo ciudades toma sus valores dependiendo del valor que tome el No. de vendedor. Es decir, podemos observar que un vendedor vende en 1 sola ciudad. Puede suceder lo contrario; que varios vendedores vendan en una sola ciudad, sin embargo la dependencia sigue existiendo.



es en este caso que decimos que un atributo -CIUDAD- es funcionalmente dependiente de otro.

Las tablas normalizadas se dividen en 4 tipos:

1FN, 2FN, 3FN y 4FN

Todas las tablas normalizadas son de 1FN, algunas de ellas son 2FN. De estas unas pocas son 3FN y de éstas otras algunas son 4FN.

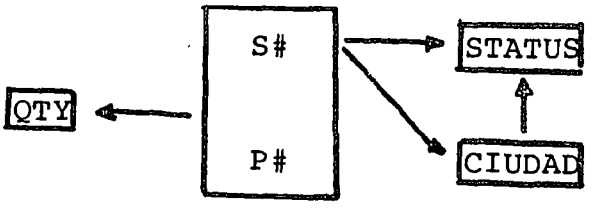
Todas estas relaciones son válidas para un modelo relacional, pero presentan algunas desventajas. De todas ellas las tipo 4FN son las mejores. Por lo tanto, el ideal será que nuestras tablas sean de tipo 4FN.

Relaciones tipo 1FN.- Una relación es de tipo 1FN si todos los dominios solamente tienen relaciones atómicos.

Consideremos la siguiente relación de tipo 1FN

UNO	S#	STATUS	CIUDAD	P#	QTY
	SI	20	LONDRES	P1	300
	SI	20	LONDRES	P2	200
	SI	20	LONDRES	P3	400
	SI	20	LONDRES	P4	200
	SI	20	LONDRES	P5	100
	SI	20	LONDRES	P6	100
	S2	10	PARIS	P1	300
	S2	10	PARIS	P2	400
	S3	10	PARIS	P2	200
	S4	20	LONDRES	P2	200
	S4	20	LONDRES	P4	300
	S4	20	LONDRES	P5	400

Las dependencias funcionales



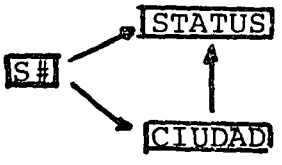
Sus desventajas:

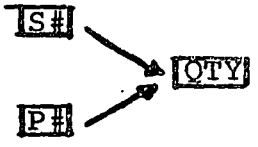
Inserción.- No podemos dar de alta un vendedor en una Ciudad hasta que éste venda al menos 1 pieza en esa ciudad.

Borrado.- Si queremos borrar a un vendedor que vende "X" pieza, también perderemos la información de en que ciudad se encuentra localizado ese vendedor.

Actualización.- Si queremos cambiar el lugar de residencia de un vendedor tendremos que hacer una búsqueda exhaustiva en la tabla.

Si descomponemos las dependencias funcionales en 2 relaciones tenemos:





DOS	S#	STATUS	CIUDAD
	S1	20	LONDRES
	S2	10	PARIS

SP	S#	P#	QTY
	S1	P1	300
	S1	P2	200

Con estas 2 relaciones nos evitamos los problemas que teníamos en la la. tabla.

-2a forma normal 2FN- Una relación es de tipo 2FN si es de tipo 1FN y además cada atributo no llave es totalmente dependiente de la llave.

Un atributo es totalmente dependiente de una llave si: la llave es compuesta y el atributo no depende de alguno de los atributos que forman la llave.

Las 2 últimas relaciones que hemos obtenido son de tipo 2FN.

Toda relación de tipo 1FN que no es tipo 2FN puede ser llevada a una forma de ese tipo haciendo proyecciones, tal como nosotros lo hemos hecho.

Las relaciones tipo 2FN todavía presentan algunos problemas, veamos:

Inserción.- Si queremos calificar a una ciudad con un Status, deberemos de esperar hasta que un vendedor se localice en ésta.

Borrado.- Si un vendedor se mueve de una ciudad, y es el único en ésta, al borrarlo perderemos el status de la ciudad.

Actualización.- Si queremos hacer un cambio de status en una ciudad, tendremos que hacer una búsqueda en toda la tabla, o de lo contrario podemos provocar inconsistencia en la información.

Estas desventajas se pueden solucionar si las dependencias funcionales de la tabla dos las descomponemos en:

S# ⟶ CIUDAD CIUDAD ⟶ STATUS

Obteniendo 2 relaciones nuevas:

SC	S#	CIUDAD
	S1	LONDRES
	S2	PARIS

CS	CIUDAD	STATUS
	ATENAS	30
	LONDRES	20

Estas tablas no presentan las mismas desventajas.

Relaciones en tipo 3FN.- Las relaciones tipo 3FN son aquellas que son tipo 2FN y los atributos son directamente dependientes de las llaves.

El tipo de relación se determina por las dependencias funcionales, por lo tanto no es posible con solo un vistazo decidir que tipo de relación tiene una tabla.

Supongamos que a los atributos de los cuales otros dependen les llamamos determinantes. Podemos definir entonces:

- Una relación es de tipo 3FN si cada determinante es una posible llave.

Esta definición tiene algunas ventajas ya que hace posible el rechazo de algunas relaciones que podrían considerarse 3FN.

Esas relaciones son aquellas donde puede existir más de una llave.

Sea la relación SSP (S#,NOMBRE,P#,QTY).

Supongamos que los nombres son únicos, es decir, a cada número de vendedor le corresponde un nombre y viceversa.

Las llaves de esta relación son (S#,P#), (NOMBRE,P#) bajo la definición primera, esta relación sería de tipo 3FN. La definición dice no indica qué atributo debe ser totalmente dependiente si es componente de otra llave.

Bajo la nueva definición, esta relación no es 3FN. S# y nombre son determinantes, pero ninguno de ellos por sí solo puede ser una posible llave.

Si hubiéramos aceptado esta relación, tendríamos problemas con las actualizaciones. Si queremos cambiar de nombre a un vendedor tendríamos que hacer una búsqueda exhaustiva, en toda la relación.

Relaciones tipo 4FN.-

Sea la relación CTX en tipo 3FN

CURSO	MAESTRO	TEXTO
FISICA	LOPEZ	MECANICA BASICA
FISICA	LOPEZ	PRINCIPIOS DE OPTICA
FISICA	GONZALEZ	MECANICA BASICA
FISICA	GONZALEZ	PRINCIPIOS DE OPTICA

Si se introduce un nuevo texto, tendremos que dar de alta dos nuevos registros (uno por cada profesor). Obviamente ésto no sería lo ideal. Si dividimos esta relación en 2:

CT (CURSO,MAESTRO) CX(CURSO,TEXTO)

Se resuelven nuestros problemas pero estas dos no son del tipo 3FN.

Si observamos nuestros problemas se deben a que: un curso determina no sólo uno, sino varios profesores (es un atributo multideterminante) y el valor de los profesores depende del curso (dependencia multi-valuada).

Definimos una relación de tipo 4FN si:

Cuando exista una dependencia multivariada en R, digamos de B en A, todos los demás atributos de R son totalmente dependientes de A.

CTX no es una relación 4FN, sin embargo

CT y CX sí son del tipo 4FN.

ENFOQUE JERARQUICO.

1.- Arquitectura.

Muchos de los sistemas de base de datos actuales están basados en este enfoque, entre ellos el IMS de IBM. Para mejor entender el modelo jerárquico se estudiará el sistema IMS.

El sistema IMS tiene varias versiones y las principales son el IMS/360 versión 1, versión 2 y la actual IMS/VS versión 1. La forma básica de este sistema sólo permite corridas "batch" y se puede extender a la forma interactiva simulando trabajo batch.

La siguiente figura muestra la arquitectura del sistema.

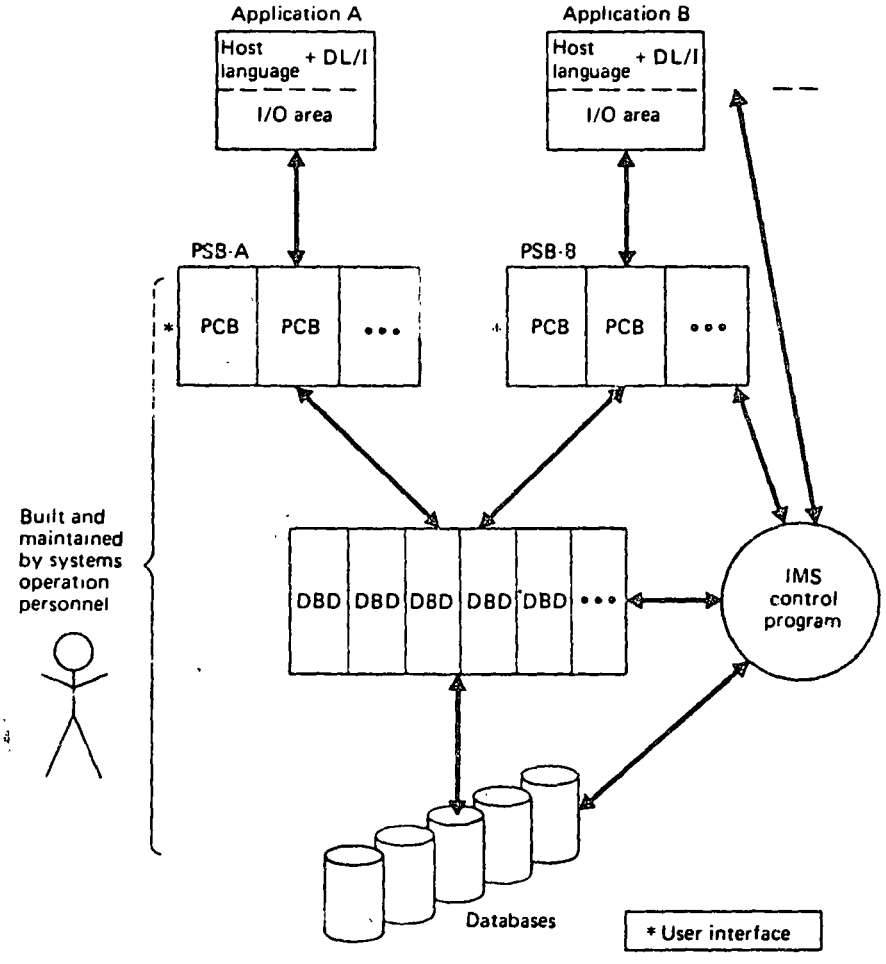


Fig. 1: Arquitectura del Sistema IMS.

Una base IMS es la representación de la información almacenada de una "base física" y una base física se puede entender como una lista o tabla (no normalizada) de datos.

El modelo de datos es una colección de bases físicas.

Esto es, el modelo consiste de varias listas o tablas de datos que estan almacenadas físicamente en varios archivos. El usuario no ve el almacenamiento físico de la información sino ve una representación o virtualización de ella, se le llama base IMS y al conjunto de estas bases se le llama modelo. Del mismo modo que en el modelo relacional existe la interfaz entre el usuario y la base física. A continuación se establecen los elementos más importantes de ella.

Cada base física esta definida por la descripción de la base de datos "DBD". Aquí también se define el mapeo a la forma interna de almacenamiento.

En esta arquitectura el usuario no actua directamente sobre el modelo antes descrito, sino que lo hace en un modelo externo. Este modelo externo consiste de una o varias bases de datos o subconjuntos de ellas. Así pues, se puede entender al modelo externo como la parte del sistema que utilizará un usuario en particular. Por ejemplo si pensamos que el modelo pertenece a una fábrica entonces el usuario "Almacen" sólo interactúa con la

parte de la información que tiene relación con sus funciones y no tiene por qué tener acceso a otro tipo de información, como por ejemplo la nómina. Entonces el usuario Almacen interactuará con un modelo externo diseñado para él.

Como se indica en la figura 1, cada modelo externo se comunica en la base física por medio del bloque de comunicaciones y el bloque de especificaciones.

La interacción con el modelo se logra por medio de llamadas a subrutinas desde un lenguaje anfitrión. Esto es, se hace un programa en COBOL o PL/1 y este contiene subrutinas que acce san a la información.

2.- El Modelo IMS. A continuación se estudiará con más detalle el modelo IMS, que como se mencionó antes es un conjunto de bases físicas "PDB's".

Una PDB es una lista ordenada de elementos, de un mismo tipo, a cada elemento se le llama segmento. Al conjunto de todos los segmentos se le llama registro de la base física.

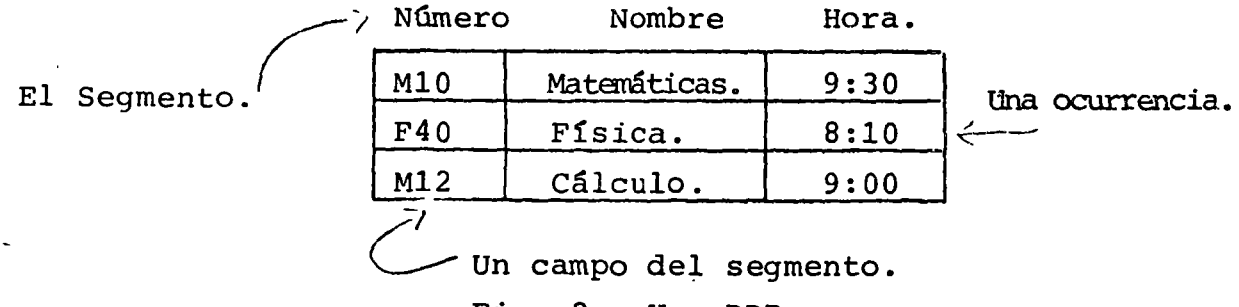


Fig. 2: Una PDB.

Como ejemplo se estudirará un sistema que describe el programa educativo de una compañía como sigue:

Se ofrecen cursos de entrenamiento en diferentes lugares. - Los empleados de la compañía pueden ser tanto estudiantes como profesores. Los detalles que contendrá cada uno de los registros de la PDB son:

- Cada curso tiene un número único, nombre y descripción.
- Los cursos pueden tener prerequisitos y cada prerequisito contiene el número y el nombre.
- Cada lugar de oferta tiene la fecha, localización, formato y todos los detalles relativos a los alumnos y profesores que participan.
- Cada profesor tiene su número y nombre.
- Cada alumno tiene su número, nombre y calificación obtenida.

La figura 3 representa al registro de la base

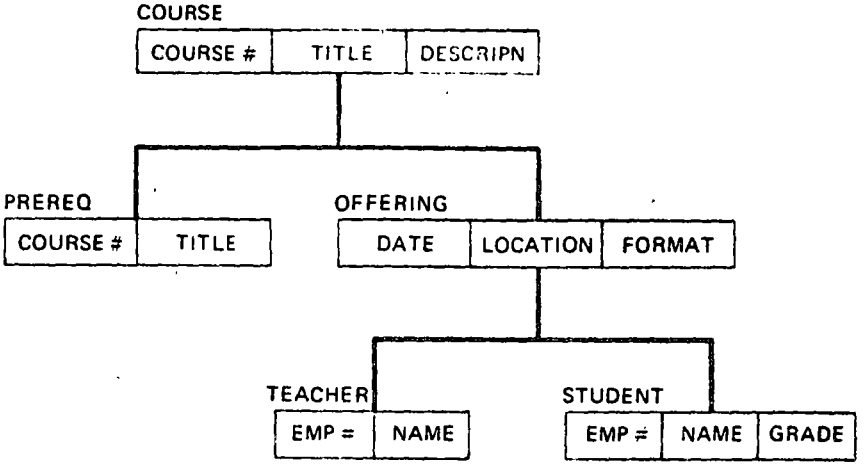


Fig. 3. El registro de la base.

Aquí se observaran 5 tipos de segmento.

1.- Curso.

2.- Prerequisito.

3.- Oferta.

4.- Profesor.

5.- Estudiante.

Al Curso se le llama "Raiz" y es el padre de los segmentos, "Prerequisito" y "Oferta".

Este árbol genera una dependencia entre los diferentes tipos de segmentos. Así, ESTUDIANTE depende de "Oferta" y "Oferta" depende de "Curso". A estudiante se le llama hijo de oferta dado que es dependiente. Nótese que la raiz no es dependiente de ningún segmento.

Una vez aceptada la existencia del registro y su dependencia, o sea el ordenamiento de los segmentos de la base física se introducen los datos propiamente dicho: Qué cursos existen, donde se ofrecen, quienes son los profesores y estudiantes. A estos datos se les conoce como ocurrencias de los segmentos del PDBR. La figura 4 representa un posible conjunto de dichos segmentos.

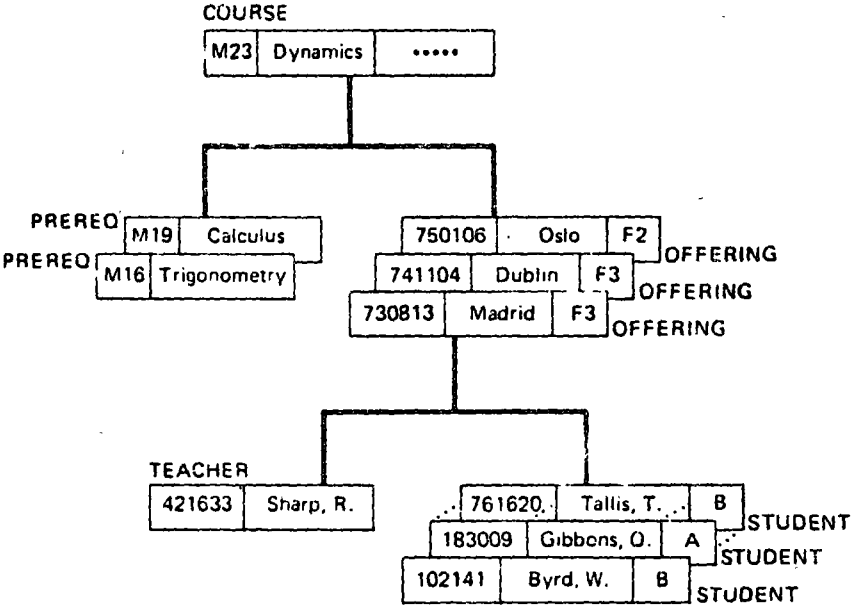


Fig. 4: Un ejemplo de las ocurrencias o datos de la base.

Aquí se observa que sólo existe un curso: Dynamics y se - ofrece en Oslo, Dublin y Madrid. El Profesor en Madrid es Sharp, y los estudiantes son Tallis, Gibbons y Byrd.

También se observa que ni en Oslo ni en Dublin, hay estudiantes o profesores.

Características del PDBR.

- Sólo tiene una raíz.
- La raíz puede tener cualquier número de tipo de hijos o descendientes.
- A su vez cada hijo puede tener cualquier número de tipo de hijos, etc.
- Para cada ocurrencia puede haber cualquier número de ocurrencias en sus hijos.
- Ninguna ocurrencia puede existir sin ocurrencia padre.

Es muy importante establecer la diferencia entre los conceptos tipo del segmento y ocurrencia. El segmento describe un tipo de información, mientras que la ocurrencia es un dato. En el ejemplo el segmento es Curso y la ocurrencia es M23, Dynamics,..... Dado que esta diferencia es siempre clara en el futuro se omitirá la palabra tipo y ocurrencia al referirnos al segmento.

La base física se define por el mapeo o almacenamiento inter

no y por su descripción. En el IMS la descripción esta dada por varias instrucciones del Ensamblador. Por el momento se omitira lo relacionado al mapeo y se tratará de presentar las instrucciones que describen a la base física. A esta descripción también se le conoce como "esquema conceptual".

```
1  DBD      NAME=EDUCPDBD
2  SEGM     NAME=COURSE,BYTES=256
3  FIELD    NAME=(COURSE#,SEQ),BYTES=3,START=1
4  FIELD    NAME=TITLE,BYTES=33,START=4
5  FIELD    NAME=DESCRIPN,BYTES=220,START=37
6  SEGM     NAME=PREREQ,PARENT=COURSE,BYTES=36
7  FIELD    NAME=(COURSE#,SEQ),BYTES=3,START=1
8  FIELD    NAME=TITLE,BYTES=33,START=4
9  SEGM     NAME=OFFERING,PARENT=COURSE,BYTES=20
10 FIELD    NAME=(DATE,SEQ,M),BYTES=6,START=1
11 FIELD    NAME=LOCATION,BYTES=12,START=7
12 FIELD    NAME=FORMAT,BYTES=2,START=19
13 SEGM     NAME=TEACHER,PARENT=OFFERING,BYTES=24
14 FIELD    NAME=(EMP#,SEQ),BYTES=6,START=1
15 FIELD    NAME=NAME,BYTES=18,START=7
16 SEGM     NAME=STUDENT,PARENT=OFFERING,BYTES=25
17 FIELD    NAME=(EMP#,SEQ),BYTES=6,START=1
18 FIELD    NAME=NAME,BYTES=18,START=7
19 FIELD    NAME=GRADE,BYTES=1,START=25
```

Fig. 5: Esquema o descripción de la base educacional.

La explicación de estas instrucciones está dada a continuación.

1.- La base se llama EDUCPDBD.

2.- La raiz se llama CURSO y tiene 256 caracteres.

3.- Los campos de Curso son:

CURSO y tiene 3 catacteres, además este campo comienza en el primer carácter de CURSO. Es importante mencionar en qué carácter comienza, dado que está permitida la superposición de campos. La palabra SEQ indica que este campo se usará como llave del registro.

4,5.- La descripción de estos campos es similar a la anterior.

6.- El segmento PREREQUISITO es hijo de CURSO y tiene 36 caracteres.

10.- En este campo aparece una M que indica que el valor de la llave DATE puede ser múltiple, esto es puede haber dos ocurrencias con la misma fecha.

Para finalizar con esta explicación se supone que el campo llave (SEQ) siempre es el primer campo de cada segmento y que es único, a menos que aparezca una M.

Ahora está bastante clara la razón del nombre jerárquico, y a continuación se dará una razón mucho más poderosa para este nombre.

A cada ocurrencia se le asigna un número llamado "llave je-

rárquica" que define un orden único de cada ocurrencia dentro de la base física. Primero se establece un orden de cada segmento dentro del registro como sigue:

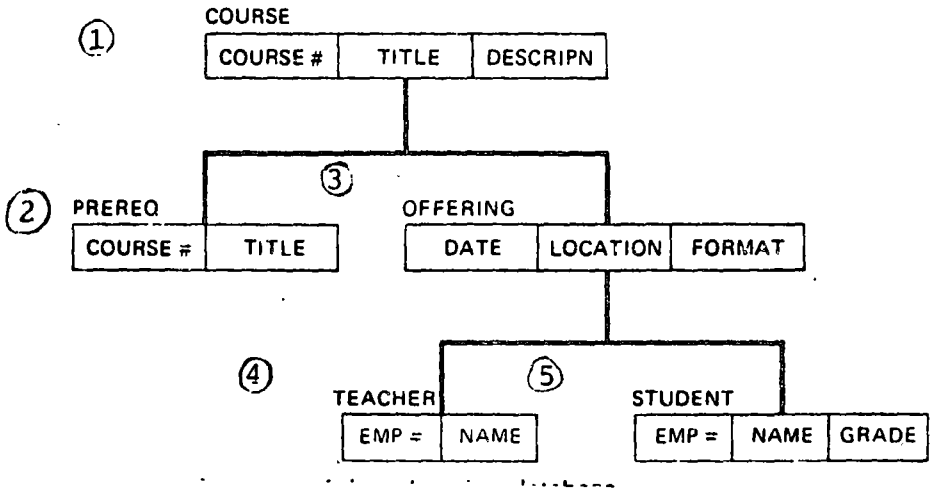


Fig. 6.

La llave jerárquica se construye escribiendo el orden y luego el valor llave (SEQ) antes descrito. Supóngase que se quiere la llave jerárquica del estudiante Byrd, entonces:

1.- Byrd tiene como llave SEQ 102141 y orden 5.

2.- El padre de Byrd, es Madrid y tiene como secuencia - 730813 y orden 3.

3.- El padre de Madrid es Dynamics y tiene como secuencia M23 y orden 1.

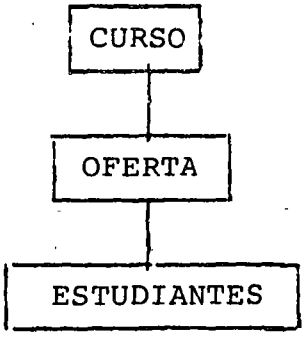
Por lo tanto la llave es

1/M23/3/73081/5/102141

O sea:

1M233730815102141

El Modelo Externo.- Anteriormente se definió como modelo externo a un conjunto de bases lógicas, y una base lógica es un subconjunto correspondiente a una base física; este subconjunto está orientado a un usuario en particular. La base lógica está descrita por su registro, que es un subconjunto 1 del registro que define a la base física.



MODELO DE LA BASE.

Como ejemplo de un modelo externo supóngase que un usuario de la base educacional sólo utilizará datos relativos a CURSOS, OFERTA y ESTUDIANTES, entonces el registro de su base será.

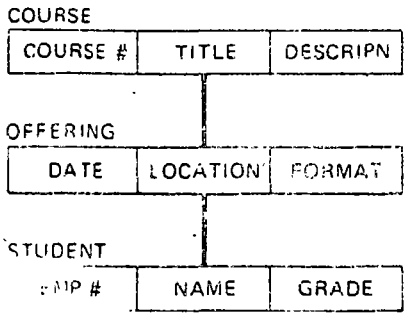


Fig. 8

A los segmentos que aparecen en el registro se les llama "sensitivos" y el usuario que utilice este modelo solamente estará conciente de la existencia de los segmentos sensitivos. Generalmente no se permite eliminar o agregar segmentos dado que cambiaría la jerarquía de la base, excepto cuando al agregar un segmento se hace en el último nivel de alguna "rama" de la jerarquía y no afecta la jerarquía de los otros segmentos.

Cada base lógica se define por medio de un bloque de comunicaciones que establece la interfaz entre el usuario y el sistema. También este bloque de comunicaciones se escribe en Ensamblador.

Hasta este momento hay dos cosas escritas en ensamblador. Por un lado está la descripción de la base y por otro está el bloque de comunicaciones para cada base lógica, o sea para cada tipo de usuario. A continuación está el bloque de la base lógica que se acaba de definir.

```
1  PCB      TYPE = DB, DBNAME = EDUCPDBD, KEYLEN = 15
2  SENSEG   NAME = CURSO, PROCOPT = G
3  SENSEG   NAME = OFERTA, PADRE = CURSO, PROCOPT = G
4  SENSEG   NAME = ESTUDIANTE, PADRE = OFERTA, PROCOPT = G
```

El Bloque de Comunicación de la base lógica.

Fig. 9

1.- Aquí se especifica que es un bloque de comunicaciones de la base educacional EDUCPVSD y la llave tiene 15 caracteres. Esta área para la llave es necesario que se defina puesto que ahí estará almacenada la dirección de la última ocurrencia solicitada, esta llave no es la misma que la llave jerárquica puesto que no contiene los valores correspondientes al orden de los segmentos.

Llave M 23 73081 3 102141

Llave Jerárquica 1M23 373081 1 5 102141

2.,3.- Aquí se especifican los segmentos sensitivos y las opciones de procesamiento, PROCOPT. Dependiendo de los valores asociados a esta variable se permitirá accesar, reemplazar, modificar o eliminar ocurrencias en este segmento.

G: Acceso.

I: Inserción.

R: Reemplazo.

D: Eliminación.

Finalmente, en caso de que los segmentos sensibles no esten unidos (hijos de o padres de) entre sí, sino que a través de otros segmentos, pero se desea que estos no aparezcan en la base lógica entonces de los de la opción K. Por ejemplo: si la base lógica consiste sólo de los segmentos CURSO y ESTUDIANTE, entonces el segmento intermedio OFERTA tendrá PROCOPT = K.

El sublenguaje IMS.- En esta sección se describirán las instrucciones que efectúan operaciones sobre los datos. Hasta ahora sólo se había considerado las definiciones de la base. Este sublenguaje esta sumergido en un lenguaje anfitrión, como lo puede ser PL/1 o COBOL.

A continuación está la parte inicial del programa, escrita en PL/1 que manipula la base CURSO-OFERTA-ESTUDIANTE, definida anteriormente.

```
DLITPLI: PROCEDURE(COSPCB_ADDR) OPTIONS(MAIN);
         .
         .
         .
DECLARE 1 COSPCB     BASED(COSPCB_ADDR),
        2 DBDNAME    CHARACTER(8),
        2 SEGLEVEL   CHARACTER(2),
        2 STATUS     CHARACTER(2),
        2 PROCOPT    CHARACTER(4),
        2 RESERVED   FIXED BINARY(31),
        2 SEGNAME    CHARACTER(8),
        2 KEYFBLEN   FIXED BINARY(31),
        2 #SENSEGS   FIXED BINARY(31),
        2 KEYFBAREA CHARACTER(15);
```

Fig. 10.

El nombre del procedimiento es fijo (DLITPLI) y todos los demás nombres son variables definidos por el usuario.

La expresión entre paréntesis representa una lista de los apuntadores a cada una de las bases utilizadas. En este caso sólo hay un valor dado que sólo existe en base.

La instrucción DECLARE, representa la estructura que define las características de la base que se usará.

1.- La base se llamará COSPCB y tendrá el apuntador COSPCB-ADDR.

2.- DBDNAME especifica el número de caracteres de la base física usada es decir EDUCPDBD.

SEGLEVEL contendrá el valor de orden del segmento accesado.

STATUS especifica si hubo o no éxito en el acceso a algún dato.

PROCOPT es la opción antes mencionada.

SEGNAME es el nombre del último segmento accesado..

RESERVED es para uso del sistema.

KEYFBLEN es la longitud de segmentos de la llave.

SENSEGS es el número de segmentos sensitivos.

KEYFBAREA tiene el valor de la llave.

Las operaciones permitidas son:

GU Obtener el único elemento.
GN Obtener el siguiente elemento.
GNP Obtener el siguiente elemento bajo el mismo padre.

GHU, GHN, GHNP } Igual que las anteriores, pero la siguiente instrucción será renueva (DLET) o reemplaza (REPL).

ISRT Inserta.
PLET Remueve.
REPL Reemplaza.

Un ejemple de una instrucción es:

```
GU  CURSO  (TITLE = DINAMICA)
    OFERTA (FORMATO = F1 o FORMATO = F3)
    ESTUDIANTE (CALIF = A).
```

que representa la pregunta obtener un estudiante del curso - DINAMICA que haya participado en un grupo con formato F1 o F3 y obteniendo la calificación A.

La respuesta a esta pregunta estará dada en las variables definidas en el DEFINE, esto es tendremos la llave de la ocurrencia en KEYFBAREA, en STATUS habrá un blanco que implica la existencia del dato, etc.

Las expresiones entre paréntesis se llaman argumentos de búsqueda y son expresiones booleanas típicas.

A continuación se presentan varias preguntas y su representación en el sublenguaje. Estas preguntas están referidas a la base lógica.

CURSO

OFERTA

ESTUD.

1.- Encuentre la primera oferta con localización en Estocolmo.

```
GU    CURSO
      OFERTA (Localización=Estocolmo).
```

La respuesta será la primera ocurrencia que tenga como localización ESTOCOLMO.

2.- Encuentre todos los estudiantes en Estocolmo.

```
GU      CURSO
        OFERTA (LOC=ESTOCOLMO)
        ESTUDIANTE

NS: GN   ESTUDIANTE
    GOTO NS.
```

La instrucción GU encuentra al primer estudiante de Estocolmo en la lista, después GN va encontrando todos los siguientes estudiantes que satisfagan el predicado LOCALIZACION = ESTOCOLMO.

3.- Igual que en el anterior pero estudiantes con calificación A.

```
GU    CURSO
      OFERTA (LOC=ESTOCOLMO)
      ESTUDIANTE (CALIF=A)

NSA:GU   ESTUDIANTE (CALIF=A)
    GOTO NSA.
```

4.- Todos los estudiantes con calificación A;

```
          GU   CURSO
NX        GN   ESTUDIANTE (CALIF=A)
          GOTO NX.
```

Estructura de almacenamiento. En esta sección se presentan las diferentes formas de almacenar la información físicamente. - El sistema IMS tiene cuatro formas de hacerlo y se llaman:

HSAM Es el método de acceso secuencial

HISAM Es el método secuencial indicado (Index sequential)

HDAM Es el método de acceso directo.

HIDAM Es el método indicado directo (Index direct access).

Antes de explicar cada uno de estos métodos de almacenamiento y por consiguiente de acceso de la información, es importante aclarar que el modelo externo no varía por la utilización de estos métodos y por lo tanto el usuario siempre ve el mismo modelo, es la implentación la que varía. También el sistema será más o menos eficiente, dependiendo de la selección de los métodos de acceso.

HSAM.- Es la representación sucesiva de la información. Un dato siempre está después del anterior. Se puede visualizar - este modelo como la información almacenada en una cinta magnética. Los registros de almacenamiento son de tamaño fijo y por lo tanto

un segmento siempre estará contenido en un registro lógico. Nóte se que la palabra registro tiene un significado diferente al de secciones anteriores. Aquí significa el tamaño de un grupo de palabras que pueden ser leídas o escritas en una sola operación. - Es similar al concepto de segmento pero el registro está dado en parte por las características del sistema de cómputo que se utilice mientras que el segmento está definido por el diseñador de la base. Es deseable que el tamaño de ambos sea "igual" para no desperdiciar espacio. La figura 11 representa la forma de almacenar parte de los datos de la base educacional por el método HSAM.

COURSE	PREREQ	PREREQ	OFFERING	TEACHER	STUDENT	STUDENT	
M23	M16	M19	730813	421633	102141	193009	..

STUDENT	OFFERING	OFFERING	COURSE	PREREQ	OFFERING	TEACHER	TEACHER	
761620	741104	750106	M27	L02	740602	421633	502417	...

Part of the education database (HSAM)

Fig. 11: Parte de la base educacional (HSAM).

Una vez que la base está creada es imposible insertar un - nuevo segmento dado que no hay espacio disponible. Por esto sólo operaciones de lectura podrán efectuarse.

HISAM.- En este método de acceso la información está - guardada de acuerdo a la estructura jerárquica a la que pertenece. Informalmente se puede decir que cada raíz o padre junto con

su descendencia está almacenado en un lugar diferente. Esto es - cierto en el momento de cargar la información, y no es absolutamente cierto después de haber efectuado operaciones de inserción o remoción.

El almacenamiento se efectuará en dos archivos el ISAM y el OSAM que tienen tamaño de registro fijo y superior al tamaño de los segmentos. Así un registro contendrá uno o varios segmentos La estructura de uno de estos registros se ve en la figura 12.

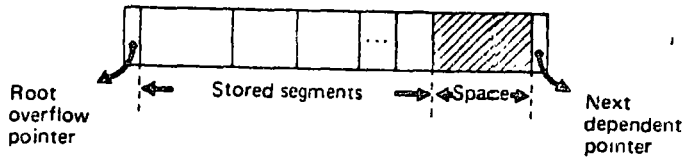


Fig. 12: Estructura de registros.

En el primer archivo (ISAM) estarán almacenadas las raíces y cada una de ellas apuntará a un registro del otro archivo (OSAM) que contendrá sus segmentos dependientes. El archivo o ISAM es fijo y nunca aumentará a pesar de haber insertado nuevas raíces.

La figura 13 representa la información en ambos archivos en ISAM este CURSO y su PRERQUISITO y apuntan a los registros de OSAM donde están todos los segmentos dependientes.

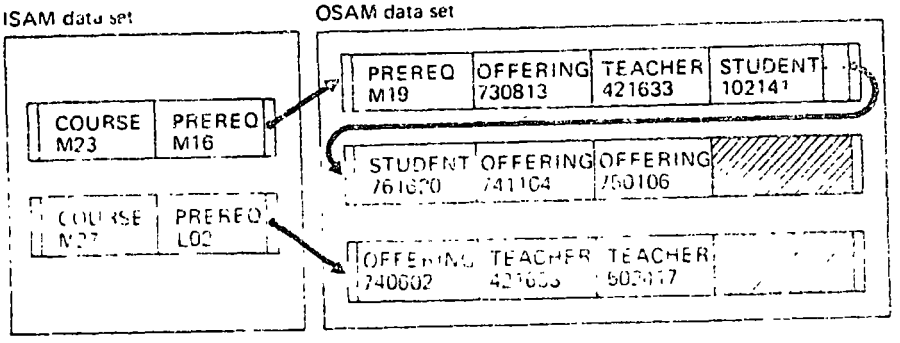
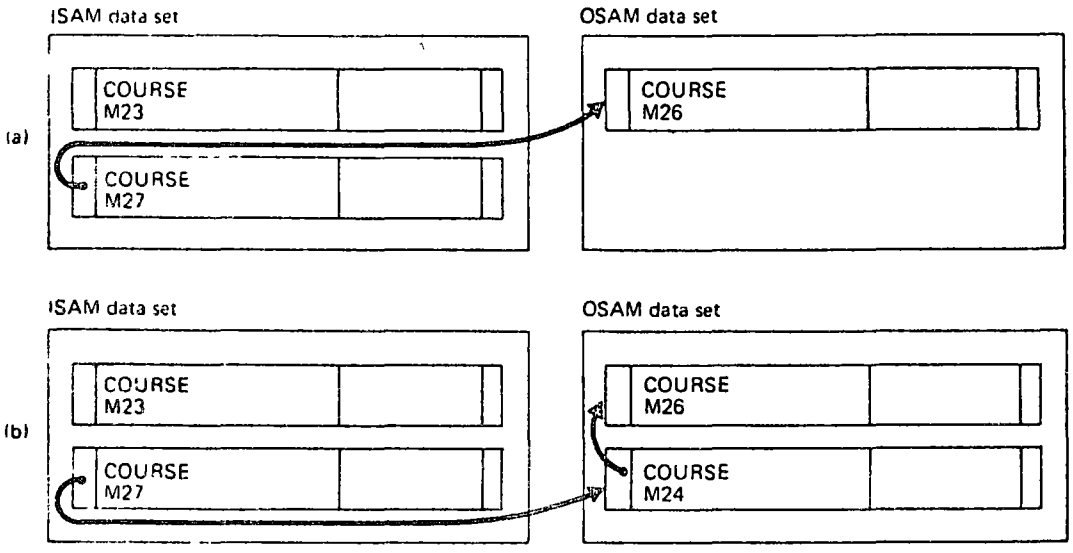


Fig. 13: Parte de la base educacional (HISAM).

Si se quiere insertar más información hay dos posibilidades: que sea relativa a las raíces contenidas en ISAM o que sea dependiente y esté asociada al archivo OSAM. Primero se considerará el caso de la información dependiente: ésta se inserta en el lugar apropiado en OSAM. Generalmente esta inserción implica un reordenamiento de la información dentro del registro. Pero si el registro está lleno, entonces se crea un nuevo registro que estará unido vía el apuntador dependiente.

Cuando se quiere insertar una raíz, ésto también se hace en el archivo OSAM y está representado en la figura 14.



Root segment insertion examples (HISAM, using ISAM/OSAM)

Fig. 14: Inserción en HISAM.

En la parte (a) se ejemplifica la inserción de una nueva raíz CURSO M26 y en la parte (b) se ejemplifica la inserción de dos nuevas raices: CURSO M26 y CURSO M24. Nótese que el apuntador en ellas proviene de la raiz inmediatamente superior en jerarquía a la raiz por insertar.

Hay otros variantes del método HISAM como lo son el permitir que ambos archivos ISAM y OSAM aumenten de tamaño o que cada uno de ellos sólo tenga un tipo de información.

Apuntadores: El uso de apuntadores esta más generalizado en los métodos de acceso directo HDAM y HIDAM y son para unir segmentos consecutivamente. A diferencia de los apuntadores del archivo

OSAM en que había un apuntador por registro, aquí habrá un apuntador por cada segmento y se usarán para representar la jerarquía de los datos. Las dos formas más comunes del uso de estos apuntadores están representadas en las figuras 15 y 16 en la primera y de acuerdo con el orden se copia la jerarquía de la base. Esto es, la raíz es curso M23 y apunta al segmento, tipo de segmento (orden = 2) en cada tipo de segmento se unen todas las ocurrencias, después se une el segmento tipo de segmento (orden=3) y así sucesivamente. A estos se les llama "apuntadores jerárquicos". Una variación común es cuando los apuntadores estan en ambas direcciones. Imaginemos flechas dobles.

La segunda figura representa los apuntadores gemelos y consiste en que cada padre apunta a todos sus hijos.

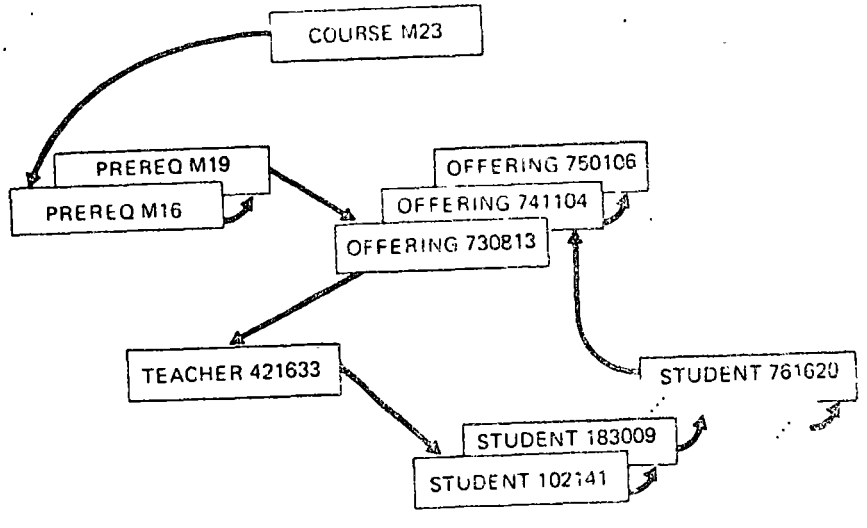


Fig. 15.

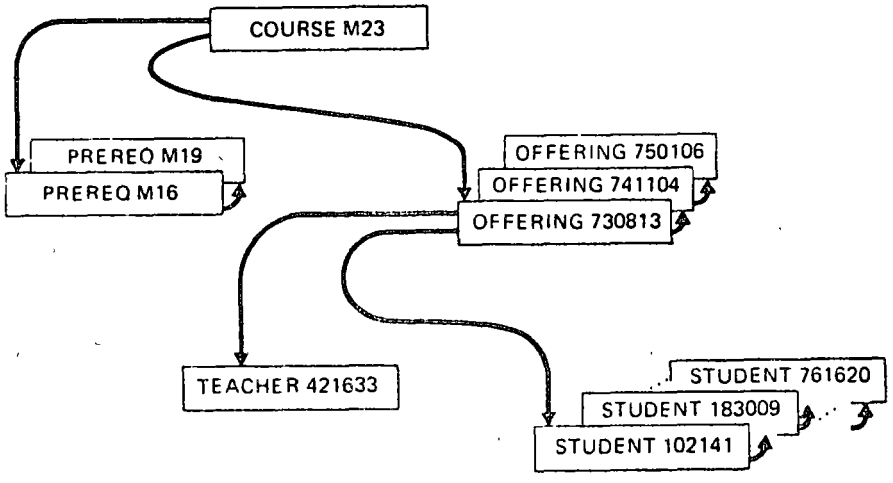


Fig. 16

HDAM.- Es el método de acceso directo a la raiz vía "hashing". En la forma más simple se puede pensar en archivo dividido en registros fijos y arreglado en dos secciones una para las raices y otra para el desborde. Tal y como se indica en la figura 17.

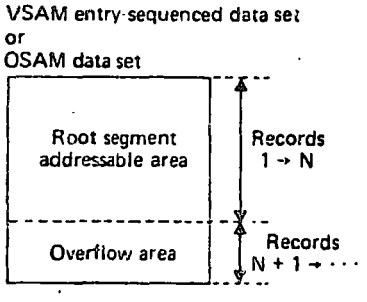


Fig. 17 Estructura HIDAM.

Una diferencia de este método con los anteriores es que la información se puede cargar en desorden ya que cada raíz esta lo-

calizada por su llave, sin embargo, no se permite que los segmentos subordinados a la raíz se carguen en desorden.

Ejemplo: Supóngase que la subrutina de "Hashing" es el cociente más uno de dividir el valor de la raíz entre 100.

- Insertar Raíz 322
  Entonces 322/100 + 1 = 23, se inserta en la localidad 23

- Inserta Raíz 522
  Entonces 522/100 + 1 = 23 y suponiendo que hay lugar en ese registro entonces se inserta en la localidad 23.

- Inserta Raíz 222
  Se insertará en el siguiente lugar disponible, que es la localidad 25.

- Inserta Raíz 422
  Dado que toda el área ya se encuentra llena se inserta en la localidad 144 del área de desborde.

La cadena de almacenamiento para estas colisiones está en la figura 18.

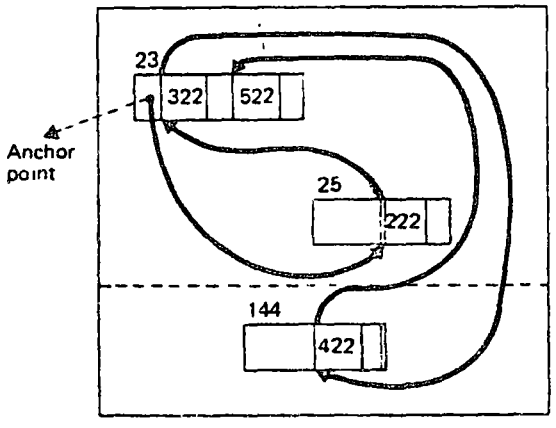


Fig. 18: Ejemplo de la colisión en 23.

HIDAM.- Por último este método tiene un indicamiento a la raiz y otro a los segmentos subordinados como en HISAM pero los apuntadores estan controlados por el sistema y son similares a HDAM.

III.- MODELO DE REDES O RETICULAR

El último enfoque de modelos de datos es el enfoque reticular.

Lo que aquí explicaremos fue propuesto por DBTG "Data Base Task Group".

Este grupo recomienda un lenguaje y las especificaciones y organización de datos para dar de alta y trabajar con una base de datos teniendo como lenguaje huésped el lenguaje COBOL.

Diseño de la base de datos.- Recordemos que el diseño de toda base de datos tiene 2 puntos críticos, el diseño de la información y el diseño de la estructura de datos compatible con un DBMS.

Primero se discutirá el diseño de la información y después el diseño de la estructura de datos.

Acto seguido hablaremos de las estructuras en forma de diagramas y después hablaremos de algunas partes en especial de el modelo reticular.

El modelo de datos que utilizaremos será un modelo presidencial.

Diseño de información.- Como nosotros ya sambemos, la información contenida dentro de una base de datos debe estructurarse respetándose las relaciones que se presentan en el modelo real, es decir, aquellas relaciones que interesan a el usuario.

Un diseñador de base de datos tiene como primera tarea identificar todas las entidades relevantes que son de interés para la organización.

Una vez que el diseñador ha identificado todas las entidades o atributos relevantes, deberá de agruparlos en conjuntos comunes a los que llamará relaciones.

Una vez hecho ésto, deberá de realizar las siguientes funciones:

- Determinar aquellos atributos que identifican a las entidades. Es decir, encontrará los atributos llave.
- Determinará todos aquellos atributos restantes que pertenecen a una relación y checará que tomen un solo valor por cada ocurrencia. Es decir, checará que los valores de la relación sean atómicos.
- En el caso de que un atributo tenga varios valores, deberá checar la posible generación de una nueva identidad.
- En el caso de relaciones uno a muchos entre identidades distintas, deberá de colocar el identificador de la "uno" con los "muchos".
- En el caso de relaciones muchos-muchos, deberá de considerar el unir estas 2 relaciones en una sola y que tendrá como identificador los identificadores unidos de las 2 relaciones.

Estos son los pasos fundamentales que un diseñador deberá de tomar en cuenta para la estructura de su información.

Estructura de datos.- Una vez diseñada la información es posible diseñar su estructura.

Esta parte nuevamente contempla la elección de una estructura física (Index secuencial, Random, etc.) y mucho de la elección dependerá de la habilidad o experiencia de el diseñador.

Existen unos diagramas bajo el enfoque reticular con los cuales podemos visualizar las distintas relaciones.

El enfoque reticular es aquel donde las distintas relaciones se unen entre sí formando redes de información, de tal suerte que para encontrar un dato específico, tal parece que "navegamos" a través de toda la base.

Con el uso de diagramas ésto se verá más claro.

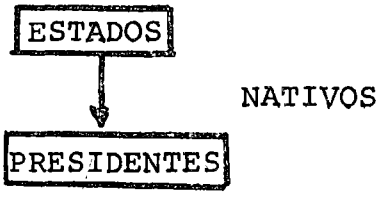
Diagramas de estructuras de datos.- Antes de discutirlo hagamos las siguientes convenciones:

- Una relación sería equivalente a una identidad o tipo de registro.
- Utilizaremos la palabra registro para nombrar la ocurrencia de una identidad.
- Llamaremos tipo de conjunto (Set Type) a la relación existente entre 2 identidades.

Los diagramas se componen de 2 símbolos:

Un rectángulo para encerrar el nombre de una identidad

Una flecha para relacionar 2 identidades.



A la identidad de la que parte la flecha le llamaremos propia (ESTADOS).

A la identidad que recibe la flecha le llamaremos miembro (PRESIDENTES).

En el ejemplo anterior, nativos sería el tipo de conjunto.

Un conjunto de ocurrencias es una ocurrencia de un tipo de registro propio con cero o más ocurrencia de tipos de registro miembros.

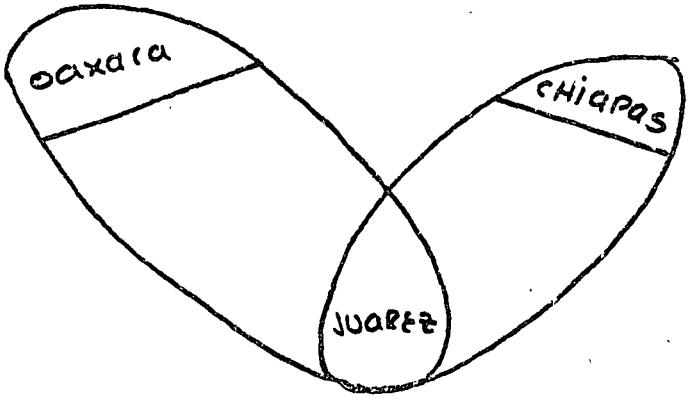
En cada conjunto de ocurrencias las siguientes relaciones existen:

- Dado un registro propio, es posible procesar todos los registros miembros asociados a ese conjunto de ocurrencias.
- Dado un registro miembro, es posible procesar el registro propio asociado a ese conjunto de ocurrencias.
- Dado un registro miembro, es posible procesar todos los demás registros miembros asociados a ese conjunto de ocurrencias.

Cualquier implementación que satisfaga estas tres reglas, será una implementación válida de el concepto de tipos de conjunto.

Imaginemos los siguientes conjuntos de ocurrencias.

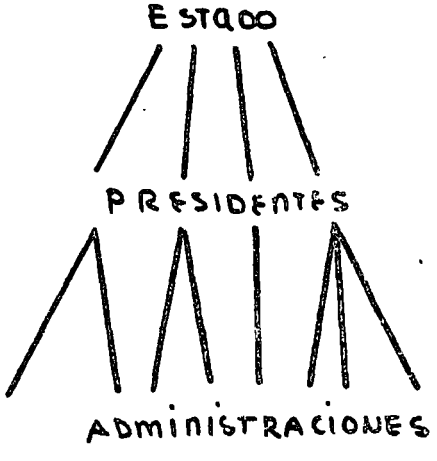
Puede suceder que un conjunto de ocurrencias tenga un registro propio pero no registros miembros. A este conjunto le llamaremos vacío.



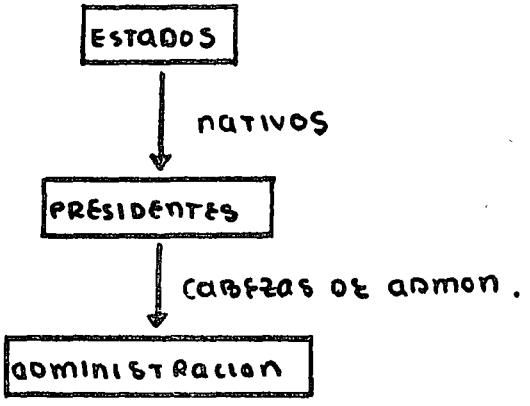
Un registro miembro no puede pertenecer a 2 conjuntos de ocu-- rrencias a la vez.

Existen muchas formas de representar diagramas,nosotros utilizaremos los anillos(modelo reticular).

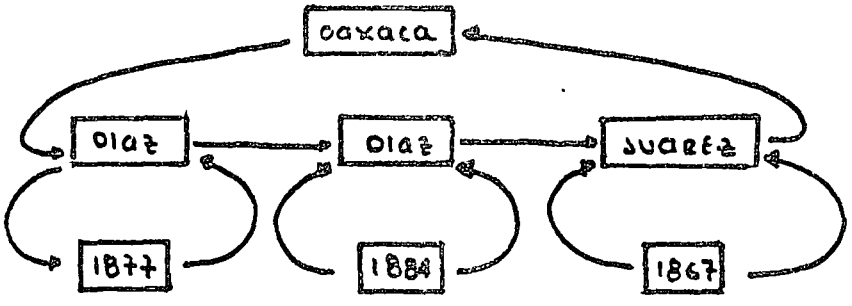
Representación de Jerarquías.-Una jerarquía represen ta relaciones 1 a n. Por ejemplo:



Esta jerarquía la representaremos asi:



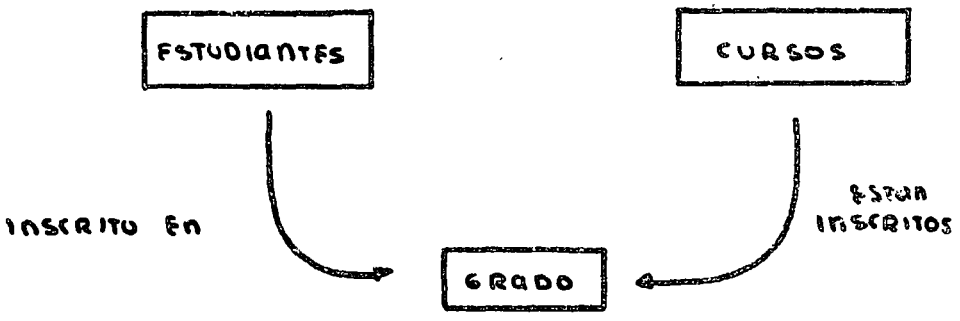
Una ocurrencia de esta jerarquía la representaremos asi:



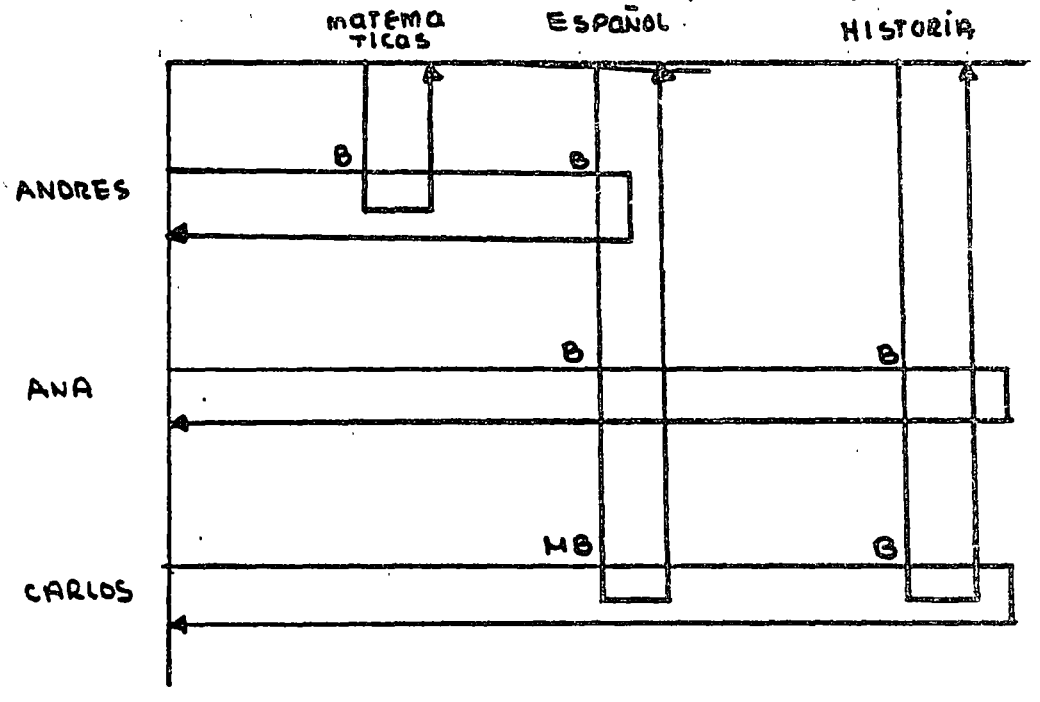
Representación de relaciónes n a n.-Un ejemplo típico de estas relaciónes son los estudiantes y los cursos como 2 identidades Un estudiante puede tomar muchos cursos y un curso puede ser tomado por muchos estudiantes.

Con estas 2 relaciones no es posible construir un diagrama,pues no se cumplirian las 3 reglas.

Usualmente,para solucionar este problema se crea una nueva identidad que servira como identidad miembro de cada una de las otras identidades.En nuestro caso,la identidad calificaciónes puede servirnos para este fin.

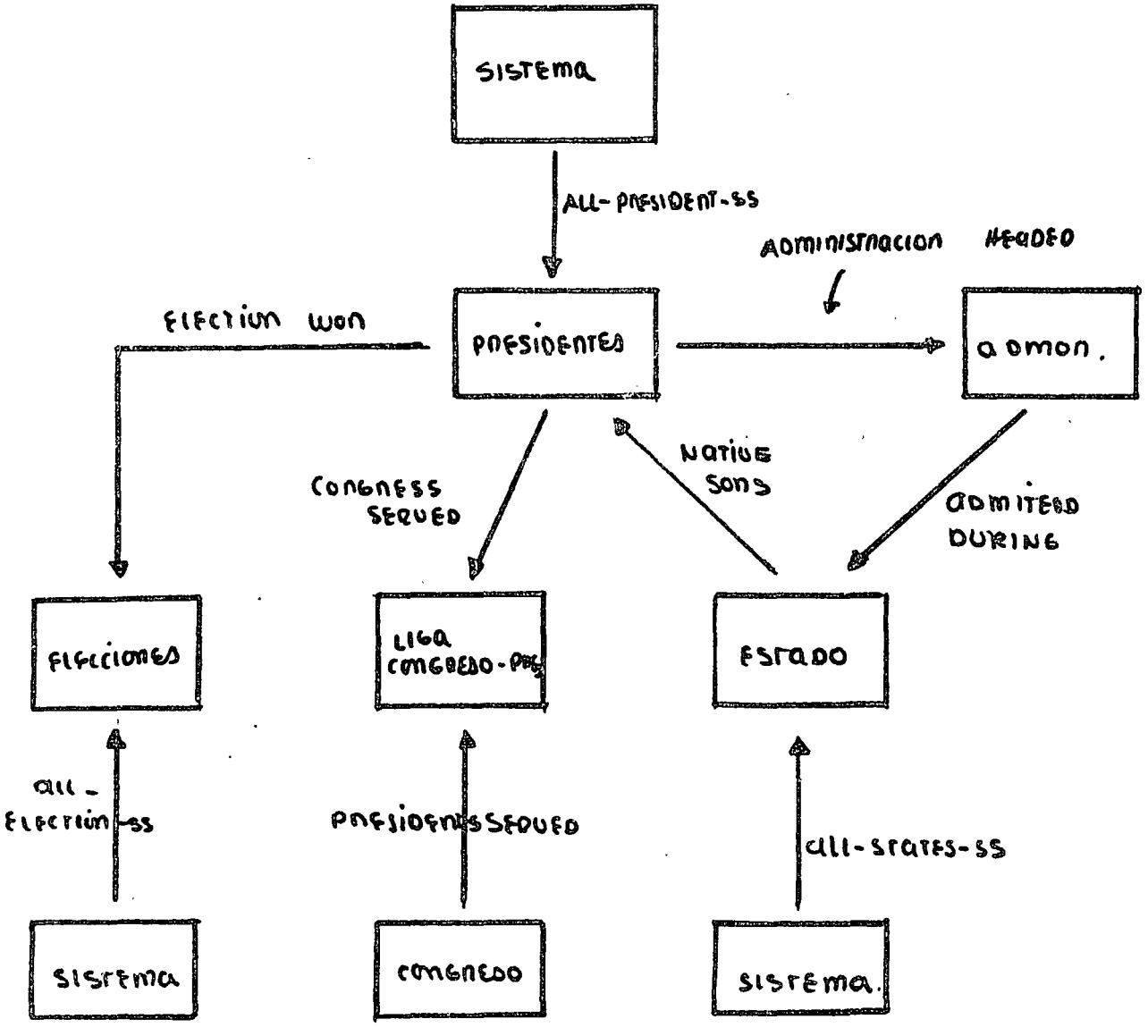


Una ocurrencia de esta relación sería:



Debemos de notar que estamos definiendo redes donde un registro puede ser tanto miembro como propio.

Estas son las notaciones más comunes para expresar relaciónes. Para ejemplificar el uso de estos diagramas, construiremos una base de datos de todos los presidentes de México.



* los conjuntos se han nombrado en Inglés para hacerlos compatibles con los sigiuentes ejemplos y ejercicios.

Dentro de esta gráfica podemos observar que:

-existen relaciónes n-n tal como las identidades presidentes y congreso las cuales estan ligadas por la identidad ligas congreso-pres.

-existen 3 registros propios a los que hemos llamados sis-- temas. Estos son conjuntos singulares y los discutiremos más adelante.

Nuestro para implementar esta base es definir el esquema. Esto lo haremos por medio de DDL.

Este lenguaje consiste de 4 secciones:

-Una cláusula introductoria.-Damos nombre a la Base y enunciamos algúnas de sus características.

-Una cláusula de áreas.-las áreas son los lugares donde guardaremos nuestros datos. Por cada área podemos definir una estructura,medios de seguridad etc, dependiendo de los datos que se guarden en ella.

-Cláusulas de registros.-sirven para definir las identidades

-cláusulas de conjuntos.-sirven para definir los tipos de conjuntos.

En nuestra Base de Datos esto se haría de la siguien te forma:

-cláusula introductoria.

```
SCHEMA NAME IS PRESIDENTIAL
PRIVACY LOOK FOR COPY IS COPY PASWORD
PRIVACY LOOK FOR ALTER IS PROCEDURE CHECK-AUTHORIZAT
                                                ION.
```

Generalmente son 3 las cosas que podemos hacer con un esquema:

-alter.-hacerle modificaciones.

-copy.-copiar el esquema a nuestro sub-esquema.

-display.-imprimir o desplegar el esquema.

Por cada una de estas acciones existen llaves de seguridad,las cuales deben de cumplirse.

-cláusula de áreas.

```
AREA NAME IS PRESIDENTIAL-AREA
ON OPEN FOR UPDATE CALL UPDATED-CHECK
```

Con esto definimimos el área y establecemos que sera actualizada por UPDATED-CHECK.

-Cláusula de registros.

```
RECORD NAME IS PRESIDENT
LOCATION MODE IS CALC
  USING LAST-NAME,FIRST-NAME,DUPLICATES ARE NOT
  ALLOWED WITHIN PRESIDENTIAL-AREA
02 PRES-NAME
   03 LAST NAME  PIC "A(10)"
   03 FIRST NAME  PIC  "A(10)"
02 PRES-DATE-OF-BIRTH
   ..........................
```

Con estas declaraciones registramos cada una de nuestras identidades.

LOCATION MODE indica la forma como seran accesados esos regis tros. CALC significa hashing.
LAST-NAME y FIRST-NAME son las llaves.No se permiten llaves duplicadas.
Las demás son especificaciones de campos.

Debera de exisitir una cláusula distinta por cada identidad.

-cláusula de conjuntos.

Antes de indicar como se declaran los conjuntos tipos hablaremos de el conjunto singular.

El conjunto singular es un conjunto donde el registro propio es el sistema. Como existe solamente una ocurren - cia de este conjunto se le llama singular.

```
SET NAME IS ALL-PRESIDENTS-SS
OWNER IS SYSTEM
ORDER IS PERMANENT SORTED BY DEFINED KEYS
 DUPLICATES ARE LAST
MEMBER IS PRESIDENT MANDATORY AUTOMATIC
 KEY IS ASCENDING LAST-NAME IN PRES-NAME
 SET SELECTION IS THRU ALL-PRESIDENTS-SS
 OWNER IDENTIFIED BY SYSTEM
```

ORDER IS indica la forma como seran presentados los registr- os miembros a el usuario.
PERMANENT indica que el usuario no podra cambiar este ordena miento.

DUPLICATES ARE LAST indica que las ocurrencias con llaves du-- plicadas se almacenaran despues de la ultima ocurrencia "nor-- mal".

MANDATORY AUTOMATIC indica que tan pronto ocurra una ocurren-- cia de los registros miembros,esta sera incluida en el conjun- to.

SET SELECTION permite almacenar todas las ocurrencias de los registros miembros en el conjunto de ocurrencias.

En el caso de identidades ligas,como la única fun - ción de estas es la de servir precisamente como ligas,su or - den no nos importa.

```
SET NAME IS CONGRESS-SERVED
 OWNER IS PRESIDENT
 ORDER IS PERMANENT INMATERIAL
 MEMBER IS CONGRESS-PRES-LINK MANDATORY AUTOMATIC
  SET SELECTION IS THRU CONGRESS-SERVED
  OWNER IDENTIFIED BY CALC-KEY
```

Una vez definido el esquema,el usuario querrá defi- nir las partes con las que desea trabajar. Es decir,definir su sub-esquema.

SUB-ESQUEMAS DE LA BASE DE DATOS.-Asi como se ha definido el esquema,también se puede definir el sub-esquema:

Primero definiremos los nombres y las llaves:

```
SS PRES-ADMIN-STATE WITHIN SCHEMA PRESIDENTIAL
PRIVACY KEY IS 'COPY PASWORD'
```

Luego son definidas las áreas necesarias:

```
REALM DIVISION
RD PRESIDENTIAL-AREA
```

Luego definimos los conjuntos que deseamos:

```
SET DIVISION
SD ALL-PRESIDENTS-SS
SD ALL-STATES-SS
.....................
```

Por último definimos los registros:

```
RECORD DIVISION
01 PRESIDENT
  02 PRESS-NAME
.................
```

Una vez definida nuestra base de datos, suponiendo además que ya ha sido cargada, los programas de aplicación ya podrán hacer uso de ella*.

* Ver apéndice B

IV.4

SISTEMAS EN TIEMPO REAL

Comenzaremos definiendo lo que es un sistema en tiempo real:

Un sistema en tiempo real es aquel que controla un medio ambiente recibiendo datos de éste,procesandolos y entregando resultados lo suficientemente rápido como para afectar el funcionamiento de el medio ambiente en ese momento.

Ejemplos de sistemas en tiempo real:

- Un sistema de reservaciones áereas.
- un sistema bancario
- etc

Es claro que dadas las características de un sistema en tiempo real,es de vital importancia el tiempo de respuesta de éste.

Entendemos por tiempo de respuesta,el intervalo de tiempo entre un evento y la respuesta de el sistema a ese evento.

En el diseño de un sistema en tiempo real existen 6 aspectos que debemos de tomar en consideración:

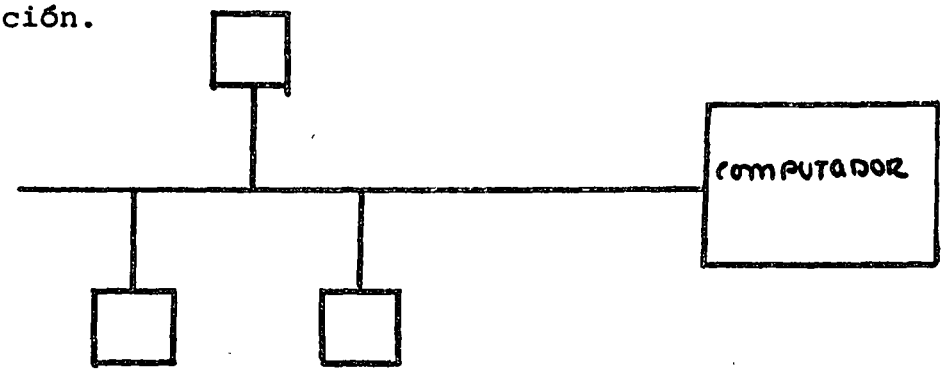
- la complejidad de el equipo.
- el tiempo de respuesta
- el tiempo promedio de arrivos.
- el número total de instrucciones
- la complejidad de los programas de aplicación

- la complejidad de los programas supervisores

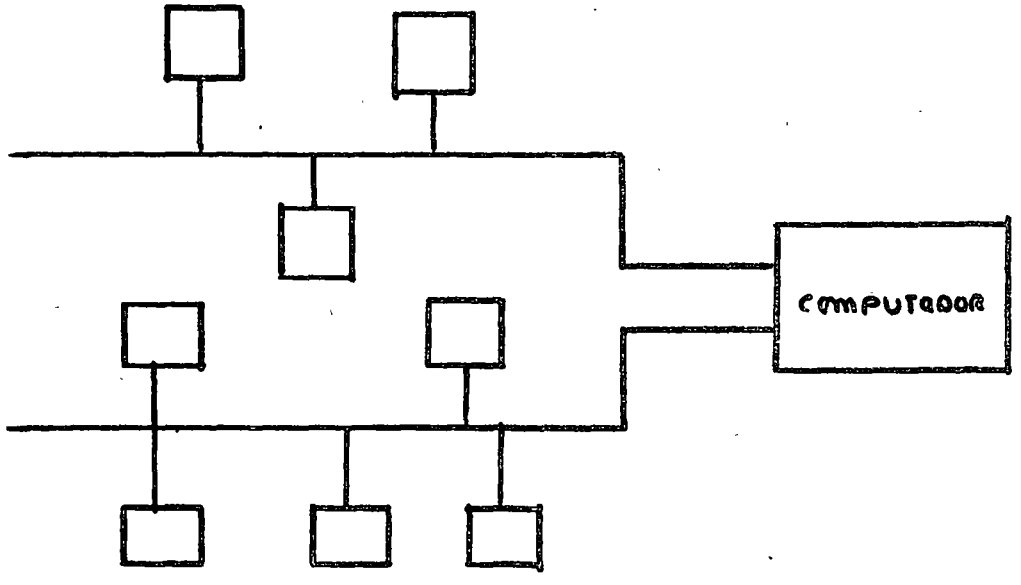
-la complejidad del equipo.

Este punto se refiere al número de dispositivos con los que cuenta nuestro sistema.

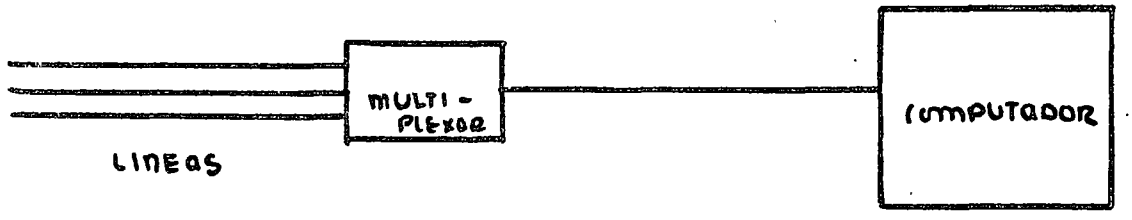
El dispositivo más simple es donde varias terminales estan conectadas a una computadora por una linea de comunicación.



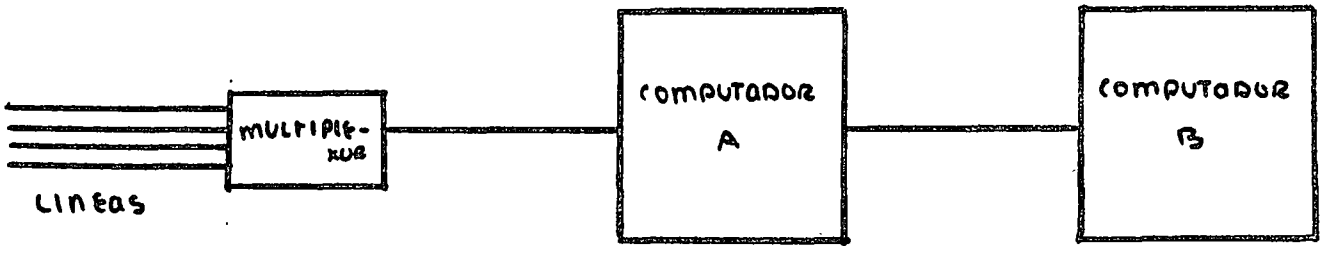
También podemos tener varias lineas de comunicación:



También puede existir un multiplexor como inter - mediario entre las lineas y la computadora.



O inclusive podemos tener 2 o más computadoras en nuestro sistema.



Estas son algunas de las configuraciones, pero no son todas. De hecho, la configuración dependerá de el objeti - vo de nuestro sistema.

Es claro también que la configuración de un sistema influye en el diseño tanto del programa supervisor como de los programas de aplicaciones.

- El tiempo de respuesta.

Generalmente el tiempo de respuesta de un sistema cambia de acuerdo a las necesidades de éste, es decir, existen sistemas que requieren un tiempo de respuesta bastante corto y otros que no.

El tiempo de respuesta más simple es aquel tiempo que tarda la computadora en interrumpir la tarea que esta ejecutando y atiende el procesamiento del evento.

Sin embargo, pueden existir varios delays:

- varios eventos pueden llegar al mismo tiempo y por lo tanto deben de hacer cola.
- puede existir una tarea de mayor prioridad procesándose en ese momento.

- Tiempos promedio de arribo.

Al tiempo promedio que transcurre entre el arribo de un evento y otro le llamamos tiempo promedio de arribo.

Generalmente los tiempos de arribo cambian. Por lo tanto debemos de diseñar un sistema que maneje loa arribos en forma eficiente, sobre todo en horas pico.

- El número de instrucciones.

Esto no requiere de mucha explicación. Es claro que un programa procesándose en tiempo real debe de contar con las instrucciones extrictamente necesarias.

- La complejidad de programas.

Se refiere a los algoritmos seleccionados para el procesamiento de el evento.

- La complejidad de los programas supervisores.

El programa supervisor también es conocido con el nombre de Ejecutivo. Su complejidad dependerá de la configuración del sistema.

Asociados a los sistemas en tiempo real existen también problemas con la programación.

A continuación definiremos algunos de los problemas más comunes.

- Despachador dínamico.-Dado que los arribos llegan en forma aleatoria y además pueden ser de longitudes distintas necesitarán - comunmente - de tareas distintas para su procesamiento.

Debido al carácter aleatorio en la ejecución de estas tareas, no podrá planearse una ejecución secuencial de ellas. Es por eso que necesitamos un despacho dínamico.

- Direccionamiento dínamico de memoria.- Debido a que las tareas necesitan espacios distintos de memoria para su ejecución y debido a su carácter aleatorio, es necesario un direccionamiento dínamico de memoria.

Es tarea del Ejecutivo proveer este direccionamiento dínamico de memoria.

Otra consecuencia de esto es que los programas deben ser de código relocalizable.

- Direccionamiento de prioridades.- Generalmente varios arribos llegarán al mismo tiempo a la computadora.

Es probable que todos tengan la misma prioridad de ejecución, teniendo todos que hacer cola.

Sin embargo el caso más común es la existencia de arribos que no deben de hacer cola. Estos deberán de tener una prioridad mayor que los demás arribos.

El problema está en las interrupciones que se causan en las tareas que se estaban procesando.

- Multiprogramación.- Debido también a los arribos, un sistema en tiempo real exige multiprogramación. Si no existiera, el tiempo de respuesta sería pésimo.

- Interrupts.- El sistema debe de manejar también interrupciones. Deben existir interrupciones para la llegada de arribos, para la operaciones de entrada y salida, etc.

Generalmente, entre más grandes sean los sistemas y además sean multiprogramados, el número de interrupciones sera mucho mayor.

- Colas.- Ya que pueden llegar varios arribos al mismo tiempo a la computadora, se hace necesario el uso de colas de espera para su procesamiento.

Existen colas para los arribos, colas para las salidas, etc.

Las colas suelen organizarse por prioridades de tareas. Es trabajo del Ejecutivo manejar estas colas.

- Sobrecargas.- Puede ocurrir que en un instante sean tantos los mensajes a una computadora, que el tamaño de las colas crece bastante, robando espacio de memoria para el procesamiento.

En este caso decimos que el sistema se sobrecarga y se pone en peligro su funcionamiento.

Un buen sistema tendrá muy pocas posibilidades de sobrecargamiento, pero siempre debe diseñarse un programa que manaje esta situación.

- Multiprocesamiento.- En los sistemas donde existe más de un procesador surgen problemas extras.

El ejecutivo será más complicado ya que deberá de controlar las operaciones de las unidades de procesamiento conectadas.

Generalmente un procesaor funciona como maestro y los otros como esclavos.

- Lineas de comunicación.- Las lineas de comunicación también representan un trabajo extra.

Tanto los mensajes que se reciben como los que salen a todas las lineas de comunicación deben de manejarse en forma paralela.

Aunado a esto se encuentra el problema causado por las interrupciones ocasionadas por los mensajes enviados por

terminales y el polling que el sistema deberá de realizar sobre todas las lineas.

- El Ejecutivo.- Sobre este punto no existe mucha discusión. Bastará con decir que el buen funcionamiento del sistema dependeraen gran parte de la eficiencia del Ejecutivo.

- Duplexing.- En algunos sistemas para ganar confiabilidad se usa el sistema de duplexamiento.

Bajo este sistema, una computadora permanece stand-by hasta que la otra computadora sufre una caída. En ese instante se hace un switcheo para que sea la otra computadora quien continúe realizando el procesamiento.

Este metodo presenta algunos problemas: Es probable que una tarea estuviera a la mitad de su ejecución cuando se cayó el sistema. Al entrar el otro computador en linea, la ejecución debera de continuarse exactamente donde se interrumpió.

- Fall Back.- Algunas vaces la caída de el sistema no es total. Puede ser un archivo en disco o puede ser una linea de comunicación solamente.

En este caso es deseable que esta caída no provoque un abortamiento del sistema sino solo una degradación.

Fall Back es el modo de operación de una computadora degradada. Una vez que la causa de la degradación es resuelta, el Ejecutivo hace una recuperación de lo perdido y todo vuelve a la normalidad.

El manejo de Fall Back puede ser muy complicado. Si un archivo ha sido la causa de la degradación, todos los programas de aplicación que accesan ese archivo deberán de ser corregidos en lo que se refiere a su direccionamiento.

- Chequeo.- En sistemas en tiempo real las funciones de chequeo son bastante complicadas.

Si el Ejecutivo no esta terminado o algunos de los programas de aplicación que interactúan con otros tampoco estan terminados, se deberán de hacer simulaciones de las partes faltantes para poder efectuar el chequeo.

Si se trata de un sistema multiprogramado, la recuperación de errores se vuelve particularmente difícil.

Estos mismos problemas se presentan en sistemas sobrecargados, fall backs o duplexamientos.

El apéndice A hace referencia a un ejemplo de un sistema en tiempo real.

SEGURIDAD, PRIVACIDAD Y PROTECCION DE SISTEMAS.

Protección.

Hay dos aspectos principales relativos a la protección de la información de un sistema.

- Prohibir el acceso a usuarios que no tienen derecho a cierta información.
- Garantizar el acceso a los usuarios, esto es, responzabilizarse a que la información almacenada no se destruya o pierda.

Ultimamente se ha discutido mucho otro aspecto de la protección y está relacionada con el valor social de proteger la información personal. Si se acepta que cada ser tiene derecho a mantener privada alguna información relativa a él, como por ejemplo el resultado de un análisis psicológico, entonces se debe de garantizar que esa información nunca debe de hacerse pública. Sin embargo, este tema no será analizado en esta sección.

Hay varios ejemplos que destacan la importancia del problema de la protección de la información.

- Privacidad individual El historial psicológico de los empleados de una compañía no debe de ser consultado por cualquier usuario diferente al médico de la compañía.

- Privacidad profesional.- Si dos empleados compiten por

un puesto superior, cualquiera de ellos no debe tener acceso al - expediente de su competidor.

- Actualidad de la información.- Esta debe de estar actualizada para que refleje la realidad.

- Vandalismo.- Un empleado decide destruir la base y la remueve.

- Fraude.- Modificar los datos para obtener ventajas, por ejemplo más ventas implican más comisión.

Siempre hay un costo asociado al diseño de un sistema de protección, es muy importante que este costo no exceda al valor de la información a proteger. La experiencia indica que se debe de gastar entre un 10 y un 30% del valor del sistema en la protección.

Los tres aspectos más importantes en la protección son:

- Protección en la lectura y escritura (sólo personal autorizado puede manipular la información).
- Privacidad (no se puede asociar un dato a un individuo en particular).
- Confidencialidad (sólo personal autorizado tiene acceso a la información).

La responsabilidad de mantener protegida la información pertenece a la dirección de la organización y sólo algunos aspectos pueden ser delegados al administrador de la base.

Hay tres elementos principales que intervienen para garantizar la protección.

- Los posibles usuarios.
- El tipo de acceso deseado.
- Los objetos por accesarse.

Los posibles usuarios.- Se identifican por alguna clave - que ha sido asignada y por un proceso de validación de dicha clave, por ejemplo, el sistema pregunta por algún código que sólo el dueño de la clave conoce, a este código se le conoce como llave secreta (password). La posibilidad que un usuario no autorizado, descubra el valor de una de las llaves secretas es mínimo. En general el tiempo necesario para descifrar la llave será:

$$T = \frac{1}{2} d^c t_e$$

d es la longitud de la llave.

c es el número de valores posibles para cada carácter.

t_e el tiempo que tarda el sistema en validar la llave.

Muchos sistemas no permiten que un posible usuario se "equivoque" muchas veces. Otros sistemas tardan bastante tiempo en ve

rificar la validez de la llave, no permitiendo así que se intenten muchos valores en un tiempo razonable.

Además de los usuarios no autorizados hay que evitar que los programas que tienen acceso a la información no sean ejecutados por cualquier persona. Muchas veces son solamente partes del programa las que deben de estar protegidas. Por ejemplo: el sueldo promedio puede ser un dato accesible a muchos usuarios, pero el sueldo individual es un dato privado. También la ejecución repetida de un programa por un usuario generalmente indica que algo prohibido se esta realizando.

El destino y la hora en la que se realice determinada consulta es un parámetro que debe de controlarse también. Una terminal en el área de producción no tiene por qué accesar información de la nómina.

El tipo de acceso deseado,- Hay siete tipos de acceso a la información que deben de controlarse.

- Lectura de datos.
- Ejecución de programas.
- Modificación de los datos.
- Remoción de los datos.
- Adición de los datos.
- Movimiento de los datos.
- Verificación de existencia.

La lectura de datos es el privilegio de copiar partes de la información al ambiente del usuario y manipularla.

La ejecución de programas permite al usuario, a través de las subrutinas del mismo programa tener acceso a partes de la información para lo cual puede o no tener autorización.

Los datos pueden ser modificados, removidos o adicionados y cada uno de estos privilegios tiene que estar considerado dado que hay usuarios que tendrán uno pero no el otro.

El movimiento esta relacionado con el hecho de transferir la información pero sin poderla analizar. Un operador tiene la obligación de almacenar ciertos archivos en cintas, pero esto no le - permite analizarla.

Por último la verificación de existencia permite al usuario saber si cierto dato está o no en la base.

Los objetos por accesarse.- Hay 5 tipos de objetos que deben de protegerse.

- Los datos, Muchas veces pedazos de papel tirados a la basura contienen información importante. Otras veces algún visitante toma una fotografía y resulta que en ese momento aparecía información importante.

- Caminos de acceso- Representan la relación que hay entre diferentes tipos de información. Son apuntadores que relacionan datos. Estos datos por sí mismos pueden ser de valor nulo, pero si se obtienen también las relaciones, entonces se podrá reconstruir información prohibida.

- Programas. Como se mencionó antes, estos no deben de estar al alcance de todos los usuarios. Muchas veces es suficiente con la protección del programa fuente, pero otras hay que proteger el programa objeto también.

- Los esquemas.- Sólo un grupo de usuarios puede utilizar un esquema. Le puede mantener una bitácora de uso de los esquemas para garantizar que sólo los usuarios permitidos lo utilizan. Aquí es importante mencionar la importancia de que en la bitácora estén anotados los datos relevantes de cada usuario que ha interaccionado con el sistema. Este es un modo eficaz de descubrir posibles infractores.

- El tiempo.- La información deja de ser confidencial conforme pasa el tiempo, hay que controlar las fechas para las cuales la información puede ser desprotegida.

Criptografía.-

Otro método de proteger una base es almacenando una transformación de los datos. Existen tres técnicas principales:

- Codificando la información.
- Desplazando los valores de los datos.
- Substitución de los datos.

La codificación significa asociar un valor a cada tipo de dato permitido. Por ejemplo:

1 - Hombre

2 - Mujer

A - Estado de México.

B - Guerrero.

C - Oaxaca.

El desplazamiento de la información consiste en sumar un valor a los caracteres.

Original: A B C D E F G H I J - - - - - - - - - -

Desplazado:D E F G H I J K L M - - - - - - - - - -

La sustitución consiste en crear una tabla que asocie diferentes valores a cada carácter, esta tabla puede ser fija o generada por una función.

INTEGRIDAD DE LA INFORMACION.

El problema de integridad consiste en que la información es correcta en todo tiempo. Hay un límite: sería prácticamente imposible que el sistema revisara que cada dato es correcto, pero puede checar que cada dato es posible. Por ejemplo la edad de un indivíduo será mayor a 0 y menor a 100, cualquier otro valor no es aceptable o posible, sin embargo sería muy costoso y difícil implementar procesos que garanticen que la edad es 35 y no 36 para un indivíduo en particular. Mantener una base íntegra se puede vizualizar como protegerla de modificaciones inválidas. Y se diferencia de protección en el hecho de que las modificaciones son involuntarias y no ilegales: la protección se logra impidiendo el acceso a ciertos indivíduos, mientras que la integridad se logra imponiendo restricciones a los valores que puede tomar la información para cada campo.

El problema de mantener la información dadas las restricciones es un problema difícil por sí mismo: en un sistema de múltiples usuarios, que estén interactuando sobre un sólo grupo de información es un ejemplo de lo difícil que es mantener la información correcta.

La pérdida de integridad puede ser causada por:

- Fallas en los componentes electrónicos.

- Error humano al operar el sistema.
- Error de programación.

Todos los sistemas están predispuestos a producir errores y por ello debe de haber forma de detectarlos y corregirlos. Esto requiere en conjunto de muchas subrutinas bien elaboradas.

En el modelo relacional cada relación tiene un conjunto de restricciones asociadas para cada campo. Si las restricciones son de la "cuarta forma normal" entonces la mayoría de los datos sólo aparecen una vez garantizado que valor único.

A continuación se presentan algunos ejemplos de las restricciones que habrán para varias situaciones:

1.- Por definición toda llave primaria es única y no puede ser nula.
2.- Si existen otros llaves también serán únicas.

Por ejemplo:

Relación S (S#, S NOMBRE, STATUS, CIUDAD)
Llave (S#)

Restricción

Rango S SX
Rango S SY
S Y S Y (SX.S NOMBRE=SY. S NOMBRE SX.S#=SY S#)

3.- Las dependencias funcionales son otra forma de integridad si una relación es de la cuarta forma normal todo atributo que no pertenece a la llave será dependiente de ella.

4.- Hay atributos cuyo valor depende de los valores de otros. Por ejemplo:

"Cantidad en almacen" y "Cantidad vendida".

5.- El valor de los campos sólo puede variar en un rango predeterminado.

6.- El formato de los datos puede detectar un error. Por ejemplo: Comenzar con una mayúscula y dejar un espacio en blanco entre signos de puntuación.

7.- La variación de un dato. Por ejemplo: La antiguedad de un empleado en la compañía nunca puede decrecer. Este tipo de restricciones se pueden incorporar con comandos del sistema, ver la siguiente sección que es en donde se detalla cada una de las instrucciones aquí consideradas.

8.- El último caso de restricción es la diferida. Por ejemplo: en un sistema bancario, cuando se transfiere dinero de una cuenta a otra el usuario lo hace mediante una

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

EL IMPACTO DE LOS SISTEMAS DE INFORMACION EN LAS AREAS DE LA ORGANIZACION EMPRESARIAL

AGOSTO, 1978.

V. EL IMPACTO DE LOS SITEMAS DE INFORMACION EN LAS AREAS DE LA ORGANIZACION EMPRESARIAL.

V.1 FACTORES A CONSIDERAR EN LA SELECCION DE EQUIPO

CARACTERISTICAS QUE HACEN UNICA LA DECISION DE SELECCION DE COMPUTADOR

En esta sección, se hace una explicación relativa al porque ciertos factores hacen de la adquisición de computador una decisión única.

MEDIDA DE COSTOS Y BENEFICIOS.

Aunque la determinación de los costos de un sistema propuesto es regularmente fácil, es muy difícil tratar de medir los beneficios que serán obtenidos por adoptar el sistema. El valor de mejor información para la toma de decisiones de la administración o el incrementar el servicio al consumidor no es fácilmente medible.

FALTA DE UNA TECNICA UNIVERSAL.

En un panorama de selección de computadores en la práctica, se encuentra que cada instalación y selección es única. Algunos facotres que causan esta unicidad son: el tamaño de la firma, industria, tipo de aplicaciones y localización.

Como resultado no puede haber una práctica universal para la selección de computador sólo delineamientos generales.

PARTICIPACION DE LA ALTA ADMINISTRACION.

Probablemente la diferencia más importante entre selección de computador y otra selección es la necesidad de que la alta administración participe en un alto grado.

Para alinear las metas de la organización con el esfuerzo de sistemas y procesamiento de datos, la administración debe estar interesada en esta área.

El porque debe tomar parte en ello, es demostrado por la siguiente lista de factores:

1) El computador cuesta una gran cantidad de dinero (función de planeación financiera).
2) La organización de sistemas debe requerir un departamento funcional separado (especialmente en la adquisición).
3) El computador puede cambiar la información y reportes que reciben los ejecutivos.
4) El computador puede crear necesidades de cambio en políticas y prácticas dentro de la organización.
5) El computador puede crear necesidades de cambio en la estructura organizacional. (relaciones con otros departamentos).

La selección de computador debe ser considerada cuando cualquier adquisición por primera vez, reemplazo, expansión o adición de facilidades en computación tenga lugar.

El llevar a cabo este proceso es una decisión en tres partes:

1) ¿Se justifica el uso de un computador?
2) Si es así, ¿Cuál?
3) ¿Cómo conviene adquirirlo?

La primera parte es comúnmente conocido como estudio de factibilidad, la segunda comprende los criterios de decisión y las técnicas de evaluación, y la tercera es el método de financiamiento usado. Cada uno de estos puntos es tratado posteriormente en forma más detallada.

ESTUDIO DE FACTIBILIDAD.

El factor primordial en el estudio de factibilidad está en determinar si una computadora puede ser usada para ayudar a lograr los objetivos de la organización. El primer paso en esta fase es definir claramente los objetivos de la organización y como se relacionan con el esfuerzo de procesamiento de datos.

¿Qué es lo que le interesa a la organización?
- Conocer eficiencia de operación.
- Contar con mejor información para la toma de decisiones.
- Mejorar servicio al cliente.

Como segundo paso se debe definir el tipo de sistema que pueda contribuir a obtener lo deseado.

La forma de cubrir lo anterior en este estudio consiste en dividirlo en las siguientes áreas: Análisis de sistemas actuales, definición de requerimientos, economías esperadas.

ANALISIS DE SISTEMAS ACTUALES.

Esta tarea comprende las fases de recopilación de hechos y análisis de la información coleccionada.

Los procedimientos actuales necesitan estar completamente entendidos y documentados en suficiente detalle para proveer una buena base para elaborar la propuesta de sistema.

Los requerimientos de información incluyen:
- Funciones propuestas que se ejecutarán en la computadora.
- Costos de métodos actuales en la ejecución de estas funciones.
- Requerimientos de información de entrada.
- Requerimientos de reportaje.
- Inter-relaciones de las actividades seleccionadas.

De gran importancia en esta fase es el método de registro de información para facilitar el análisis. Algunos de los métodos y formas de presentación incluyen:
- Diagrama de organización.
- Diagrama de operación y descripción de trabajos.
- Lista de reportes y formas.
- Diagrama de distribución de reportes y formas.
- Diagrama de utilización de gente y equipo.
- Tablas de cargas de trabajo.
- Diagrama de distribución de cargas de trabajo.
- Tablas, gráficas y estadísticas.
- Declaraciones narrativas.

DETERMINACION DE REQUERIMIENTOS.

La administración puede sentir que las facilidades de procesamiento de datos deben ser expandidas, pero usualmente esta en una posición tal que la definición de estas necesidades es en términos vagos. En esta fase del estudio deben especificarse los requerimientos en forma completa definiéndolos en función de las consideraciones a corto y largo plazo de la firma, teniendo en cuenta potencial para:

- Mecanización de aplicaciones actuales.
- Expansión de aplicaciones ya mecanizadas.
- Mecanización de nuevas aplicaciones técnicas y comerciales.
- Centralización de la mecanización para otras unidades de la compañía.

Es decir que las facilidades solicitadas para procesamiento de datos deben ser de tal naturaleza que permitan agregados por futuras expansiones en volúmenes de información o cambios en el panorama de esta área.

ECONOMIAS ESPERADAS.

Uno de los principales propósitos en esta fase del estudio es determinar el costo de las operaciones que serán eliminadas por los sistemas propuestos.

Uno debe considerar:
- Aplicaciones potenciales o sujetas a investigación.
- Estimados de ahorro en pesos, o beneficios potenciales a ser derivados.
- Indicación de tipos de mejoras esperadas.

PRUEBAS DE FACTIBILIDAD.

Todo estudio de factibilidad debe de considerar cierto tipo de pruebas como son:
1) TECNICA.- Que las aplicaciones propuestas son posibles dentro de los límites de recursos y tecnología disponibles.
2) ECONOMICA.- Que los beneficios a obtener son mayores que los costos.
3) OPERACIONAL.- Que las aplicaciones desarrolladas pueden ser y serán usadas por la organización.

CRITERIOS DE DECISION Y TECNICAS DE EVALUACION.

El usuario debe definir las especificaciones del sistema clara y precisamente. El debe saber el método de evaluación a seguir.

Es esencial recordar que la selección de computador será sólo tan buena como la información proporcionada al vendedor y los criterios de evaluación usados.

La opinión de definir las necesidades en sistemas y la técnica de evaluación a usar forza al usuario a revisar el criterio de decisión y ver como influye en la selección de un computador específico. Si el usuario deja que el vendedor determine sus necesidades, este orientará su consejo enfatizando puntos fuertes de sus sistemas disponibles.

INVESTIGACION SOBRE CRITERIOS DE DECISION.

Antes de examinar el criterio usado para tomar la decisión, debe ser hecha una referencia adicional a alguna investigación sobre dicho criterio.

Quien toma decisiones trata de minimizar inconvenientes y no maximizar utilidad como se piensa.

Mucha información que se dice necesaria, no es realmente usada. Esto debe considerarse ya que podemos establecer una larga lista de criterios que actualmente tengan poco efecto sobre la decisión a realizar.

CRITERIOS OBLIGATORIOS VS. CRITERIOS DESEABLES.

Relacionado el punto anterior se hace la designación entre lo que se llama criterios obligatorios y criterios deseables en consideración de un sistema de computación.

Un criterio obligatorio es una característica que se marca esencial para que el sistema cumpla completamente con su misión.

Los criterios deseables son esas características o requerimientos que facilitan al sistema el cumplir con su tarea.

Deben tenerse precaución para evitar fijar ciertos criterios deseables como obligatorios; ya que al establecer muchos requerimientos obligatorios un usuario se está limitando a sí mismo a un pequeño número de sistema de computación, cuando existen otros proveedores que pueden satisfacer sus necesidades.

El orden que se les dá en la presentación no es necesariamente el orden de importancia de los mismos. La lista incluye tanto criterios objetivos como subjetivos y para su discusión son separados dentro de cuatro grupos:
Consideraciones de costos, de Hardware, de Software y de Soporte del proveedor.

CONSIDERACIONES DE COSTOS.

Existen dos tipos principales de costos que deben ser considerados en la selección de computador:

1) Costos de una sola vez.- que pueden ser costos de aduanas, transportación, preparación del lugar, instalación, costos de conversión.
2) Costos continuos.- que deben considerarse como: rentas, mantenimiento, seguros, energía, sueldos y salarios de personal requerido, artículos y suplementos de mandados por el nuevo sistema.

Algunos costos pueden ser continuos, de una sola vez o no existir. Esto depende del tipo de adquisición que se haga. (compra, renta o arrendamiento).

CONSIDERACIONES DE HARDWARE.

El principal interés del usuario en el hardware está en conseguir la capacidad de rendimiento necesaria al menor precio.

Una medida de rendimiento es el "throughput" del sistema o sea la cantidad de carga de trabajo que puede ser manejada en un período de tiempo determinado. Los factores que afectan el "throughput" incluyen:
1) Velocidad del procesador central.
2) Velocidad de I/O
3) Facilidad de multi-proceso.

La modularidad de un sistema de computación puede proporcionar la habilidad de incrementar "throughput" en el futuro sin tener que reemplazar el sistema.

CONSIDERACIONES DE SOFTWARE.

PROGRAMAS ACTUALES: El usuario puede tener programas de aplicación escritos en un lenguaje de bajo nivel. Un ahorro en costos resulta si el usuario puede correr esos programas en el equipo propuesto en una emulación. En esto también afecta la velocidad de conversión. El usuario debe estar conciente que los programas que tiene en uso fácilmente convertidos para obtener una eficiencia óptima.

COMPILADORES: Otra consideración en software es que el vendedor ofrezca los compiladores para los lenguajes que la organización desee usar.

PAQUETES DE APLICACIONES: El usuario debe también considerar los paquetes de aplicación extras o especiales que puede el proveedor ofrecer. Bastante tiempo de programación y administración de proyectos puede ser disminuído al obtener utilities como sorts y merges, rutinas de programación lineal o técnicas de camino crítico.

SISTEMA OPERATIVO: El sistema operativo y su versatilidad deben ser considerados. Por ejemplo la cantidad de memoria que requiere puede afectar la cantidad que el usuario debe comprar. La protección de memoria será necesaria si la característica de multiprogramación es parte del sistema. La efectividad del sistema operativo será considerada como parte básica en lo referente a técnicas de evaluación.

SOPORTE DEL PROVEEDOR.

Es frecuente que lo que el usuario espera del proveedor esté en desacuerdo con lo que recibe por parte de éste. Algunos puntos a considerar en este aspecto son los siguientes:

SISTEMAS.

En el soporte de ingeniería de sistemas que ofrece el proveedor, debe tenerse precaución en considerar no sólo cantidad de personal, sino también su calidad. El usuario debe checar los antecedentes o experiencia en aplicaciones similares en el área.

ENTRENAMIENTO.

Una área donde el proveedor puede ser de gran ayuda es en el entrenamiento. Usualmente el proveedor tiene disponibles el personal y facilidades para la enseñanza en sistemas.

MANTENIMIENTO.

El usuario debe también considerar el soporte de mantenimiento que ofrece el proveedor con el sistema computacional. Se debe examinar el programa de mantenimiento preventivo del proveedor y tratar de tener contacto con otros usuarios para determinar como son de efectivos dichos programas.

Debe considerarse también un plan de reparaciones sobre lugar o sobre llamada (casos especiales). Un sistema no es bueno si no está trabajando.

"BACK-UP"

El usuario será grandemente beneficiado si puede usar otros sistemas similares dentro de su área geográfica en caso de sobre-carga de trabajo o fallas de su equipo por largo tiempo.

Debe también considerarse la localización de las oficinas o instalaciones del proveedor. Otra área de "back-up" es el empuje del proveedor.

Un punto importante es la garantía que ofrece el proveedor en las fechas de entrega.

TECNICAS DE EVALUACION.

Comúnmente en la selección de computador, no es hecha una evaluación real por parte del usuario.

Favoritismos o subjetividades de alguien en la organización determinan que un proveedor "X" parece bueno o ha servido bien en el pasado y se evitan los problemas de evaluación.

Otra técnica que no es producto de un tomador de decisiones racional es una donde ciertas características estandard son establecidas en reuniones con proveedores y el que las cubre a menor precio asegura el contrato.

Nada podría ser más placentero que poder contar con una técnica sencilla, completamente probada, que garantice la selección del equipo más adecuado para sus necesidades.

Desafortunadamente, tal técnica no está disponible actualmente y ninguna se espera en un futuro próximo. El desarrollo de tal técnica no ha sido posible porque muchos factores que tienen un efecto importante sobre el rendimiento y economía global del sistema no son fáciles de expresar en forma cuantitiva.

Existen, sin embargo, un número de técnicas disponibles actualmente que pueden ayudar muy significativamente a determinar el equipo y Software más apropiado para ciertas necesidades en particular. Algunas de esas técnicas serán descritas aquí presentando ventajas y limitaciones que tiene cada una de ellas.

INSTRUCTION MIXES.

Para comparar velocidades de procesadores centrales, han sido desarrolladas varias mezclas de instrucciones las cuales consisten simplemente de un promedio ponderado de los tiempos de ejecución de un conjunto de las instrucciones más comúnmente usadas. Se le asigna un factor de peso a cada instrucción de acuerdo con la frecuencia de uso en programas de un tipo dado.

Las mezclas de instrucción son fáciles de calcular y comparar, pero sólo pueden medir las velocidades de los procesadores centrales. Además, este método ignora que las frecuencias de diferentes tipos de instrucciones varían significativamente en programas para diferentes aplicaciones, y que una sola instrucción en cierto computador puede ejecutar funciones que requieren varias instrucciones básicas en otros computadores.

KERNELS.

El Kernel es un problema simple, presumiblemente representativo de aplicaciones típicas de negocios o científicas, el cual se codifica y se le toma el tiempo en cada uno de los computadores que se están comparando.

Los Kernels permiten que el repertorio de instrucciones de cada computador se use de la manera más ventajosa, pero, como los Mix Instructions, generalmente ignoran consideraciones de In put/Out put y factores de eficiencia de Software.

SIMULACION.

Otra posible forma de comparar equipos es el uso de la simulación, se toman en cuenta las características de cada equipo (velocidad de I/O, tiempo de proceso, etc.), y se toman cargas de trabajo típicas iguales comparando luego los resultados obtenidos de cada uno de ellos.

La comparación por simulación tiene una serie de desventajas, algunas son:
a) El uso de tiempos de ejecución promedio no es muy significativo.
b) Los resultados dependen mucho de los algoritmos usados.
c) Puede llegar a ser muy costosa.

BENCHMARKS.

Los Benchmarks buscan representar la carga total de trabajo por medio de programas representativos. Estos programas se corren en la máquina real usando los sistemas operativos y compiladores propuestos por los proveedores.

La ventajas de este método son: que el sistema operativo real, el compilador real, y la máquina real son probados.

Desventajas significativas son: falta de certeza de que los Benchmarks y los datos de prueba sean representativos de la carga real de trabajo, el costo de desarrollo del Benchmark y de adquirir datos de prueba, el esfuerzo requerido para convertir el programa de Benchmark para cada una de las máquinas, el costo de correr el Benchmark, y la dificultad de obtener la configuración precisa de el sistema propuesto.

FUENTES DE INFORMACION SOBRE LOS CRITERIOS.

Para hacer cualquier tipo de evaluación de lo que los requerimientos de sistemas serán satisfechos el usuario debe tener fuentes de información acerca de los criterios que el siente que son importantes.

OTRO USUARIOS.

Una fuente de información pueden ser otros usuarios de equipo del mismo proveedor.

Esto es especialmente verdadero en factores subjetivos tales como confiabilidad del proveedor o habilidad de cumplir con lo prometido. Los proveedores no están para juzgarse a sí mismos, pero otros usuarios seguramente tendrán opiniones y sentimientos que puedan ser examinados para indicar la satisfacción con el proveedor.

Debe tenerse la precaución de mantener la mente abierta cuando se entrevista algún usuario porque alguna de sus opiniones puede no estar bien fundada.

REPORTES ESTANDARIZADOS.

Una fuente de información acerca de los diferentes modelos y sus rendimientos y diseño son los reportes estandarizados tales como los publicados por el Auerbach Corporation.

Los reportes no proveen al usuario con un proceso de decisión, pero con información a menudo necesaria para hacer la decisión. Es recomendable para el usuario el efectuar revisiones periódicas de alguna revista técnica.

PROVEEDORES.

Puede también obtenerse información de los proveedores. Se recomienda para ello que el usuario proporcione al proveedor un cuestionario sobre los aspectos que considere más importantes. Debe tenerse cuidado al diseñar el cuestionario que no sean una preguntas tipo general sino lo más específicas posibles para evitar que la información proporcionada sea la adecuada.

RELACIONES USUARIO-PROVEEDOR.

La comunicación entre el usuario y el proveedor es importante dado que el proveedor debe saber exactamente que necesita el usuario, y este debe saber que le ofrece el proveedor para satisfacer sus necesidades.

- REQUERIMIENTO DE PROPUESTA DE SISTEMA.

Un método para comunicar las necesidades del usuario al proveedor es lo que se conoce como "requerimiento de propuesta de sistema" y que debe incluir lo siguiente:

1.- Requerimientos de sistema. Que debe ser capaz de hacer el sistema.
2.- Resumen del soporte que ofrece el proveedor.
3.- Preguntas técnicas y tablas de tiempos.
4.- Información de Benchmark.
5.- Demostración y presentación del proveedor.
6.- Fechas tentativas propuestas.
7.- Condiciones de contrato.
8.- Comentarios generales.

Las ventajas que proporciona este requerimiento de propuesta de sistemas pueden ser sumarizadas como beneficios tanto del proveedor como del usuario.

- BENEFICIOS DEL PROVEEDOR.

1.- Puede más fácilmente decidir que es lo que el usuario quiere y si él debe competir.
2.- Son más fácil de hacer las propuestas si se entiende lo que el usuario quiere.
3.- Tiene oportunidad de comentar sobre la técnica de evaluación a usar.

- BENEFICIOS DEL USUARIO.

1.- Sólo los proveedores que puedan satisfacer sus necesidades harán propuestas.
2.- Cada proveedor no hará propuestas alternativas al saber que busca el usuario.
3.- Tiempo de evaluación será reducido dado que todos los proveedores responderán las mismas preguntas.
4.- Las revelaciones de la técnica de evaluación permite observar lo apropiado de la técnica.

METODOS DE ADQUISICION DE COMPUTADOR.

Se asume que el análisis de evaluación ha sido completado, y que una configuración específica y un proveedor han sido seleccionados.

Entonces existen varias alternativas para adquirir el equipo:
1) Compra
2) Renta
3) Arrendamiento financiero

Cada una de ellas tiene sus ventajas y desventajas como se indican a continuación:

COMPRA.

El usuario compra directamente del proveedor el equipo. Paga adicionalmente mantenimiento mensual y seguros.

Ventajas:
- Uso ilimitado, sin gastos adicionales.
- No afectan los cambios en moneda.
- Existe un valor de recuperación cuando el equipo es reemplazado.
- Depreciación.
- Cuando se espera larga vida en el uso de ese equipo es económicamente el mejor método.

Desventajas:
- Problema de obsolecencia física y técnica.
- Usuario no puede presionar por servicios de otro tipo de soporte requerido del proveedor.

RENTA.

El usuario renta el computador al proveedor. Existe un pago de renta mensual y existe usualmente un pago por sobre-tiempo.

Ventajas:
- Fácil de financiar.
- Deducible completamente de impuesto.
- Puede tener cierta presión en servicios con el proveedor.
- No existe el problema de obsolecencia.

Desventajas:
- Afectan cambios en moneda.
- Uso limitado, gastos adicionales.
- Generalmente el método más caro.

ARRENDAMIENTO FINANCIERO.

El usuario paga a un tercero mensualmente una cantidad determinada durante el período establecido en contrato (3 ó 5 años) al final del cual tiene opción a compra.

Ventajas:
- Pagos deducibles de impuesto.
- Pagos menores que los hechos en caso de renta.
- Puede ser reemplazado el computador al expirar el contrato.
- Puede ser presionado el proveedor por servicios.
- No hay costo de sobre-tiempo.

Desventajas:
- En nuestro medio poco conocido este método, y muy caro y restringuido por ley en cuanto a su deducibilidad.

METODOS PARA EVALUAR ALTERNATIVAS DE ADQUISICION.

Para efectuar el análisis económico de cada una de estas alternativas se cuenta con las técnicas de: Punto de equilibrio y flujos de efectivo descontados.

PUNTO EQUILIBRIO.

Esta técnica consiste en determinar la función de costos acumulados por uso del computador para cada alternativa y encontrar los puntos de tiempo donde sea indiferente el uso de cualquiera de los métodos comparados.
No considera el valor del dinero a través del tiempo.

FLUJOS EFECTIVO DESCONTADOS.

Una técnica más común es la de determinar el flujo de efectivo actual (valor presente) de cada alternativa descontando de acuerdo con una tasa de interés predeterminada. Los resultados netos son comparables obteniendo la selección aquel método que tenga un menor flujo de efectivo.

V.2 CENTRALIZACION VERSUS DESCENTRALIZACION

A pesar de que el primer computador digital fue construido desde 1940, no fue sino hasta la decada de los cincuentas cuando su uso se generalizó en los negocios; en este lapso, los problemas fundamentales del proceso de datos en computador se referían metodológicamente a comprobar si los sistemas prioritarios en la organización eran computarizados y si los resultados obtenidos documentalmente eran útiles.

Actualmente, cuando el tamaño y el costo del computador (y su importancia dentro de la institución) ha crecido considerablemente, el problema de la selección, la programación y distribución de los recursos de sistemas informáticos ha cobrado relieve; es en este contexto en el que la disyuntiva entre centralizar o descentralizar los recuros de cómputo tiene lugar.

Quienes abogan por la centralización afirman que ese enfoque permite las "economias de escala" en operaciones de Hardware, en desarrollo de sistemas y en capacitación de recursos humanos; así es posible evitar toda duplicidad de esfuerzos, asegurar la mejor utilización del personal y facilitar considerablemente el control sobre el proceso de datos como un todo.

Por otra parte, quienes se inclinan por la descentralización alegan ofrecer con ella un más adecuado servicio a los administradores a nivel operativo, construri sistemas más efectivos con mayor 'familiaridad' con problemas locales, y también un más efectivo manejo de los sistemas desarrollados.

Reducido a estas dimensiones el problemas parecería concentrarse en la elección de eficiencia frente a eficacia: en el primer caso son consideraciones económicas y técnicas las predonimantes, y en el segundo resaltan los argumentos prácticos y operativos. La polémica parece deslizarse paralelamente a la controversia tradicional en la teoría organizacional según la cual la administración (genérica) centralizada resulta más eficiente, más controlada, en tanto que la descentralizada responde mejor a las necesidades del mercado.

En la práctica, tanto las ventajas como las respuestas al dilema son obscuras, lo único claro es que no hay una sola respuesta, y por esta razón nos abocamos aquí a delinear un marco de referencia sistemático y analítico que permita obtener una guía práctica sopesando los pros y los contras de la cuestión (1).

Convencionalmente dividiremos la presentación en seis secciones:
1) Criterios de evaluación de las opciones de organización.
2) Formas en que el trabajo de sistemas puede segmentarse para un mejor análisis.
3) Rangos variables de alternativas de organización.
4) Operaciones de proceso de datos.
5) Diseño y desarrollo de sistemas.
6) Planeación y control.

Evidentemente las tres primeras secciones ofrecen un panorama de los factores a considerar en la decisión,en tanto que las tres últimas cubren aspectos más específicos, que se mencionan genéricamente en la sección dos.

1) Criterios de Evaluación: En la búsqueda de respuestas es indispensable contar con una clara apreciación de los objetivos que deban ser cubiertos por los recursos de cómputo, las necesidades a satisfacer y los criterios que deban usarse para decidir como cubrirlas (objetivos y necesidades).

Podemos distinguir con claridad seis criterios susceptibles de consolidarse como punto de partida para examinar las opciones y determinar la organización apropiada:

a) Costos.
b) Servicio a usuarios.
c) Estilo de la organización general de la institución.
d) Posiblidad de administración.
e) Productividad de personal y su control.
f) Adaptabilidad al cambio.

Una primera regla de oro sería si los dos primeros criterios apuntan a la misma opción, esta puede ser elegida sin mayores consideraciones.

De hecho, los dos últimos criterios sólo son considerados en caso de 'empate' en los cuatro anteriores; en la mayoría de estos 'empates' generalmente se opta por mantener el STATUS QUO.

Aún cuando no es el criterio óptimo, en muchos casos la organización es definida de acuerdo al tercer criterio: en instituciones cuya alta gerencia tiende a absorber la toma de decisiones (incluso a nivel operativo) sin delegar facultades, la opción de centralización obtiene un importante handicap.

Por otra parte, si la premisa fundamental resulta ser "los mismos resultados, al menor costo", la centralización ofrece más ventaja que frente a una política de "al mismo costo, los mejores resultados".

Es más factible descentralizar en una institución que viene sufriendo cambios impresionantes (absorción de nuevas empresas, invasión de mercados diferentes con productos diferentes, etc.) que en aquellos relativamente estables en términos de productos y tamaño, para los que sería más aconsejable la centralización.

2) Segmentación de los trabajos de procesamiento de datos: Esquemáticamente podemos segmentar (para fines de análisis) los trabajos en tres grandes
a) Operación del computador.
b) Diseño y desarrollo de sistemas.
c) Planeación y control.

Las operaciones del computador se refieren fundamentalmente a los locales físicos (SITE) en que puedan instalarse los recursos de cómputo, sobre todo en lo referente a las decisiones que se toman durante las operaciones cotidianas; el diseño y desarrollo de sistemas se refieren al esfuerzo requerido para el ciclo de vida de un sistema en relación con la ubicación (o rotación) de un grupo de profesionales de apoyo (staff) y del tipo de decisiones que pueden tomarse localmente, sin previa aprobación central.

Finalmente, los trabajos de planeación y control deben considerarse en la medida en qie sea factible (dada una organización, centralizada o no) conservar los estandares definidos, y las facultades de selección de proyectos, control de tiempos, etc.

Una discusión más detallada de los tres aspectos (ventajas y desventajas de centralizar) será tratada en las secciones 4, 5 y 6; por ahora basta tenerlas presentes en este primer enfoque.

3) Alternativas de Organización: Una vez enjuiciados los elementos de trabajo a considerar y los criterios a utilizar, puede iniciarse la construcción de alternativas que tengan viabilidad operativa.

En realidad, la disyuntiva no suele ser tan radical como la hemos presentado hasta aquí; más bien suele optarse por configuraciones relativamente centralizadas o relativamente descentralizadas.

El rango en que vacían las alternativas depende normalmente de las facilidades que ofrezcan los locales físicos y de la filosofía administrativa de la institución; así, puede darse el caso de que el Hardware esté disperso en múltiples unidades locales, pero permanezcan bajo un estricto control central; igualmente puede existir una concentración de los recursos de cómputo en un solo local,pero con el administrador del mismo reportando a un "directorio" de usuarios locales con amplio poder de decisión.

La figura VI.1 ejemplifica un amplio rango de opciones, haciendo referencia a los "segmentos del trabajo" que citamos en la sección anterior.

4) Operaciones de Proceso de Datos:

a) Ventajas económicas de la centralización:
Probadamente (2) la relación de costos beneficios es considerablemente mayor (casi 7 veces) para equipos más grandes, que para varios equipos más pequeños.

b) Ventajas en recursos humanos de la centralización:
Un número reducido de operadores, y personal especializado es requerido; un grupo bien remunerado y mejor preparado puede ser contratado centralmente.

c) Ventajas en financiamiento de la centralización:
En la medida en que el número de proveedores y/o contratos que se tengan sea reducido, el número de arreglos financieros será menor y resulta menos costoso que realizar una operación con múltiples proveedores, en momentos distintos.

d) Ventajas en adquisición de Software de la centralización:
A menudo es posible justificar la compra de sofisticados paquetes de Software en la medida en que múltiples usuarios los requieran a una sola unidad de proceso.

e) Ventajas de seguridad en la centralización:
Por definición, los procedimientos de seguridad deberán encargarse a un muy reducido grupo, lo que se facilita más, en menor número de unidades de cómputo.

FIGURA V.2.1

		Descentralización			Centralización
Operación de Computador	Facilidades de cómputo	Hardware en cada unidad	Red de computadoras	Terminales locales CPU central	Hardware en el local central
	Producción	En cada unidad	Procesos previos en unidad	Captura de datos en la unidad	Sólo en la central
	Poder de Decisión	Completo en cada unidad	Central supervisa decisiones tomadas	Decisiones autorizadas centralmente	Ninguno en unidades
Desarrollo Sistemas	Control de Personal	Completo en cada unidad	Central supervisa	Central aprueba	Ninguno en cada unidad
	Grupos de análisis	Uno por unidad	Un analista senior en unidad	Grupo por unidad en central	Uno sólo en central
Planeacion	Políticas y control (estandares)	Central coordina	Central desarrolla recomendaciones	Central aprueba y rechaza	Central desarrolla y supervisa

Si de todos estos incisos sacamos conclusiones ventajosas en la centralización, entonces ¿Cuándo no centralizar?:

a) Cuando el total de requerimientos de cómputo de un local remoto puede ser tan pequeño o menor que una terminal "inteligente" muy sofisticada. En estos casos es frecuentemente más económico procesar localmente que usar iguales recursos para comunicarse con la unidad central. Ejemplo: Un sistema unitario frente a un teletipo y sus moderns, etc.

b) Cuando el tamaño de la institución y su volumen de información, no permita asegurar centralmente tiempos de respuesta y adecuación de modalidades en los sistemas.

c) Cuando se trate de consorcios con problemas específicos y muy tipificados,con requerimientos de Software diametralmente opuestos.

d) Cuando las economías a escala derivados de la descentralización sean asimiladas o rebasadas por gastos de comunicación provocados por distancias geográficas o sofisticaciones de transmisión (Ejemplo: un gran número de demandas de proceso en tiempo real).

e) Cuando los recursos humanos disponibles requieran de un arduo proceso de capacitación para dar soporte técnico adecuado a las unidades usuarias.

f) Cuando se parte de una situación de descentralización que dificulte en extemo la conversión y/o arroje costos muy elevados; esto suele suceder en corporaciones que al expedirse, asimilan empresas que ya tienen un centro de cómputo en funciones.

g) Cuando por problemas de control laboral pueda no ser recomendable "tener todos los huevos en una sola cesta"; la dependencia irrestricta de un sistema a una sola persona implica su carácter de "indispensable" y dificulta la productividad del centro; lo mismo puede decirse a nivel de "un solo grupo" de sistemas.

5) Diseño y desarrollo de sistemas: Los arguemtnos aquí son muchos más complejos que en la sección anterior; resumamos algunos:

a) El monitoreo y la rotación de analistas es más agil en una instalación centralizada; la compenetración y resolución de problemas suele ser más agil en la descentralización.

b) Es siempre más conveniente concentrar recursos humanos bien remunerados y altamente capacitados, que atomizar el presupuesto relativo, en muchas unidades de análisis.

c) El uso de estandares y rutinas comunes es más efectivo y amplio en instalaciones centralizadas.

6) Planeación y control: Las dos previas secciones representan los "segmentos de los trabajos de proceso de datos" con mayor peso específico en la definición de la organización. Sería sin embargo, un error costoso dejar fuera de ella a la responsabilidad de determinar políticas y gobernar de hecho el esfuerzo de todos los sistemas.

En la centralización, las decisiones acerca de la planeación y el control, así como la ubicación del poder de decisión, deben ser totalmente claras: también deben centralizarse.

En las instituciones descentralizadas, la ubicación de los procesos mencionados no puede contestarse tan radicalmente.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

LABORATORIO DE SISTEMAS

AGOSTO, 1978.

Palacio de Minería Calle de Tacuba 5, primer piso. México 1, D. F.

VI. LABORATORIO DE SISTEMAS.

E N V I R.

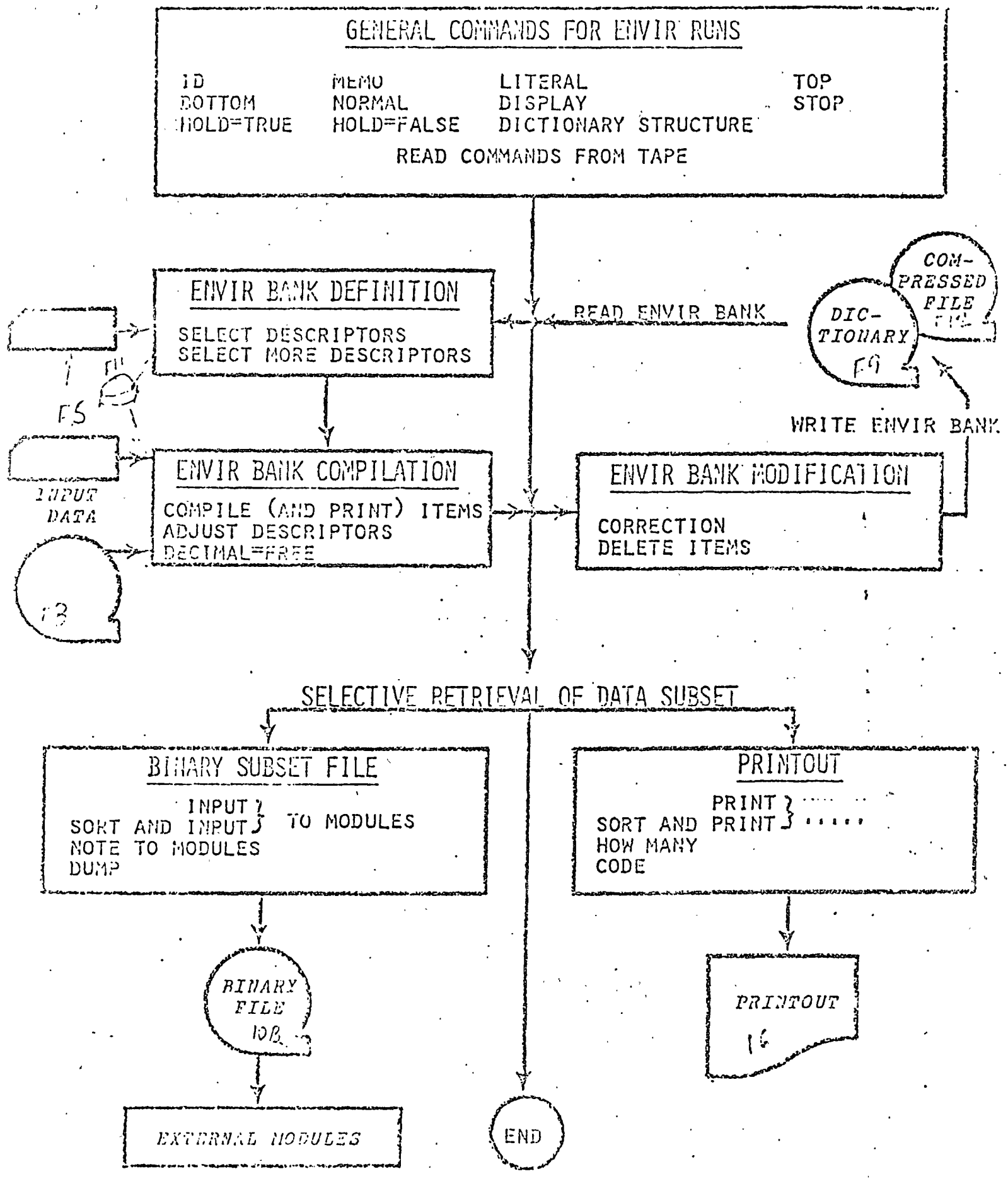
Para ilustrar los conceptos de generación y utilización de una base de datos, usaremos el sistema ENVIR que se caracteriza por estar escrito en FORTRAN y manejar parte del modelo relacional. El modelo que se utilizará sólo maneja una relación o tabla, pero es suficiente para ejemplificar muchos de los conceptos estudiados en el curso.

Las características principales del sistema son:

- Sólo maneja una tabla o relación.
- Puede haber tantos dominios como se desee.
- Los dominios pueden ser de tres tipos.
  - Numéricos.
  - Alfanuméricos.
  - Valores de una tabla de codificación.
- En cualquier momento se puede solicitar la composición de la tabla.
- Se puede agregar en cualquier momento más dominios en la relación.
- Se pueden agregar datos en cualquier momento.
- Se pueden eliminar datos.
- Se pueden modificar datos en cualquier momento.

- Dado un predicado se puede accesar la información que lo satisfaga.
- Cuenta cuantos elementos que satisfagan un predicado - existen.
- Hay ciertas facilidades de edición de los resultados.
- Durante la ejecución existe una copia del banco en el área de trabajo del usuario, en cualquier momento se puede proteger en disco.
- No tiene ningún mecanismo de protección de la información excepto por la posesión física del programa y el acceso a una computadora.
- La integridad se controla ya que al especificar los dominios el usuario especifica las restricciones para los - mismos.
- La información es con sus unidades.
  Por ejemplo: "500 pesos".
- En la versión actual se ha probado hasta con 100,000 registros, teniendo un tiempo de acceso en promedio de 20 segs en una B6700, con menos de 10,000 segmentos el tiempo se reduce a 1 segmento máximo por pregunta.

A continuación se reproduce parte del manual.


GENERAL COMMANDS FOR ENVIR RUNS
ID
MEMO
LITERAL
TOP
BOTTOM
NORMAL
DISPLAY
STOP
HOLD=TRUE
HOLD=FALSE
DICTIONARY STRUCTURE
READ COMMANDS FROM TAPE
ENVIR BANK DEFINITION
SELECT DESCRIPTORS
SELECT MORE DESCRIPTORS
READ ENVIR BANK
COM-
PRESSED
FILE
DIC-
TIONARY
F9
WRITE ENVIR BANK
F5
INPUT
DATA
ENVIR BANK COMPILATION
COMPILE (AND PRINT) ITEMS
ADJUST DESCRIPTORS
DECIMAL=FREE
F3
ENVIR BANK MODIFICATION
CORRECTION
DELETE ITEMS
SELECTIVE RETRIEVAL OF DATA SUBSET
BINARY SUBSET FILE
INPUT
SORT AND INPUT
TO MODULES
NOTE TO MODULES
DUMP
PRINTOUT
PRINT
SORT AND PRINT
HOW MANY
CODE
BINARY
FILE
10
PRINTOUT
16
EXTERNAL MODULES
END

ADJUST DESCRIPTORS descriptor sequence numbers separated by field separators *

BOTTOM

CODE

COMPILE AND PRINT ITEMS FROM CARDS / TAPE / DISK

COMPILE ITEMS FROM CAREDS / TAPE / DISK

CORRECTION (D,S) (D,S) .....(D,S) noise WITH / LLAVE Boolean Expression *

DECIMAL=FREE

DELETE ITEMS noise WITH / LLAVE Boolean Expression *

DICTIONARY STRUCTURE

DISPLAY

DUMP

END

HOLD=FALSE

HOLD=TRUE

HOW MANY noise WITH / HAVE Boolean Expression *

ID the remainder of the card may contain any string of characteres

INPUT TO MODULES

SORT AND INPUT TO MODULES

Noise: descriptor list FOR noise WITH / HAVE Boolean Expression *

LITERAL field separator

MEMO user supplied message *

NORMAL

NOTE TO MODULES
The next card may contain any string of characters as information to external modules.

PRINT
SORT AND PRINT noise: descriptor list FOR noise WITH HAVE Boolean Expression *

READ COMMANDS FROM TAPE

READ ENVIR BANK

SELECT DESCRIPTORS n descriptor name (specifications) ..... *

SELECT MORE DESCRIPTORS n descriptor name (specification) ..... *

SORT AND INPUT TO MODULES
INPUT TO MODULES noise: descriptor list FOR noise WITH / HAVE Boolean Expression *

SORT AND PRINT
PRINT noise: descriptor list FOR noise WITH / HAVE Boolean Expression *

STOP

TOP

WRITE ENVIR BANK

When ENVIR recognizes an error in a command or data record, it prints a diagnostic message before reading the next command or record. The message contains a number that identifies the error and the number of the card-column in which the error was recognized. A list of all error numbers with a brief explanation of each follows below.

A fatal error causes the immediate termination of ENVIR; a message to that effect is printed. A recoverable error in a command causes that command to be skipped entirely. An error in an input data record causes that record to be skipped. No part of the faulty record enters the compressed file, even though any new states of NAME descriptors appearing in that record before the error, will be entered in the dictionary. If a command or data record contains several errors, only the first is recognized.

When the nature of the error makes its immediate detection difficult, the content of the error message is unreliable. For instance, ENVIR skips by scanning for an asterisk in the input string of characters, and consequently when a required asterisk has been omitted the error is recognized only when reading the succeeding command(s) or record(s). Both the faulty and the following record or command will be cancelled as a result.

Some error messages are duplicates of others. They have been sustained in this error list because they originate in different parts of the program.

ENVIR ERROR MESSAGES

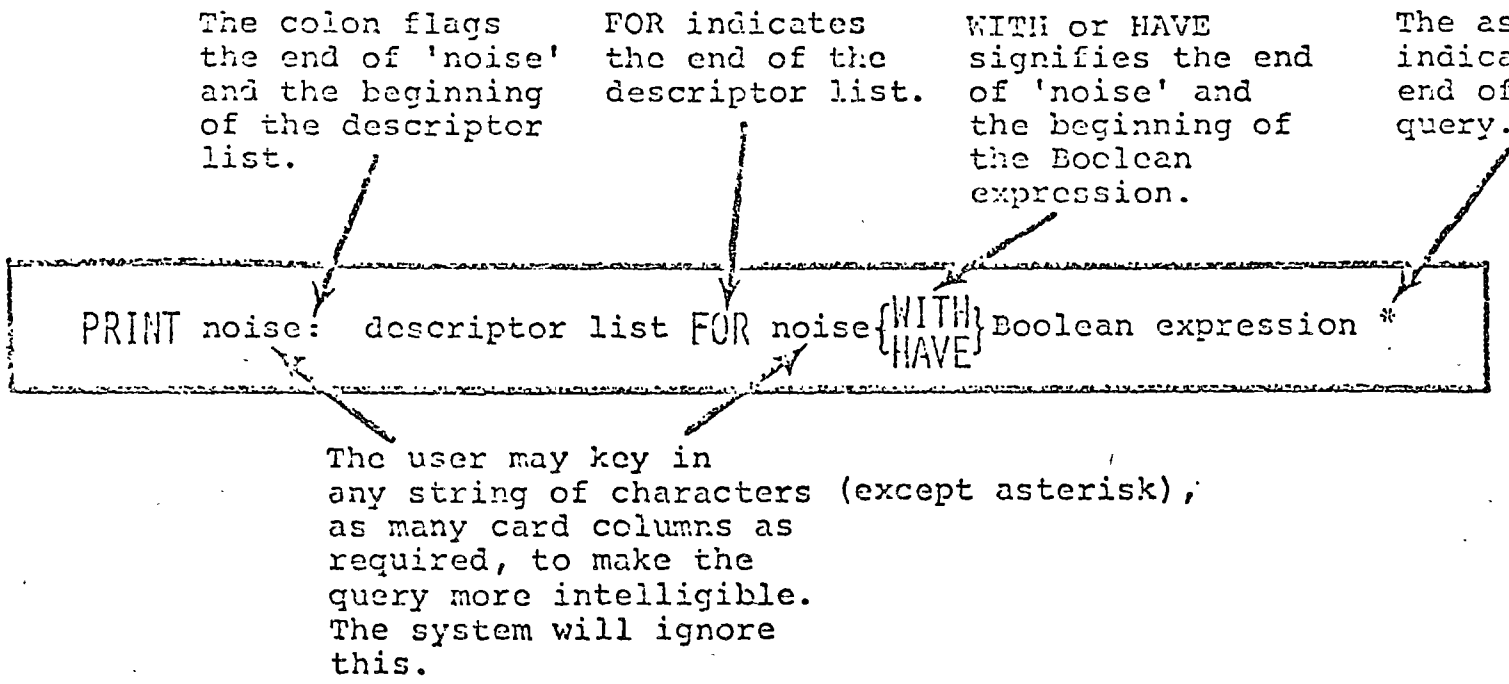
1. UNRECOGNIZABLE COMMAND
2. NOT USED
3. ERROR NOT ISOLATED BY ENVIR (SELECT DESCRIPTORS COMMAND)
4. DESCRIPTOR LIST TOO LONG. DIMENSION PARAMETER 'I' EXCEEDED
5. DESCRIPTOR NAME CANNOT BEGIN WITH A LEFT PARENTHESIS
6. TOO MANY DESCRIPTORS. DIMENSION PARAMETER 'A' EXCEEDED
7. TOO MANY DICTIONARY OVERFLOWS. DIMENSION PARAMETER 'D' EXCEEDED
8. TOO MANY STATES IN DICTIONARY. DIMENSION PARAMETER 'B' EXCEEDED
9. CODED DESCRIPTOR STATES OUT OF RANGE DEFINED
10. DESCRIPTOR SEQUENCE NUMBER GREATER THAN DIMENSION PARAMETER 'E'
11. ILLEGAL DESCRIPTOR OPTION SPECIFIED
12. NUMBER EXPECTED BUT NOT FOUND
13. MAXIMUM NAME LENGTH EXCEEDED (see Appendix to Section 4)
14. ALL BLANK NAME
15. DUPLICATE DESCRIPTOR NAME
16. DESCRIPTOR NAME NEVER DEFINED
17. DUPLICATE DESCRIPTOR-STATE NAME
18. DESCRIPTOR-STATE NAME NEVER DEFINED
19. MISSING RIGHT PARENTHESIS
20. COMPRESSED DATA TOO LONG. DIMENSION PARAMETER 'F' EXCEEDED
21. TOO MANY PRINT LINES PER RECORD
22. SAME AS 19
23. NO COMMA OR ASTERISK AFTER DESCRIPTOR STATE
24. NUMBER OF STATES EXCEEDS ESTIMATE
25. NUMBER OF PRINT POSITIONS ON A SINGLE LINE EXCEEDS THE MAXIMUM (see Appendix to Section 4)

26. SAME AS 18
27. RECORD HAS TOO MANY OR TOO FEW DESCRIPTORS
28. ILLEGAL ESTIMATE OF NUMBER OF STATES
29. NOT USED
30. BOOLEAN EXPRESSION TOO LONG - USE HOLD OR INCREASE DIMENSION PARAMETER 'J'
31. BOOLEAN EXPRESSION TOO LONG - USE HOLD OR INCREASE DIMENSION PARAMETER 'K'
32. UNBALANCED PARENTHESES
33. NO DATA BANK LOADED
34. DUPLICATE DESCRIPTOR SEQUENCE NUMBERS
35. NOT USED
36. CANNOT HOLD DATA BANK AS DIMENSIONED
37. INPUT SOURCE NOT SPECIFIED
38. ILLEGAL USE OF DECIMAL
39. ILLEGAL FROM-TO RANGE
40. SAME AS 39
41. AND or OR BEGINS BOOLEAN EXPRESSION
42. ILLEGAL USE OF PARENTHESIS
43. SAME AS 39
44. ILLEGAL BOOLEAN OPERATOR
45. NOT or FROM EXPECTED
46. NOT EXPECTED
47. ERROR NOT ISOLATED BY ENVIR (BOOLEAN EXPRESSION)
48. SAME AS 46
49. SAME AS 42
50. NON-NUMERIC CHARACTERS FOUND
51. NUMBER OUT OF RANGE
52. SAME AS 32
53. TO or RIGHT PARENTHESIS MISSING
54. ILLEGAL TERMINATION OF BOOLEAN EXPRESSION
55. AND, OR, NOT or TO EXPECTED
56. SAME AS 42
57. BOOLEAN OPERATOR EXPECTED
58. AND or OR EXPECTED
59. SAME AS 32
60. ERROR NOT ISOLATED BY ENVIR (BOOLEAN EXPRESSION)
61. NO LEFT PARENTHESIS
62. NO DESCRIPTOR OR STATE NAME FOUND
63. ASTERISK OR LEFT PARENTHESIS OUT OF PLACE
64. ILLEGAL NEW DESCRIPTOR STATE
65. TOO MANY NUMERIC DESCRIPTORS. DIMENSION PARAMETER 'C' EXCEEDED
66. NO TO PARAMETER
67. ILLEGAL NUMBER OF BCD FIELDS. DIMENSION PARAMETER 'K' OR 'L' EXCEEDED
68. ILLEGAL EQUALS REFERENCE
69. SAME AS 39
70. NON-NUMERIC SKIP or COPY COMMAND
71. NUMERIC SKIP or COPY COMMAND OUT OF RANGE
72. ILLEGAL ADJUST DESCRIPTOR COMMAND
73. NOT USED
74. NOT USED
75. NOT USED
76. NOT USED
77. NOT USED
78. CANNOT DELETE ITEMS. DIMENSION PARAMETER 'H' EXCEEDED.
79. DECIMAL PLACEMENT OUT OF RANGE
80. DECIMAL PLACEMENT DIFFERENT THAN DEFINED

SECTION 3

FORMAT OF ENVIR COMMANDS

The format of the commands is illustrated, along with a brief explanation of their function. Those parts of the command which are in bold letters must be keyed in verbatum and the rest is user input. There are no card column requirements. Some of the commands must be terminated by an asterisk. Certain commands use field separators. Commas are normally used as field separators, however the LITERAL command allows the user to choose other symbols as field separators. A colon, and the words FOR, WITH, and HAVE, signify the end of certain operative components within some commands, as shown below.



DEFINITION OF TERMS

RECORD: (Sometimes called an ITEM in other ENVIR documents).

INPUT DATA RECORD: A logical record of data in the input stream.
COMPRESSED FILE RECORD: The binary representation in the compressed file, of the selected fields of each input data record.

DESCRIPTOR: A field in the input data record, or the binary representation of that in the compressed file.

DESCRIPTOR-STATE: The data contained in one field. (STATE is an abbreviation of DESCRIPTOR-STATE).

Example of the above: In a bibliographic bank, each book is represented by one RECORD. AUTHOR, TITLE, PUBLISHER, YEAR, etc. are DESCRIPTORS. JONES, C.B. is a STATE of the DESCRIPTOR AUTHOR.

DESCRIPTOR LIST: A string of descriptor names that appears in several ENVIR commands. It serves to specify the fields that are to be retrieved. See PRINT command.

BOOLEAN EXPRESSION: A logical algorithm appearing in several ENVIR commands. It serves to specify the records that meet the conditions for retrieval. See PRINT command.

NOISE: An arbitrary string of characters (except an asterisk) that the user is free to insert in designated places of some ENVIR commands. These insertions are intended to improve the legibility of the query and are ignored by ENVIR when processing the command.

FIELD SEPARATOR: A character used exclusively to separate fields in data records and commands. A comma is assumed by ENVIR, unless any other character is substituted, by means of the LITERAL command. In the examples of this document, a comma is used.

COMPRESSED FILE: A two-dimensional bit matrix that is ENVIR's internal representation of the input data.

ENVIR BANK: A pair of files containing 1) the compressed file and 2) the dictionaries and other information applying to the same data.

KEYWORDS: Eight words, listed below, which have a particular significance to ENVIR and may be used only in commands, as directed. The LITERAL command removes this restriction.

FOR: signals the end of a descriptor list in a command.
FROM: precedes the minimum value of a FROM-TO descriptor's range in a descriptor specification or Boolean expression.
HOLD: references the data subset defined by the preceding query.
SAME: references the descriptor list of the preceding query.
TO: precedes the maximum value of a FROM-TO descriptor's range in a descriptor specification or Boolean expression.
UNKNOWN: the value, different from zero, stored by ENVIR as the representation of a field found blank on the input data record, provided that the corresponding discriptor had been defined.

The keywords WITH and HAVE differ from the above in that they can always be used in the input data. In a command, either one signals the beginning of a Boolean expression.

BOOLEAN OPERATORS: The three words AND, OR, NOT used to represent intersection, union, and complement in Boolean expressions. They may not be used otherwise, unless a LITERAL command is issued, in which case they are recognized as operators only if immediately preceded by a period.

ADJUST DESCRIPTORS descriptor sequence numbers separated by field separators *

This command serves to modify the correspondence between the descriptors in the descriptor definition and the fields of the input data records. It must immediately precede the COMPILE ITEMS or (COMPILE AND PRINT ITEMS) command whenever a batch of new records to be read do not exactly match the last descriptor definition made in the bank (by means of SELECT DESCRIPTORS or SELECT MORE DESCRIPTORS commands).

The ADJUST DESCRIPTORS command must contain as many fields as the new input data records. The new correspondence is established by entering a descriptor's sequence number into the appropriate field. (Descriptor sequence numbers are printed by ENVIR in response to a DICTIONARY STRUCTURE command).

Not only the fields affected by changes, but all fields to be read, must contain the corresponding descriptor sequence number. Fields left empty in the command will be ignored on the data records.

BOTTOM

This cancels out the TOP command.

CODE

When this command is issued, the states of all CODE and NAME descriptors printed by ENVIR in response to PRINT or SORT AND PRINT commands, will be represented by their numeric code instead of their alphabetic name.

Numeric codes of both kinds of descriptors are printed by ENVIR in response to a DICTIONARY STRUCTURE command.

COMPILE AND PRINT ITEMS FROM {CARDS / TAPE / DISK}

Causes ENVIR to read data card images on the input medium, build the compressed file and print a list of all input data records.

When ENVIR recognizes an error on an input data record, an error diagnostic code is printed but that record is not entered in the compressed file.

COMPILE ITEMS FROM {CARDS / TAPE / DISK}

Causes ENVIR to read data card images on the input medium, build the compressed file and list only the card images that contain recognizable errors.

CORRECTION (D,S) (D,S)....(D,S) noise {WITH / HAVE} Boolean Expression *

This command allows the user to access any entry in the compressed file and alter it as specified. In each parenthetic pair 'D' represents the descriptor name and 'S' is the state of that descriptor that must be substituted for the existing state in all compressed file records that satisfy the Boolean expression. Particular caution in setting up the Boolean expression is recommended since an error here could damage or destroy the ENVIR bank.

DECIMAL=FREE

When this command is issued ENVIR accepts in the input data records real numbers with any number of decimal places (up to 9) and normalizes each entry to the number of places specified in the SELECT DESCRIPTORS command, by rounding off or adding trailing zeros as needed.

When this command is not issued, any real number appearing on input data records with a number of decimal places differing from its specification, causes the entire record to be rejected.

DELETE ITEMS noise {WITH / HAVE} Boolean Expression *

All the records in the compressed file which satisfy the Boolean expression will be deleted. ENVIR shrinks the compressed file, freeing the space taken by the deleted records. Particular caution in setting up the Boolean expression is recommended since an error here could damage or destroy the ENVIR bank.

DICTIONARY STRUCTURE

This command causes ENVIR to print out the name and the specifications of all descriptors in the ENVIR bank, originally specified by the user in a SELECT DESCRIPTORS or SELECT MORE DESCRIPTORS command.

In the printout the descriptors are arranged in a sequence which corresponds to ascending descriptor sequence number. Within each NAME and CODE descriptor the state names, and the numeric codes assigned by ENVIR to each state, are printed. For NAME descriptors the states are listed alphabetically, and for CODE descriptors the states are listed in the same sequence as originally specified by the user in SELECT DESCRIPTORS or SELECT MORE DESCRIPTORS command.

DISPLAY

This command, which is to be used only when executing ENVIR from a demand terminal, causes the automatic display of the following message:

ENVIR WAITING FOR INPUT

whenever ENVIR expects an input card-image.

DUMP

Once this command has been issued any INPUT TO MODULES or SORT AND INPUT TO MODULES command issued thereafter will produce a hard copy listing of the retrieved data in addition to the binary subset file.

END

Terminates the execution of ENVIR.

The occurence of this as the last command of an ENVIR execution is a requirement for normal termination of the program.

HOLD=FALSE

This cancels out HOLD=TRUE command.

HOLD=TRUE

This command causes ENVIR to retain the data subset defined in each query, for reference in the query immediately following. The retained data subset is referenced by the keyword HOLD. However, the keyword HOLD can be used to this effect without prior issuance of a HOLD=TRUE command when the compressed file consists of only one bufferload.

HOW MANY noise {WITH / HAVE} Boolean Expression *

The NUMBER of records in the ENVIR bank which satisfy the given Boolean expression is printed out.

ID the remainder of the card may contain any string of characters

Page heading. The string of characters supplied by the user will be printed out on top of every printout page until a new ID command is issued. This command should not exceed one 80-column card.

The print image of the command immediately succeeding an ID command will automatically go to a new page.

INPUT TO MODULES
SORT AND INPUT TO MODULES } noise: descriptor list FOR noise {WITH / HAVE} Boolean Expression

This command generates data subsets in the form of binary files legible to several external modules associated with ENVIR.

The generated data subsets consist of all records in the ENVIR bank that satisfy the given Boolean expression.

The descriptor list and Boolean expression are governed by the same rules explained under the PRINT command, except that any pair of parentheses appearing in the descriptor list will be ignored.

INPUT TO MODULES writes the selected records on the output medium in the order in which they occur in the compressed file.

SORT AND INPUT TO MODULES sorts them hierarchically according to the descriptor list. States of NAME descriptors are sorted alphabetically, while those of FROM-TO and CODE descriptors are sorted in ascending numeric order.

LITERAL field separator

Allows the use of a character other than a comma as a field separator in input data records and commands. The selected character is entered following the command word LITERAL. The new field separator may not be any of the characters " > < or blank.

When a LITERAL command is issued, the keywords FROM, TO, HOLD, FOR, SAME, UNKNOWN and the Boolean operators AND, OR, NOT are released for use on input data, and are only recognized in their operational capacity if immediately preceded by one period.

The effect of a LITERAL command remains in force from run to run throughout the life of the ENVIR bank, unless cancelled by the NORMAL command.

MEMO user supplied message *

The user-supplied message may contain any string of characters except an asterisk, and occupy any number of cards.

An asterisk terminates the message.

The message is printed in the output stream following the execution of the last command preceding MEMO.

NORMAL

Cancels out LITERAL command.

NOTE TO MODULES

The contents of the first card following the one containing this command are written by ENVIR at the beginning of the subset data file created in response to { INPUT TO MODULES / SORT AND INPUT TO MODULES } commands. This allows the user to pass on any information required by the module invoked. Such information must be contained within one 80-column card.

PRINT / SORT AND PRINT } noise; descriptor list FOR noise { WITH / HAVE } Boolean Expression *

This command allows the user to define a subset of the ENVIR bank, that will be written on the printer. It provides for some variation in the output format, as discussed below. It identifies the records which satisfy the BOOLEAN EXPRESSION appearing in the command and prints out, from those records, the fields specified in the DESCRIPTOR LIST.

DESCRIPTOR LIST and formatting rules

1. The DESCRIPTOR LIST is a list of descriptor names separated by field separators.
2. If no parenthesis appears in the DESCRIPTOR LIST the descriptors are printed in the order in which they appear in the DESCRIPTOR LIST, one to a printout line, each descriptor indented 5 print positions with respect to the previous. Identical states of a descriptor in consecutive records will be printed only once unless the records differ in the state of a previous descriptor.
3. Any group of descriptors enclosed in a pair of parentheses will result in the corresponding descriptor-states being printed out on the same line, with the remaining descriptors or groups of descriptors printed out on the following lines with a progressive indentation of 5 print positions on each line. Each output line is compared to the preceding one of the same indentation. If they coincide entirely the second one is omitted provided that there are no lines of a lesser indentation that differ between the two records. ENVIR allows for all descriptors grouped on a line the number of print positions required by their longest descriptor-state plus a blank to separate them. If an output line requires more than the maximum allowed print positions (see Appendix to Section 4), that query will not be executed and an error message will be printed.
4. If one query is to contain a DESCRIPTOR LIST identical in all respects, including formatting, to that of the previous query, it suffices to enter the keyword SAME in place of the entire DESCRIPTOR LIST.

BOOLEAN EXPRESSION

A Boolean expression is a string of descriptors and descriptor-states separated by field separators, Boolean operators, or keywords. They must obey the following rules.

1. D,s
   is a Boolean expression if D is a descriptor name and s is a state within the range of that descriptor. The ranges of a FROM-TO and CODE descriptors are specified in the SELECT DESCRITORS statement. The range of a NAME descriptor is formed by the entire set of names appearing in the dictionary of that descriptor. The range of all descriptors include the state UNKNOWN, which is represented in the print out by three dashes.
2. HOLD
   is a Boolean expression representing the subset defined under the command HOLD=TRUE.
3. NOT b
   is a Boolean expression if b is a Boolean expression
4. D, FROM s_1 TO s_2
   is a Boolean expression if s_1 and s_2 are states within the range of the descriptor D and s_1 is smaller than s_2. This form may be used only for FROM-TO and CODE descriptors and in the latter case the s_1 and s_2 are handled by their numeric codes although the user is free to enter their names instead.
5. D, s_1 OR s_2 .....
   is a Boolean expression if s_1 and s_2 are any states within the range of descriptor D.
6. b_1 op_1 b_2 op_2 b_3 .....
   is a Boolean expression if b_1, b_2, b_3 ... are Boolean expressions and op_1, op_2 ... are the operators AND or OR.
7. (b_1 op_1 b_2) op_2 (b_3 .....
   is a Boolean expression as a corollary of the above. Unlimited nesting with parentheses is allowed in the ENVIR Boolean expressions.

SORTING

The SORT AND PRINT command is identical to the PRINT command except that the output is sorted as explained under SORT AND INPUT TO MODULES, while for the PRINT command it is ordered in same sequence as that of the records in the compressed file.

READ COMMANDS FROM TAPE

This command allows the user to execute a series of ENVIR commands stored on magnetic tapes or mass storage file.

READ ENVIR BANK

In any execution of ENIVR that makes use of a ENVIR bank defined in an earlier execution, this command is required once, before undertaking any operation that involves that bank. However, in the execution in which an ENVIR bank is first defined this command must not be issued at all unless a WRITE ENVIR BANK command has been issued.

SELECT DESCRIPTORS n descriptor name (specifications) ..... *

Where n is the number of fields in the input data record. However, see NOTE under SELECT MORE DESCRIPTORS command.

SELECT DESCRIPTORS is the command by which the user enters all specifications of the descriptors in building the compressed file as explained below. This command can be used only once in the life of one bank. Additional descriptors can be specified only by SELECT MORE DESCRIPTORS command.

Every descriptor is specified by an entry consisting of its name followed by a pair of parentheses enclosing its specifications. The specifications are of the form:

(s TYPE 1)

Where s is a unique positive sequence number (Descriptor Sequence Number) indicating the place of that field in the input data record. TYPE is the keyword NAME, CODE, or FROM identifying the type of the descriptor. The parameter 1 is:

.. for NAME descriptor the maximum number of unique states anticipated for this descriptor.
.. for CODE descriptors the list of all anticipated names in the desired sequence separated by field separators. ENVIR will assign to each a positive code equal to its sequence number in this list starting from 1 for the first name.
.. for FROM-TO descriptors 1 takes the form

i TO j

Where i and j are the minimum and maximum values respectively in the range of that descriptor. Both may be of either sign but the positive sign must be implicit, and in all cases j must be greater than i. If the descriptor is a real number the specification must also include, before the closing parenthesis, the following:

DECIMAL K

Where K is the number of decimal places expected in this field in all records. Also in this case i and j must be entered with all their digits but without their decimal point.

The DECIMAL specification may also be followed by an optional specification of LABEL which may be used for integer descriptors as well, and serves to specify units of measurement. This LABEL is of the form

IN A

Where A is a string of characters, one computer word long or less.

.. when two descriptors have identical specifications the name of the second descriptor may be followed by a specification of the form

s=r

Where s is the sequence number of the current descriptor and r the sequence number of the previous one that has the same specifications.

The first descriptor entry may but need not begin in the same card containing the SELECT DESCRIPTORS command words. Other descriptor entries may follow anywhere on the same or succeeding cards. Any character other than a blank, or an asterisk following a closing parenthesis of one specification will be considered as part of the following descriptor name. The command is terminated by an asterisk.

It is not necessary to specify as many descriptors as there are fields in the input data record. However only the fields whose descriptors have been specified will be entered in the compressed file.

SELECT MORE DESCRIPTORS n descriptor name (specification) ..... *

This command can be used to add previously unspecified descriptors to the compressed file. Its form is identical to that of the SELECT DESCRIPTORS command. The parameter n may not be smaller than the value it had before issuing this command. The names and sequence numbers appearing in the descriptor specifications may not duplicate those of previously specified descriptors. All records entered in the compressed file before issuing this command automatically receive the UNKNOWN state for the new descriptors.

NOTE ON THE PARAMETER n. Since the descriptor definitions appearing in SELECT DESCRIPTORS and SELECT MORE DESCRIPTORS commands are cumulative, the use of the latter command may eventually result in defining more descriptors in the bank than there are fields on the forthcoming input data records. The parameter n may not be smaller than the total number of descriptors specified in the bank, regardless of the number of fields on the input data records.

NOTE ON DESCRIPTOR SEQUENCE NUMBERS. When the SELECT MORE DESCRIPTORS command is issued in conjunction with a new batch of input data records whose contents differ from those of the previous batch, a sequence-number conflict may develop in that a new descriptor may now be read from a field previously specified to contain another descriptor. However, the new descriptor must be assigned an as yet unused sequence number, and an ADJUST DESCRIPTORS command must be used to read that descriptor in the proper field.

SORT AND INPUT TO MODULES } noise; descriptor list FOR noise {WITH / HAVE} Boolean Expression *
INPUT TO MODULES

See INPUT TO MODULES command.

SORT AND PRINT } noise: descriptor list FOR noise {WITH / HAVE} Boolean Expression *
PRINT

See PRINT command.

STOP

Cancels the command READ COMMANDS FROM TAPE.

TOP

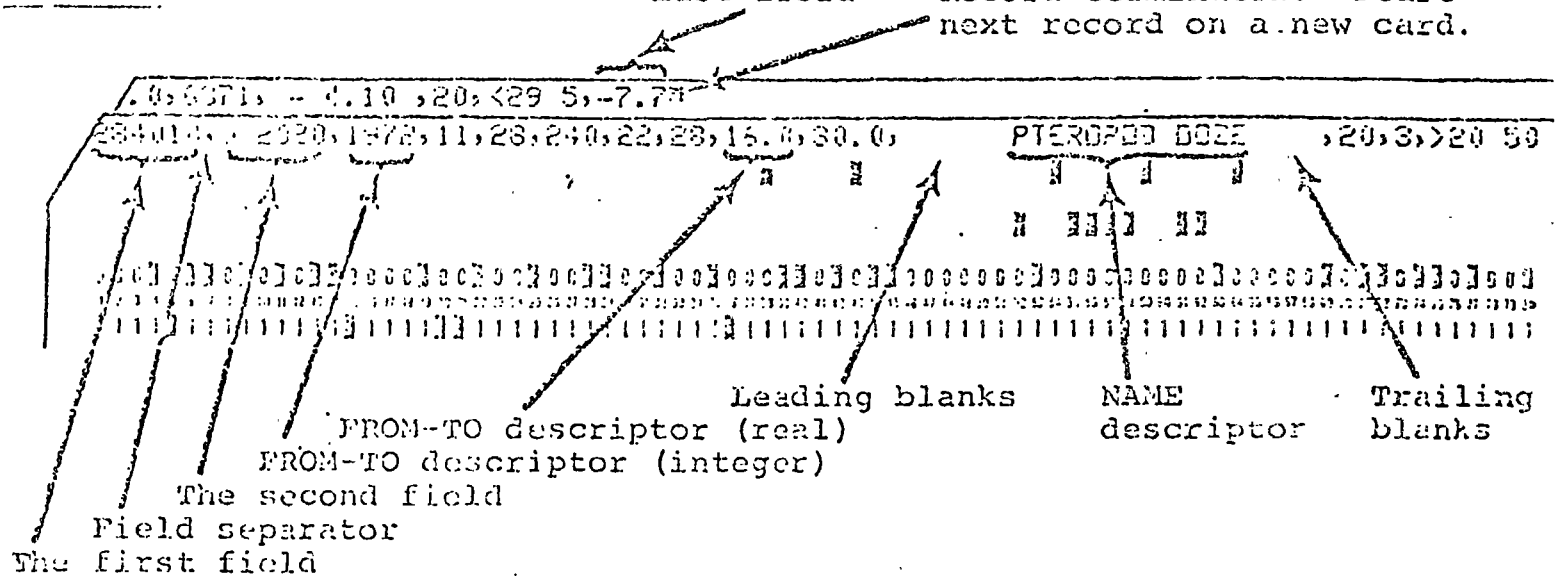
Sets a flag so that the printout of each command and its output (if any) will begin at the top of a new page in the printout.

WRITE ENVIR BANK

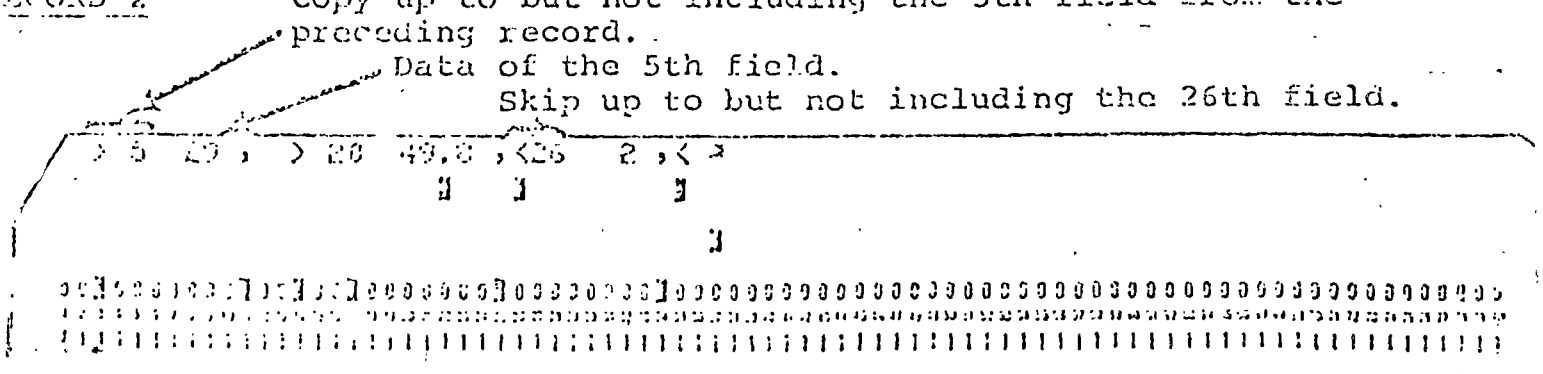
The latest version of the ENVIR bank is written on mass storage for subsequent savi[illegible] on magnetic tape.

COPY - Any number of consecutive fields on an input record can be declared to contain the same data entered in these fields on the preceding record, by punching the single character > immediately followed by the sequence number of the next field to be read. The contents of this field follow the sequence number with at least one intervening blank. If the copied fields are at the end of the record, the sequence number is replaced by the asterisk that terminates the record. If other fields in the record must be read before the copied ones, a field separator must appear between the copy character and the field preceding it. Copy characters can be repeated over any number of consecutive records, to copy the same data, in the pertinent fields, into all these records.

RECORD 1



RECORD 2



Section 5 -- ENVIR INPUT-DATA FORMAT

ENVIR accepts input data in free-field format on punch-cards or as BCD card images on magnetic tape. Data fields must be separated by one field separator. Starting with the first non-blank character in a field the remainder of that field cannot exceed the MAXIMUM NAME LENGTH specified in the Appendix in Section 4. Each logical record is terminated by one asterisk, which follows the last field without intervening field separator. Any record may take any number of cards. A file mark is required after the last record to signal the end of input data.

Leading and trailing blanks in any field are always ignored.

NAME descriptor states may contain any character except the field separator, an asterisk, parentheses and the SKIP and COPY characters (see below). Blanks embedded in the field are kept.

FROM TO descriptor states may contain only numeric characters, one decimal point and one minus sign. Embedded blanks, except following the sign, are considered errors.

CODE descriptor states are handled as NAME descriptor states when a state's *name* appears on input (only names previously specified in the descriptor definition can be used). When a state's *code* appears on input, only numeric characters are accepted.

All records in a file must contain the same number of fields, in the same sequence, as specified in the latest SELECT DESCRIPTORS, SELECT MORE DESCRIPTORS or ADJUST DESCRIPTORS command.

A field need not be of the same length on different records. If a field is empty on input, its value is stored by ENVIR as "unknown", which is different from zero. A field is empty on input when two field separators appear in succession, or with only blanks in between.

SKIP -- Any number of consecutive fields on an input record can be declared empty by punching the single character < immediately followed by the sequence number of the next field to be read. The contents of this field must follow the sequence number with at least one intervening blank. If the empty fields are at the end of the record, the sequence number is replaced by the asterisk that terminates the record. If other fields in the record must be read before the empty ones, there must be a field separator between the skip character and the field preceding it.

Ejercicio:

En el archivo o "datos" se encuentra información acerca de de un grupo de obreros, su registro federal de causantes, el estado y municipio de residencia y el tipo de empleo que desarrollan.

1) Cargar los datos en un sistema imponindo restricciones en los datos.
2) Imprimir el nombre de datos , los empleados del Estado de Guerrero.
3) Efectuar modificaciones a algunos datos.
4) Contar el número de empleado que satisfacen algunas condición e imprimir sus nombres, forma alfabética.
5) Medir el tiempo de respuesta para las diferentes preguntas.
6) Agregar un nuevo dominio llamado PAIS, y agregar el nombre México a todos los empleados.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

Anexo: MATERIAL DE CLASE DE LOS CAPITULOS I Y II

LIC. FERNANDO ARENAS ESTRADA

AGOSTO, 1978.

CIBERNETICA

- Estudio analítico del ISOMORFISMO de la estructura de las comunicaciones en los mecanismos, los organismos y las sociedades.

SISTEMA

- Agrupamiento de partes ordenadas entre sí con un objeto dado
- Conjunto de órganos que intervienen en alguna función vegetativa

SISTEMA : Definición funcional

- Conjunto de cosas o partes
- integradas e interdependientes
- cuyas relaciones entre sí y con sus atributos las hacen formar un todo unitario y organizado
- que cumple un propósito
- y que mantiene un grado mínimo de estabilidad

SISTEMA : Componentes conceptuales

- organización
- integración
- interdependencia (interacción)
- ESTABILIDAD

TEORIA GENERAL DE SISTEMAS

- Serie sistemática de postulados que describen y explican el comportamiento de los sistemas (cualquier clase de sistemas), permitiendo predecirlo con un alto grado de certeza.

TEORIA GRAL. DE SISTEMAS

- POSTULADOS ó PRINCIPIOS -

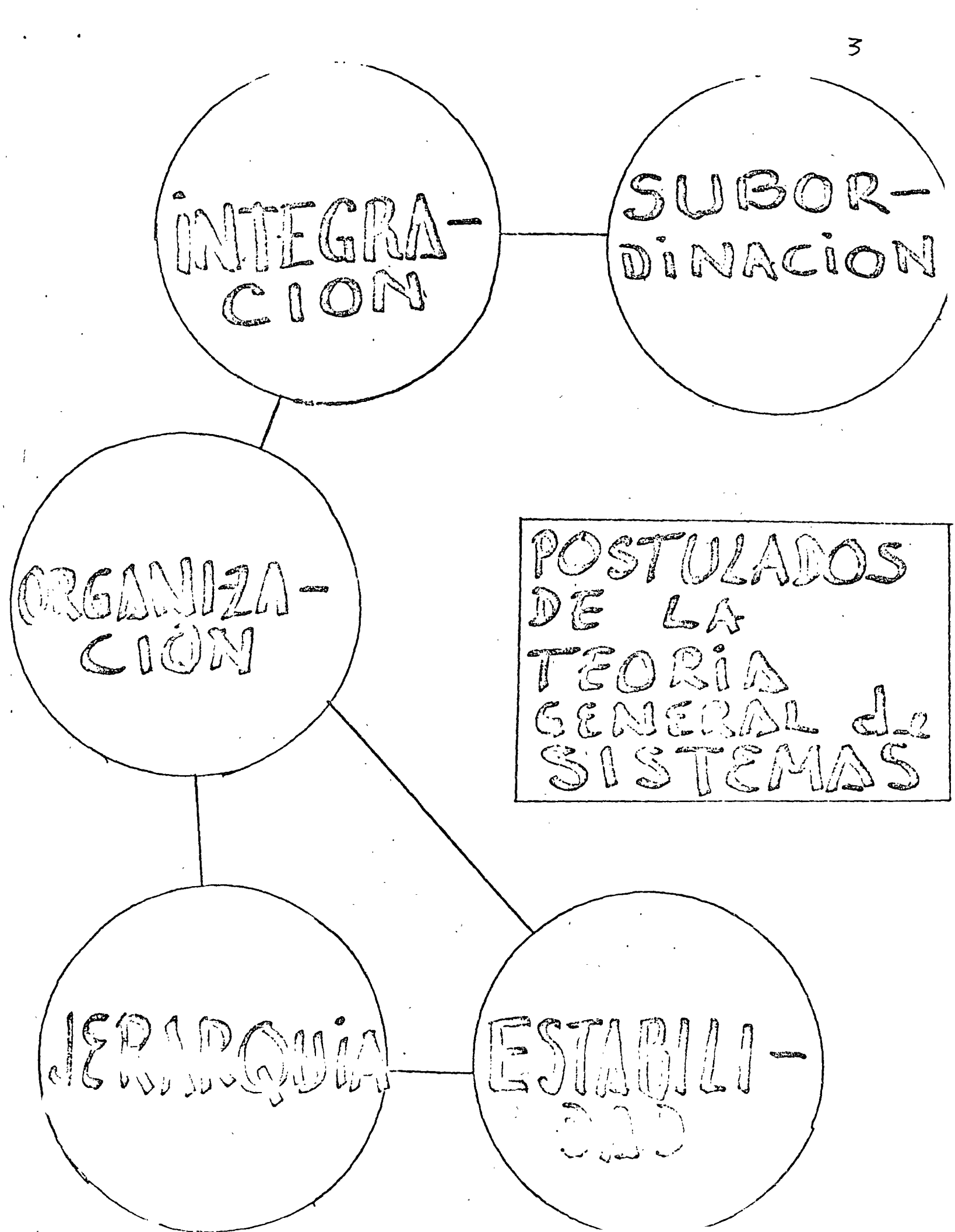
1) INTEGRACION (UNIDAD): UN sistema es un TODO indisoluble cuyas partes están interrelacionadas, son interactuantes e interdependientes; ninguna de las partes puede afectarse sin afectar al resto. El todo se conduce como una unidad, sin importar lo complejo que sea,

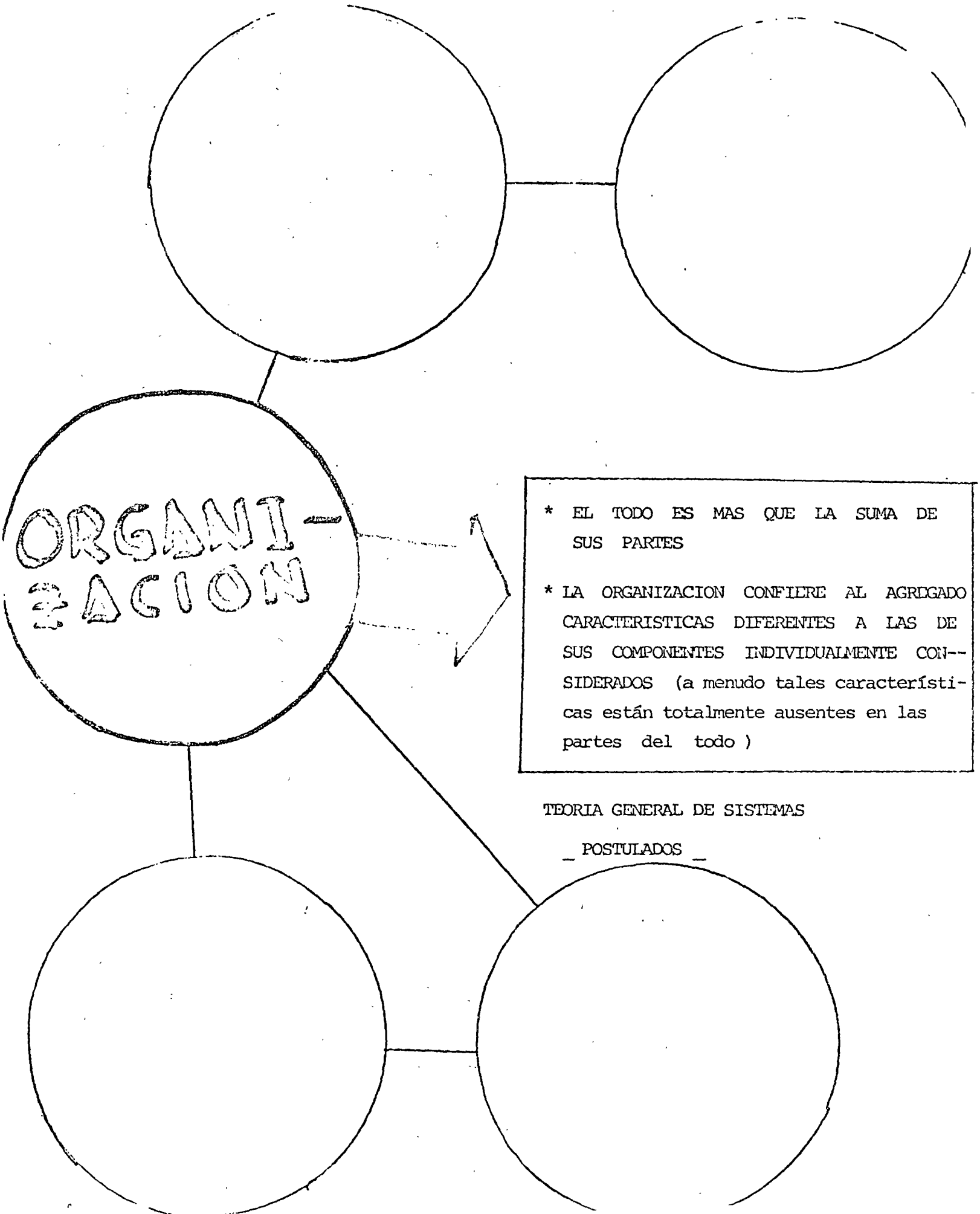
2) SUBORDINACION : El todo es primario y las partes secundarias. El papel que juegan las partes depende del propósito para el que existe el todo. La naturaleza de la parte y su función, se deriva de su posición dentro del todo y su conducta es regulada por la relación del todo a la parte.

3) ESTABILIDAD : La identidad del todo y su unidad se preservan, pero las partes cambian. El todo se renueva a sí mismo constantemente.

4) ORGANIZACION : El todo es <u>más</u> que la suma de sus partes. La organización confiere al agragado características diferentes a las de sus componentes individualmente considerados.

5) JERARQUIA : Las partes de un sistema pueden ser ellas mismas (Subsistemas del sistema), un sistema compuesto a su vez por subsistemas.


INTEGRA-
CION
SUBOR-
DINACION
ORGANIZA-
CION
POSTULADOS
DE LA
TEORIA
GENERAL de
SISTEMAS
JERARQUIA
ESTABILI-
DAD

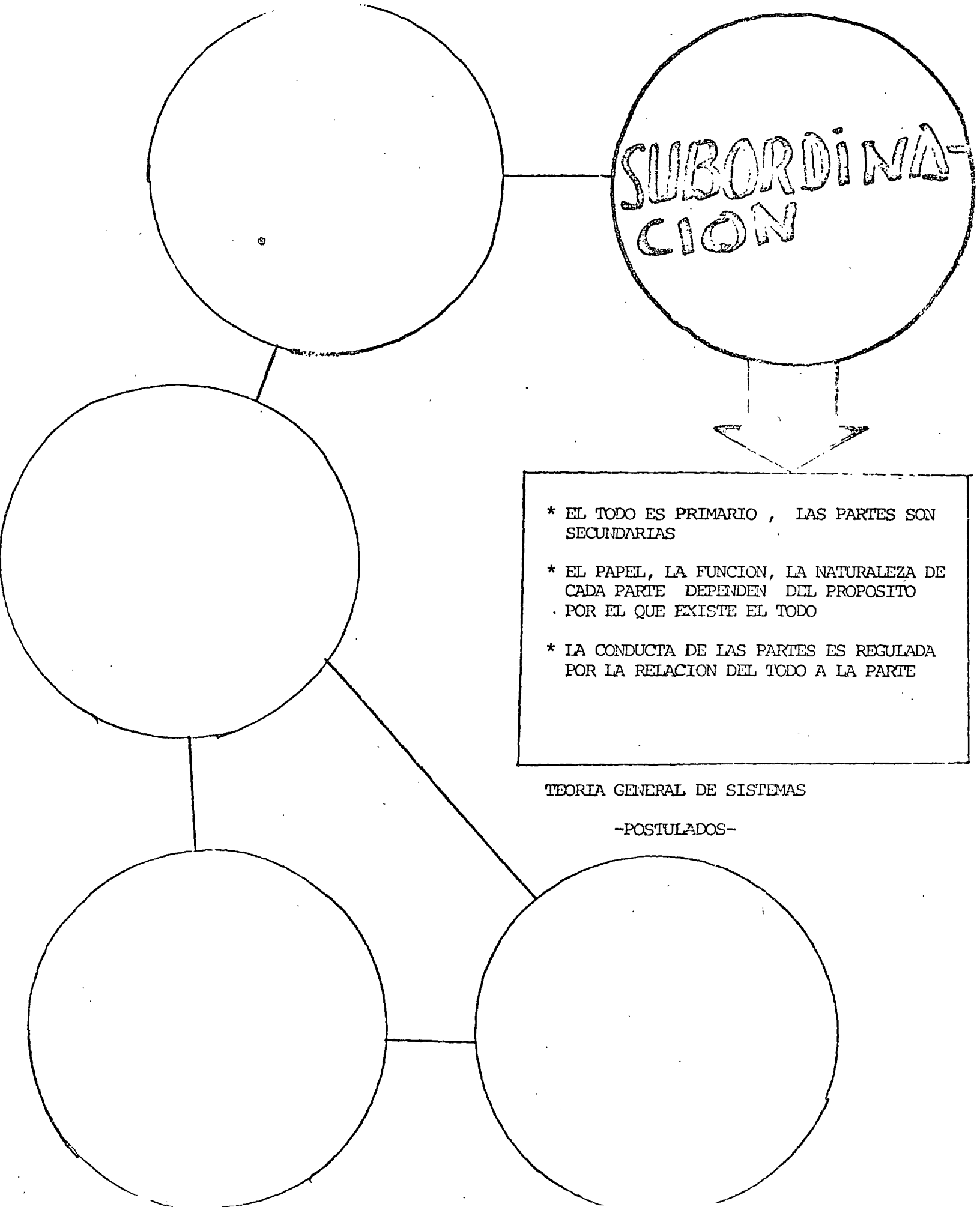


* EL TODO ES MAS QUE LA SUMA DE SUS PARTES

* LA ORGANIZACION CONFIERE AL AGREGADO CARACTERISTICAS DIFERENTES A LAS DE SUS COMPONENTES INDIVIDUALMENTE CONSIDERADOS (a menudo tales características están totalmente ausentes en las partes del todo)

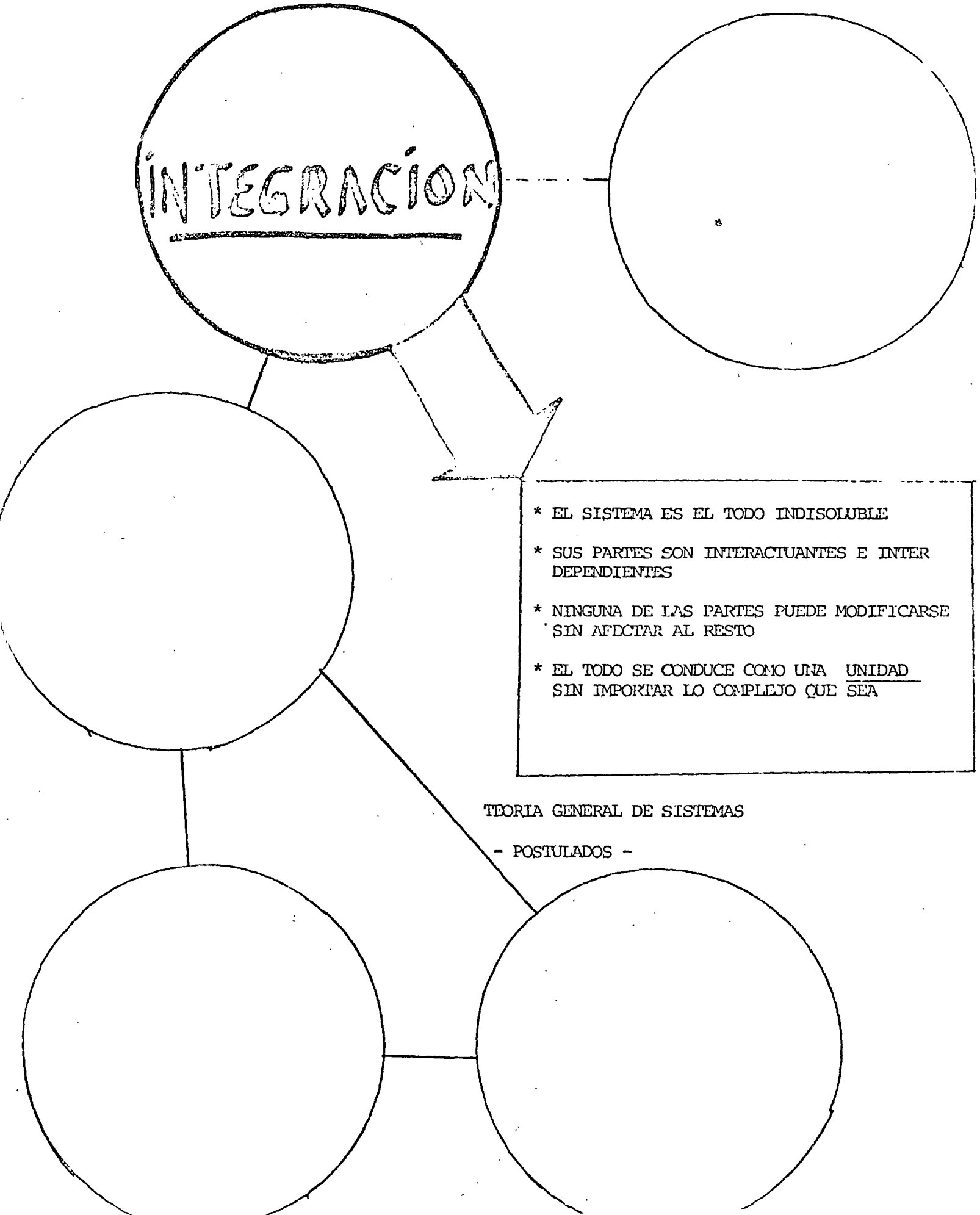
TEORIA GENERAL DE SISTEMAS

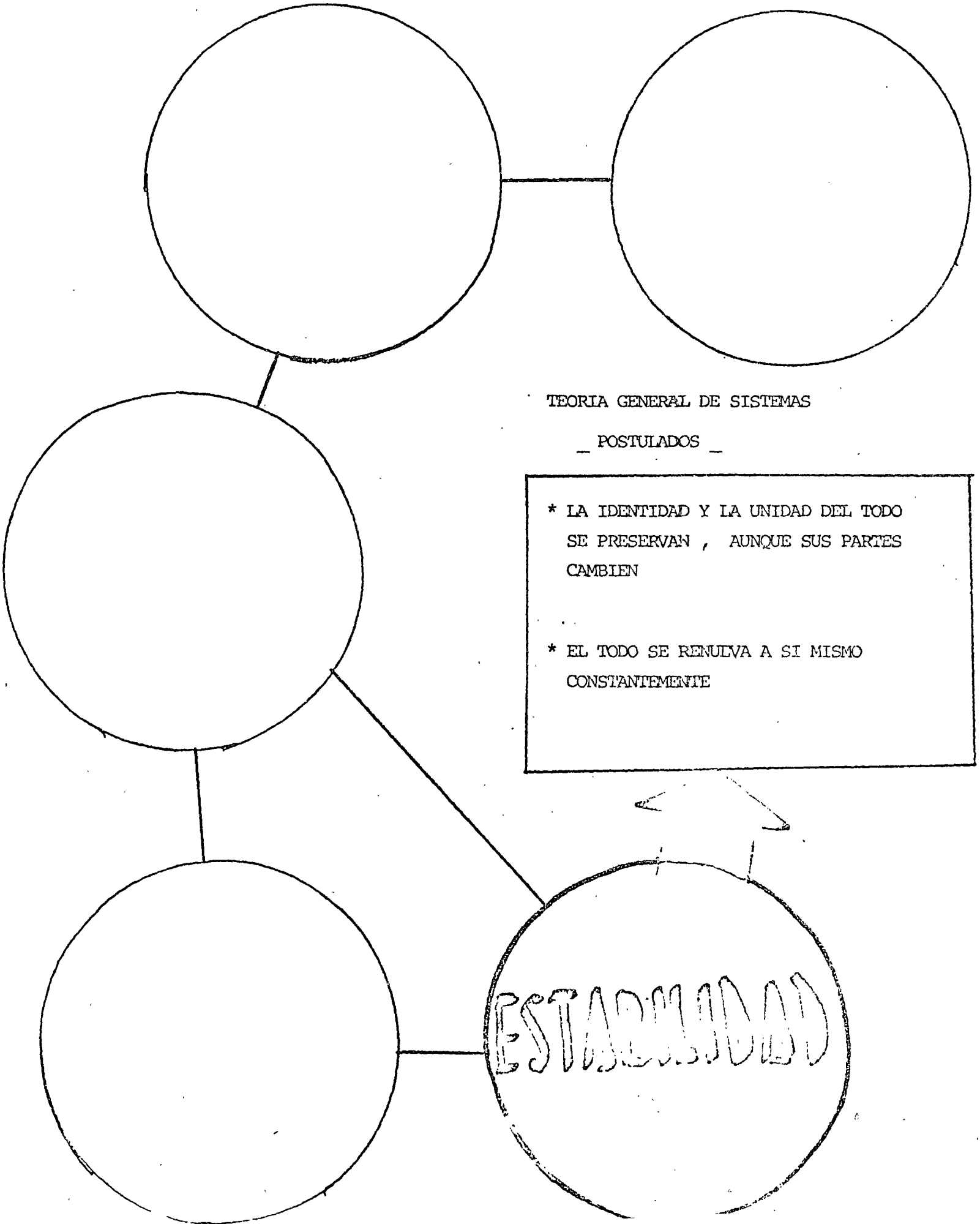
_ POSTULADOS _



TEORIA GENERAL DE SISTEMAS

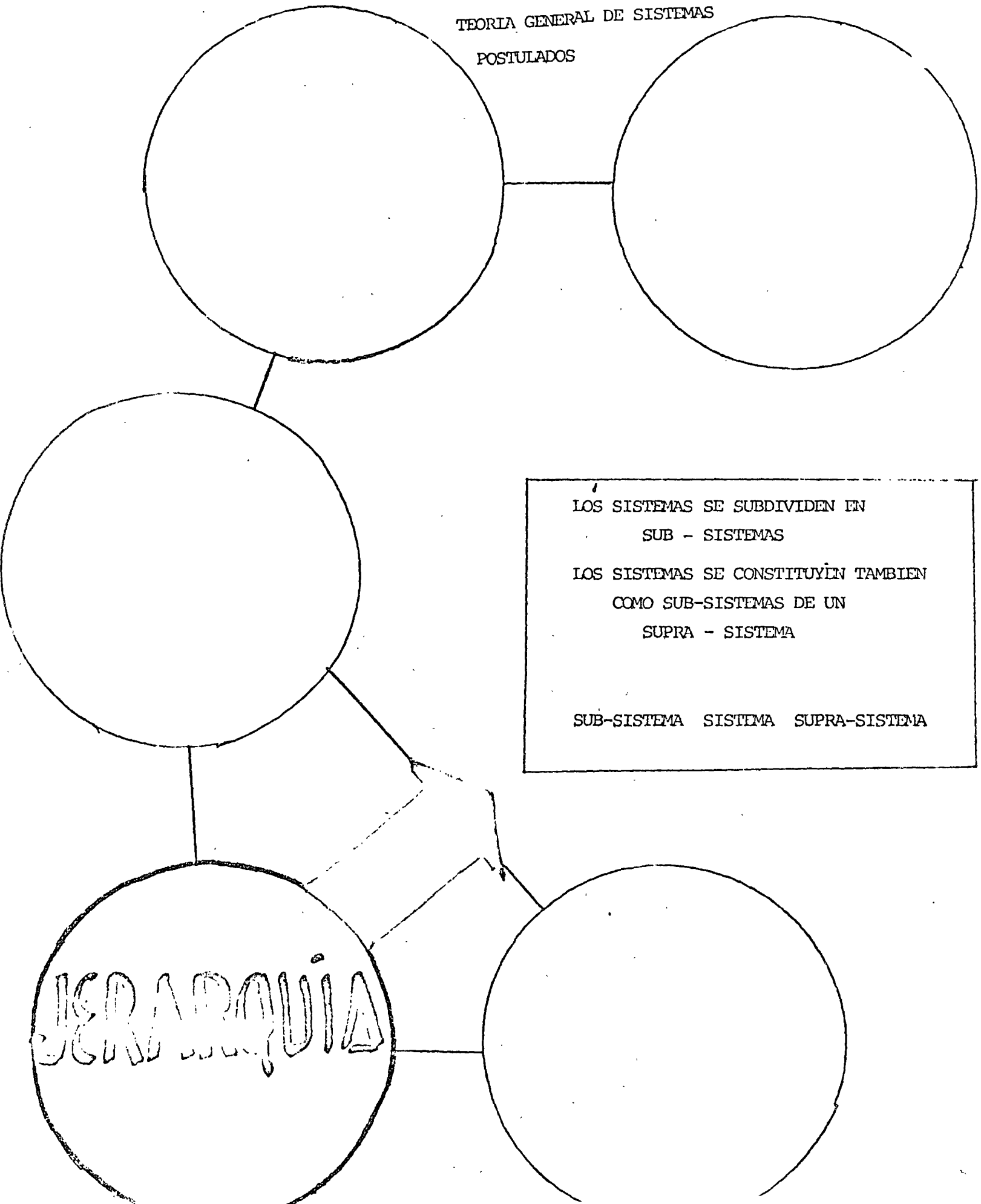
-POSTULADOS-


INTEGRACION
* EL SISTEMA ES EL TODO INDISOLUBLE
* SUS PARTES SON INTERACTUANTES E INTER DEPENDIENTES
* NINGUNA DE LAS PARTES PUEDE MODIFICARSE SIN AFECTAR AL RESTO
* EL TODO SE CONDUCE COMO UNA UNIDAD SIN IMPORTAR LO COMPLEJO QUE SEA
TEORIA GENERAL DE SISTEMAS
- POSTULADOS -


TEORIA GENERAL DE SISTEMAS
_ POSTULADOS _
* LA IDENTIDAD Y LA UNIDAD DEL TODO SE PRESERVAN , AUNQUE SUS PARTES CAMBIEN
* EL TODO SE RENULVA A SI MISMO CONSTANTEMENTE
ESTABILIDAD

TEORIA GENERAL DE SISTEMAS

POSTULADOS



LOS SISTEMAS SE SUBDIVIDEN EN
SUB - SISTEMAS

LOS SISTEMAS SE CONSTITUYEN TAMBIEN
COMO SUB-SISTEMAS DE UN
SUPRA - SISTEMA

SUB-SISTEMA SISTEMA SUPRA-SISTEMA

JERARQUÍA

E L E N F O Q U E S I S T E M I C O

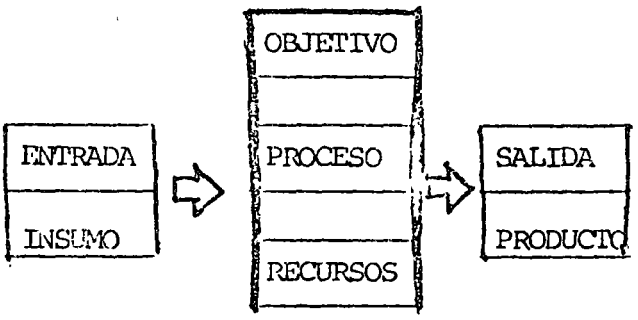
-- Todo fenómeno observable , sujeto de conocimiento puede ser conceptualizado como un SISTEMA; sin dejar de entenderlo como un todo , puede descomponerse todo universo en partes (análisis) para su estudio.

El enfoque sistémico es una nueva metodología científica que tiene su fundamentación en la teoría general de sistemas; puede decirse que en ella se encuentra el germen de un nuevo paradigma científico.

El enfoque sistemico tiene aplicacion no solo en las ciencias básicas como la física, sino -cada vez más- en ciencias sociales y naturales.

DIAGRAMA DE ENFOQUE SISTEMICO EN LA ADMINISTRACION

Elementos de un Sistema



ELEMENTOS DE UN SISTEMA ADMINISTRATIVO

-- OBJETIVO : Finalidad que se pretende alcanzar con la acción administrativa.

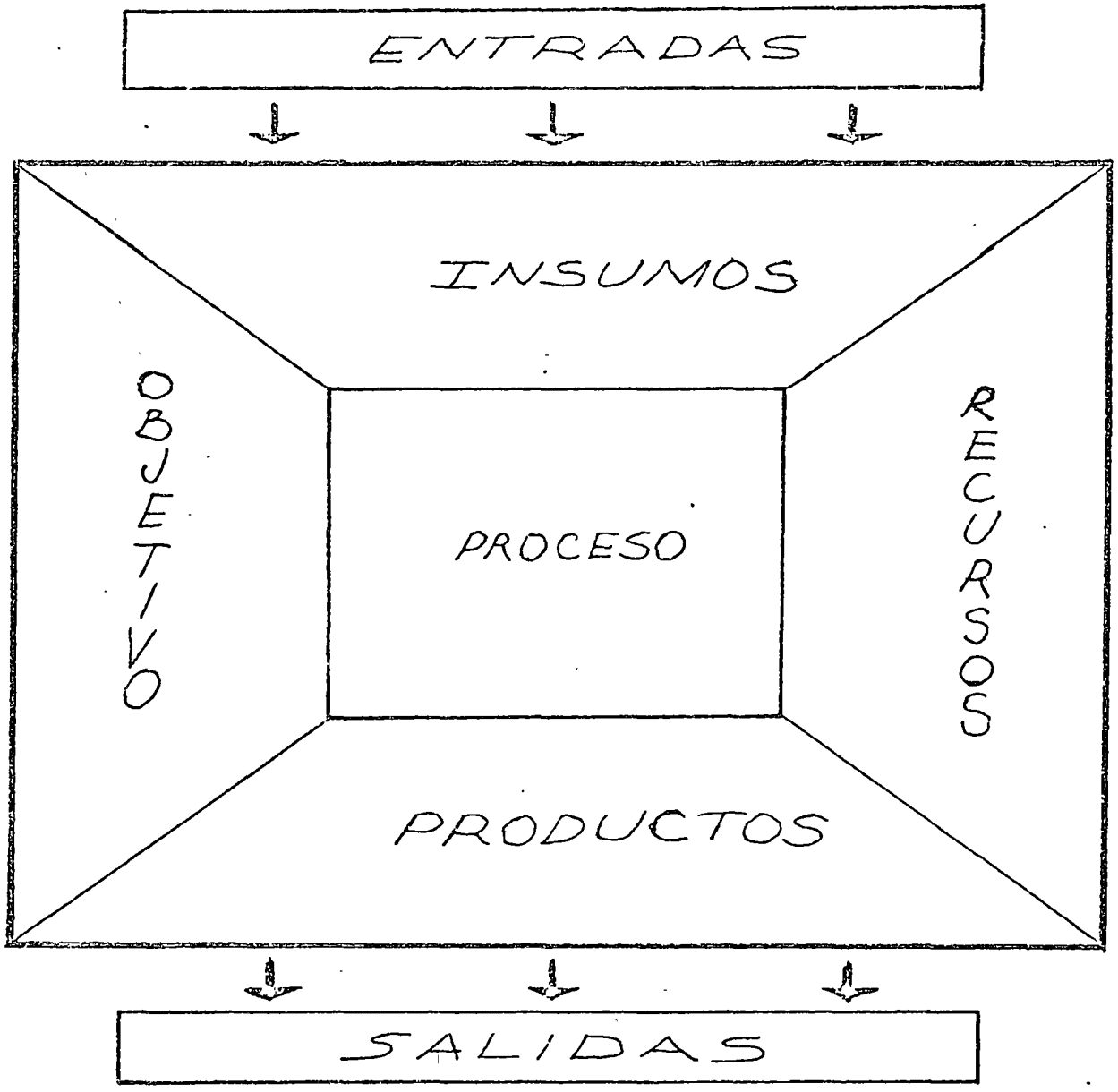
-- INSUMO : Elementos (físicos o informativos) que alimentan al sistema con el fin de ser utilizados en el proceso

-- PROCESO : Secuencia ordenada de los pasos o etapas que comprende la -- transformación del insumo, incluyendo la retroalimentación y el control.

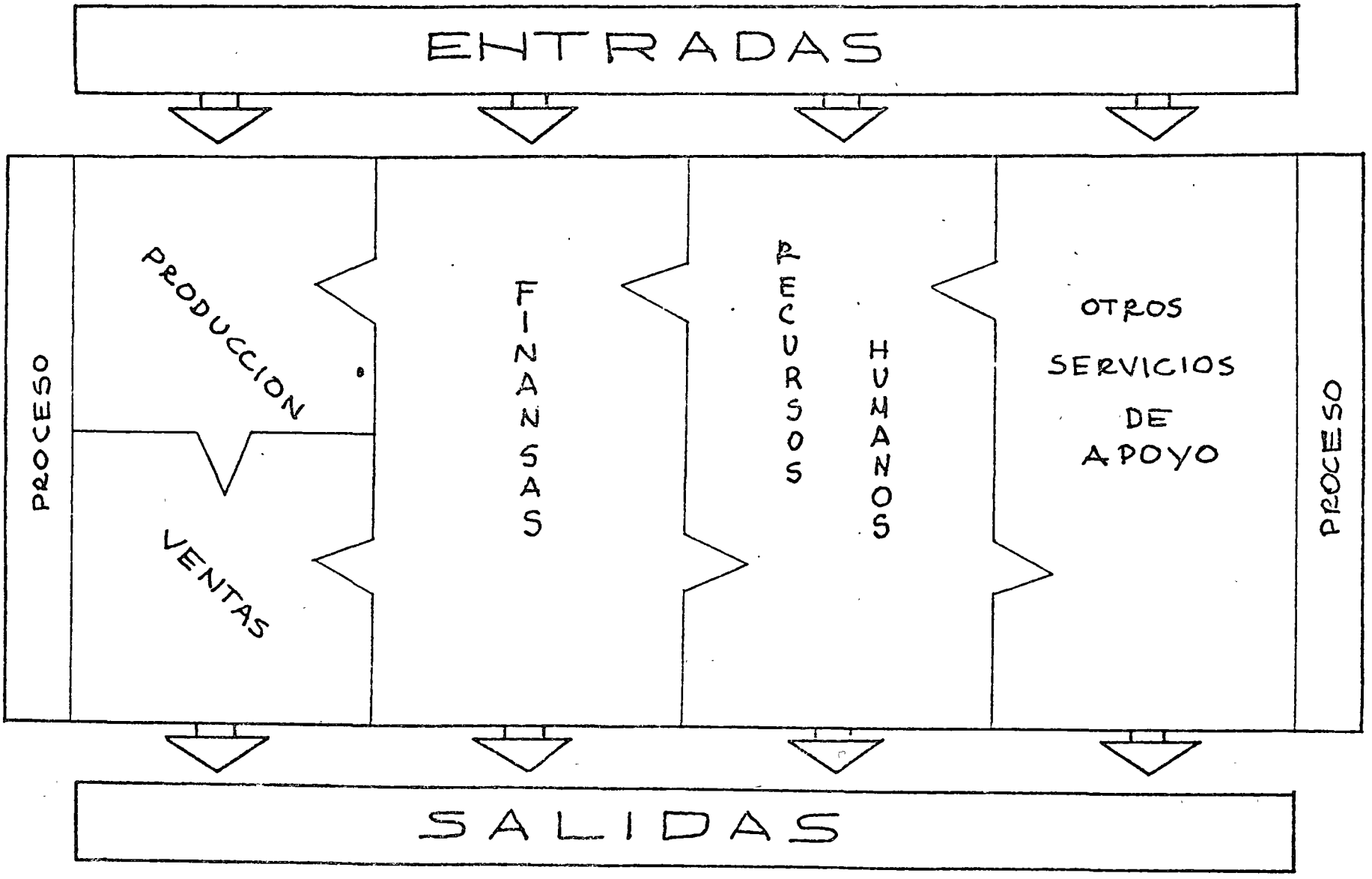
-- PRODUCTO : Son los elementos (físicos o informativos) resultantes de procesar el insumo.

-- RECURSOS : Humanos, físicos y ambientales que sirven como agentes del - proceso.

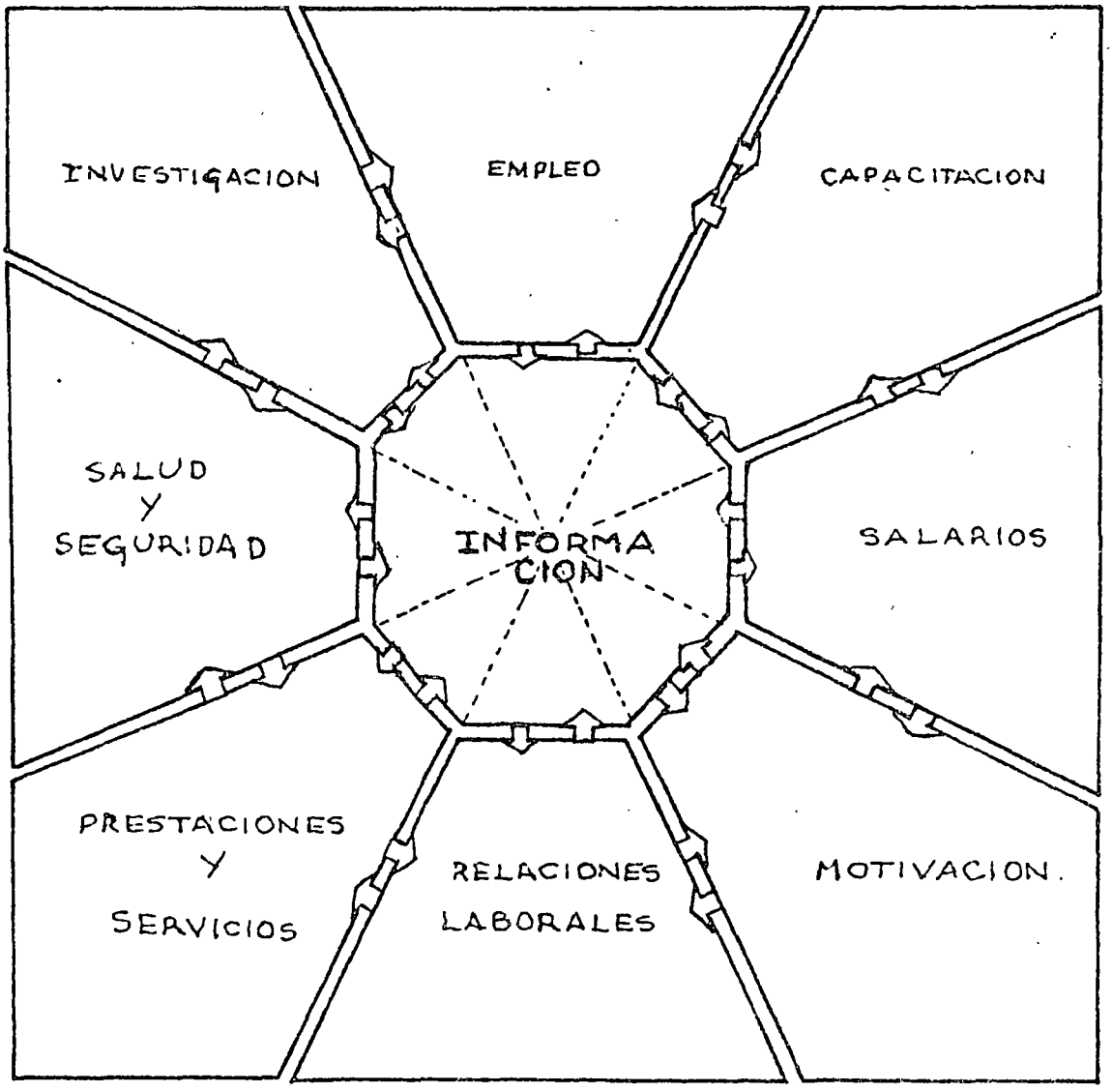
ELEMENTOS DE UN SISTEMA ADMINISTRATIVO



SUBSISTEMAS DE UN SISTEMA ADMINISTRATIVO



SUBSISTEMAS DE UN SISTEMA DE ADMINISTRACION DE RECURSOS HUMANOS



Se llama PARADIGMA al conjunto de logros científicos mundialmente reconocidos que durante una época proporcionan problemas y soluciones modelos a una comunidad científica.

Todo paradigma posee tres características:

1) es importante y atractivo en grado de poder reunir un grupo numeroso y duradero de partidarios en la comunidad científica.
2) es elemental en el sentido de permitir la redefinición y resolución de muchos problemas.
3) es constituyente de una base sólida, la que dándose por descontada, permite al investigador concentrarse en la resolución de problemas concretos, particulares.

UN CAMBIO de paradigma entraña una revolución científica.

El Paradigma de Newton en Física: describe un Universo único en el que - todo discurre con estricto apego a una ley susceptible de conocerse.

-- la causalidad como categoría mensurable

-- la exactitud en la predicción

-- la investigación descubre lo que sucederá en el universo invariablemente

-- existe un determinismo absoluto

El Paradigma de Gibbs en la física: describe una serie de universos (o un universo contingente) en los que se dan probables respuestas a los cuestionamientos del hombre

-- la contingencia como factor de distorción creciente

-- la probabilidad en la predicción

-- la investigación descubre lo que sucederá en el universo con una probabilidad muy grande

-- existe un determinismo incompleto

La ABSTRACCION de un investigador conforma el universo que investiga

La CIENCIA INTENTA DISCERNIR : en que medida son probables (en un número mayor de universos) las respuestas que damos a algunas cuestiones para algunos de ellos

Se llama ENTROPIA al logaritmo de esa probabilidad, y su característica principal es la de ser siempre creciente.

La ENTROPIA SE MANIFIESTA NITIDAMENTE en la COMUNICACION, tendiendo a degradarla

LA ENTROPIA OBLIGA A LA RETROALIMENTACION

LA ENTROPIA es una medida de desorganización

LA INFORMACION es una medida de organizacion

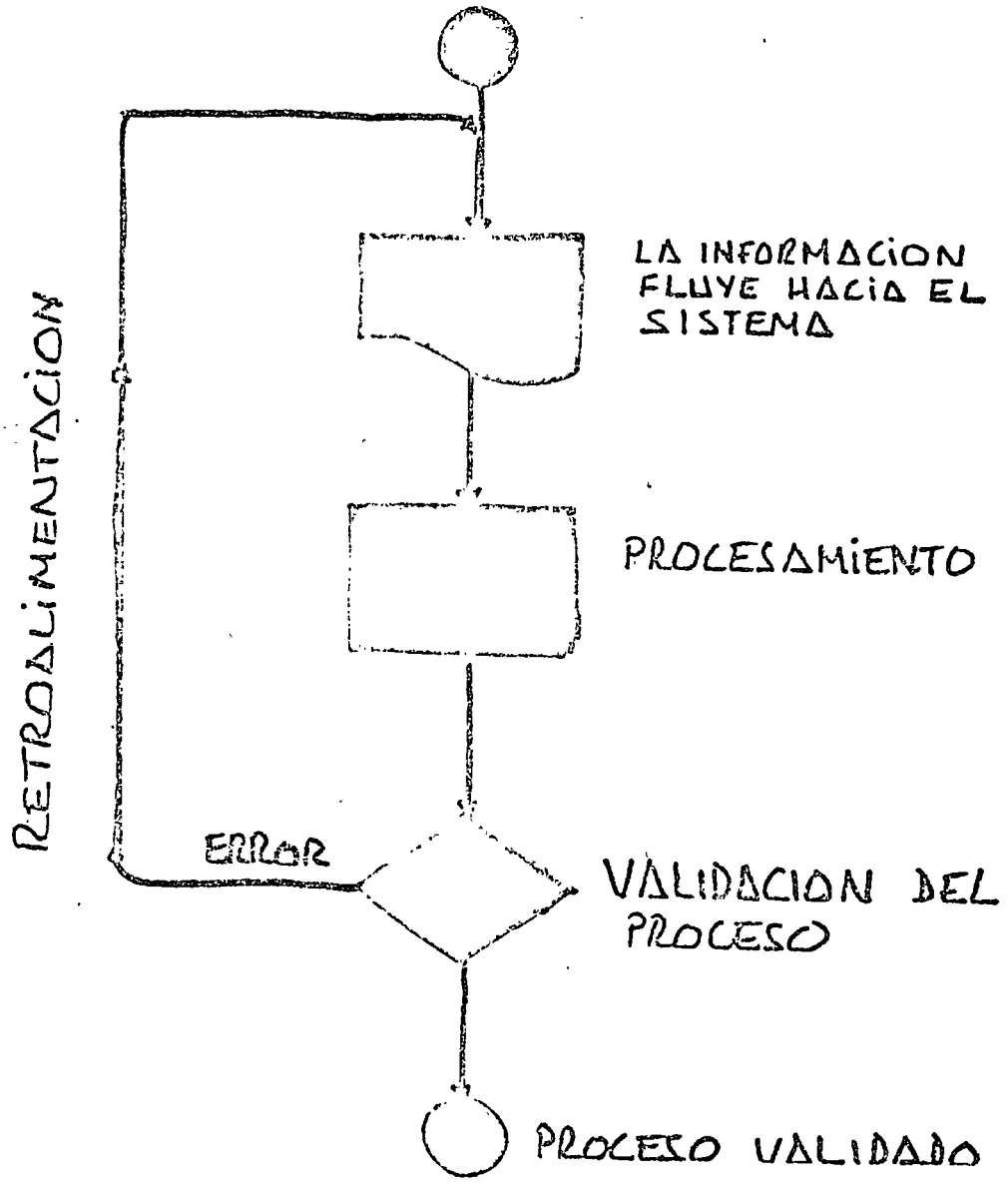
" cuando mas probable es un mensaje, menos información contiene:
un cliche contiene menos información que un gran poema"

TODO SISTEMA DEBE INFORMARSE ACERCA DE LA ACTIVIDAD VERDADERAMENTE EJECUTADA

LA RETRO ALIMENTACION :

* Permite rectificar o ratificar procesos,
* disminuye la tendencia hacia la desorganización "contingente"
* invierte la dirección espontánea de la entropía

LA RETROALIMENTACION



2ª ETAPAS del CICLO de Vida de un SISTEMA

ANALISIS

* SEPARACION DEL TODO (problema) EN SUS PARTES PARA SU ESTUDIO
* COMPRENSION INTEGRAL DEL PROBLEMA
* PROPUESTA DE SOLUCION

DISEÑO

* DESGLOSE DE LA SOLUCION PROPUESTA EN SUS COMPONENTES TECNICOS
* DESCRIPCION DETALLADA DE LOS ELEMENTOS (insumos , procesos y productos) DEL SISTEMA PROPUESTO COMO SOLUCION

PROGRAMACION

* RESOLUCION LOGICA DE LOS PROGRAMAS DEL SISTEMA A ELABORAR
* PREPARACION Y CODIFICACION DE INSTRUCCIONES AL COMPUTADOR
* TRADUCCION DE LAS INSTRUCCIONES (programas) AL LENGUAJE DEL COMPUTADOR Y PRUEBAS METICULOSAS DE CADA UNO DE ELLOS

DOCUMENTACION

* REDACCION DETALLADA DEL SISTEMA --SUS PROGRAMAS --SUS COLECCIONES DE DATOS
  --SUS PROCEDIMIENTOS DE OPERACION Y
  --SUS POLITICAS DE APLICACION

INSTALACION

* PUESTA EN MARCHA DEL SISTEMA EN PARALELO CON EL SISTEMA A SUSTITUIR
* SUSTITUCION DEFINITIVA DEL SISTEMA "VIEJO" POR EL "NUEVO".

OPERACION

* VIDA "REAL" DEL SISTEMA INSTALADO

  (desarrollo normal)

OBSOLESCENCIA

* EL SISTEMA ES REBASADO POR LA DINAMICA DE CAMBIO EN LOS OBJETIVOS
* ESTA ETAPA IMPLICA EL MANTENIMIENTO Y LA REVISION DEL SISTEMA

FASES del

ANALISIS

I IDENTIFICACION

¿EXISTE REALMENTE UN PROBLEMA?

¿SURGIO SUBITAMENTE?

¿QUIENES ESTAN INVOLUCRADOS?

II DEFINICION

¿CUAL es exactamente?

¿QUE LO PROVOCA?

¿DE QUE MAGNITUD ES?

¿COMO PUEDE RESOLVERSE?

III SOLUCION

LA OPTIMA SOLUCION es PROPUESTA

H E R R A M I E N T A S D E T R A B A J O DE LA FASE I

IDENTIFICACION DEL PROBLEMA :

A) MATRIZ DE FUNCIONES INVOLUCRADAS EN EL PROBLEMA Y SUS OPINIONES ANTE EL

* quienes estan inmersos en el problema
* que difusión alcanza el problema
* que actitudes se toman al respecto
* que contradicciones lo agudizan

B) MATRIZ DE CAUSAS / EFECTOS

* ratificación de los aspectos sobresalientes en la Matriz A
* enumeración preliminar de los orígenes NO TECNICOS del problema
* certificación de la existencia del problema mismo
* evolución del problema (calificación de ella)

DOCUMENTO DE LA FASE I

CONTENIDO DE LA

SOLICITUD FUNDAMENTADA DE UN SISTEMA

A) Propósito

B) Perfil del sistema actual

1) Objetivos 2) Funciones 3) Organigramas y Layouts 4) operaciones

C) Perfil del nuevo sistema

HERRAMIENTAS DE LA FASE II

-- ESTUDIO DEL FLUJO DE INFORMACION (Nivel operativo)

-- ENTREVISTAS DIRECTAS (Nivel ejecutivo)

ESTUDIO DEL FLUJO DE INFORMACION

-- descripción de documentos (que, quién, cuando, como, donde)

-- descripción de comunicaciones verbales

-- volúmenes (picos)

-- casuística de resolución; excepciones

-- Matriz (función/actividad) de eventos secuenciados

-- diagramas y ejemplos

ENTREVISTAS DIRECTAS

-- ratificación del ESTUDIO del flujo de información

-- evolución del problema en el sistema vigente

-- confirmación de objetivos frente al sistema vigente

DOCUMENTO DE LA FASE II

MEMORANDA DE COMPENSION

A) Resumen del sistema vigente

B) Resumen de los objetivos del nuevo sistema

C) Inventario de sistemas alternativos

D) Relación de recursos a asignar

E) Aprobaciones

ACTIVIDADES DE LA FASE III

-- revisión de elementos de juicio

-- formulación de Hipótesis para la solución

-- discusión interna de la propuesta

-- discusión abierta de la propuesta

-- presentación de la propuesta

DOCUMENTO DE LA FASE III

A) Alternativas de Sistemas

B) Proposición del Sistema

LAS ALTERNATIVAS DE SISTEMAS :

a) El sistema vigente con ligeras adaptaciones

b) El sistema vigente modificado sustancialmente

c) Un nuevo sistema con capacidad para cubrir la "solicitud fundamentada"

d) Un nuevo sistema que posibilite el desarrollo de otras funciones no mencionadas en la "solicitud fundamentada"

e) Uso de paquetes externos

FASE III DEL ANALISIS

PERFIL DE LAS ALTERNATIVAS (contenido)

A) DIAGRAMAS DEL SISTEMA (entradas procesos salidas)

B) FORMATOS DE INSUMOS Y PRODUCTOS (entradas y salidas)

C) CONTENIDO DE LAS COLECCIONES DE DATOS (archivos) Y ORIGEN DE LA INFORMACION - sobre todo en la utilización de BASES DE DATOS -

D) DESCRIPCION DE LOS PASOS (procesos) PRINCIPALES

E) ESTIMACION DE VOLUMENES (proceso captura almacenamiento)

F) ESTIMACION DE TIEMPOS Y COSTOS DE DESARROLLO Y OPERACION

G) ESTIMACION DE BENEFICIOS (economicos e intangibles)

H) ANALISIS (breve) DE LOS RIESGOS INHERENTES A LAS ESTIMACIONES

I) RELACIONES CON (Y EFECTOS SOBRE) OTROS SISTEMAS.

LA PROPUESTA DEL SISTEMA

Es desarrollada por el grupo de SISTEMAS y presentada al usuario

Debe ofrecer una solución OPTIMA .

Debe mostrar en detalle la CAPACIDAD de la propuesta y sus CARACTERISTICAS OPERATIVAS.

**********SERA DESDE SU APROBACION, EL DOCUMENTO RECTOR DE LAS ETAPAS SUBSECUENTES EN EL CICLO DE GESTACION DEL SISTEMA

CONTENIDO DE LA PROPUESTA:

a) Resumen del problema

b) Breve descripción de las inadecuaciones del sistema vigente

c) Descripción del sistema propuesto

   c.1) objetivos específicos

   c.2) alcances

   c.3) logística

        * Diagrama general simplificado
        * Fuentes de información
        * Configuración de archivos
        * Procesos generales
        * Productos
        * Tiempos-respuesta
        * Volúmenes
        * Control
        * Equipo

d) Resumen del Plan de Desarrollo

   d.1) descripción por Tareas mayores; Puntos de control

   d.2) recursos

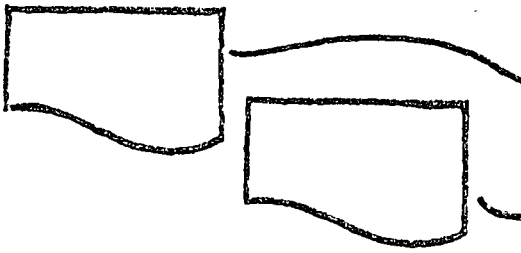
   d.3) responsabilidades

   d.4) costos

documentos presentes en la etapa de ANALISIS

FASE I

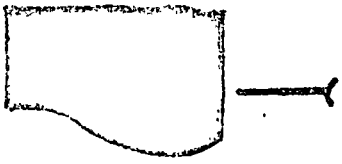
identificación



MATRICES DE

'PROMOTORES / OPINIONES'

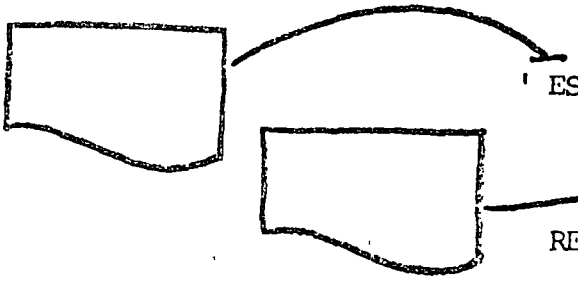
'CAUSAS / EFECTOS'



" SOLICITUD FUNDAMENTADA DEL SISTEMA "

FASE II

definición



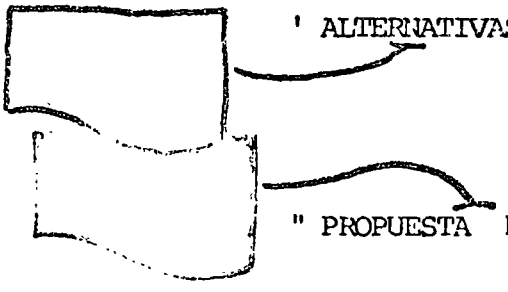
' ESTUDIO DEL FLUJO DE INFORMACION '

RECOPILACION DE ENTREVISTAS DIRECTAS

" MEMORANDA DE COMPRENSION "

FASE III

solución



' ALTERNATIVAS DE SISTEMAS '

" PROPUESTA DEL SISTEMA "

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

REFERENCIAS BIBLIOGRAFICAS

AGOSTO, 1978.

Palacio de Minería Calle de Tacuba 5, primer piso. México 1, D. F. Tels: 521-40-23 521-73-35 5123-123

REFERENCIAS BIBLIOGRAFICAS

Capítulo I.

(1) El concepto de 'Cibernética' nació en 1948 al publicarse el libro de Norbert Wienner titulado precisamente "CIBERNETICS', editado por THE TECNOLOGY PRESS; la definición de Cibernética que se proporciona aquí -sin embargo- parte del libro "THE HUMAN USE OF HUMAN BEINGS" publicado en 1950 por Houghton Miffin Co. en Boston, Mschsts., del propio Norbert Wienner.

(2) Véase "CIBERNETICA Y SOCIEDAD", especialmente el prólog y los capítulos II y III; Ed. Sudamericana, Buenos Aires, Arg. 1958 (2a. ed.).

(3) Diccionario de la Lengua Española, de la Real Academica Española. Madrid 1970.

(4) Duhalt Krauss, Miguel, "La Administración de Personal en el Sector Público: un enfoque sistémico"; I.N.A.P., México 1974; pág. 35.
Definiciones diferentes semánticamente pero comprendiadas conceptualmente en la que adoptamos, pueden consultarse en:
D.N. Chorofas "la Investigación en la Empresa", Aguilar, Madrid 1964; pág. 123, o bien en W. Buckley: "La Sociología y la Teoría Moderna de los Sistemas",Ed. Amorrortu, Buenos Aires, Arg. 1970; pág. 70.

(5) Véase S. L. Opner "Análisis para Empresas y Solución de Problemas Industriales", Ed. Diana, México 1968; pág. 47.

(6) Wilson and Wilson: "Information Computer and Systems Designs", Ed. J. Willey and Sons; New York 1967; pág. 3.
Véase también:
F.W. Martin: "The System Concepto" en "Systems, Organization, Analysis, Management", Ed. Mc.Graw Hill; New York 1969; pág. 49.

(7) L. Von Berthalanffy en conjunción con otros sociólogos y pensadores como Doug Brown, A. Rapaport y otros, fundaron en U.S.A. esta sociedad y editaron bajo el título de la propia sociedad: "Society for the Advancemente of General Systems Theory" recopilación de artículos en torno a este enfoque; Ann Arbor; Michigan 1957.

(8) Frank, H.: "Cibernética, un puente entre las ciencias"; Ed. Zeus, Barcelona, España 1966; pág. 21 y 22.

(9) Remitimos a quien desee profundizar sobre el debate a los trabajos de:
Hall, Arthur,D., "Ingeniería de Sistemas", CECSA, México 1964, pág. 98.
Greniewsky, H., "Cibernética sin matemáticas", FCE, México 1965.

(10) Véanse las publicaciones:
Shoderbek, P.: "Management Systems", J. Willey and Sons, New York 1967.
Martin, E.W.; Op. Cit.
Boulding, Keneth: "General Systems Theory: The Skeleton of Science", Ed. J.Wiley and Sons, New York 1967.

(11) Duhalt Krauss, Miguel; Op.Cit. pág. 41

(12) I.B.M.: "Study Organitation Plan The Approach F208135",Mimeografía 1972;pág.7

(13) Una sólida argumentación al respecto está en el prólogo de:
Wiener Norbert, "Cibernética y Sociedad", Op.Cit.

REFERENCIAS BIBLIOGRAFICAS

CAPITULO II.

(1) Micheal J. Samek, "Business Systems Anaylsis: Problem Difinition" en "The Information Systems Handbook" recopilación a cargo de F. W. Mc.Farlan y R. L. Nolan; Ed. Down Jones-Irwuin, New York 1975, Pag. 530.

(2) Michael J. Samek, Op. Cit. Pág. 534

(3) Boldo-Gaspa, M. Dolors; "Metodología de Análisis, Diseño y Programación" en la revista "Informática" No. 3, Octubre 1975, pág. 34.

(4) Knutsen, K. Eric., "Business System Analysis:...." Op. Cit. Pág. 539.

(5) Burch, John G., "Information Systems: Theory and Practice", Ed. John Willey and Sons, New York 1974, pág. 270.

(6) Burch, John G., Op. Cit. Pág. 270-305

(7) Burch, John G., Op. Cit. Pág. 330

(8) Mills, Harland, "Software Engineering" en la revista "Science" volumen 195 de Marzo 1977.
Véase también, del mismo autor: "Software Development" en la revista IEEE (Institute of Electrical and Electronic Engineers, Inc.) volumen 4, Diciembre 1976, así como "Programing Standard and Control" en "The Information Systems Handbook"...Op. Cit.

REFERENCIAS BIBLIOGRAFICAS

CAPITULO III.

(1) Cyert, Richard y March, James; "A Behavioral Theory of the firms"; Ed. Prentice Hall; Englewood Cal. 1963; Pág. 123.

(2) Lineamientos más concretos acerca de la metodología adecuada para la formulación del cuestionario puede consultarse en:
Seward, Henry H.; "Evaluation Information Systems"; Ed. Down Jones, New York 1975; Pág. 140-145.

REFERENCIAS BIBLIOGRAFICAS

CAPITULO V.

(1) Golub, Harrey; "Organizing Information System Resources: Centralization vs. Decentralization" artículo recopilado en "The Information Systems Handbook"; Op.Cit. Pág. 65.

(2) Martín B. Solomon; "Economics of Scale and IBM System 360" en la revista "Comunications of the ACM" de junio de 1966.

BIBLIOGRAFIA: SISTEMAS EN TIEMPO REAL

- Design of on Line Computer Systems
  Edward Yourdon
  Prentice-Hall

- Design of Real Time Computer Systems
  James Martin
  Prentice-Hall

- Programming Real-Time Computer Systems
  James Martin
  Prentice-Hall

BIBLIOGRAFIA: MODELO RETICULAR

- Computer Data Base Organization
James Martin
Prentice-Hall
2a. Edición.

- ACM Computing Surveys
Volumen 8 Número 1 Marzo 1976

- An Introduction to Data Base System
C.J. Date
Addison-Wesley

BIBLIOGRAFIA: TECNICAS DE DIRECCIONAMIENTO

- Computer Data Base Organization
  James Martin
  Prentice-Hall
  2a. Edición

- Design of on Line Computer Systems
  Edward Yourdon
  Prentice-Hall

BIBLIOGRAFIA: MODELO RELACIONAL

- An Introduction to Data Base Systems
C.J. Date
Addison-Wesley

- ACM Computing Surveys
Volume 8 Número 1 Marzo 1976

- Computer Data Base Organization
James Martin
Prentice-Hall
2a. Edición

BIBLIOGRAFIA

- The Information Systems Handbook

Edited by F. Warren McFarlan & Richard L. Nolan

Dow Jones - Irwin Inc.

- Data Base Design

910 Wiederhold

McGraw-Hill

- Techniques for Direct Access

Keith R. London

Petrocelli Books

- File Structures for on Line Systems

David Lefkovits

Hayden Book Co.

- Data Base Management Systems

Donald A. Jordine

North-Holland

- Management Data Bases

R. Clay Sprowls

John Wiley & Sons

- Manual de Usuario para la Explotación de un Banco de Datos Geográficos

Ernesto Bribiesca y Adolfo Guzmán Arenas

Centro Científico de la America Latina, IBM de México

APENDICES

APENDICE A: DESIGN OF REAL-TIME COMPUTER SYSTEMS
JAMES MARTIN
PRENTICE-HALL
PAG. 195-212

APENDICE B: ACM COMPUTING SURVEYS
VOLUMEN 8 NUMERO 1
PAG. 84-100

APENDICE C: THE INFORMATION SYSTEMS HANDBOOK
F. WARREN McFARLAN
RICHARD L. NOLAN
PAG. 686-712

APENDICE A

15 AIRLINE RESERVATIONS

Many of the world's major airlines now have a real-time system for seat reservations. These systems, however, differ very widely in scope, function, complexity, and cost. Some airlines are on their second generation of real-time system, having started with a relatively simple one and later changed to a more complex and comprehensive system, often from a different manufacturer.

With the hardware available today it can be economic to include functions other than reservations in an airline real-time system. The planning now being done in several airlines is toward an airline *operating* system, of which reservations handling is a subset. The computer complex might handle real-time and non-real-time functions of revenue and cost accounting, crew scheduling, operation scheduling, scheduling of maintenance services, passenger check-in, load and trim calculations, spares inventory control, cargo handling, and so on. It might assist in flight planning and update flight plans while a flight is in progress. An airline operating system may include different computers in different parts of the world connected by communication lines. At the time of writing no such comprehensive system is operating and the problems ahead in achieving such a goal are enormous.

Building the programs and organization for such an integrated system will take many years. Some of the above functions will probably be working individually, possibly on separate machines, before such a comprehensive system is installed. This chapter discusses the data processing associated with reservations—from the moment a passenger first asks what flights are available to the time when he climbs on board the aircraft.

OBJECTIVES

There are three main objects in designing a reservation system:

1. To improve the service given to passengers and potential passengers.
2. To save staff in sales offices and in control offices where reservations are processed and space on aircraft is controlled.
3. To improve the load factor on flights.

If it handles cargo reservations, the objectives will be to save staff and to optimize the loading of cargo. Some systems have been planned with staff savings being the main or sole justification. However, a major payoff, where it can be achieved, is the potential increase in the loading of flights. Most airliners take off today with a large number of seats empty. Many airlines have an average load factor of only about 50 per cent. When this load factor can be increased by better reservation procedures or advertising, it gives a direct increase in revenue for the airline because the cost of flying a full plane is little more than that of flying a half-full plane. If a medium-sized airline can increase its load factor by one passenger per flight this represents an increase in annual revenue of more than a million dollars.

As reservations are being made, certain flights will become fully booked or so close to being fully booked that restrictions must be placed on further bookings. The flight is then referred to as a *critical flight*. However it is a characteristic of the airline business that in the two or three days before the flight leaves there will be a flurry of bookings and cancellations, and on many airlines the average seat sold is sold $2\frac{1}{2}$ to 3 times because of cancellations. For this reason booking limits need to be applied with caution. If the mechanisms for booking were perfect, the critical flights would take off 100 per cent full. However, in fact, they are commonly less than 90 per cent full. It might reasonably be expected that a computer using real-time space-control methods would increase the loading of critical flights from 90 to 95 per cent. If 10 per cent of all flights become critical, this represents an increase of 0.5 per cent in the airline's revenue. For a medium-sized airline, this may represent again about a million dollars per year. This argument applies more strongly to international flights than to flights within America, for example, for which there may be unbooked travellers waiting at the airport.

In addition the airline management should obtain valuable statistics for future route and flight planning. The system could indicate for example, how much business is turned away and on what flights.

The introduction of a relatively simple real-time system in Eastern Airlines enabled it to cut its costs of passenger reservations by about 25 per cent and increase reservation operator efficiency by about 20 per cent. The load factors on criticial flights were increased from 90 to nearly 95 per cent.* Eastern Airlines subsequently ordered a more complex and comprehensive reservation system.

*Figures quoted in *Electronics Weekly* (London), Sept. 4, 1963.

FUNCTIONS OF THE SYSTEM

The reservation process starts when a potential passenger walks into a sales office or telephones to ask what seats are available or to ask for other flight information on a given route.

The first function of the system will therefore be to enable a sales agent to give up-to-date information to the public. Many such people may be asking for seats on the same route all over the country or all over the world. A sales agent may say that seats are available when in fact they have just been sold by a different and far-away agent. To prevent overbooking, high-speed communications to a central coordinating office are required. The inventory of seats available may be kept up to date in a distant computer, and enquiries are made on this from the sales locations.

The potential passenger may then make a booking or several bookings that enable him to complete a journey. A ticket is made out, and the booking must be recorded. The booking will affect the number of seats available for other customers, and so the central inventory of seats must be updated. This is in effect a stock-control problem. It is a critical one because the stock item, an airline seat, is perishable. Once a flight departs, any seats that are left cannot be sold! It is further complicated by the large geographical area over which seats are sold. This is at its most extreme with an international airline. Seats on a flight from London to New York may be sold in Rome, Los Angeles, or Tokyo. The only way to have tight control over this kind of stock is by using fast communication facilities. They need to be fast because in the twenty-four hours before the plane takes off there will be a flurry of activity, many seats being cancelled, rebooked, confirmed from a waitlist, and so on. The sale of one seat can make the difference between a reasonable profit and a loss on a flight, and so control of the last-minute transactions is important.

Unfortunately, simple numerical stock control of airline seats gives rise to many errors. A passenger in his eagerness to ensure that he has a seat may become booked twice. His secretary may make a booking and he, himself, may ring up to confirm it. In one case in the author's experience a passenger was booked no less than fifteen times for one flight. Again, a passenger may cancel his bookings and by a mistake, on the part of the passenger or the airline clerk, they may be cancelled from the wrong flight. This can easily happen over the telephone. It leaves one flight with empty seats not open for sale and another in danger of being overloaded. A remedy for this problem is to use the passenger's name when modifying the inventory. When a booking is made on a flight, the names already booked are scanned to check that the booking has not already been made, and, when a cancellation is requested for a given flight, a check can be made that the passenger in question is, in fact, on that flight. It may be too big a job to scan the other passengers' names at the time the booking is made, and so they will be scanned at a later

time, off line, and an appropriate authority notified of any suspected duplicates or mistakes.

This will reduce the number of "no-shows," that is, passengers booked who do not arrive at the airport desk to check in, and "no-recs," that is, passengers who do arrive but for whom there is no record.

Controlling the selling of seats in an optimum manner is much more difficult on airlines which have flights with several stops, and particularly difficult on long international flights which stop at points around the world and have few alternative flights. Consider a lengthy flight *A-B-C-D-E* for which the airline gains a considerable revenue from passengers who fly from *A* to *E*, but much less from those flying *A* to *B*, or *B* to *C*. It therefore wants to restrict the selling of one-, two-, and three-leg flights if there is a chance that they will prevent the sale of four- and five-leg flights on this route. However there probably will not be a plane full of passengers flying *A* to *E*, and so the restriction must be applied with caution. The pattern in which bookings build up and are cancelled will be quite different for different flights, and so each situation must be judged on its merit. Optimizing the revenue on these bookings is a complex problem in probability analysis that needs continuous real-time monitoring. It is a problem that no airline has yet solved completely satisfactorily.

EXISTING METHODS OF SELLING SEATS

Even in an airline which has installed a real-time system, some cities may be without terminal sets which tell them whether or not to sell seats at a given moment. There may be many locations in the world to which the computer will not be on-line in the foreseeable future, because the communication facilities are either nonexistent or too expensive. Again, a location may not have access to an on-line terminal because its volume of sales is too low. Some other means of controlling sales is needed in these locations.

The old, established methods used for this would still carry on in these places after the introduction of a computer. Methods in common use for this are as follows:

1. *Free Sale*

Space is sold freely at certain locations on certain flights without space-availability information being maintained. This cannot be allowed to occur at a location which may sell too large a volume of seats, and it must be stopped when a flight is near to becoming critical.

2. *Quota Sale*

A central space-control section gives allotments of seats on aircraft to various locations to sell. The size of the allotment is determined by a forecast

of how much each location in question is likely to sell. When a location has sold all of its allocation, it may make a request for more space. A certain time, two or three days perhaps, before the flight takes off the location may change from quota sale on that flight to "sell and report."

3. *Sell and Report*

In this system each sale is reported to a central space-control section. When the seats on a flight become filled beyond a certain point the "flight controllers" observe this and may send status messages to the sales points placing restrictions on further selling. The restrictions may say, for example, that the office can sell no more seats on a flight without requesting permission or that it can sell no more than four seats without requesting permission.

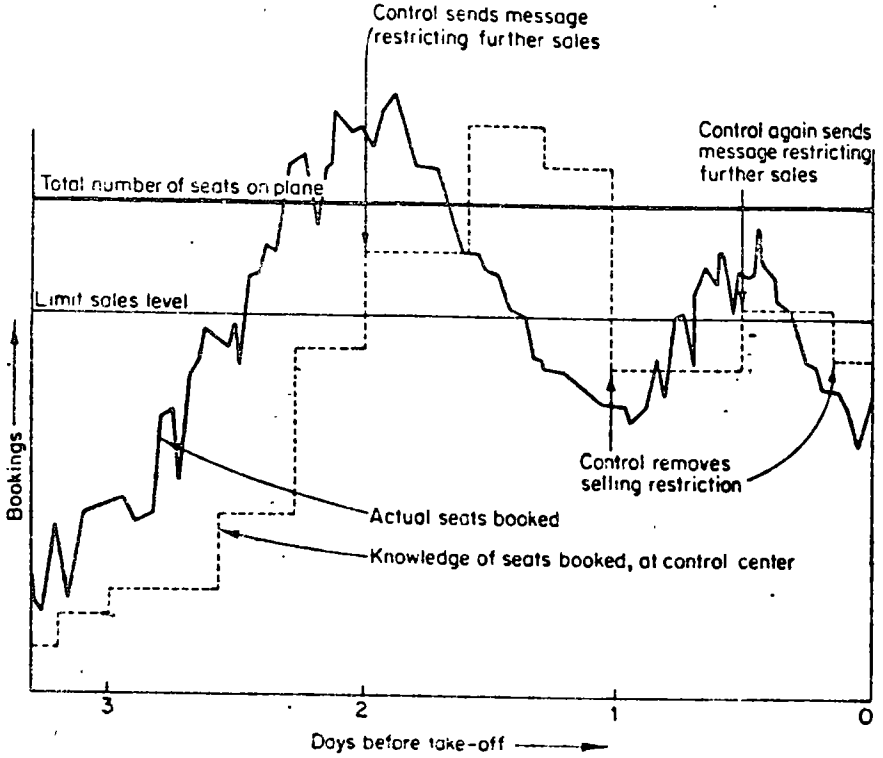


Fig. 15.1. The control of airline seat selling on a non-real-time system. A flurry of bookings and cancellations take place on a flight in the few days before it takes off. Where a time delay exists between the booking of seats and the placing of appropriate restrictions on booking from the control center, oscillations will occur in the manner described in the last chapter. The situation would be much more complex on a flight making several stops, with some passengers flying short journeys and others long, higher-revenue trips.

An internationally recognised standard code format is normally used for these messages.

Requesting permission may be a long process. Teletype messages from the sales office may take half an hour to reach the flight controller, or they may take half a day, depending upon the communication facilities available and the nature of the switching centers. The communication connections across the North American continent, Europe, or across the trans-Atlantic cable are generally fast and efficient; but this is not so to the Far East or Africa, and here "sell and report" has a long time lag.

Where there is a time lag before selling restrictions are placed or before they are effective, the bookings can swing beyond the authorized level. The restrictions are placed to correct this, and then too low a swing may occur before the control center knows about it, as shown in Fig. 15.1. Oscillations of this type occur in a similar manner to those described in the last chapter.

PASSENGER FILES

The airline sales offices must maintain a file on the passengers to whom seats are sold or who are waitlisted. The passenger is likely to contact the office again to cancel his flight or change his routing or ask questions about the booking made. He may have special requirements, such as the need to take a pet or baby or the need for special meals on the plane. The passenger may go to a different office from that at which he made the booking. If he is an American touring Europe, he may go into the Rome office to change his itinerary. The Rome office will have no record of his booking. They may be able to contact the office at which the booking was made but there will be considerable delay as teletype messages are sent and answered. In the worst case the passenger may not know the office where the booking was made. The total clerical labor involved in maintaining the passenger files in all the numerous sales offices is considerable. In the larger office difficulties can arise in this when manual filing is used. The filing clerks sometimes are unable to retrieve records on a customer.

It can be economic to automate this filing process, keeping the records centrally in a distant computer. The terminals and communication facilities used can be the same as those used for making bookings and answering availability queries. It is a large and complex step, however, because there are so many different functions that must be carried out to meet the variety of situations that can arise.

PASSENGER CHECK-IN

The final act of the reservations process takes place when the passenger arrives at the airport and checks in. A manifest is drawn up giving the names of all the passengers booked on the flight in question. This is

transmitted to the airport check-in desk, and passengers are ticked off the list as they arrive. Last-minute alterations are liable to occur at the airport. Passengers may arrive for a given flight when they are not on the manifest. A flight may be cancelled and an attempt made to fit the passengers on other flights, possibly of other airlines. A passenger may arrive too late, and so on. A terminal at the airport desk connected to the distant computer can help in settling these problems and help in allocating the passenger a seat.

WEIGHT AND BALANCE

The passenger check-in process can be of more value if the weight and distribution of the load on the aircraft are taken into account. The freight to be carried by the aircraft, as well as the numbers of passengers, may vary in the brief period before takeoff. If fewer passengers travel, more cargo can be carried, and vice versa. It is valuable to have available an up-to-the-minute assessment of the permissible cargo load.

Records concerning the weight and balance in the aircraft may be set up in the computer files as soon as the flight is opened for check-in. An average value may be taken for the weight of a man, woman, and child. The bookings recorded will originally be used for setting up the weight and balance record. This is adjusted as the passengers check in with the weight of their baggage. Any updating of the booking records causes an updating of the weight and balance records. There is now no need to close the flight for weight and balance calculations half an hour or more before takeoff. Weight and balance control and passenger check-in take place simultaneously. When last-minute passengers arrive, the system will tell the check-in clerk whether they can be accepted for first or tourist class or whether they are acceptable as standby. If an attempt is made to check in a passenger too late, the reply will indicate that the flight is closed.

The advantages of having terminals at the airports to perform these functions are, then, firstly, a considerable saving in manpower is achieved by systems handling these functions. Secondly, it might be possible to close the flight later than otherwise and so last-minute occurrences may be dealt with. In some cases this will result in a greater passenger or cargo loading on the plane. Customer service is improved as the delay is less. Confusion caused by passengers arriving when there is no record of their booking can be dealt with more easily.

The calculation of the load and trim of an aircraft brings in fuel and cargo as well as passengers and their baggage. It therefore ties up with cargo reservations. It is of value to control the booking of cargo space on the same system as that for passenger bookings.

The outputs of the load control system may be warning of overload, cargo loading instructions, load and trim calculations given to the captain

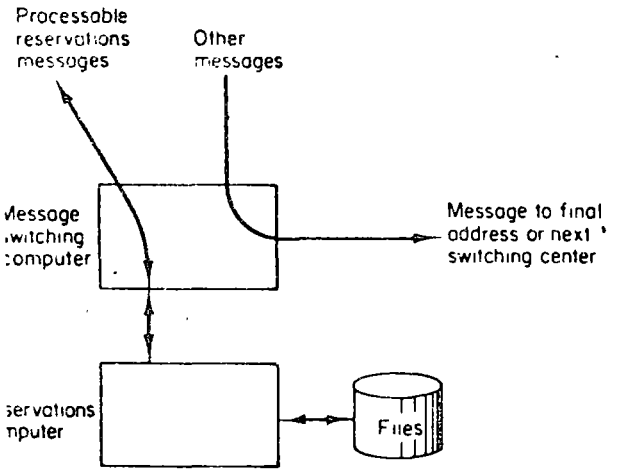


Fig. 15.2. Combination of message switching and reservations.

of the plane, boarding instructions telling the passenger-handling staff how many passengers should board, and messages about the load on the plane to stations on its route. All of this activity requires a large amount of data-message transmitting over telephone and teletype lines. This may need concentrators of the type described in Chapter 20. It will also involve message-switching centers handling the large amount of teletype traffic required. This may be within one country or in various parts of the world for an international airline. They may handle the traffic of one airline only, or they may handle several airlines. They will handle reservations traffic and other administrative traffic, so that some of the messages passing through them will be processable by computer; others not. Figure 15.2 illustrates the relationship of a message-switching computer to a reservations computer in a typical system.

SUMMARY OF SYSTEM FUNCTIONS

To summarize the problem, we list functions that may be automated in airline reservations. Although interrelated, these functions can be thought of separately and in many cases have to be automated separately.

Function 1. *Giving flight information* to distant sales points and, especially, answering availability requests.

Function 2. *Centralized inventory control* of seats booked and cancelled at distant locations. This has been done without passengers' names being used, but to be done efficiently these are needed to eliminate duplicate bookings and invalid cancellations.

Function 3. *Control of space allocation* to offices not directly on line to the computer. From a knowledge of the booking patterns on given routes, allotments of seats to be sold by various offices must be set and, as the bookings build up, status limits must be set.

Function 4. *Mechanization of passenger files.* These have previously been maintained manually in the sales offices. To keep them in the files of the distant computer and maintain them in a real-time manner, using the facilities needed for the above functions, will give major labor-cost savings and improve the service given to passengers.

Function 5. *Waitlisting, reconfirmation, checking ticket time limits,* and other operations concerned with manipulation of the passenger files. If, for example, a passenger does not reconfirm or collect his ticket when he should, the computer notifies the appropriate sales office.

Function 6. *Provision of special facilities for the passenger,* such as renting a car or booking hotels in a distant location or providing a baby carrier, wheelchair, or facilities for pets.

Function 7. *Message switching.* Airlines need large message-switching centers for routing off-line teletype bookings and other messages to the control points. Many of the teletype booking messages will be from other airlines. A computer can be used for this as part of the reservation system.

Function 8. *Passenger check-in at airports.* Manifests giving details of passengers are sent to the airport check-in desks. The check-in clerks have the facility to make enquiries on the distant computer and use this in allocating seats.

Function 9. *Load and trim* calculations done in a real-time fashion just before the airplane takes off can be combined with passenger check-in. Weights of passengers' baggage and estimated weights of passengers are used for the calculations. The results determine last-minute acceptance of cargo and passengers.

Function 10. *Cargo reservations* may also be controlled by the system, and the weight and approximate volume of cargo used in the load and trim calculations.

Not all of these functions are mechanized on all airline reservation systems. Today, however, it is probably economic to put all of them onto the same system in large airlines. Some airlines only mechanize function 3; others, functions 1 and 3; others, functions 1, 2, and 3. Many are now taking the complex step of mechanizing 4, 5, and 6. There is therefore a wide variation in cost and complexity between one airline reservation system and another. Some airlines use separate computers for message switching. Some use a separate system for functions 8 and 9. However, the more the functions can be combined onto one system, the more economic the operation becomes.

In thinking about each of these functions, the systems analyst or system designer must ask for each operation, "What is the response time needed for this?" and, "What degree of reliability is needed for this?" The answers to these questions have differed from one airline to another, and so here again there is a difference in the cost and complexity of the systems.

RESPONSE TIMES

Function 1 above needs a fast response time. The availability of seats or other information must be sent to the terminal while a customer waits behind a counter or at the other end of a telephone line, in conversation with a sales agent. It would be inconvenient if this took longer than twenty seconds. On most systems it takes less than three.

A man in Albany telephones the local office of an international airline and asks about seats on a journey, say to Cairo. The call is routed to New York on Telpak lines and reaches a sales agent with a real-time terminal connected, perhaps, to a computer in Europe. The agent sends an appropriate message to the computer as she talks to the man, and three seconds later she receives a reply indicating on which flight seats are available and on which the customer may be waitlisted.

If the man wishes to make a booking, there is slightly less need for doing this with a fast response time. It would be adequate if the booking were made in five minutes after the customer has rung off or even in two hours in most cases. In some airlines the inventory of seats booked is updated once per day—though this could result in lost passengers. If the inventory records are updated on a purely numerical basis, the terminal making availability requests can update them in three seconds or so with little extra cost. On the other hand, if passenger names are used when the inventory is updated, this would be much more expensive to do with the three-second response time. Some airlines keep two inventories, one updated numerically as rapidly as possible, and the other updated when teletype messages are received containing passenger details. The two inventories are compared at night or at off-peak periods, and the numerically updated one is adjusted for any discrepancies.

The response time for function 3 is geared to the speed at which messages can be sent to the distant sales offices over teletype. The response time for function 4, passenger-information retrieval, needs to be low if the data about a passenger are to be obtained while he is standing at the counter or is on the telephone. The response time for the various operations must be thought about in conjunction with the cost of the terminals and other equipment that is to be used. Some airlines give a response less than three seconds to almost every operation, and this is probably going to be the rule rather than the exception in the future.

RELIABILITY

The computers or the files or other devices will fail every so often and will take a certain time to repair, perhaps two hours. The *mean time to failure* and *mean time to repair* can be predicted for the various units in the system, and so the overall failure rate of the system calculated as in Chapter 6.

Failure of the machinery is likely to cost the airline a certain amount of money. It may result in lost business, annoyance to passengers, inadequate loading of cargo, and so on. The probability of failure can be reduced greatly by duplicating certain parts of the equipment or designing fallback procedures so that vital parts of the work continue at the expense of nonvital parts. It must therefore be decided what degree of reliability it is economic to build into each of the operations.

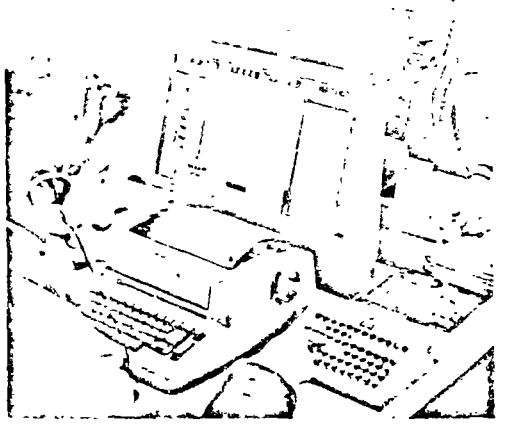
In reservations the actions concerned with planes taking off in the next few hours are vital. Those concerned with the next few days are fairly important, and a quick answer is desirable. However, those relating to seats on planes for next week, next month, or next summer do not demand infallibility and no one will be very greatly inconvenienced if these cease to function for two hours or so. The bulk of the files do not hold data relating to the next few days, and so, perhaps, these need not be duplicated. If a customer rings up about his booking on a flight next week and the file holding his records is out of action, the sales agent can arrange to ring him back later. Similarly, if some of the communication lines are out or the computers are operating in a degraded fashion and only a fraction of the normal transactions can be transmitted and processed in real time, then the nonvital work may have to wait.

Of the functions listed, one that demands highest reliability is perhaps the load and trim calculating. This must be done shortly before takeoff, and it must be reliable, because if it fails the plane cannot take off in safety. The computer center may use two compatible computers, one large and one small. The small one is large enough to handle *only* the vital jobs if the large one fails. Many airlines, however, duplex *all* of the equipment on their system.

EXAMPLES OF TYPICAL SYSTEMS

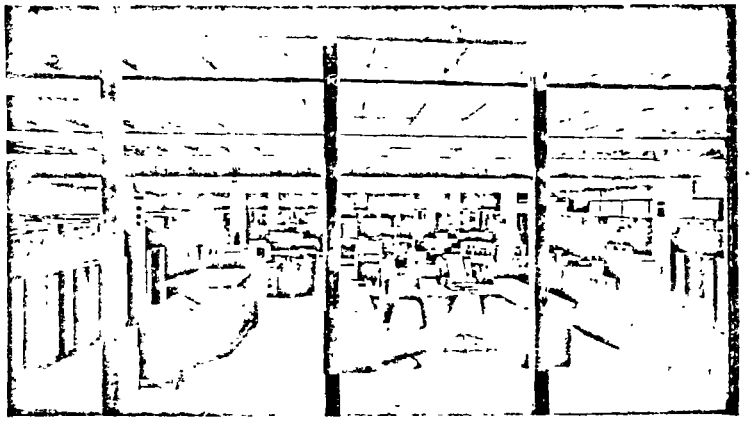
Figure 15.3 illustrates the processing of transactions on American Airlines SABRE System.

As a booking is made the passenger's name, home and business telephone numbers, ticketing arrangements, and other pertinent information are sent to the computer and stored on its disk files. This is done while the passenger is standing at the sales desk or is on the telephone to the sales agent. The computer checks the name with the booking to ensure, at that time, that it is not a duplicate booking. It checks also that all the desirable details for filing have been

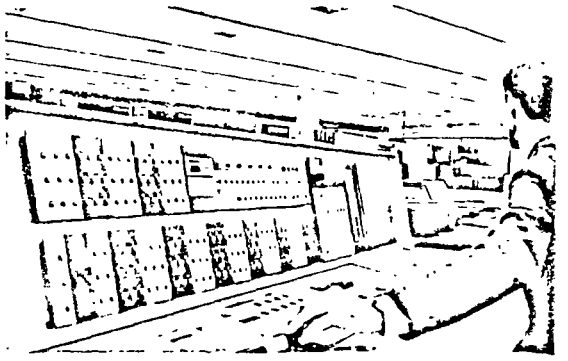


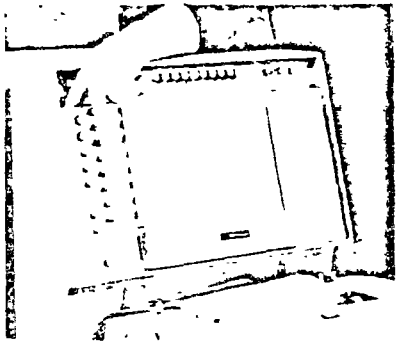
(1) Passenger requests seat reservation by telephone or in person at ticket counter from any of nearly 1,000 American Airlines' agent positions serving more than 50 cities. (2) Agent places card listing all flights to AA designation specified by customer on display rack of desk-size console, keys in number of seats and date requested and presses "need" button.

(3) The operator sends a message over long distance lines to the SABRE computing center at Briarcliff, New York, asking for a seat on a specific flight.

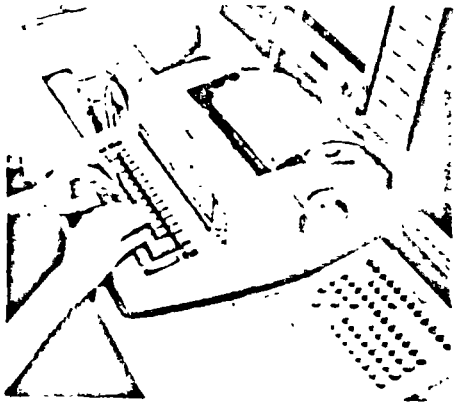


(4) Computer instantly selects appropriate inventory record from its files. If requested seat is available, computer immediately flashes confirmation to agent and at same time automatically records the reservation and subtracts seat from inventory for the particular flight and date.

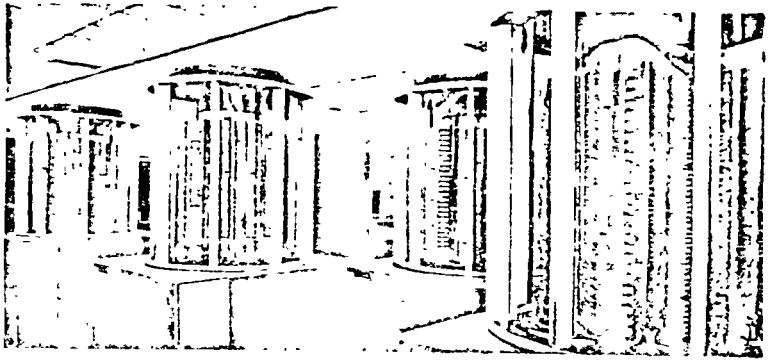




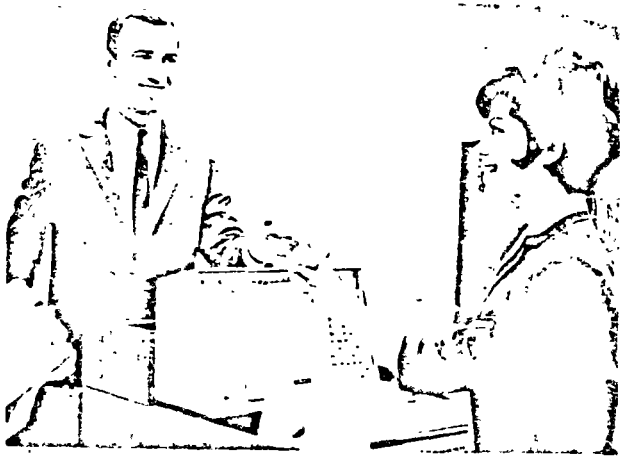
(5) If requested flight is not available, computer responds instantly by activating lights along side the card, thus informing a of



(6) Briarcliff center confirms sale by automatically typing out on printer in front of agent the flight number, date, number of passengers, departure and destination cities, and scheduled departure and arrival times. Agent then uses the console keyboard to type into computer record the passenger's name, home and business telephone numbers, ticketing arrangements, and other pertinent information.



(7) Computer automatically checks and confirms this additional data for completeness and electronically files the information as part of the passenger's record until his itinerary is completed, changed, or cancelled.



(8) The computer checks when the customer picks up his ticket, possibly at a different office, and ensures that the ticket time limit is observed.

Fig. 15.3. American Airlines SABRE system. [In addition to controlling seat inventory and maintaining passenger records, American's IBM system also: (A) Notifies agents when special action is required such as calling a passenger to inform him of a change in flight status. (B) Maintains and quickly processes waiting lists of passengers desiring space on fully-booked flights. (C) Sends teletype messages to other airlines requesting space, follows up if no reply is received, and answers requests for space from other airlines. (D) Provides arrival and departure time for all today's flights.]

entered. In doing this, it enters into a conversation mode with the girl on the terminal many miles away.

Passengers will ring later to confirm, or change or cancel a booking. This happens with the majority of seats sold. While a passenger is on the telephone, the details of his booking will be retrieved from the system, and the agent will make the desired changes. The appropriate seat inventory will be reduced when cancellations or changes are made. The system will check that this is a valid reduction: that the seat being cancelled has in fact been booked. Tight checks and controls ensure that the many errors which occur on a noncomputerized airline reservation system cannot happen here.

The response time of the system to each agent's action must be low enough to permit efficient conversation with the computer while talking to a passenger on the telephone. The American Airlines' contract with IBM says that 90 per cent of all messages must have a response of less than three seconds. In fact, on the working system most of them receive a reply in about one second.

The reliability of the system must be very high. It would lose American Airlines' business if the system was "down" and details of passenger bookings were unobtainable. Every component at the computer center is duplicated, and almost all offices have more than one terminal.

The processing and filing of passenger names in real-time is expensive and complicated. Four-fifths of the massive random-access files are devoted to passenger details. Numerous programs are needed because so many exception conditions arise, and such a variety of situations must be catered for.

Other airlines followed American's lead in setting up a system that handles all details of the reservations process in this way. Pan American has installed a system that is even more spectacular because it is worldwide (see Fig. 15.4). Today many airlines are installing what is sometimes thought of as a "third generation" airline system using graphic display terminals instead of terminals operating at typewriter speed. The cost of such a scheme is high, although it is rapidly becoming less.

Many smaller airlines are thinking of a cheaper, less comprehensive system. This is particularly so with airlines outside the United States and Europe, for example, in South America and Africa. Here many airlines are less profitable than those in the United States. They are often supported by their country's government and prestige caters into the economics.

Figure 15.5 shows a reservation scheme less costly than SABRE and Fig. 15.6 shows an inexpensive terminal which handles numeric transactions only. This system handles reservations and also load control at major airports. It maintains a passenger detail file but does not enter into a conversation mode with the sales agents.

An on-line sales office has a fast response from the computer for availability requests and numerical bookings. The terminal for obtaining this is

She's got your number.

She's got your name. Telephone numbers. Ticket. And your diet, if it's a special one. She's got all the poop on you, if you're flying with us.

Who is our brainchild? She's Boadicea. The most advanced computer system in the entire airline business. (We named her after a British queen who lived back in AD 60. A real lady tiger who proved to be somebody to have on your side.)

How does she work for you? Well, let's say you want to fly our VC10 to Britain. She'll tell you what's available on the day you want to leave. Reserve and confirm a seat for you. All in 3 seconds. Around the world she can do the same. And, if you ever have to use a connecting airline, she'll make arrangements in seconds.

But, say, suddenly you need to change flight dates. Up to 11 months in advance she can tell you what planes and seats are available, plus their arrival and departure times. Then reschedule you in moments.

You wish hotel reservations and a hired car? Okay. They're made.

You want information on international currency, entry visas and vaccinations? She'll tell you all.

You want to take your baby with you? Or Fido? She'll arrange for a bassinet. Let you know if there's available space for Fido in the cabin.

And when you take off, she'll check your baggage in, count how many pieces you have, and make sure it all flies with you. She'll make sure your plane has the most favorable winds so that you can arrive in the shortest time possible. (If your plane is delayed in stack-up, she'll connect anyone who'll be waiting for you.)

Come meet Boadicea. The most sophisticated woman in the whole world. Maybe the most beautiful, too.

BOAC

British Overseas Airways Corporation. 530 Fifth Ave. (& Broadway) New York. MU 7-1900. 262 Broad Street, Newark. MI 3-0660

Fig. 15.4. Customer service: an example of an airline's advertising. The original design and proposal for this system was done by the author.

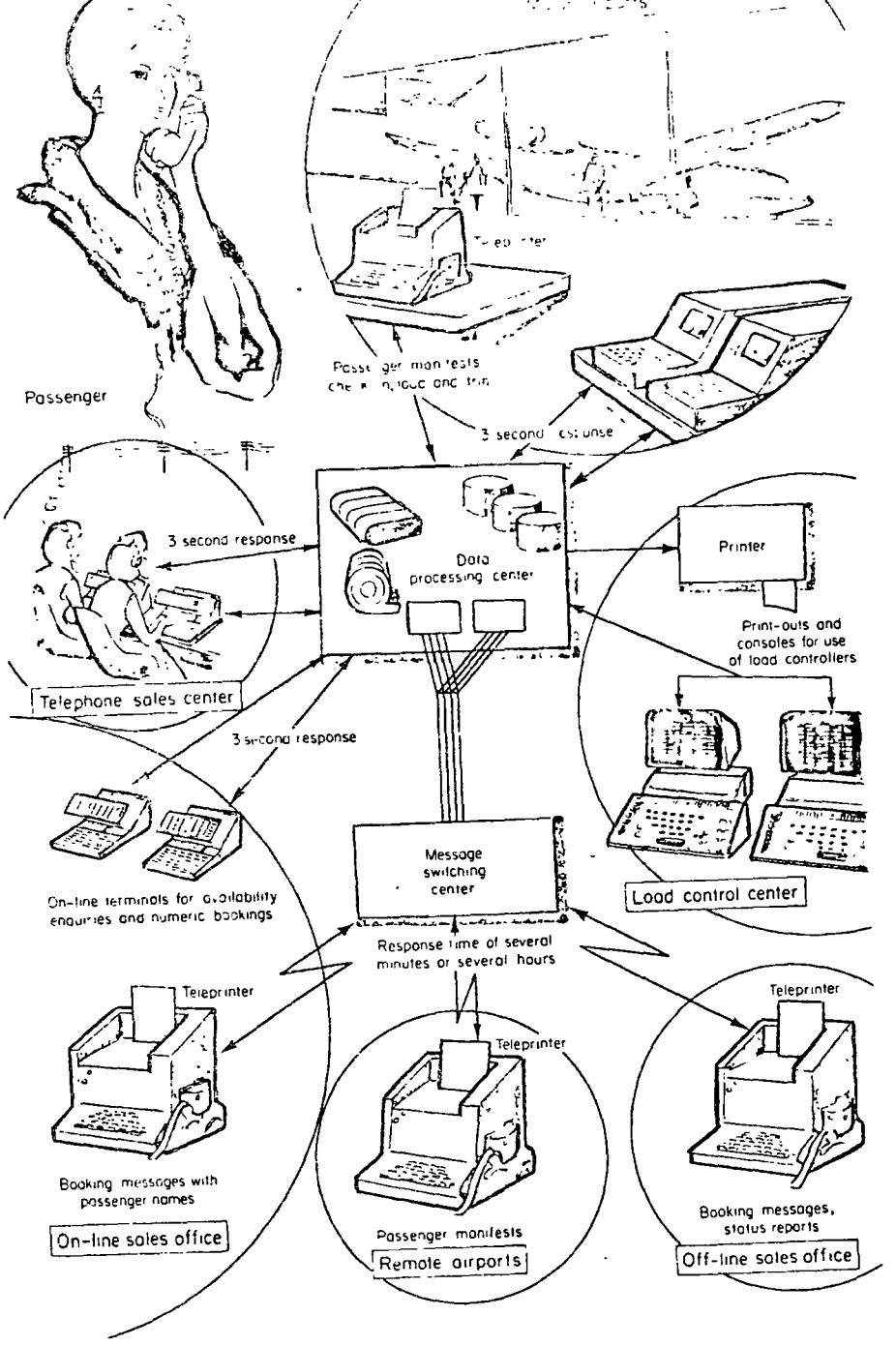


Fig. 15.5. Illustration of an airline reservation system which gives fast answers to availability enquiries but does not give fast access to passenger detail files.

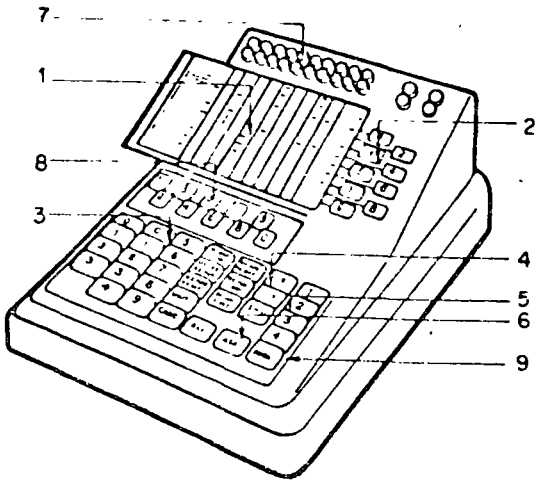


Fig. 15.6. A simple and inexpensive real-time terminal manufactured by Standard Telephone & Cables Ltd., London. It is installed in SAS and BOAC offices and in both these airlines it is soon to be replaced by more advanced terminals.

When operating, the user inserts a flight plate (1) giving flights to the destination in question. She keys in the segment of the flight (2), date (3), class (4), and number of seats (5). She depresses the ASK button (6), and the computer will send back lamp signals (7), thus:

Green lamp: *The seats requested are available on these flights*
Red lamp: *Not available*
Red and green lamp: *Make teletype request*

She selects a flight indicated by the green lamp using key (8) and depresses the BOOK key (9) to make a booking. The booking must then be confirmed by teletype.

used by the sales agent while she talks to a customer. The passenger name and other details, however, will be written down manually as they always have been; when the booking is completed, they will be transmitted over a teletype network to the computer. This may take some time, as the message must pass through switching centers. If there is a query on the message, the computer will send it back, also over the relatively slow teleprinter network.

The computer will check the names of passengers booked on an off-line run to ensure, as far as possible, that no duplicate bookings are made or invalid cancellations deducted from the seat inventory. This will be a less exact process as errors are no longer caught at their source.

It may be necessary to keep two separate inventories: one up-to-date second by second, but likely to be inaccurate because purely numerical booking figures are used; the other hours out-of-date but more accurate. These would be correlated at off-peak periods.

With this system the sales office must still keep its own file of passenger records. Part of the justification of SABRE was the manpower saved in doing this. If a sales office wants to obtain information about a passenger filed in the computer or ask whether certain bookings can be made, it must do this by teletype and the answer may take several minutes or several hours to come back.

In the system illustrated, sales offices without availability terminals on line still have a teleprinter, as indeed they probably had before the computer was installed. This gives them the quota-sale and sell-and-report facilities described above. These will now be controlled by status messages from the computer, which automatically sets selling limits when the sales reach certain figures.

Handling exceptional conditions and possibly the sending of certain status messages will be done by load controllers. The diagram shows these having terminals with cathode-ray tubes. They also have a printer for providing them with flight-status listings. The load controllers are men with considerable experience of the airline's reservations problems. Certain situations needing more human experience than can be built into the programs may be referred to them. A small proportion of the teletype messages reaching the computer will not be machine processable. These are displayed or printed so that the load controllers can take the required action. Again, certain error conditions in messages received can be dealt with by the load controllers with returning the message in question to its source.

Whereas, on this system, the distant sales offices cannot operate in conversation mode with the computer, the load controllers *can*. As with many data-processing situations, the best solution is one which combines the abilities of a machine and an experienced man in the optimum combination.

APENDICE B

sub-schema:

```
RECORD DIVISION
01  PRESIDENT.
    02  PRES-NAME
        03  LAST-NAME  PIC A(10)
        03  FIRST-NAME  PIC A(10)
01  ADMINISTRATION.
    02  ADMIN-KEY  PIC XXX.
    02  ADMIN-INAUGURATION-DATE
        03  MONTH  PIC 99
        03  YEAR  PIC 9999.
01  STATE
    02  STATE-NAME  PIC X(10)
    02  STATE-YEAR-ADMITTED  PIC 9999
```

In this record section we not only eliminate unnecessary records from the sub-schema, but include only those data items of interest to an application program using this sub-schema.

Sample Retrieval Program

Once the schema and sub-schema are defined, application programs can be written to store and access data. The DBTG specifications do not include a special data-base population function; therefore the first program written is normally a data-base load program. We shall, however, assume that a data base does exist for our examples.

The first program presented here finds all of the States that have *more* than one President as a native son. Then we print out the name of the State with its number of Presidents. Queries like this, which involve traversing almost the entire data base and performing counting are well suited for DBTG-type systems.

The DML presented in this section is designed to augment the COBOL programming language. To save space (and to aid those who are not COBOL programmers), the examples use English language descriptions for nondata-base functions such as input/output and computation. For those who know the COBOL programming language, the corresponding code should be obvious.

Our COBOL/DML program begins with the standard COBOL IDENTIFICATION and ENVIRONMENT DIVISIONS:

```
IDENTIFICATION DIVISION
PROGRAM-NAME. SAMPLE-QUERY
ENVIRONMENT DIVISION
  identification of machine environment and
  declaration of non data-base files
  (i.e., standard COBOL files)
```

The IDENTIFICATION and ENVIRONMENT DIVISIONS remain unchanged from those used by standard COBOL. Within the ENVIRONMENT DIVISION, we assign an internal COBOL file to the actual print file, for output of our query.

The COBOL DATA DIVISION incorporates the link to the data base:

```
DATA DIVISION
FILE SECTION.
DB  PRES-ADMIN-STATE-INFO WITHIN PRESIDENTIAL.
FD  REPORT-FILE.
    remainder of data item entries which make
    up a standard COBOL file.
WORKING STORAGE SECTION.
77  PRESIDENT COUNT USAGE COMPUTATIONAL  PIC 999
77  DONE  PIC 9(5) VALUE "01021".
77  NO MORE-SONS  PIC A(5).
77  NO MORE-STATES  PIC A(5).
```

The DB entry specifies which sub-schema and schema this program is referencing. While not required by the specifications, the effect of such a statement in most implementations is to cause the record descriptions from the sub-schema (augmented by schema information) to be copied into the COBOL application program, thus reserving space within the program for each data-base record type (and selected items) which this program may access. The record and data item names declared in the sub-schema are therefore referenceable from the COBOL program. DML verbs cause the DBMS to transfer data to/from the buffers reserved by copying the sub-schema record descriptions into the program.

The working storage section remains unchanged; here we define local variables to be used in the program. PRESIDENT-COUNT is used to keep a count of the native sons of a particular state. DONE is used as a mnemonic device for the status condition of a data-base. It is initialized to the correct status value. While using a DBTG-like system, it might be common practice to define a library of common status codes and to include them in the working storage section by using the COBOL COPY facility. NO-MORE-SONS and NO-MORE-STATES are status variables used within the program logic.

The procedures for accessing the data base

are specified in the COBOL PROCEDURE DIVISION:

```
PROCEDURE DIVISION
DECLARATIVES
EXPECTED ERROR SECTION
  USE FOR DATABASE EXCEPTION ON "04021".
EXPECTED ERROR HANDLING
  EXIT
UNEXPECTED ERROR SECTION.
  USE FOR DATABASE EXCEPTION ON OTHER
UNEXPECTED ERROR HANDLING.
  here we would process unexpected error
  conditions
END DECLARATIVES
```

The first part of the PROCEDURE DIVISION is the DECLARATIVES section. In the DECLARATIVES section we state that the processing is to take place when the DBMS determines that an error has occurred. The DBMS maintains a status location that can be referenced in the program by the name DATABASE-STATUS. DATABASE-STATUS, upon return from a DML command, will contain a value of "00000" if the command was successfully executed. A nonzero code represents an error condition, with the value for the code indicating the nature of the error.

In the event that an error code results from a DML operation, control is returned to that section within the DECLARATIVES that corresponds to the error code. For every error status code listed explicitly in a USE FOR DATABASE-EXCEPTION ON "error status code", control is passed to the paragraph that follows the USE statement. In the preceding example, if a DATABASE-STATUS of "04021" were returned, the paragraph labeled EXPECTED-ERROR-HANDLING would be invoked. Any DATABASE-STATUS codes that are not explicitly listed cause a transfer to the paragraph following the USE FOR DATABASE-EXCEPTION ON OTHER statement. In the preceding example, control would be returned to the paragraph UNEXPECTED-ERROR-HANDLING after an unlisted DATABASE-STATUS statement resulted.

As implied by the paragraph and section names in the preceding example, there is generally a distinction between error situations that we expect to happen as part of our normal processing, and error situations that are totally unexpected. An example of an expected error situation occurs when we sequentially traverse through a set occurrence and reach the end of the set occurrence. Such an error situation usually means that we should proceed to another part of our program to continue processing. An example of an unexpected error condition is an input/output error (for example, bad parity detected).

In this example the DATABASE-STATUS code of "04021" corresponds to the end-of-set occurrence condition just mentioned. When such an error occurs, the processing specifies a return to the statement following the DML command which caused the error (EXIT). The program would then examine DATABASE-STATUS and, if an end-of-set occurrence condition had occurred, could branch to another part of the program.

If any DATABASE-STATUS code other than "04021" occurs, control is passed to UNEXPECTED-ERROR-HANDLING. Here we would specify the processing to take place for these unexpected error situations. The code can examine DATABASE-STATUS as well as other status locations in an attempt to determine what caused the error and how the program should attempt to recover from it.

Following the DECLARATIVES section are the normal processing procedures:

```
INITIALIZATION.
  READY PRESIDENTIAL-AREA.
  OPEN non-database COBOL files.
  MOVE "FALSE" TO NO-MORE-STATES
  FIND FIRST STATE IN ALL-STATES-SS.
  PERFORM PROCESS-STATE THRU FINISH-STATE
    UNTIL NO-MORE-STATES = "TRUE".
  GO TO FINISH UP
PROCESS STATE
  MOVE 0 TO PRESIDENT COUNT
  IF NATIVE SON IS EMPTY
    MOVE "TRUE" TO NO MORE-SONS.
    ELSE MOVE "FALSE" TO NO-MORE-SONS
  PERFORM COUNT-NATIVE-SONS
    UNTIL NO MORE SONS = "TRUE"
  GO TO FINISH-STATE.
COUNT NATIVE SONS.
  FIND NEXT PRESIDENT IN NATIVE SON
  IF DATABASE STATUS = DONE
    MOVE "TRUE" TO NO-MORE-SONS
    ELSE ADD 1 TO PRESIDENT-COUNT.
FINISH-STATE
  IF PRESIDENT-COUNT IS GREATER THAN 1
    FIND STATE CURRENT,
    GET STATE,
    write out state name and president count.
  FIND NEXT STATE IN ALL-STATES-SS
  IF DATABASE-STATUS = DONE
    MOVE "TRUE" TO NO-MORE-STATES.
FINISH UP.
  FINISH PRESIDENTIAL-AREA.
  CLOSE non-database COBOL files.
  STOP RUN.
```

The initialization of this program involves READYing the PRESIDENTIAL-AREA realm (area in the Schema language), which makes this realm available to the application program. Following this, any standard COBOL files are opened (for example, the report file). NO-MORE-STATES is used to indicate that we have finished processing all of the states, and is initialized to "FALSE".

Our algorithm used to solve this query is to traverse the STATE-record occurrences (by sequencing through the singular set ALL-STATES-SS). As we find each new STATE record, we check to see whether its NATIVE-SON set is empty (i.e., whether there are no native sons of that state), in which case we move to the next state. Otherwise, we traverse the occurrence of NATIVE-SON, counting the PRESIDENT records that participate in the set occurrence. If the count is equal to or greater than one, we write out the state name and number of native sons. If not, we continue by selecting the next STATE record until we have finished processing all states.

The FIND statement in the COBOL DML is used to locate a specified record occurrence in the data base. It does not cause the contents of the found record to be transferred to the working storage of the program. For example, the statement FIND FIRST STATE IN ALL-STATES-SS causes the system only to locate the record occurrence. The FIND statement changes the status of several *currency indicators* so that they point to the first record occurrence in ALL-STATES-SS. The member record which is "first" depends on the member-record order which was specified in the Schema description of the set.

By finding an occurrence of a STATE record, we have identified an occurrence of the NATIVE-SON record. We now PERFORM (execute) the COBOL paragraph labeled PROCESS-STATE through FINISH-STATE until the variable NO-MORE-STATES is set to the value of "TRUE", at which point we branch to FINISH-UP.

Within PROCESS-STATE we initialize PRESIDENT-COUNT to zero. If the current occurrence of NATIVE-SON is empty, we set NO-MORE-SONS to "TRUE", otherwise, NO-MORE-SONS receives the value "FALSE". When the set is not empty, we PERFORM the paragraph COUNT-NATIVE-SONS until NO-MORE-SONS is set to "TRUE".

Within COUNT-NATIVE-SONS we FIND the NEXT occurrence of PRESIDENT. NEXT is relative to the current record occurrence of NATIVE-SON. If the current record of NATIVE-SON is the owner record (STATE), "next" is the first member record. When a PRESIDENT record is the current record of NATIVE-SON, then the "next" record is the PRESIDENT record which follows (unless the set is now exhausted).

If the program has traversed through all occurrences of PRESIDENT within the current occurrence of NATIVE-SON, the system sets DATABASE-STATUS to indicate this and passes control to the DECLARATIVES. Within the DECLARATIVES we have specified that for an end-of-set condition return is to be passed back to the statement after the FIND command. The program then sets NO-MORE-SONS to "TRUE", which causes the execution of COUNT-NATIVE-SONS to terminate.

If we have successfully found another occurrence of PRESIDENT, we increment PRESIDENT-COUNT and continue another iteration through COUNT-NATIVE-SONS.

The paragraph FINISH-STATE is entered when we have finished counting all of the native sons of the current STATE. We check PRESIDENT-COUNT to see whether its value is greater than one; if it is, we re-locate or again find the current STATE record (FIND STATE CURRENT) and GET it. The GET command causes the transfer of the record occurrence from secondary storage/system buffers to the internal work space of the program. After GETting the current STATE record, we write out the state name and the number of native sons.

We then proceed to select the next STATE record within ALL-STATES-SS. If we have finished processing all STATE records, NO-MORE-STATES receives the value "TRUE", which causes the termination of PERFORM PROCESS-STATE THRU FINISH-STATE. Otherwise, we continue with another iteration beginning at PROCESS-STATE.

When we have finished processing all of the STATE records, we enter the FINISH-UP paragraph; when we FINISH (release) the realm PRESIDENTIAL-AREA, CLOSE, and non-data-base (COBOL) files, we terminate the program.

Sample Update Program

While the Presidential data base is designed primarily for retrieval, its updating is still necessary, for example, at election time or when a new state is admitted to the Union. We now define a program to admit a new state. Referring to the discussion of selection criteria for the ADMITTED-DURING set, we recall that in order to enter a new occurrence of STATE in the data base, we must supply values for both LAST-NAME and FIRST-NAME in PRESIDENT, and for ADMIN-KEY in ADMINISTRATION.

In this program the IDENTIFICATION, ENVIRONMENT, and DATA DIVISIONs are basically the same as before, and therefore we specify only the PROCEDURE DIVISION:

```
PROCEDURE DIVISION
DECLARATIVES
UNEXPECTED-ERROR SECTION
   USE FOR DATABASE EXCEPTION
UNEXPECTED ERROR HANDLING
   here we process unexpected error conditions
END DECLARATIVES
INITIALIZATION
   READY PRESIDENTIAL AREA,
      USAGE MODE IS UPDATE
   OPEN COBOL files
STORE-NEW-STATE
   here we read in from standard COBOL input files
   the values for LAST NAME and FIRST NAME in PRESIDENT
   ADMIN KEY in ADMINISTRATION, and values for the new
   STATE record
   FIND ANY PRESIDENT.
   FIND ADMINISTRATION IN ADMINISTRATIONS-HEADED
      USING ADMIN KEY
   STORE STATE
   CONNECT STATE TO ADMITTED-DURING
FINISH-UP.
   FINISH PRESIDENTIAL-AREA
   CLOSE COBOL files
   STOP RUN.
```

Because STATE is a manual member of ADMITTED-DURING (see subsection Presidential Data Base in the DDL), we must explicitly tell the system which ADMINISTRATION record occurrence is to be the owner of the new STATE (in the ADMITTED-DURING set). We first locate the appropriate PRESIDENT. The FIND ANY PRESIDENT statement specifies that the system is to locate (using CALC) the occurrence of PRESIDENT based on its key value (LAST-NAME, FIRST-NAME). Then, within the occurrence of ADMINISTRATIONS-HEADED owned by the PRESIDENT thus selected, the system is to search for an occurrence of ADMINISTRATION with a value for ADMIN-KEY equal to that supplied to the program. We have now identified the necessary occurrence of ADMINISTRATION. When we request that the STATE information be STOREd in the data base, we complete the process by CONNECTing the newly stored State record to the current occurrence of ADMITTED-DURING. If the State had been an AUTOMATIC member of ADMITTED-DURING, the sequence of FIND statements would have been performed by the DBMS.

Traversing an m:n Relation in the COBOL DML

The relationship between CONGRESS and PRESIDENT is a many-to-many relationship (Section 1, subsection Many-to-Many-Relationships) and necessitated the introduction of a link record (CONGRESS-PRES-LINK). The introduction of such a link record complicates traversals from an occurrence of PRESIDENT to his CONGRESSes, and vice-versa. We present here a program segment designed to traverse such an $m:n$ relation.

We assume that LAST-NAME and FIRST-NAME in PRESIDENT have been set to a desired PRESIDENT in Dislay 2 below:

Display 2

```
   FIND ANY PRESIDENT.
FIND-NEXT LINK
   FIND NEXT CONGRESS-PRES-LINK IN CONGRESS-SERVED
   IF DATABASE-STATUS = DONE GO TO COMPLETE-RUN.
FIND-CONGRESS
   FIND PRESIDENT-SERVED OWNER
   here we have located a CONGRESS record occurrence over which the PRESIDENT
      served
   GO TO FIND-NEXT-LINK.
COMPLETE RUN.
   continuation of program
```

This program segment is designed to FIND all CONGRESSes over which a specified PRESIDENT served. After FINDing the desired PRESIDENT, we traverse through each of the CONGRESS-PRES-LINK records owned by this PRESIDENT. When we reach a DATABASE-STATUS condition of DONE we have found all CONGRESSes over which this PRESIDENT served.

After locating a CONGRESS-PRES-LINK, we look for the owner of the record in the PRESIDENT-SERVED set. The owner of PRESIDENT-SERVED is the CONGRESS record. By FINDing each CONGRESS-PRES-LINK owned by a particular PRESIDENT within the CONGRESS-SERVED set and then by FINDing the owner of the link in the PRESIDENT-SERVED set, we can traverse from a PRESIDENT to all CONGRESSes over which he served. We can similarly traverse from a given CONGRESS record to all PRESIDENTs who served over that CONGRESS.

Other COBOL DML Facilities

The examples presented here illustrate some of the more important COBOL DML additions to the COBOL programming language. Primarily because of the static nature of the Presidential data base, it is difficult to create meaningful examples to illustrate several additional COBOL DML facilities. Several are briefly discussed here.

In addition to storing a new record occurrence in the data base, an existing record occurrence may need to have some of its values changed. The MODIFY verb is available for this. This operation (which is performed by the system) may be very complex because the effect of a key change may alter the position of the record or even affect the sets in which it resides (through set selection).

A record occurrence which exists in the data base may be deleted by use of the ERASE verb; once again, this may be a very complex operation, because the MANDATORY option may cause multiple deletions of many member records.

Finally, the second example showed the use of CONNECT to place a member record in a set. By use of the DISCONNECT verb an optional member record may be removed from a set occurrence. Note that DISCONNECT may only be used on optional member records.

3. ADVANCED FEATURES

Section 2, Sample Data-Base Applications introduced the basic facilities offered by the DBTG systems. The paper by Sibley and Fry [see page 7] pointed out, however, that there is more involved in the design and maintenance of a data base than the specification and manipulation of complex, interrelated data-base structures. In particular, any complete system must provide facilities that enable a data-base administration staff to write various utility routines for loading the data base, to check its validity, to collect statistics about record frequencies and clusterings, and to reallocate groups of record occurrences to improve performance. This list is by no means exhaustive. It should come as no surprise that the same logical structure can be realized on a computer in a variety of ways, each with varying efficiencies in different situations. Users occasionally need access to a "low" level of data (one close to bits on the storage media); there must be such a system interface.

An additional requirement in any system designed to serve a wide community of users concerns tuning and tailoring: means must be offered to allow various system services the option of either having their performance improved for a particular application mix, or having their services bypassed when such usage would lead to unacceptable performance. Much has already been said about data independence; certainly it is generally a major design goal. But data independence ultimately involves some execution time binding of decisions, and the extra computation involved may be intolerable. Yet it may still be desirable to allow certain programmers who knowingly sacrifice data independence to use the data-base system. The full recovery services of the system may be needed; its ability to manage multiple indexes may be needed; or

its ability to be used in a "customized" access method written in a procedure-oriented language may be needed.

The DBTG-like system architecture and language facilities makes such uses possible. The proposed facilities can be categorized in two parts. First, the DDL provides facilities for a data-base administrator to control various details of the storage strategies. For example, a user may declare that the system should create and maintain an index on specified items within a record type. The index would speed processing in appropriate situations. Examples of this and other DDL declarative facilities are given later in this section. In addition, the mechanism known as a data-base procedure allows the normal system facilities to be extended. Extremely detailed control can be attained, if desired, by the data-base administrator, and hence performance tuning is possible. Second, other facilities provide a set of data manipulation verbs (and a few more DDL options) which allow designated users to access data in a way that is less data independent. For example, there are facilities to obtain a data-base key (a logical "pointer" to a record occurrence) and subsequently to use this key to FIND a record. Examples of some uses of such a facility are included later. It is clear that use of these facilities should be controlled carefully, since data independence could suffer. A preprocessor or extended compiler could enforce such installation-defined standards.

The subsections that follow illustrate some of these advanced features. We augment the schema developed in Section 2, Sample Data-Base Application, and provide fragments of programs that use the advanced versions of various DML verbs.

Data-Base Procedures

The previous structural examples are largely fixed at the time schema definitions are made. For example, the record types, set types, and several of their attributes (e.g., AUTOMATIC) are all determined before data-base processing begins. One reason for this is, of course, that run-time interpretation generally reduces efficiency; hence record formats and accessing strategies are often fixed before processing begins. However, any flexible system provides a means whereby some parameters can be set (or certain extra processing can be triggered) during execution of a request. The DBTG system provides for this with data-base procedures.

A data-base procedure is logically a part of the data-base definition (the schema). It is a procedural augmentation of the (largely) declarative schema. The conditions under which a data-base procedure is to be invoked are given in the schema, either explicitly or implicitly. The procedure often operates as an ON condition functions in PL/I and may accomplish some change to the data-base state or control parameters. Data-base procedures can also be used to collect statistics and to enforce privacy by checking passwords. We use the concept of data-base procedures in many of the examples in the remaining subsections.

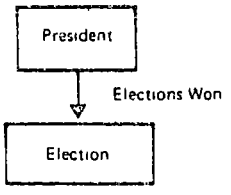
Data-base procedures are commonly used to derive data values. Frequently, certain data items are computable from other items in the data base. For example, the value of the total salaries paid to the people in a department is equal to the sum of the separate salaries of each of the employees of the department; or a person's age can be derived from his birth data and from today's date. Whenever such a functional relationship exists, the data-base administrator can provide a data-base procedure to compute the result. But two possibilities still remain. The functional value may be computed each time the record occurrence is retrieved by a program (VIRTUAL), or the functional value may be stored (ACTUAL) in the record occurrence and updated each time one of the values on which it depends is changed. Depending on the situation, performance can vary widely (e.g., where an ACTUAL result depends on time). The DBTG specifications provide the data-base administrator with the facilities for declaring items as either ACTUAL or VIRTUAL results of other items in the same record, or functions of items within selected members of sets owned by the given record. As an example, suppose we wished to include, in PRESI-

DENT, the maximum number of electoral votes ever obtained in any election of that president. This may be accomplished by including an item in PRESIDENT:

```
02  MAX-ELECT-VOTES, IS ACTUAL RESULT OF
        FIND-MAX-ELECT-VOTES
        ON MEMBERS OF ELECTIONS-WON.
```

where FIND-MAX-ELECT-VOTES is a data-base procedure name. ACTUAL was chosen here since updates should be infrequent. FIND-MAX-ELECT-VOTES would be called each time a new ELECTION record is added to the data base.

It is also possible to "propagate" items from an owner to a member record by using a SOURCE statement. This statement names the item within the owner-record type from which the propagated value should be drawn; once again, the data-base administrator can either save storage (virtual) or insure common copies of data values among the various records in the set (actual). For example, consider the structure:



where the data-base administrator has decided to include the name of the winning President as an item in the ELECTION record (one reason for doing this might be compatibility with a relational view of data). One way of implementing is:

```
RECORD NAME IS ELECTION

02  WINNING-PRESIDENT IS VIRTUAL AND
        SOURCE IS PRES-NAME
        OF OWNER OF ELECTIONS-WON.
```

where PRES-NAME is a qualified name of the item that will serve as the SOURCE item when ELECTION is fetched.

Data-base procedures provide a number of flexible facilities to control the behavior of the data base. Additional examples involving data-base procedures are given in the following subsections.

Areas

The various record occurrences that are the subject of the STORE command must, of course, be placed on actual secondary storage media. Naturally, if a data-base administrator is to have any influence in this placement strategy, there must be constructs in the DDL that allow this policy to be stated. Such control over physical placement seems to be necessary for reasonable performance, as demonstrated in many environments, including those not necessarily involving data bases. For example, Moler [G2] shows that numerical analysis programs which took advantage of the clustering of array values on the same page had significantly improved performance in a paged environment; for example, in FORTRAN implementations with column major order that scan by columns, not rows, when possible. The same phenomenon is as important in a data-base application.

The DBTG system provides basically two mechanisms for influencing record-occurrence placement: *areas* and *location mode* (which is treated in subsection Location Mode). "An area is a named subdivision of the data base" [S2, p. 2.23]. As was previously illustrated, each area is named in the schema, and there may be one or more areas declared. Many implementors have found it convenient to associate each area declared in the schema with a file (a catalogable entity) in the associated operating system. However, this need not be the case; hence our previous stress that the area is a logical rather than a physical concept.

Areas give the data-base administrator a mechanism for clustering or separating different record occurrences (possibly of diverse types). A given record occurrence of a given type may reside in only one area, but other occurrences of the same record type may reside in different areas, if desired. Also, occurrences of different record types can coexist in the same area. The area concept allows data-base designers flexibilities such as the following:

- If certain (or all) occurrences of a record type are known to be archival, they can reside in an area which is associated with a less expensive storage

medium. It may be that this area need not always be mounted.

- If records of diverse types are often used together, their occurrences can be clustered for high performance.
- By designating that all occurrences of a record type reside in one area, and by reserving that area for that record type, the effect of homogeneous files can be achieved.
- By omitting certain areas from a subschema, a measure of privacy is attained.
- Records may be processed consecutively within an area (using the FIND NEXT IN AREA verb), and processing can proceed at essentially sequential speeds, if the implementor allows areas to be allocated sequentially. This is because each "page" within an area will usually follow the previous one in terms of residing on the same cylinder on a disk. Such performance can be important, especially in utility and certain bulk "statistical" application programs in which record ordering is not important.
- An area may be used as a unit of recovery. It is then possible to vary the checkpoint policy for different portions of the data base. Some implementations may allow dual copies of designated areas (maintained by the system) both to reduce contention and to serve as added protection against catastrophe.

Since record occurrences are placed in areas, there must be a mechanism for specifying in which area a record occurrence of a given type must be stored. In Section 2, Sample Data-Base Application, we illustrated the constructs necessary to store all records in a single area. Since every record type had a unique area to contain it, no special action was needed. If, however, distinct occurrences of the same record type can potentially be stored in more than one area, the situation is more complex. For example, if the data-base administrator declares that STATE records may be stored in either of two areas by writing

```
RECORD STATE
WITHIN EASTERN-AREA, WESTERN-AREA
  AREA-ID IS STATE-AREA
```

then the variable STATE-AREA must be initialized with the proper alphanumeric area name at time of store by a command like:

```
MOVE 'EASTERN-AREA' TO STATE AREA.
STORE STATE.
```

In this example, the applications programmer has control over area placement; the MOVE (COBOL) statement is used to initialize the AREA-ID variable. If the variable is left "null," then the area placement algorithm is left to the system implementor. Another option is to use a data-base procedure:

```
RECORD STATE
WITHIN EASTERN-AREA, WESTERN-AREA
  AREA ID IS STATE-AREA
  USING PROCEDURE PICK-STATE-AREA
```

In this case, the procedure PICK-STATE-AREA is invoked to load the variable STATE-AREA, and the programmer need not know about the multiple areas.

Areas, then, give rather explicit control over major subdivisions of the data base and possibly their association with storage media. If this control is given to the applications programmer, there will probably be some loss in data independence. However, the decision concerning whether areas are or are not transparent to an application is in the hands of the data-base administrator.

Areas play an important role in data manipulation in the DBTG system. They are the basic unit (like files) which is OPENed and CLOSEd. As discussed in Section 2, Sample Data-Base Applications, areas are called REALMs in the COBOL subschema. The operations corresponding to OPEN and CLOSE are called READY and FINISH. The verb form

```
FIND NEXT record-name IN realm-name
```

provides a means whereby records can be scanned in ascending "physical" order within an area. For example, if the transaction that retires employees moves each retired employee record to a REALM called RETIRED-EMPLOYEES, then a "batch" program to scan sequentially for

all the recently retired employees would be as follows:

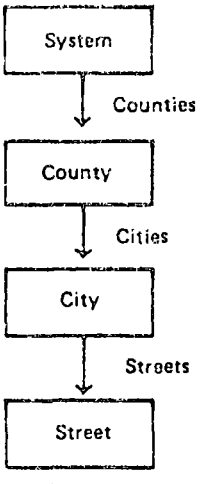
```
READY RETIRED EMPLOYEES
FIND FIRST EMPLOYEE IN RETIRED-EMPLOYEES.
check that at least one exists, then
PERFORM PROCESS EMPLOYEES UNTIL (DATABASE-STATUS =
END OF REALM)
STOP RUN
PROCESS EMPLOYEES
  Process an Employee record.
  FIND NEXT EMPLOYEE IN RETIRED-EMPLOYEES.
```

Location Mode

As illustrated in Section 2, Sample Data-Base Application, the DBTG-like systems allow a data-base administrator to have considerable control over the storage strategies and interrecord clusterings among records within a given area. This control is achieved through proper use of the LOCATION MODE clause, which appears in the declaration of each record type in the schema. It should be emphasized that the LOCATION MODE of a record type designates the strategy to be used for initial record placement when a new record occurrence is STOREd. Knowledge of how a record was initially stored can also be used in subsequently FINDing a record; but as the examples have shown, there are many ways of FINDing a record in the DBTG system. Thus, one should *not* associate LOCATION MODE with FINDing, but rather with "setting in a particular place" i.e., with the STORE command. As might be expected, there is a version of the FIND verb which depends on a knowledge of the location mode of the record being sought. Improper use of this form could, of course, lead to a loss in data independence.

There are four LOCATION MODEs defined. SYSTEM specifies that an implementor-defined algorithm be used in storing the record. Any area control specifications would, of course, be used by this algorithm.

LOCATION MODE VIA set-name (where the record type is a set type member) specifies that the system place the new record occurrence as close as possible to its "appropriate" place in the set occurrence in which it (potentially) will become a member. This implies that the system will use the SET SELECTION clause of the appropriate set to find the proper owner-record occurrence. Having done so, the system will use the insert properties of the set (first, last, ordered, etc.) to place the new record. Using the LOCATION MODE VIA mechanism, it is possible to achieve a record clustering that is efficient for a "depth-first" search. For example, consider the following data structure diagram:

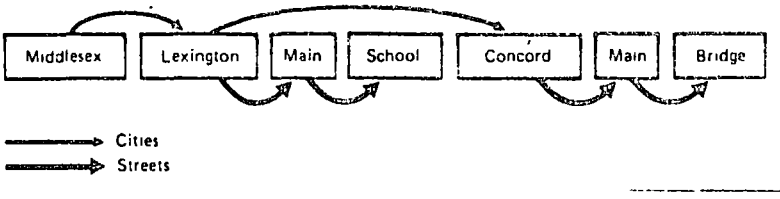


This diagram defines a hierarchical structure of a geographical region. By specifying the following statements in the record declaration section

```
RECORD COUNTY
LOCATION MODE IS SYSTEM
WITHIN GEO-AREA
  .
  .
RECORD CITY
LOCATION MODE IS VIA CITIES SET
WITHIN AREA OF OWNER

RECORD STREET
LOCATION MODE IS VIA STREETS SET
WITHIN AREA OF OWNER
  .
```

and declaring the sets, it is possible to effect a storage structure:



Here, we introduce another option of the WITHIN clause. This option is meaningful only when the LOCATION MODE of the record is VIA some set-name. The area decision for a given record occurrence follows the area decision of the associated owner-record occurrence.

If a data-base administrator wishes more explicit control over record placement, then either of the two remaining location modes, CALC or DIRECT, may be appropriate. We illustrated, in Section 2, Sample Data-Base Application, how LOCATION MODE CALC may be used to place records according to an implementor-defined randomizing routine. The data-base administrator is also free to designate a data-base procedure that can serve as an algorithm for developing "addresses," based on the value of the identifier or identifiers. Whether such an algorithm attempts to behave pseudorandomly over the space of possible identifiers is, of course, determined by the algorithm developer (the data-base administrator). It is therefore possible to incorporate arbitrary record placement algorithms into the system.

It is important to understand the relationship between the location mode of a record type and the FIND verb. As illustrated in Section 2, there are many ways to FIND a record occurrence. These may be classified into two types. Either one FINDs a record occurrence based on its participation in a set (possibly a singular set), or one FINDs a record occurrence based on knowledge of how it was initially STOREd. These two methods provide a number of potential flexibilities. For example, by making every record type a member in a singular set and by having programmers FIND record occurrences using the form

FIND record-name IN singular-set-name USING identifier(s)

it is possible to provide access to all records by specifying their partial contents. A programmer need not know the location mode of a record type in order to use this verb. In addition, the data-base administrator can enhance performance in this case by defining indexes. The details are illustrated in subsection Search Keys. On the other hand, a knowledge of the method used in storing a record initially can provide excellent performance when applied again. For example, a search for a record with location mode VIA should follow the "clustering hierarchy" used in its initial storage. Similarly, knowledge of the identifier or identifiers used by the CALC routine can, in carefully designed applications, yield performance close to one secondary storage access per record retrieval [G3]. The form

FIND record-name IN set-name USING identifier(s)

can be used in a hierarchical search. This is illustrated by the example shown in Figure 16 which finds the first MAIN street in a city in MIDDLESEX county. This example differs from previous ones because there is a "don't care" condition on the CITY record occurrence.

As illustrated in Section 2, the specifications allow usage of the form

FIND ANY record-name

when a location mode of CALC has been declared for the sought record type. By properly initializing the identifier or identifiers which were used in initially storing the record, that record will be found. This procedure requires, of course, that programmers know which record types in the data base have location mode CALC and what item or items constitute their respective CALC-KEYS. It may therefore be difficult to change location modes (or the declaration of items forming the key) without affecting some program and hence introducing a lack of data independence. There have been suggestions [E4] for avoiding this potential problem by using a precompiler or data-base procedures.

The fourth location mode for a record

```
MOVE 'MIDDLESEX' TO COUNTY NAME
MOVE 'MAIN' TO STREET NAME
MOVE 'FALSE' TO SUCCESS

FIND COUNTY IN COUNTIES USING COUNTY-NAME
IF DATABASE STATUS = NOT FOUND
  Process for county not found

FIND FIRST CITY IN CITIES
PERFORM SEARCH FOR STREET UNTIL
  (SUCCESS = 'TRUE') OR (DATABASE STATUS = DONE)
IF SUCCESS = 'TRUE'
  Process for city found
  ELSE Process for city not found

SEARCH FOR STREET.

  FIND STREET IN CURRENT OF STREETS USING STREET NAME
  IF DATABASE STATUS = FOUND
    MOVE 'TRUE' TO SUCCESS
                              ELSE
    FIND NEXT CITY IN CITIES.
```

FIGURE 17. Hierarchical search.

type is DIRECT. In order to present this mode properly, we must introduce the notion of a data-base key. A *data-base key* is a unique identifier which is associated with every record occurrence in the data base. This data-base key is associated with the record occurrence when it is initially stored, and remains the unique identifier of the record occurrence throughout its lifetime in the data base. Although the specifications leave the detailed structure of a data-base key to be decided by the implementor, it is common for a data-base key to have some physical implications; data-base keys are often used in the implementation strategies of a given set. For set traversal to be efficient, the mechanism used must have some physical implications. For example, in many implementations, a data-base key will designate the area, the page within the area, and the record number within the page of the record occurrence. Areas (subdivisions of the data base) are often composed of fixed length pages, and it is easy to note the similarity to a segment/page addressing scheme in a virtual memory environment. However, in contrast with most virtual memory schemes, the actual placement of records within a page is often governed by a local "on-page" index, which maps the record number portion of the data-base key to a displacement within the page. In this way, space within a page can be garbage collected, and records, whose size can in general vary with time, can be moved within the page, all without disturbing pointers pointing at the object being moved. Of course, if a record grows to the extent that it must be moved to an overflow page, then a small "overflow pointer" must remain on the original page. As long as the number of records on overflow pages is not a great fraction of the total records, this scheme can behave almost like direct address pointers while still allowing record movement. If too many records are moved to overflow pages, then the corresponding area can be expanded, and either the page size or the number of pages can be increased.

We now return to the discussion of LOCATION MODE DIRECT. In the DIRECT mode, the program which STOREs occurrences of the given record type may specify the data-base key which the system will try to use. This is in contrast to other location modes where the data-base system determines the data-base key based on calculation keys or the proximity to a logical insertion point in a set. Assuming the data-base key is not already used, the system will use the program-designated key. If that data-base key is already used, the system chooses the next (higher) unused data-base key. With care, the program may have a great deal of control over record placement.

Clearly, DIRECT location mode may be much closer to the physical levels of data. If a high level of data independence is a goal, then the usage of this location mode must be carefully controled. However, such a facility can be quite useful, especially when writing data-base maintenance procedures. As an illustration, consider the method of loading the data base. One popular technique is to let the system, during loading, operate under a schema that is different from the run-schema. In this load-schema, the location mode of the records is DIRECT. If the run-schema specifies a location mode of CALC, the load program can then use the following algorithm:

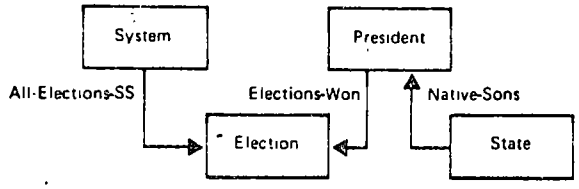
1) For each record to be loaded, invoke a local copy of the CALC subroutine to compute a data-base key using the designated CALC-key items within the record.
2) Sort all records by ascending computed data-base key.
3) Use the load-schema (which has location mode DIRECT) to store the records in one sequential pass over the area.

So-called batch-random operation [G4] in a data-base load program has been found to improve performance by as much as four times. Location mode DIRECT has allowed such a program to be written in a high-level language.

Location mode DIRECT may also be used when direct accessing of records is desired or when building specialized access methods on top of the basic system facilities.

Search Keys

Consider the data structure diagram



Let us suppose that the data-base administrator has determined that the dominant usage of this portion of the data base is to answer the question: Print the elections won by President X. In this case, it would be appropriate to give ELECTION records a LOCATION MODE VIA ELECTIONS-WON set in order to cluster them close to the related President. On the other hand, if we consider the question: Find the state whose native son was the winner of the election of date X, it is clear that the speed of response depends primarily on how fast the election record of a given date can be found (after that, the answer involves two FIND OWNER statements). The question may then be posed:

```
FIND ELECTION VIA ALL-ELECTION-SS USING ELECTION-YEAR.
```

Since an exhaustive search of all ELECTION record occurrences very likely would be slow, it may be appropriate, if query volume is sufficient, to define an index on occurrences of the ELECTION record type. Indexes in the DBTG-like systems are specified by using the SEARCH-KEY clause. By declaring:

```
SET NAME IS ALL-ELECTIONS-SS
  OWNER IS SYSTEM
  MEMBER IS ELECTION
    MANDATORY AUTOMATIC
SEARCH KEY IS ELECTION-YEAR USING INDEX
  DUPLICATES ARE NOT ALLOWED
```

the system builds and maintains an index on the ELECTION-YEAR item. When the FIND statement is issued, the system uses this index to speed the search; it uses the ELECTION-YEAR item provided by the program to search the index to find the first (and in this case only) record having the given year. This ELECTION record is then accessed. The success of the FIND operation does not depend on the continued existence of the index; indexes can be added or deleted as appropriate.

In general, use of the SEARCH KEY clause implies that the DBMS will build an index on the member records (of a given type) within a set occurrence. Its use implies the existence of many indexes, one for each set occurrence. The use in a singular set is a special case.

Introduction of the SEARCH KEY clause allows simulation of the traditional indexed sequential file. By declaring:

```
SET NAME IS ALL-STATES
  OWNER IS SYSTEM
  ORDER IS PERMANENT INSERTION IS
    SORTED BY DEFINED KEYS
  MEMBER IS STATE
    MANDATORY AUTOMATIC
  KEY IS ASCENDING STATE NAME
    DUPLICATES ARE NOT ALLOWED
    NULL IS NOT ALLOWED
  SEARCH KEY IS STATE NAME USING INDEX
    DUPLICATES ARE NOT ALLOWED
```

the data-base administrator can: 1) specify an ability to scan sequentially through all record occurrences of a given type; plus 2) provide a direct path (through an index) to a particular occurrence, given a value of its primary key. To obtain a performance similar to the traditional indexed sequential file, the data-base administrator would have only one record type in the associated area. Thus logically adjacent records would tend to be physically adjacent, as in an indexed sequential file.

SEARCH KEYs can also be used to support an arbitrary number of inversions (secondary indexes) over the members of a set occurrence. The data-base administrator has the option of allowing or disallowing duplicates for items or concatenations of items among members of the set occurrence. Thus, in a singular set, there is a mechanism for guaranteeing unique values across all occurrences of a given record type.

Set Selection

As explained in Section 2, Sample Data-Base Application, each declared set in the

schema has an associated set (occurrence) selection clause. The basic purpose of this clause is to inform the system how to select a particular set occurrence of the particular set type. This is necessary, for example, when a record type is an AUTOMATIC member in a set type. As previously illustrated, when a new record of this type is stored, the system must automatically include it in a set occurrence; the set selection clause tells which is the appropriate one.

The other major use of set selection is in the FIND verb:

```
FIND record-name IN set-name USING id-1, id-2, . .
```

When this version of the FIND verb is executed, the set selection clause of the named set is invoked to find the set occurrence. Searching for a record of the designated type within the set occurrence then proceeds. If the optional USING clause is omitted, the first such record is selected. Otherwise, the system selects the first member of the set where all identifiers (in the record) are equal to the initialized identifiers. If no such record exists, the search fails and control is returned to the DECLARATIVES section of the program.

The set selection clause, while defined in the schema, may be changed or augmented in the subschema. That is, the particular set occurrence selection can vary from one subschema to the next, under control of the data-base administrator. The set selection clause in the schema is a default, which will be used unless overridden by the subschema.

As an illustration of these concepts, we search for the first MAIN street in LEXINGTON in MIDDLESEX county (see subsection Location Mode). The relevant piece of program code is:

```
MOVE 'MIDDLESEX' TO COUNTY
MOVE 'LEXINGTON' TO TOWN
MOVE 'MAIN' TO STREET-NAME
FIND STREET IN STREETS USING STREET NAME
```

This contains considerably less code than the previous example (though the example is slightly different). Clearly, the extra searching is performed by the data-base system with only one FIND command. The specifications for doing this are declared in the associated subschema, as:

```
SD STREETS
  SET SELECTION FOR STREET IS
    VIA COUNTIES  OWNER SYSTEM
    VIA CITIES    OWNER VALUE OF
                    COUNTY-NAME IS COUNTY
    VIA STREETS   OWNER VALUE OF
                    TOWN-NAME IS TOWN
```

In the set selection specification, the system is instructed to first locate a COUNTY by going to the singular set COUNTIES and to search forward until a matching county name (i.e., MIDDLESEX) is found; then the system is instructed to descend into the CITIES set to search for a matching town; finally the system is instructed to search for a matching street by the USING phrase on the FIND verb.

It should be clear that a set selection declaration is basically a specification for a data-base procedure that performs an effectively hierarchical search in order to establish a set occurrence. An entry point is established, either through singular sets, CALC entry points, or currency (see subsection Currency Indicators); then, if this is not the desired set occurrence, a hierarchical traversal can be carried out starting at this point until the proper set occurence is found.

One can also note that the only "match arguments" currently allowed are item equalities or concatenations of item equalities. The "don't care" condition and potential for backtracking, as expressed in subsection Location Mode, is not possible. This greatly eases the implementation, as opposed to making tree traversals against arbitrary Boolean expressions. To achieve the same effect produced by the example given in subsection Location Mode, a data-base procedure would be written. The system could then be told to use the procedure by the statement:

```
SET SELECTION IS BY PROCEDURE FIND-ANY-MAIN-STREET.
```

Support for the more complex traversals can be defined by installation using data-base procedures, though of course there is only one possible data-base procedure per set declaration per subschema. If the traversal is more specific to applications or transactions, it must be specially programmed.

Currency Indicators

The notion of a current record was discussed briefly in Section 2, Sample Data-Base Applications, but our examples have not emphasized currency. Normally, the system behaves in an expected way and the programmer does not need to take special note of currency. There are certain complex traversals, however, where the application programmer must be aware of the exact status of the currency indicators. In this section, we discuss currency and show when care must be exercised.

There are many currency indicators associated with a typical program that executes with a subschema (this is called a run-unit):

- one currency indicator designating the current record referenced by the run-unit;
- one currency indicator for each record type, indicating the current record of that type;
- one currency indicator for each set type, indicating the current set occurrence of that type by "pointing" at the referenced record occurrence, which is either the owner or a member of the given set; and
- one currency indicator for each realm, indicating the current record referenced in that realm.

Thus in the Presidential data base, if a program can access the entire data base, the system would maintain seventeen currency indicators—six for the six record types, nine for the nine sets (including each singular set), one for the single area, and one for the current record of the run-unit.

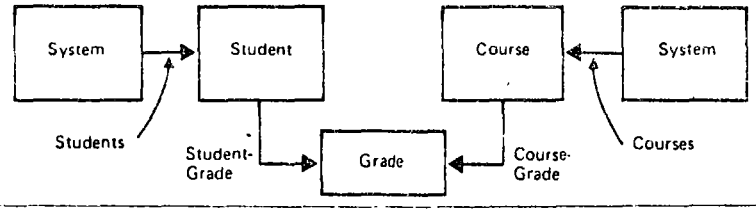
The currency indicators always make the concept of "next" (and prior) well defined, regardless of how the currency has been established; for example, next within a realm means the record in that realm with the next higher data-base key, and next within a set means the next record in the "forward" direction in that set.

It is important to understand when currency indicators change. Currency indicators always change on execution of some DML verb. Also, the object of certain actions must be the current record of the run-unit. The following rules apply:

- Only the current record of the run-unit may be the subject of the GET, ERASE, CONNECT, and DISCONNECT verbs.
- When a new record of any type is found or stored (except for special exceptions discussed later), it becomes the current record of the run-unit and realm in which it resides; it also becomes the current record of its type and of all sets in which it participates as either owner or member. Thus the currency of many sets may be affected.

Because the currency indicators of sets can potentially change when a new record of a given type is found, a programmer must be careful when doing traversals that involve "backtracking." By backtracking, we mean a situation which (for some reason) makes it necessary that a previously established position be reestablished. If, in the meantime, a new record has been stored, the set currency indicators may have potentially changed, making the reestablishment of position difficult or impossible. The programmer must anticipate such changes to the currency indicators and take steps to avoid such situations. The following example illustrates that, even during a pure retrieval, the currency indicators must be handled properly. The example has been adapted from Date [G5].

Given the STUDENT/COURSE data structure diagram below



answer the following question: For each student taking MATH, find all the other courses being taken by that student and print the student's name and the names of the other courses. Note the situation here. Given that we have found a student who takes MATH (found by locating the MATH record and by performing the "switch and find owner" traversal as illustrated in Section 2), we must reenter the same structure to find the courses which are not MATH being taken by that student and print them. But this retraversal of the STUDENT/COURSE structure will destroy the previous currency associated with the COURSE/GRADES set. When we try to find another student taking MATH, the results are potentially unpredictable.

To allow a currency saving facility, two approaches are provided. One is the ACCEPT statement.

ACCEPT identifier FROM {realm name / set name / record-name} CURRENCY

moves the designated currency indicator to a run-unit variable. Subsequent use of the FIND verb

FIND record name, DATABASE-KEY IS identifier

would restore the relevant currency indicator.

Another method has also been provided. Both FIND and STORE have an optional RETAINING CURRENCY clause. By using this clause, the currency for designated record types, set types, or realms is not changed by the execution of the verb; that is, the normal currency updates, as explained previously, are not performed (except for the current record of the run-unit). As an illustration of this, the program shown in Figure 18 answers the STUDENT/COURSE question already discussed. We assume course name is (at least virtually) part of the GRADE record. The reader will note that a GO TO statement is used, since it is felt that a simpler program results.

Two other examples illustrating when special treatment of currency indicators is necessary are the parts explosion traversal (see section 1, Complex Relationships

```
    MOVE 'MATH' TO COURSE-NAME
    FIND COURSE IN COURSES USING COURSE-NAME.
**  We now have the Math record  We assume it exists.
    FIND FIRST GRADE IN COURSE GRADE
    PERFORM PRINT-STUDENTS-OTHER-COURSES UNTIL
      (DATABASE-STATUS = DONE)
    STOP RUN
  PRINT-STUDENTS-OTHER-COURSES
    PERFORM FETCH-AND-PRINT-ONE-STUDENT.
    PERFORM PRINT-THAT-STUDENTS-COURSES.
**  Now check for more students by trying to
**  FIND more grades associated with the given course
**  (Math in our example)
    FIND NEXT GRADE IN COURSE-GRADE
  END-OTHER-COURSES.
  FETCH-AND-PRINT-ONE-STUDENT.
    FIND STUDENT OWNER
    GET NAME OF STUDENT.
    Print it
  END-ONE-STUDENT
  PRINT-THAT-STUDENTS-COURSES.
**  Note the use of the RETAINING CLAUSE Below
    FIND NEXT GRADE IN STUDENT GRADE
      RETAINING CURRENCY FOR COURSE GRADE
    IF DATABASE STATUS ≠ DONE
      GET GRADE
      IF COURSE IN GRADE ≠ COURSE-NAME
        Print the other course name
      GO TO PRINT-THAT-STUDENTS-COURSES.
  END-STUDENTS-COURSES
```

FIGURE 18. Illustration of currency retention.

Using Data Structure Diagrams) and the relatively uncommon question: Find the employees who earn more than their managers. The reader is encouraged to think through these two examples with respect to currency indicators.

4. IMPLEMENTATIONS OF THE DBTG SPECIFICATIONS

As mentioned in the Introduction, the specifications initially published by the DBTG are under continuing development and refinement by groups within the CODASYL organization. It is difficult to state whether system "X" does, or does not, follow the specifications. There are, however, a number of commercially available systems that have used one or more versions of the specifications as a basis for implementation. While these system may employ syntax that is slightly different from our examples, they follow the same basic data model. Some of the commercially available systems which are generally deemed to be "DBTG type" systems are:

- DBMS/10 (Data Base Management System/10), marketed by Digital Equipment Corp. for use on DEC System 10 computers
- DMS/1100 (Data Management System/1100), marketed by UNIVAC

for use on UNIVAC Series 1100 computers

- EDMS (Extended Data Management System), marketed by XEROX Data Systems for use on XEROX SIGMA 6, 7 and 9 computers
- IDMS (Integrated Data Management System), marketed by Cullinane Corp. for use of IBM System/360 and System/370 computers
- IDS/II (Integrated Data Store/II), marketed by HONEYWELL Information Systems for use of the HONEYWELL 6000 series computers
- PHOLAS, marketed by PHILIPS ELECTROLOGICA.

Though this section is not meant to be detailed, it is worthwhile to consider what parts of the specifications are successfully implemented, for such an examination would indicate which features the implementors have found easy (or difficult) to implement, or which features the implementors thought would be of use to their customers.

As a general rule, the implementors have allowed full generality to be used in the data model presented in Section 1, Design of a Data Base, as well as in most of the DML functions presented in Section 2, Sample Data-Base Application, and Section 3, Advanced Features. The part of the data model most frequently omitted is that of singular sets. The rationale for this seems to be that singular sets can be very easily simulated by the user.

While the basic data model is maintained in all of the systems, various implementors have left out many of the more sophisticated features in the Schema DDL. The facility for privacy locks and keys is most frequently dropped, as are data-base procedures. In addition, none of the available systems provide for VIRTUAL items.

In approaches to the sub-schema facility, systems vary widely. The initial implementation of some systems did not provide for a separate sub-schema language (indeed, in 1969 the DBTG specifications did not include one); instead, the systems relied on the COBOL program. Such a facility provides for inclusion or exclusion of certain schema record types from the program. Even in cases where a separate sub-schema language is provided, many implementors have still allowed only for the inclusion or exclusion of entire record types or set types. In a minority of the systems the sub-schema is allowed to select only certain data items from a record, or to reorder or change the data-item type.

GUIDE TO FURTHER READING

The DBTG class of systems will continue to evolve, as will the state of various implementations. In an introductory paper such as this, it is impossible to cover all the options and characteristics of these DBTG-like systems. For these two reasons, it is necessary to read further in the literature for an in-depth understanding of the total system architecture, data model concepts, and data-base design techniques. This section presents an annotated guide to DBTG literature. We concentrate here only on literature whose principal focus is the DBTG specifications, or literature which compares, in-depth, the DBTG approach to some alternative. Discussions of data-base management in general, or tutorial treatments, or references to specific vendor manuals appear in the general bibliography in the companion paper in this issue by Fry and Sibley.

The most recent developments with respect to the Schema language appear in the CODASYL Data Description Language Journal of Development [S2]. Specifications for the Data Manipulation Language of COBOL appear in the CODASYL COBOL Journal of Development [S3]. These documents are issued periodically, and announcements of availability appear in various professional journals. There is also a CODASYL Committee that is developing Data Manipulation Language specifications for FORTRAN [S5, see also S6]. All three references are primarily language specification manuals; as such they are not written in tutorial fashion, but are intended primarily for implementors and as a final arbiter regarding details of program/data-base system semantics. However, [S2] does contain sections that describe concepts of the Schema language.

On a more general level, a discussion of the evolution of "navigational" systems, of which the DBTG systems are a prime example, is given in the ACM Turing Lecture by Charles W. Bachman [N3]. In [N2], Bachman describes how data structure diagram notation can be used to illustrate the organization of lower levels of data—specifically illustrating the access method and storage medium levels.

A collection of more advanced examples, based on the 1971 DBTG report has been published by Frank and Sibley [E1]. Another example by Sibley, using the 1973 syntax, is available as a National Bureau of Standards report [E3]. Additional examples appear in vendor manuals, especially [E2].

There has been a continuing debate concerning the merits and disadvantages of the DBTG architecture. Aspects of this debate are covered in the companion paper by Michaels, Mittman, and Carlson in this issue. Comparisons of the DBTG proposal relative to the relational model appear in many places; one of the most complete discussions is contained in the proceedings of a debate [D1, D2] in which C. W. Bachman and E. F. Codd are the principals. There have also been critiques and technical evaluations of various aspects of DBTG. One such critique [D5] was presented when the 1971 report was published. In 1975, an IFIP Working Conference was devoted specifically to an in-depth evaluation of various constructs in the Schema language. Proceedings of that conference have been published, and various revisions to the DDL are proposed [D8, R1-R5]. The volume also contains articles that illustrate how to use DBTG systems to support a relational view and discuss the use of concepts from the relational model (e.g., normalization) in the context of DBTG systems. Papers on the proper design of data bases by using data structure diagrams and on the proper use of the Data Manipulation Language appear in [A1-A7].

There have been discussions of the possibility of designing systems which could support any data model a user might wish—whether network, relational, hierarchic, or other types. Nijssen's article "Data Structuring in the DDL and Relational Data Model" [C1] outlines the possibility of the coexistence of data models. The article "On the Equivalences of Data Based Systems" by Sibley [C4] also explores this point.

A SHARE Working Conference held in Montreal, Canada, contains papers describing user experiences with various commercially available implementations of the DBTG systems [U3–U5].

Some aspects of implementation are discussed in [I1–I5].

There have also been a number of papers dealing with the features required by data-base management systems. References [M4–M6, M8] are of particular interest.

CLASSIFICATION OF REFERENCES

S Syntax and System Specifications
N Data Structure Diagram Notation, Navigational Systems
E Example Schemas, Sub-schemas, and Programs
D Critiques and Debate Position Papers
R Suggested Revisions to the Specifications
C Comparison of the DBTG Model to Other Data Models
A Designing Data Bases Using DBTG Systems
U User Experience with Commercial Implementations
I Implementations
M Other Modeling Papers
G Referenced Papers

REFERENCES

This bibliography collects and classifies references to various articles concerning the DBTG specifications. It also includes articles which debate the merits of various features, articles which discuss implementational aspects, and articles which discuss data-base design in the context of a DBTG system. The following abbreviations are used in the bibliography.

SIGMOD/SIGFIDET — The ACM Special Interest Group on the Management of Data (formerly named the Special Interest Group on File Description and Translation) holds an annual Conference. The Proceedings of these conferences are available from ACM, New York.

IFIP TC-2 — A Special Working Conference, "An In-Depth Evaluation of Codasyl DDL," was held in Belgium in January 1975. Proceedings of that conference are available in the book *Database Description*, B.C.M. Dougue

APENDICE C

33

Organization and control of the data acquisition function

Raymond Ferrara*

Introduction

Data acquisition may be defined as the process of obtaining information in computer-readable form. The purpose of this chapter will be to examine the various data acquisition methods and types of equipment, and to discuss the implications each will have on the data acquisition function.

Ten years ago, the scope of this chapter would have been limited to keypunch-oriented systems. Today, most IS (information systems) managers are faced with a situation that has grown far more complex and correspondingly more difficult to analyze. On the one hand, many equipment manufacturers have concentrated on "building a better mousetrap" by improving the keypunch itself. Today's wide variety of buffered keypunches, key-tape units, and key-disk systems attest to the fact that a good many manufacturers, including the major mainframe manufacturers, feel the data processing community needs better mousetraps. Still others have focused their attention toward better ways of catching mice. For one, optical character recognition (OCR) systems have taken several steps downward in price—making them possible alternatives in organizations that employ as few as eight or nine keypunch operators. Secondly, the increasing viability of telecommunications has prompted many companies to reassess their methods for the acquisition and distribution of computer-processed data. Even relatively small businesses can think in terms of hooking into a data communications network if not of establishing their own. Finally, some very innovative special-purpose equipment has been and is continuing to be developed to meet the needs of specific industry applications. Data collection systems have been evolving since the late fifties for efficient gathering of data from the factory floor. Point-of-sale systems are revolutionizing the role of the computer in retail businesses, while automated teller terminals are playing a similar role in the banking industry. Voice-actuated input (VAI), though still in its infancy, could make the biggest splash of all—imagine talking to a computer instead of pecking away at a keyboard.

* President, Systems Consultants, Inc. Boston, Mass.

From the IS manager's viewpoint, data acquisition is changing from an area guided by well-established axioms to one which demands periodic reevaluation in order to insure that the organization is taking maximum advantage of what technology can and will offer.

Axioms of ten years ago	*Corresponding questions of today*
Send all forms to be keypunched	Should data be captured nearer to or at the point it is first generated?
Key-verify all data; then double-check it through the use of computer-resident validation programs	Are conventional accuracy-insuring techniques appropriate to new methods of data entry? If so, should some validations be performed while the data is being entered, and thus eliminate the need for subsequent key-verification or computer validation?
Run validated, batched data through computer applications programs, then distribute printouts to all concerned.	Should data be entered directly to the computer? Should there be an immediate response to newly entered data?

There are no universally applicable guidelines for answering these questions. While there are many factors that can be predicted with reasonable accuracy when evaluating data acquisition alternatives (see Figure 1), the more important factors are likely to be judgmental.

FIGURE 1
Checklist of evaluation criteria for data acquisition alternatives

Predictable factors	*Judgmental factors*
Operating and conversion costs	Support by corporate top management, operations management, and/or user departments for the alternatives
Timeliness of data entry	Impact of intangible benefits (e.g., improved customer service with a real-time system)
Accuracy of data entry	Ease of conversion and training
Equipment reliability	Vendor stability and maintenance policies
Availability of backup procedures	Likelihood of more attractive alternatives in the near future

In order to assist the IS manager in getting a handle on some of these factors, the next section of this chapter will outline the important features of various types of data acquisition systems and lay the groundwork for the second section—a discussion of the equipment appropriate to each type of system. The third section will suggest frameworks for analyzing data acquisition alternatives.

I. Basic types of data acquisition systems

Data transcription systems

This term denotes the more familiar, keypunch-oriented concept of data acquisition in which an intermediate operator keys information from source documents or spread sheets (source documents re-transcribed for easier keying) onto computer-readable media like cards or tape.

As a good rule of thumb, the direct costs of preparing a punched card with all 80 columns keyed and verified are on the order of eight to ten cents per card. The cost structure for a typical keypunch installation is shown in Table 1. Note that all indirect costs (floor space, supervisory and other indirect labor allocations, and so on) have been excluded. In many cases, these indirect costs are 50 to 100 percent of direct costs.

TABLE 1
Cost structure for typical keypunch installation

Cost category	*Percentage of direct cost*	*Assumptions*
Operator wages	74	$500/mo ; 22 days/mo., 8,000 keystrokes/hr.; 7 productive hrs/day.
Equipment rental	17	Keypunch $125/mo.; Verifier $100/mo.
CPU charges	7	200 cpm processing speed, $40/CPU hr.
Cards	2	$1 per thousand cards
Total	100	

Assuming that an installation is working with trained, experienced operators, the accuracy obtainable with keypunch systems depends primarily on the legibility and content of the source documents—strictly numeric work can usually be keyed 10 to 20 percent faster than mixed alphanumeric work but the error rate will be substantially higher. On average, an error rate of 1 to 2 percent on per-character basis is typical for numeric work. After verification, this rate should drop to .0 percent or lower.

Improvements in both operator productivity and accuracy have been reported by many installations switching to keypunch replacement equipment. The improvements in operator productivity, which may be as high as 30 to 40 percent, tend to stem more from indirect causes—reductions in the number of characters necessary for each record, faster skipping and duplicating speeds, and so on—than from any sustained increase in actual keystroke rate. Accuracy improvement, on the other hand, results primarily from incorporating logic checks right into the machines rather than leaving error-checking to subsequent computer validation runs. A straightforward method of determining the increase in accuracy that can be obtained with a given model of keypunch replacement equipment is to compare the error-checking features of the model against the errors that are actually being detected during computer validation runs.

Source data automation

The essential feature of source data automation is the elimination of the need for intermediate operators. Information is captured in computer-readable form at or nearer to the point it is first generated, rather than being funneled through the data transcription process. Typically, this is accomplished by having user department personnel type or handwrite data in a form suitable for optical recognition, or by locating keyboard terminals in user departments.

The pros and cons of source data automation versus data transcription systems will be discussed throughout this chapter. But, in general, the expected benefits of source data automation are (1) reduced operating costs, since operator wages and most other indirect costs can be eliminated; (2) increased accuracy, since there will be less intervention between recording the data and processing it; and (3) faster information flow, since incoming data will not have to be channeled through the data transcription process. The drawbacks to source data automation are also considerable. Aside from generally requiring a large initial investment and being less able to guarantee complete reliability, source data automation systems tend to be much harder to implement. The personnel responsible for preparing data will be outside the direct control of the data processing department; successful implementation will require top management support combined with a rapport between EDP and user department personnel.

Service bureaus

Service bureaus are perhaps more useful as a means of handling overloads or large one-time jobs such as file conversions, but some organizations may find them more cost-effective overall than in-house

data preparation. This is particularly true in smaller organizations when the service bureau can bring the advantages of OCR or a remote terminal network to companies which do not have the money or the volume to justify in-house development of these facilities. For strictly keypunch work, the service bureau still may be able to do it cheaper. How? Some do it by relying on part-time, less expensive labor, but most gain economies by selecting only the best operators (the productivity of individual operators may vary by a factor of two or more), by close supervision, and by fully utilizing their facilities.

Basic system attributes

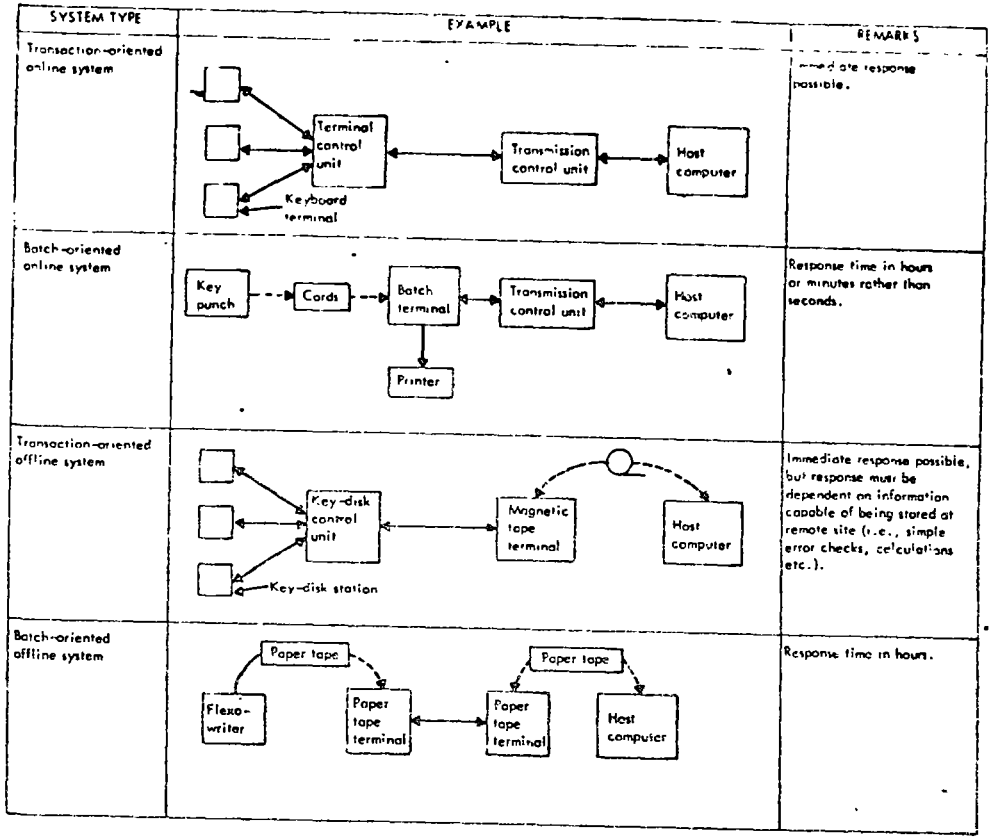
Turnaround systems. In a turnaround system, the source document is itself an output of an earlier data processing operation. Some information will be prepunched or precoded on the document, thereby obviating the need for this information to be rekeyed or retyped when the document is resubmitted for processing. A good example of a turnaround document would be the return stub of a punched-card or OCR utility bill.

Turnaround systems may be classified as either source data automation or data transcription systems, depending on whether intermediate operators are employed to key the remaining data on the document. While no company has successfully utilized turnaround documents filled out by the general public, there are numerous instances of successful intracompany source data turnaround applications. The utility companies, again, employ this concept on meter-reading forms filled out by their meter-reading personnel. In any case, turnaround systems offer the possibility of "streamlining" the data acquisition process.

Interactive systems. A truly interactive system is a lot more than just a data acquisition system. Proper analysis will include a number of topics outside the scope of this chapter—data communications, data base management, security. However, it is helpful to classify the spectrum of data acquisition alternatives by the degree to which they approach real-time interaction. Four basic system categories are shown in Figure 2. The various types of equipment appropriate to each system will be discussed in the next section.

The four categories are based along obviously functional lines, but, by and large, they are economic guidelines as well. Any off-line system will require considerably less investment than an on-line system. There will be no need for transmission control hardware and, more important, no need for hardware or system software modifications to the host computer. Application software development, in addition to being less complex, can proceed without being tied to the development of an on-line system.

FIGURE 2

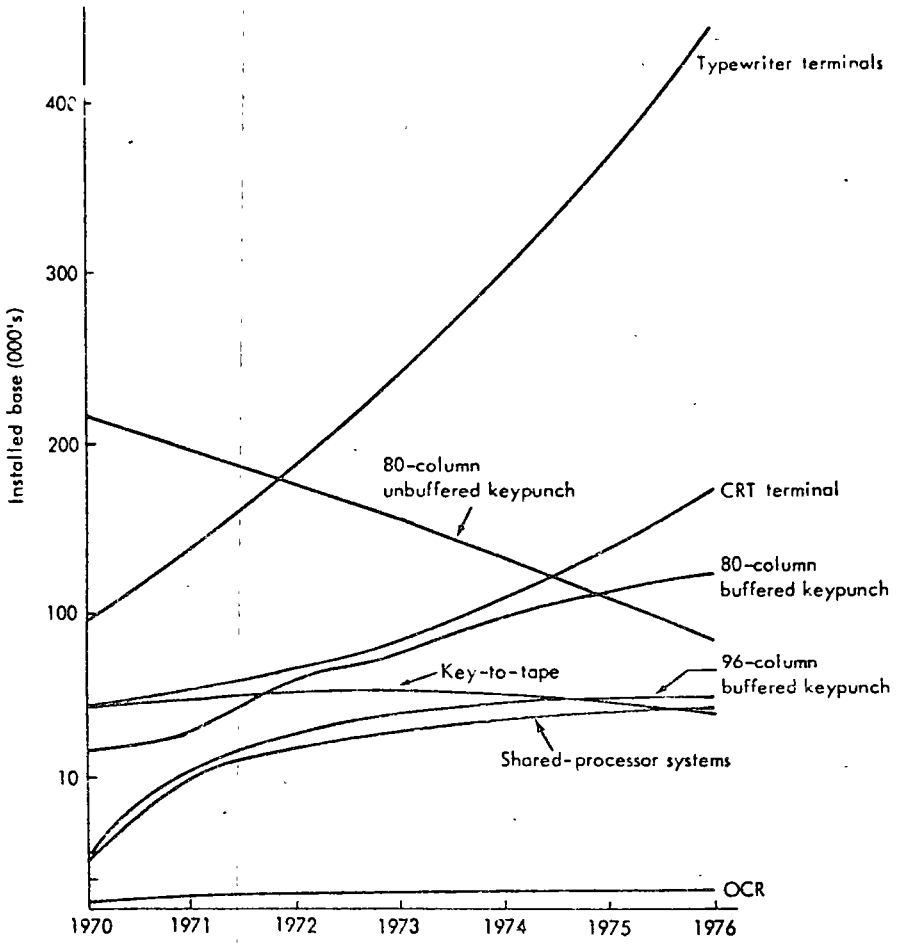


II. Data acquisition equipment

Figure 3 presents some well-researched market projections for the major categories of data-entry-related equipment.[1] These projections clearly indicate a strong trend away from the standard keypunch to more advanced data transcription equipment, such as the buffered keypunch or key-disk systems. There is apparently an even stronger trend toward keyboard terminals, particularly for teletypewriter terminals, but the reader should bear in mind that most of these terminals are being used in time-sharing and message switching rather than volume data entry applications. Currently, an estimated 20–25 percent of data entry requirements are being met via the terminal route, another 10 percent by optical scanning, and the remainder by data transcription methods.

[1] Figure 3 is based upon the author's individual research and on published data by International Data Corporation, *Data Entry Equipment*, 1972

FIGURE 3
Market projections



Standard keypunch

Burroughs and Univac still market standard keypunch equipment, but this segment is unquestionably dominated by IBM's 029 keypunch and 059 verifier. Compared with most keypunch replacement equipment, the two prime disadvantages of this type of equipment are (1) error corrections cannot be made on the verifier; instead, the card must be recycled back to be re-keypunched, and (2) skipping, duplicating, and card registration speeds are comparatively slow. Two lesser-known options usually thought to be available only with keypunch replacement—check digit generation/verification and high-speed skip—can help reduce this performance differential, but

probably not to a significant degree, given the work loads of most installations.

Buffered keypunch

With the buffered keypunch, characters are punched on the card *after* it is released. Because the operator's keying need not be gaited by the mechanical movement of the card, there is no "wait" time associated with actions like skipping, duplicating, or card registration.

Furthermore, errors detected while keying may be corrected by backspacing and re-keying. While many installations discourage operators from making corrections on the initial keypunch phase, all have found this feature helpful in simplifying the verification phase. Unlike the 029/059 system, the verifier operator can make corrections on the spot without recycling mispunched cards; a new card with entirely correct data is automatically inserted into the deck while the error card is directed to a reject hopper.

IBM, Univac, Burroughs, Decision Data, and Tab Products market 80-column buffered keypunches; IBM and Decision Data have buffered versions for the more compact System/3 96-column cards. Available features include check digit generation/verification, accumulation of batch totals and productivity statistics, and up to six pre-stored formats in contrast to the 029's two formats.

Key-to-tape encoders

Mohawk, Sanders, and few others have produced multistation key-tape systems, but for the most part key-tape devices are individual, stand-alone units. Almost all key-tape units feature a buffered IBM 029-style keyboard, a selection of ten or more possible formats, and accumulation of batch totals and productivity statistics. Key-tape devices may be functionally classified according to whether data is recorded on a computer-compatible magnetic tape reel or on a cassette cartridge. With the latter, another device usually called a *pooler* will be necessary to re-record data into computer-compatible form. Pooling may be necessary even with reel-recorded data in order to combine and sort the work load from a group of operators. In some cases, key-tape manufacturers provide off-line poolers; in others, pooling must be performed on the host computer. In either case, pooling can be a time-consuming process.

A number of manufacturers have marketed viable key-tape encoders, but key-tape use appears to be declining due to competition from the buffered keypunch, key-disk systems, and direct entry. Key-tape units with communications interfaces (and possibly attached

printers) are still desirable for users who want to develop off-line transmission networks.

Key-disk systems

In a key-disk system, several buffered keyboards will be connected to a local controller or minicomputer. As the data is keyed, it is extensively checked and stored on disk. When a complete batch of information is accumulated, it is re-recorded on magnetic tape for input to a central computer. There is considerable variety among commercially available key-disk systems on such parameters as the number of key-stations supported (6 to 64), disk space per station (if too small, the disk may have to be dumped several times a day), display features (some display only the last character keyed; others, the entire record plus status information), and supervisory console features.

The advantages of key-disk over other data transcription methods boil down to the generally superior accuracy checks and reformatting capabilities that can be obtained with this type of equipment. In addition to batch totals, check digits, and alphanumeric field checks, most key-disk systems can accommodate range checks (a field's value must lie between two preassigned values), value checks (a field may take on only certain preassigned values), and conditional checks (the field is checked according to the value of the previously keyed field).

The key-disk market has had nearly as many entrants as the key-tape market. Inforex and CMC have been the traditional market leaders, but recent introductions make it hard to define the key-disk category precisely. IBM's new diskette-recording 3740 data entry system, although it lacks full error-checking logic, certainly competes in the key-disk market. Mohawk, Data 100, and Four-Phase Systems market equipment that is essentially remote batch, but which can accommodate key-disk style data entry. Lastly, the key-disk category must include the OCR/key-disk hybrids such as the Scan-Data or Cummins Keyscan systems. The basic feature of these systems is that data can be entered either via the scanner or the keyboard. Once data is present on disk, all verifications, insertions, corrections, and so forth, can be handled through the keyboards. Although this has been hailed as a significant breakthrough in data acquisition systems, conventional key-disk systems with record search capability (an important feature) have been used in conjunction with OCR equipment long before any announcements. What the hybrid system does is simply obviate the need to enter an OCR-produced type on the system disk.

Miscellaneous data transcription equipment

Small portable keypunches range in price from as low as $20 to almost $500. Another portable device, somewhat easier to use than a portable keypunch, is the digital cassette recorder. These run from $100 up and are proving quite popular for inventory stock count applications.

Paper-tape equipment is occasionally used in data transcription environments, more so in Europe than in North America. In general, paper tape is an inferior medium compared to punch cards for most centralized data transcription facilities. Correction is more difficult, frequently requiring splicing. Although paper tape is a more compact medium, filing is usually more difficult. Lastly, paper-tape reels cannot be used in turnaround systems. However, a variant of paper tape—the edge-marked card—can be processed as turnaround documents on equipment such as Friden Flexowriters.

Optical scanning equipment

Figure 4 shows samples of three basic types of optically readable information. Each type lends itself readily to source data automation techniques. Selectric typing elements are available for OCR "A" and several bar codes, while optical mark recognition requires only pencil or ink marks with a comparatively low surface reflectivity. Several optical character recognition devices can also read handprinted numeric characters (see Figure 5). As of this writing, no firm has yet produced equipment which accepts all handprinted alphabetic characters. While some machines which rely on software recognition techniques can be "programmed" to learn one or two personal handprint styles, they are still not sufficiently reliable for general-purpose use.

Optical reading techniques almost invariably require paper with special optical properties. Fortunately, its use has become sufficiently widespread so that a number of printing firms can supply acceptable forms. When a computer printer is used to prepare turnaround documents, the printer's alignment and paper reflectivity are the critical factors. For this reason, most impact printers, particularly chain printers, can produce acceptable images, while very few nonimpact printers will.

1. OPTICAL CHARACTER RECOGNITION (OCR). Conventional OCR readers are usually classified as either document readers or page readers. Document readers accept only small documents (4 inches by 8 inches, or less) and are generally much slower than page readers.

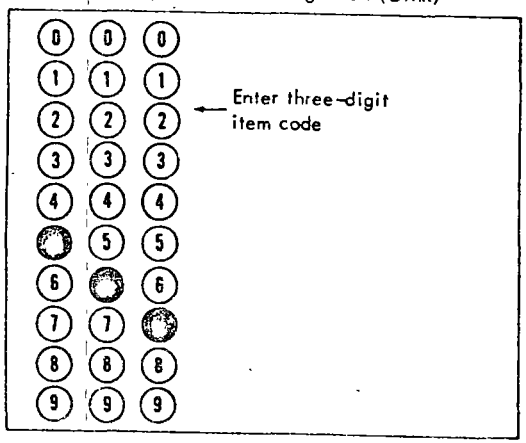
FIGURE 4

Optical character recognition (OCR)

ABCDEFGHIJKLM
NOPQRSTUVWXYZ
0123456789
• ¬ : ; = + / $ * " & !

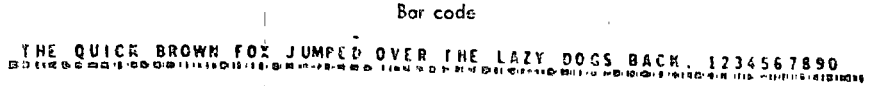
A partial listing of characters in the OCR "A" font style.

Optical mark recognition (OMR)



A three-digit field from a sample OMR document. In the above case, the field's value would be "567," since the corresponding digits have been marked.

Bar code



A sample of the bar code readable by Datatype equipment. Character interpretations are printed over the bar code.

They are limited in their scanning ability; that is, the document may contain OCR printing on only a limited number of lines, sometimes only one (journal tape readers for cash register tapes), or at most, five or six lines. The page reader is much more flexible in this respect; information can usually be picked anywhere on the page. The distinction between the two types of readers developed because inherently simpler transport mechanisms and recognition capability are required for document reading compared to page reading. A price differential exists, but mainly within a vendor's product line, not so much across the industry. Several manufacturers produce combined document/page readers.

FIGURE 5

IBM 1287 handwriting rules

Rule	Correct	Incorrect
1. Write big.	0 2 8 3 4	0 2 8 3 4
2. Close loops.	0 6 8 8 9	0 6 8 8 9
3. Use simple shapes.	0 2 3 7 5	0 2 3 7 5
4. Do not link characters.	0 0 8 8 1	0 0 8 8 1
5. Connect lines.	4 5 T	4 5 T
6. Block print.	C S T X Z	C S T X Z

Note. C, S, T, X, and Z are the only alphabetic characters which may be handwritten.

Another distinction is the number of fonts (type styles) which the reader can accept. Virtually all recent OCR machines are capable of reading OCR "A," which is emerging as the domestic standard, but the OCR industry has seen an abundance of font styles. In the early days, a newly announced reader would probably include a new font, plus several older ones. Multifont capability was considered an advantage, though it is hard to see why, given the restricted use of each font. Today, the important features along this line are (1) numeric handprint, and (2) omnifont capability. The latter is associated with software-recognition readers and implies the ability to decipher any font, provided that a sample of the type style is loaded into the machine's memory beforehand.

In any optical-scanning-based system, there will be three potential sources of inaccuracy: (1) incorrect data interpretation by the machine, (2) document rejection by the machine, and (3) incorrect source data. To detect incorrect source data (a problem common to keypunch

systems as ll) OCR documents can be sight-verified as they are prepared, but any remaining source data inaccuracies must generally be handled through mainframe-resident validation routines. A few of the more sophisticated, minicomputer-based readers do provide for redundant entry of critical information, check digit checking, and batch total accumulation.

Character-substitution accuracy refers to the machine's ability to translate input characters correctly. With the diversity of recognition methods, there is no across-the-board error rate applicable to all readers. Some manufacturers quote error rates for their own readers, but controlled experiments from users are understandably difficult to conduct. Auerbach reports that the Veterans Administration, in a large test of handprinted input, experienced a character-substitution

TABLE 2
Reject rates

	Typed input	*Handprinted input*	*Computer-printed input*
Supervised preparation	1–2%	4–8%	Negligible
Unsupervised but trained in OCR rules	3–5%	10–15%	

error rate less than 0.02 percent. Other large service bureaus and commercial users who have run experiments have found error rates significantly less than 0.1 percent, a figure comparable to the post-verification error rate for most keypunch installations.

The third source of inaccuracy, document rejection, is the one which typically assumes make-or-break status so far as overall accuracy and economics are concerned. Rejects will usually occur an order of magnitude more frequently than any other type of error and will require at least retyping the documents if not recycling them back to the source. Two equipment features which help to cut down on rejects are automatic rescan and on-line display with keyboard correction. Regardless of equipment features, however, it is essential that clear, convenient methods exist for identifying and reinserting rejects while maintaining data integrity. Table 2 presents reject rates typical of most users' experiences.[2]

The most important development in the general-purpose OCR market has been price reductions. Several systems now on the market sell for less than $100,000 – bringing this method of data entry within

[2] *AUERBACH On Optical Character Recognition* (Philadelphia: Auerbach Publishers, 1971).

the range of most medium-sized installations. OCR is particularly appropriate for installations characterized by a few high-volume, preferably turnaround, applications, it may not fare so well in installations with numerous smaller-volume jobs where control and scheduling (including reruns for rejects) can become major problems.

OPTICAL MARK RECOGNITION (OMR). Due to the inherently simpler recognition logic, OMR devices sell for considerably less than OCR equipment. Some are priced as low as $2,000 per unit, but the average is closer to $25,000. However, OMR can be an inaccurate, clumsy way of recording alphanumeric data, since preprinted areas must be provided for all possible values of each character position. Optical Scanning Corporation was the first to come up with a remote terminal system that alleviates some of the difficulty when dealing with strictly numeric data. In addition to conventional OMR, the OpScan systems can accommodate stylized digits within preprinted, sectioned boxes.

Punched cards are also a popular OMR medium. There are at least five manufacturers of combined mark sense/punched-hole card readers. These devices are comparatively inexpensive ($3,000 up) and can be used in a remote communications environment.

BAR CODE RECOGNITION. Bar code readers tend to fall between OCR and OMR devices in price. Compared to OMR, most bar code systems have the advantage that "human-readable" characters may be printed above or below the corresponding character code. Unlike OMR, bar code requires mechanical devices (computer printers, typewriters, embossers, and so forth) for source document generation. Bar code is frequently employed in specialized credit card and supermarket point-of-sale applications but has not been extensively promoted as a general-purpose data entry method.

MAGNETIC INK CHARACTER RECOGNITION (MICR). Though not strictly an optical recognition method, MICR is otherwise very similar to OCR/OMR. (Some bar code readers also rely on magnetic ink recognition.) But except for its entrenched position in the banking industry, MICR is completely unsuited for general-purpose data entry. Document control is at least as difficult as with conventional OCR; the machines are just as expensive; and most American readers can read only font E-13B, a numeric font used and developed for the banking industry.

Terminal equipment

The categories of equipment discussed in this section will correspond to the types used in each of the four categories of data com-

munications systems referred to earlier. From a technical viewpoint, evaluation of terminal equipment must take into account the terminal's compatibility within an organization's data communications network as well as its data acquisition characteristics. (Although the advent of programmable communications controllers and line concentrators is relieving much of the emphasis of maintaining terminal/network compatibility.) This coverage will assume the reader already has an understanding of the factors which affect terminal selection in his own environment.

TRANSACTION-ORIENTED, ON-LINE EQUIPMENT. Except for special-purpose terminals, the hardware in this category is virtually synonymous with the two major types of keyboard terminals: printing terminals (or teletypwriters) and alphanumeric CRT terminals. The third type, graphic CRT terminals, are of course inappropriate to most data entry applications by virtue of their price and specialized features.

Printing terminals. Excluding IBM's rather expensive 3735, most printing terminals are asynchronous, unbuffered models with no local intelligence. Except for "hardware" transmission errors, all error-handling functions must be performed through mainframe-resident software routines. While this is not necessarily an undesirable feature, the fact is that printing terminals are found much more frequently in interactive time-sharing rather than in volume data entry applications. In the latter situation, alphanumeric CRT's outnumber printing terminals, even though printing terminals boast a much wider overall use. Although both types of terminals are capable of transmitting information more quickly than the operator can key it, information cannot be received and displayed as quickly via the printed page as the electronically-buffered CRT, particularly for messages requiring multiple lines. In addition, most error-correction procedures will be more difficult with a printing terminal than with a CRT. For example, in order to correct a previously keyed character on an unbuffered printing terminal, the operator must normally key a delete character for every position back to, and including, the incorrect character and then rekey the whole string. On a CRT, this function can be handled by a two- or three-key operation. Furthermore, the operator can see the corrected version on the screen before it is transmitted.

Aside from any hardware features necessary to insure compatibility in a data communications environment, some important characteristics to look for in printing terminals are line width (70 to 132 characters); answerback generation (a string of pre-stored characters may be transmitted by depressing a single key to make sign-on procedures easier); character set (upper/lower case and control characters, such as tabs); sprocket-feed options (for printing forms);

and automatic send-receive (ASR). This last feature means the terminal may be used in unattended operations with the data read from/to paper or magnetic tape. Paper-tape peripherals are available with almost all printing terminals, while a few newer ones have higher-speed cassette, diskette, or cartridge peripherals. Unfortunately, some of these newer peripherals have had serious reliability problems, causing some well-founded user resistance.

All of the lightweight portable terminals and most of the higher-speed, 30 to 120 character-per-second printing terminals feature nonimpact printers. Although nonimpact printers are quieter and generally more reliable, the special paper they require is comparatively expensive and is not adaptable to multiple copy forms.

Alphanumeric CRT terminals. Although the installed base of A/N CRT's represents less than a third of that of printing terminals, this category has seen more vendors, more diversity, and more promising developments for the data-entry-minded user. Just a brief functional outline of this equipment on the market today includes at least three generations of CRT: intelligent terminals (programmable, minicomputer-based), smart terminals (certain editing features are programmable), and all the others (only "mechanical" editing features are performed by the terminal hardware).

Before looking into the different types of CRT's, some characteristics common to all can be examined. Unlike the large single-unit keyboard printing terminals, most CRT's are available either as single units with a built-in controller or in clusters of up to 64 keyboard displays driven by a common controller. In clusters, each incremented CRT may cost no more than $1,500 to $3,000, while basic single-unit versions tend to be priced in the $2,500 to $5,000 range. Many CRT's now sell for less than all but a few Teletype models, and prices are continually dropping.

The maximum allowable distance between the CRT and its controller has been gradually extended to the point where some CRT's may run up to 5,000 feet from the controller. (Subject to timing constraints, even a "maximum" may be lengthened by the use of digital repeaters along the connecting cable.) One obvious implication of this physical capability is that a company may require only one CRT controller at each geographic location. And as direct-entry applications grow, additional displays may be inexpensively added to the controller up to its physical performance limits.

Whenever hard copy is desired as a by-product of data entry, the printer's characteristics will be important. All too frequently, relatively slow peripheral printers are employed, causing a bottleneck in an otherwise smooth operation. Some of the newer smart and intelligent terminals are better in this respect than earlier versions; more than one printer may be connected to the controller.

Some other minor but occasionally important CRT features are:

character and line insertion; light-pen options; numeric keypad (an additional group of numeric keys), and the clarity and capacity of the display screen. Although the smaller the display capacity, the more displays the controller may support, economizing on display capacity may cause problems whenever multiple screens are required for a transaction. Because each screen is transmitted to the host on completion, multiple-screen error-checking and recovery procedures can introduce complexity for both the operator and the programmer.

CRT terminals with mechanical editing features. This group includes TTY-replacement and IBM 2260-compatible terminals. Aside from giving the operator the ability to delete and insert lines and characters, this type of terminal offers no intrinsic error-handling capabilities. All processing must be handled by the host computer.

One important data entry attribute which the CRT supports much better than printing terminals do is the concept of masking (i.e., displaying sufficient field identifiers and instructions so that the operator merely "fills in the blanks"). Masking, of course, is an extremely useful, if not necessary, concept in source data entry applications since it allows a relatively untrained operator to work with a variety of input records. With 2260-type terminals, however, the cursor (the symbol indicating where the next input character will be displayed) cannot be readily programmed to automatically skip over mask segments. For this reason, it is extremely important that some thought be given to the mask layout. Unless the fields in which data is to be entered are readily accessible (say, by locating data fields at the beginning of lines with identifiers following, or locating them consecutively across a line with identifiers on an upper line), the operator may spend half the time just locating the cursor properly. In a few instances, free-form keying in which the operator keys both field values and field-identifying symbols may be preferable to a mask layout requiring a lot of cursor manipulation.

CRT terminals with programmable editing features. In contrast to 2260-type equipment, the terminals in this category have the ability to perform limited logical functions. More precisely, these functions are: (1) selective retransmission: the terminal can be programmed to send only the data fields; on a 2260, everything between the start-of-message symbol and the current cursor location will be sent when the TRANSMIT button is depressed, (2) protected formatting (a mask layout can be retained in terminal memory and need not be transmitted to the terminal at the start of each new screen; furthermore, mask segments can be "protected" with automatic cursor bypass), and (3) performing limited error-checking routines locally (alpha-only or numeric-only checks; and checks for the presence of mandatory data fields).

All these features add up to a terminal that can be made much easier to operate and, by cutting down the number and size of messages which must be exchanged with the host computer, lead to significant savings in CPU utilization and line charges. The principal products in this category are IBM's 3270 system and Burrough's TC-series. There are also a number of ASCII-compatible CRT terminals which can support protected formatting and selective retransmission.

Programmable, minicomputer-based, CRT terminals. Since these terminals are based on general-purpose minicomputers, there are no effective hardware limitations; every conceivable editing function may be performed locally; and the controller may generally be equipped with any standard mini peripherals for secondary storage, hard-copy output, and so on. Available software will determine how effectively these terminals can be used.

In the present stage of development, these systems are being sold either as custom-programmed versions or, with software emulation packages, as replacements for standard 2260 or 3270 models. As a replacement for conventional terminal equipment, any potential advantages stemming from the terminal's local intelligence are, of course, irrelevant and the system must be judged on cost and reliability factors. But looking into the future, with the trend toward large, centralized data banks on large mainframe processors, intelligent terminals in branch offices could be used as the means for efficient data entry and inquiry to central files while simultaneously performing local problem-solving or text-editing functions without tying up the time-constrained big box at headquarters. Intelligent CRT's are a natural hardware vehicle for this sort of "distributed processing," since they can, and do, provide the features necessary for transaction-processing, stand-alone batch processing, and text-editing, all rolled into one reasonably inexpensive system. However, very few of the intelligent systems now on the market are capable of *simultaneously* functioning in all these roles. The feature to look for is support program development within a multiprogramming operating system.

BATCH-ORIENTED, ON-LINE EQUIPMENT. With suitable auxiliary storage peripherals, virtually any of the previously described CRT or printing terminals could be used for data preparation in a batch-oriented environment. Likewise, any of the previously described data transcription equipment could be used to prepare input for transmission via a physically separate batch terminal such as an IBM 2780 or CDC 200 User Terminal. Rather than review these devices again, this section will just briefly review the more popular alternatives for remote, batch-oriented data entry applications.

1. *Paper tape/cassette/diskette.* Any ASR terminal can be adopted for off-line data preparation with subsequent batch transmission.

In the IBM line, this function is usually fulfilled by the disk-based 3735 or the diskette-based 3740 and 3770 series terminals.

2. *Punched card.* Punched card transmission terminals are available from component manufacturers such as Documation and Hewlett-Packard for less than $3,000 per unit. If these are not suitable, the user can step up to conventional remote batch terminals with punched card peripherals.
3. *Magnetic tape.* A communications interface is available with most key-to-tape and key-to-disk equipment.

With any batch-oriented data acquisition system, there is no possibility for accessing central computer files while the data is being prepared. Consequently, provisions have to be made for subsequent computer validation and, probably, retransmission of incorrect data back to the originating terminal. To offset these procedural difficulties, the on-line batch system can result in significantly lower operating costs than the transaction-oriented system. Constant interaction with the host computer is unnecessary, so CPU overhead will be much lower. And because data is transmitted in a single batch, there is usually no justification for fixed-cost leased lines. Transmission can generally occur on a dial-up basis, preferably overnight when rates are lowest.

With any on-line communications terminal, of course, the prospective buyer should check whether the terminal's communications discipline is supported by standard system software on the host computer. A good source for this type of compatibility information—or any other technical specifications—is the *Auerbach Reports* (Auerbach Publishers, Philadelphia, Pa.). These regularly updated notebooks contain summaries of equipment classes as well as individual descriptions of nearly every available model of data transcription and terminal equipment.

OFF-LINE EQUIPMENT. In contrast to an on-line system, the entirely off-line system does not require a transmission control unit or any changes to the host CPU's hardware or software configuration. For these reasons, the off-line system is fairly popular among smaller installations without the money or the expertise needed to develop an on-line system. Like the on-line batch system, the off-line system precludes any access to host computer files during data preparation and consequently requires the same type of provisions for subsequent computer validation and error retransmission. But except for the necessity for manual intervention to transport files between the receiving station and the host CPU, the off-line system possesses no real disadvantages in comparison to the on-line batch system. In fact, the separation of data transmission and computer processing might be a real blessing, since, the network can function whether or not the computer is functioning. And it is a lot less expensive to obtain a second backup receiving terminal than a backup CPU.

Most key-disk systems can be used for transaction-oriented data entry at a remote site, storing the data for transmission to another key-disk system or tape terminal located at the CPU site. A few key-disk systems, notably General Computer System's 2100 and Data 100's Keybatch, obviate the need for a controller at each remote site. These models allow the keystations themselves to be situated at remote sites, communicating to a central controller via conventional modems and line facilities. For an entirely batch-oriented off-line system, most of the previously mentioned on-line batch equipment can be adapted to transmit terminal-to-terminal instead of terminal-to-CPU. In these situations, the control or "master" terminal can be situated at the CPU site.

SPECIAL-PURPOSE EQUIPMENT. This category encompasses data entry equipment which is designed for specific applications rather than general-purpose use. While this coverage cannot hope to provide sufficient information with which to evaluate even one category, the following list should be helpful in establishing the principal areas where specially-designed equipment is proving acceptable.

1. *Industrial data collection systems.* These systems may incorporate badge or card readers and simple keyboards or control dials for inputting data directly from the factory floor. The earlier versions were off-line, but more expensive on-line versions have been available for several years.
2. *Point-of-sale/credit verification systems.* The two retail functions of recording sales transactions and verifying/posting credit accounts may be accomplished on separate systems, but there is a tendency toward including both on the newer, on-line systems. The exception is supermarket point of sale systems where the trend instead will be toward including devices for reading a universal product code stamped on grocery items.
3. *Digitizers.* Digitizers are instruments for recording coordinate readings or contours from an image projected onto the digitizer's work surface. Useful application areas include drafting and cartography. This equipment may be used as a stand-alone, or integrated into real-time, minicomputer-based systems.
4. *Voice-actuated input.* Experimental airline baggage handling and supermarket checkout systems have shown this concept to be technically feasible, given a fairly small recognition vocabulary (10–50 words) while VAI could turn out to be another OCR—never quite fulfilling its grand promise—no one can doubt its potential. Perhaps more than any other input technique, VAI deserves close watching over the next several years [illegible] should

note, however, that VAI promises to be most helpful not just in replacing keyboard activity - a good operator will be able to key information faster than it can be enunciated—but in freeing personnel from the keyboard and thereby allowing them to perform manual tasks simultaneously with verbal data entry.

5. *Other special-purpose equipment.* There are several applications areas, particularly in hotels and publishing where special-purpose equipment is being developed at such a rapid pace that it is difficult to predict future trends. Aside from each industry's trade journals, the only publication which I am aware of that regularly tracks developments in new forms of data acquisition is *AUTOTRANSACTION*, industry report a biweekly newsletter published by International Data Corporation.

III. Frameworks for evaluating alternatives[3]

With such a large array of data entry equipment and methods, selecting the reasonable, cost-effective system can be a complex process. Here we will focus mainly on the economics of equipment selection—the operating and conversion costs for each alternative. Of course, noneconomic factors should enter into any selection process. Reliability, operator convenience, and ease of conversion may not find their way into the numbers, but are very real considerations nonetheless. What a cost comparison of alternatives can do is provide the background against which these noneconomic factors can be assessed.

Examining the keypunch replacement decision first provides a sense for the basic economics of data entry. Then we will develop a more general framework, a "top-down" approach suitable for evaluating a wider variety of data entry alternatives.

The direct cost structure for a typical keypunch installation was illustrated in Table 1. Since no indirect costs should be affected by keypunch replacement, we may focus only on the changes likely to occur in each of the four direct cost categories.

Card costs are a real though minor savings with key-tape and key-disk, but reduced CPU utilization may not be. Unless the installation can get rid of the computer's card reader or rents computer time on the outside, CPU savings are more apparent than real. The exception is the conversion of former unit record machine applications (e.g., sorting) to the main computer. Although computer utilization might actually increase, the reduction in unit record equipment rental could

[3] This section first appeared as part of the article "A New Look at Computer Data Entry" in the February 1973 issue of the *Journal of Systems Management*. It is reprinted here by permission of the authors, Raymond Ferrara and Richard Nolan, and the publisher, the *Journal of Systems Management*

more than balance this, and be included as a cost savings in this category.

Productivity increases are potentially the most important source of savings. Does keypunch replacement result in productivity increases? Yes, almost invariably. Productivity increases can result from four separate factors, three of which are susceptible to some degree of measurement and prediction.

The first measurable factor is the reduction in operator "idle" time. The standard keypunch duplicates at 18 card columns per second, skips up to 80 columns/second, and takes approximately a quarter of a second for card registration. For an operator keying, say, 1400 cards per day with an average of 20 columns skipped and 20 columns duplicated per card, this means 12 percent of the time (50 minutes) is spent waiting on the machine. With a buffered keyboard, the operator can spend virtually all this time actively keying.

Improved control and verification procedures also affect productivity, albeit indirectly. A typical keypunch installation might spend 5 percent of its time on error correction. All of the sources we have referred to indicate 10 percent to 90 percent first-pass error reduction with keypunch replacement. Usually this reduction is not significant enough to bypass the need for verification, but it does make for easier verification. And the verification step itself is simplified, since the verifier operator can make corrections on the spot without recycling the record back to be re-keyed. Similarly, the automatic generation of productivity statistics eliminates the need for operators or supervisors to take the time to prepare these statistics.

The third "measurable" factor, applicable to most key-tape and key-disk systems, results from reformatting card records for tape input. Extraneous columns with blank fill or, in multiple card records, card identifiers and repetitive information can all be eliminated. Essentially, operators can prepare the same information with less keystrokes.

The unpredictable factor is operator acceptance. In general, most trained operators seem to react favorably to key-punch replacement equipment, particularly if the keyboard arrangement and audible keypunch "click" remain much the same as on the standard keypunch. Due to intangible "operator image enhancement" features of keypunch replacement, it is not at all unusual to see productivity increase more than one would expect from a consideration of the three measurable factors. However, any intangible factors leading to abnormally high productivity may not be persistent, as many early key-tape users have noted.

Any savings from productivity increases must be realized indirectly through decreased personnel and machine requirements. This is common sense. If the same number of operators are kept

working on the same number of machines as before, the net result will be a jump in direct costs - keypunch replacement equipment is always more expensive on an equivalent unit basis than standard keypunches and verifiers.

Determining whether keypunch replacement equipment will pay off overall is relatively straightforward if productivity increases are predicted with a reasonable degree of accuracy. In an installation with cost characteristics similar to those in Table 1, suppose one can calculate an expected 10 percent productivity increase with buffered keyboards, and another 10 percent from reformatting some multiple card records to tape. Before conversion costs and rental charges for replacement equipment are taken into account, monthly savings as a percentage of direct costs can be calculated as shown in Table 3.

Suppose the sample installation is presently employing 15 operators on 15 keypunch/verifier machines and incurring $10,000 per month in direct costs. The bottom line of Table 3 tells us that the installation could spend up to $2,440 for monthly rental of buffered keypunches or up to $3,380 for key-disk or key-tape and still realize a net savings. How much the savings would be depends on the rental for the equipment actually selected. If we chose, say, 14 buffered keypunches with an aggregate monthly rental of $1,750 (average unit rental = $125), net direct cost savings would be on the order of $700 per month. If we choose a 12-station key-disk system renting for $1,800, savings should approximate $3,380 minus $1,800, or $1,580 per month.

Factors other than rental cost may be taken into consideration at this point. With the buffered keypunch, there are no conversion problems other than some minimal operator training. With key-type and key-disk, operator training may be slightly more complicated, but still not a major problem. Any manufacturer will provide help in this respect with short on-site training sessions. Because operator productivity should be the critical factor in determining whether keypunch replacement is economical, a trial period of several weeks duration is usually justified. But even if this cannot be arranged, potential customers will benefit by taking a close look at the projected work load. If most jobs cannot take advantage of reformatting or the increased skipping and duplicating speeds, then there may be no justification for keypunch replacement.

Converting computer programs to accept tape input can be a difficulty, though certainly not an insurmountable one. Another tape-oriented consideration is how errors detected during the computer validation run will be corrected. If the device does not have a full-record display (only a few key-disk systems do) and search capability, post-validation correction can be a recurring headache. In general,

TABLE 3
Sample calculation of monthly savings before equipment rental

Cost category	*Cost Category (percent of direct cost)*	*Buffered keypunch*			*Key-tape*			*Key-disk*		
		Source of savings	*Savings as percent of direct cost*		*Source of savings*	*Savings as percent of direct cost*		*Source of savings*	*Savings as percent of direct cost*	
	A	B	(A × B)		C	(A × C)		D	(A × D)	
Operator wages	74	10% from buffered keyboard	7.4		10% from buffered keyboard and 10% from reformatting	14.8		10% from buffered keyboard and 10% from reformatting	14.8	
Keypunch and verifier rental	17	100%. New equipment costs will be taken into account later.	17		100% New equipment costs will be taken into account later	17		100% New equipment costs will be taken into account later	17	
CPU charges	7	None	0		None	0		None	0	
Punch cards	2	None	0		Approximately 100% No cards, tapes are reusable	2		Approximately 100% No cards, tapes are reusable	2	
Totals	100		24.4			33.8			33.8	

tape loses its attractiveness as an input medium if the installation processes many small jobs.

Finally, help in judging factors like system reliability, vendor maintenance policies, and vendor stability should be sought whenever possible. Comments from other users usually prove invaluable.

Generalizing the framework

Quite clearly, there is no single best approach to, or best equipment for, the data acquisition function in general. Each of the approaches and equipment types mentioned in this article has to be judged within the context of an organization's specific EDP and business environment. Most IS and operations managers should be familiar enough with the characteristics of their own organizations to grasp which methods and equipment are most appropriate and cost-effective. For these managers, the only guidelines this article proposes to offer are those referred to in Figure 1—the checklist of items to consider when evaluating data acquisition alternatives.

However, in a large organization where the data acquisition workload consists of several different input streams with different accuracy and turnaround requirements, the process of "optimizing" the data acquisition function can get quite complex. More than likely, several different types of equipment might be justified, and each might be applicable to more than one input stream.

Table 4 shows a worksheet method for analyzing the economic implications of this type of decision. Three types of costs are considered: one-time conversion costs, monthly variable costs, and monthly fixed costs. Meaningful comparisons between alternatives can be made by defining these as follows:

1. *Monthly fixed costs.* Those costs which will be continuously incurred regardless of the number and type of jobs processed using that alternative (e.g., OCR equipment rental).
2. *Monthly variable costs.* Those costs which will be continuously incurred, over and above any fixed costs, if a particular alternative is used on a particular type of job.
3. *Conversion costs.* (*a*) Fixed: One-time costs which will be incurred if an alternative is used, regardless of the number and type of jobs processed. (*b*) Variable: One-time costs which will be incurred, over and above any fixed conversion costs, if a particular alternative is used on a particular type of job.

Some of the major items which would be included in each cost category are identified in Table 4. In this case we assume comparisons are being made relative to an existing central-site keypunch operation.

Four major alternatives are listed in this example. In practice, we would list only the particular alternatives being considered and

TABLE 4
Evaluating data entry alternatives

Alternative	Type of cost	Monthly fixed cost	Conversion fixed cost	Job type "A" Monthly variable cost	Job type "A" Conversion variable cost	Job type "B" Monthly variable cost	Job type "B" Conversion variable cost
Present key-punching	Operators	0	0	Actual direct expense	0	ETC.	
	Equipment	0	0	Actual direct expense	0		
	CPU charges	0	0	Actual direct expense	0		
	Supplies	0	0	Actual direct expense	0		
	Other	Supervisory and occupancy expense	0	Indirect expense preparation and mailing to central site	0		
OCR	Operators	OCR operators	Training	OCR typists*	Training		
	Equipment	OCR equipment	0	Typewriters*	0		
	CPU charges	0	0	CPU overhead	Testing		
	Supplies	0	0	Forms	Forms design		
	Other	0	Add. core?	Indirect expense and reject handling	Reprogramming for OCR direct or tape input		
Integrated on-line system	Operators	0	0	Terminal operators*	Training		
	Equipment	Transmission control equipment	0	Keyboard terminals, control units, modems	0		
	CPU charges	0	0	CPU overhead	Testing		
	Supplies	0	0	0			
	Other	OS conversion TP monitor	Education for programming staff	Communications charges	Reprogramming		
Composite on-line system (remote batch terminals with keypunch)	Operators	Batch terminal operators	0	Operator wage and keypunches at remote location	0		
	Equipment	Transmission Control equipment	0	0	0		
	CPU charges	Batch terminals	0	CPU overhead	Testing		
	Supplies	0	0	Cards	0		
	Other	OS conversion TP monitor Leased lines	Education for programming staff	0	Reprogramming		

* Most of the cost associated with this item would not apply if the job is susceptible to source data entry

eventually include specific manufacturers' data entry equipment. Since monthly and conversion variable costs are dependent on the type of job for which the alternative is used, these would have to be listed separately for each type of job included in the evaluation.

Getting reasonable cost estimates should be the most difficult part of all, especially for large on-line networks. As with any cost analysis some common sense rules must be observed. Any cost items which vary across the set of alternatives and jobs being considered should be included. Those which do not vary should not be included and may bias the results if they are. For example, in a department with existing batch terminals, expenses for transmission equipment and software should be included in evaluating new applications only to the extent additional charges for these items would be incurred.

Setting cost estimates in this framework may not make economic implications immediately obvious. But once the framework is established, the mechanics of estimating total costs are straightforward. The way in which the cost structure has been defined permits the evaluation and comparison of total costs for any single alternative or mix of alternatives over any given time span.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

DISEÑO DEL SISTEMA PROGRAMACION

LIC. JESUS CARRANZA

AGOSTO, 1978.

RECOPILACION DE DATOS

OBJETIVOS.

Desarrollar una base de datos comprensivos, representativa de la situación actual, que permita obtener toda una serie de conclusiones después del análisis de esa información.

Durante la recopilación de datos el analista de sistemas debe actuar como un investigador objetivo, colectando la información pertinente, sin adelantar juicios, desarrollar conclusiones o teniendo ideas preconcebidas. En la etapa siguiente "Análisis de datos", podrán obtenerse las conclusiones necesarias.

Las técnicas presentadas a continuación, pueden ser utilizadas en la investigación durante el análisis preliminar o en la etapa del análisis y diseño detallados.

Existen dos métodos para recabar información: el resumen de operaciones y la entrevista. Ambos métodos permitirán al analista recabar la siguiente información:

1.- Relación existente entre las diversas operaciones.
2.- Tipos y niveles de actividades realizadas.
3.- Volúmenes y documentación de los datos.
4.- Procedimientos de operación.
5.- Limitaciones de tiempo.
6.- Controles requeridos.
7.- Reportes requeridos.
8.- Retención de los datos.
9.- Uso de los datos.
10.-Costos.

RESUMEN DE OPERACIONES:

El resumen de operaciones es un documento compuesto por una serie de ilustraciones, figuras y tablas, que el analista prepara después de observar las distintas operaciones, revisar toda una serie de documentos y estudiar los archivos existentes. En otras palabras, es el resumen por escrito de una investigación ocular de la situación presente.

Los componentes generales del resumen de operaciones son:

- Organigrama funcional.
- Descripción narrativa de las diversas unidades en operación.
- Personal involucrado y sus características.
- Equipo utilizado.
- Flujo de información.
- Diagramas de procesos.
- Ejemplos de formas y reportes.

ENTREVISTAS.

La gerencia, supervisión y el personal en general de las actividades afectadas, son una fuente de información vital. Sin embargo, el levantamiento de datos a través de pláticas es muy subjetivo, ya que la información recabada puede verse afectada por factores tales como: ambiciones, experiencias, conocimientos, limitaciones y gustos, del personal entrevistado.

La información podrá ser muy valiosa si posteriormente es correlacionada e interpretada.

A fin de que la información que se recaba mediante entrevistas esté bien organizada, es necesario que las entrevistas se planeen y ejecuten en forma precisa, por lo que se recomienda lo siguiente:

1.- Planeación de la entrevista.

a) Concertar la entrevista.-
Hacer cita con la persona señalando el objeto y el tiempo estimado.

b) Determinar el tipo de entrevista.-
Existen dos tipos:
1) Recabar información.- primeras entrevistas que se hacen.
2) Informar, lograr aprobación y recabar más información.- entrevista posteriores.

c) Planear preguntas abiertas.-
Es aquella breve que trae una contestación amplia, o sea, que traigan consigo una gran respuesta, por ejemplo: que opina, que sugiere, etc.

d) Investigar la persona a entrevistar.

2.- Ejecución de la entrevista.

a) Secuencia a seguir.-
Recomendaciones:
1.- Llegar a tiempo.
2.- Si no somos conocidos presentarnos.
3.- Exponer lo que sabemos del asunto, cual es la fuente de información y solicitar que nos complementen la información.

b) Recomendaciones.-
1.- Saber escuchar.
2.- Tomar nota sin interrumpir.
3.- Si hay alguna duda interrumpir para pedir aclaración.
4.- No atacar métodos establecidos.
5.- Si son útiles aún, hacer las preguntas planeadas.
6.- Mostrarnos del lado del entrevistado buscando puntos afines.
7.- Mostrar seguridad.
8.- Terminar a tiempo la entrevista.

c) Lugar de la entrevista.-
Dos lugares:
1) La oficina del entrevistado.-
Estará toda la información a mano, pero podemos tener muchas interrupciones.
2) Fuera de la oficina.-
No habrá interrupciones, pero faltará información, será más difícil concertar esta entrevista.

3.- Después de la entrevista.

1.- Dar gracias por tiempo concedido.
2.- Dejar las puertas abiertas para futuras entrevistas.

ANALISIS DE DATOS

A) ORGANIZACION Y CORRELACION DE LOS DATOS.

Para organizar y correlacionar los datos se debe clasificar la información en tres grupos que son:

a) Procesos.
b) Archivos.
c) Reportes.

Se debe hacer una hoja de trabajo distinta para procesos, archivos y reportes.

a) Para procesos, la información que se registra incluye:

- Nombre/descripción
- Tipo de proceso
- Periodicidad
- Funciones de control
- Archivos requeridos
- Reportes a producir
- Procesos dependientes

b) Para archivos la información registrada incluye:

- Propósito
- Tipo de orden
- Tamaño
- Requisitos de retención
- Porcentaje de información fija
- Porcentaje de información variable
- Frecuencia de acceso
- Controles de validación
- Procesos usando archivos
- Reportes dependientes
- Grado de mecanización presente

c) Para reportes la información registrada incluye:

- Propósito
- Tipo de reporte
- volumen
- Retención
- Porcentaje de crecimiento
- Caracteres por línea
- Controles de validación
- Proceso de producción
- Archivos requeridos
- Distribución
- Secuencia
- Extensión de mecanización (hasta que grado está mecanizado)

Los procesos, archivos y reportes, mencionados anteriormente son identificados y definidos durante el análisis preliminar por lo que, al realizar la recopilación y análisis de información, la hoja de trabajo representa una guia para el analisis individual, conforme se efectúan los pasos detallados, procesos adicionales, archivos y reportes, se pueden aumentar para completar el perfil de todo el sistema en operación.

B) METODOS PARA ANALIZAR Y CONCENTRAR INFORMACION.

a) Redacciones.-
Son difíciles pues se prestan a confusiones por la redacción.

b) Gráficas.-
En ejes cartesianos.

b_1) Diagramas.-
Bloque
Flujo

c) Tablas de decisiones.-

c_1) Tablas de acciones.-
Se identifica por una sola condición por una o más acciones. (no tiene una combinación de condiciones).

1.- ¿Qué son?
Es un método tabular para representar procesos simples o complejos en tomas de decisiones.
Una serie de alternativas con decisiones a tomar.

2.- Ventajas.-

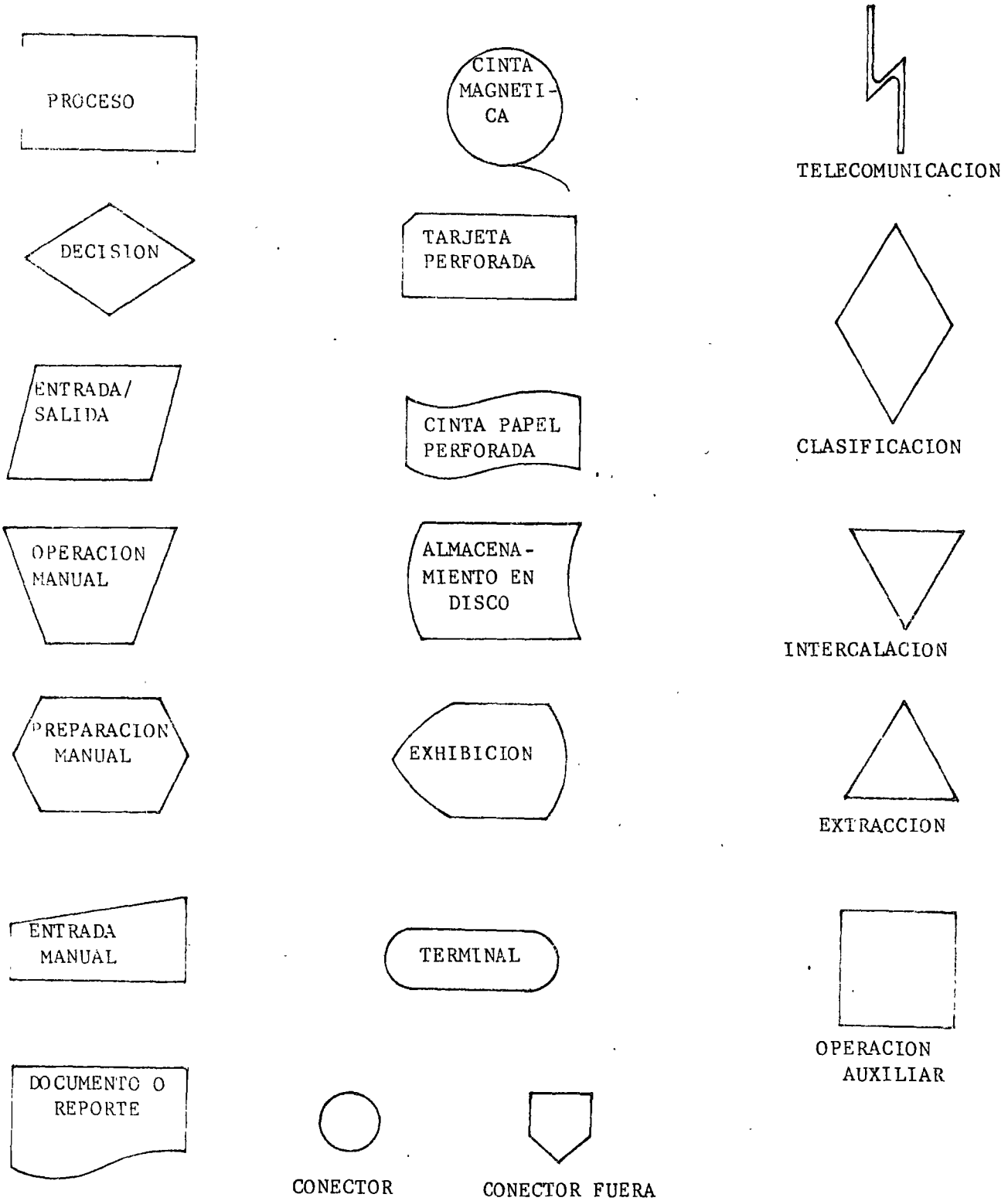
- Ayudan a romper la barrera en las comunicaciones.
- Ayudan en los distintos pasos del proceso de desarrollo de un sistema.
- Ayudan a entender los problemas en la recopilación de información.
- Ayudan o sustituyen a los diagramas de bloque o de flujo.

II.- DIAGRAMAS DE FLUJO

Existen dos tipos de diagramas:

a) Los diagramas de flujo, muestran paso a paso los puntos por donde ha de pasar información, la naturaleza de ésta, y la forma en que se procesa para ser reportada.

b) Diagrama de bloque, es la lógica del programador de la representación detallada, paso a paso.



EXPOSICION DEL PROBLEMA:

Si el crédito del cliente es aceptable (OK), y la cantidad ordenada es igual o menor que el límite disponible, embarque la partida. Si el crédito del cliente no es aceptable o la cantidad ordenada es mayor que la cantidad disponible, rechace la orden. Si la cantidad ordenada es mayor que la cantidad disponible, retrase la orden.

TABLA DE DECISION ---

El crédito del cliente es aceptable	S	N		
La cantidad ordenada es menor o igual que el límite de la orden	S		N	
La cantidad ordenada es igual o menor que la cantidad disponible	S			N
Embarque la partida	X			
Rechace la orden		X	X	
Retrase la orden				X

SECCIONES DE LA TABLA DE DECISION:

SI (EXPOSICION DE CONDICION)	TALON DE CONDICION	ENTRADA DE CONDICION
ENTONCES... (EXPOSICION DE ACCION)	TALON DE ACCION	ENTRADA DE ACCION

TABLA DE DECISION:

El límite de Crédito es correcto	S	N	N	N
La experiencia de pago es favorable		S	N	N
Es obtenida una compensación especial			S	N
Apruebe la orden	X	X	X	
Regrese la orden a ventas				X

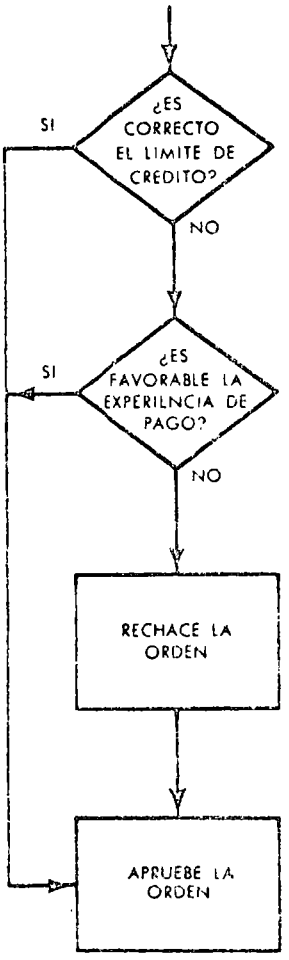
EXPOSICION DEL PROBLEMA:

"Si el límite de crédito es correcto, o la experiencia de pago es favorable, entonces apruebe la orden".

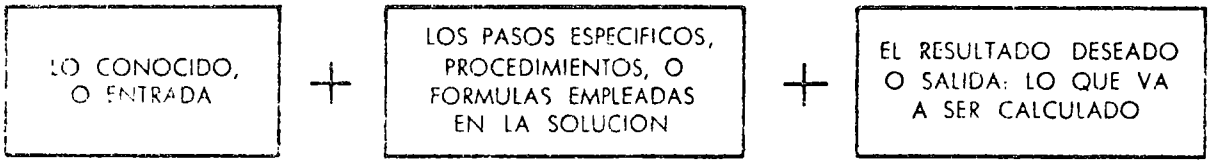
TABLA DE DECISION:

El límite de crédito es correcto	S	N	N
La experiencia de pago es favorable		S	N
Apruebe la orden	X	X	
Rechace la orden			X

ORGANIGRAMA DE PROGRAMA (parcial):



LA EXPOSICION DE UN PROBLEMA CONTIENE:



DISEÑO DEL SISTEMA

INTRODUCCION

Durante el diseño del sistema, el trabajo del diseñador consiste en elaborar un diagrama de ejecución del sistema de procesamiento de datos que se va a implantar con la computadora.

OBJETIVO GENERAL

Cuando se diseña un sistema de procesamiento de datos, - se tienen en mente ciertos objetivos referentes al perfeccionamiento de las operaciones y la reducción de los costos. Algunos de esos objetivos son los siguientes:

1 La intención de estandarizar las unidades. Por ejemplo, si una compañía tiene dos plantas, hacer el sistema de nóminas de ambas tan similar como sea posible. Uno de los lemas del diseñador es estandarizar.

2 Eliminación de funciones innecesarias -operaciones de la compañía que no tengan un servicio prolongado a - cualquier propósito. Integrar otras que normalmente están separadas.

3 Eliminación de reportes, registros y formas innecesarias.

4 Eliminar datos superfluos -partes de reportes que no tienen contribución.

5 Establecer los controles necesarios y eliminar los excesivos. Por ejemplo, si los datos estadísticos se - guardan en tarjetas perforadas, preguntarse la conveniencia de verificar la perforación.

6 Eliminar la sutileza innecesaria de la calidad de los requisitos. Por ejemplo, si la información mensual y anual es adecuada, no es necesario. También obtenerla diariamente. Un supervisor de una agencia del gobierno eliminó cuatro meses de trabajo hombre en tarjetas perforadas al año, suprimiendo una entrada diaria superflua en una forma.

   Otro ejemplo de refinamiento innecesario en la calidad de los requerimientos tomó la forma de una exposición - del transporte en una de las más grandes ciudades de - los Estados Unidos. El boletín dice "75,544,868 + pasa

jeros utilizaron el servicio de tránsito rápido de la ciudad en los primeros cinco años". Traducir esta figura a los dígitos fue refinamiento excesivo. Una figura de 75,000,000 o quizá de 75,500,000 pudo comunicar la misma información. Como cosa real esa imagen pudo dar mayor información. El signo + delante de la cifra, indica que el suministrador de la misma no está seguro de su exactitud, y la figura presentada deja al lector con cierta duda en lo que especta al grado de inexactitud: si está en las unidades o en las decenas, o ¿en dónde? Una cantidad redondeada a la siguiente unidad o decena de millar transmite un grado de exactitud también como una magnitud.

7 Eliminación de duplicación de funciones. Por ejemplo, una situación en la cual un artículo que no sólo se adquiere en compra, sino que tanto el almacén como el usuario de la compañía también lo compran, es una duplicación de función. La duplicación no es necesariamente una cosa negativa, en algunos casos puede ser más económico que no hacerlo, en cuyo caso se sugiere aceptar tal situación. Decir que se debe intentar que la duplicación de función se suprima no es decir que se deba eliminar dondequiera que aparezca. Significa, en parte, que el diseñador tiene una posición que lo hace sensible a la duplicación de función, dado que ha sido capaz de afectar la economía en tales áreas anteriormente.

8 Eliminar duplicación de objetivos. Uno de los casos de este tipo de duplicación que más extensamente se publicaron en la historia fue el desarrollo paralelo de proyectiles en las fuerzas armadas. El reflejo de esta situación da énfasis al principio de que la duplicación de objetivos es un campo fértil para la economía. Alguna duplicación puede tener un fin determinado, como es el compromiso deliberado para la competencia, para los cortes de cuentas de cheques, etc.

9 Eliminación de duplicación de operaciones, tal como el archivo múltiple de reportes iguales en una sola oficina.

10 Eliminación de duplicación de información y formas. En el caso de que dos formas se integren de la misma manera, o si su información se superpone. Esta acción es otra de las consignas del diseñador. En el contenido de un reporte pueden existir leves variaciones

en diferentes departamentos. La integración de función tenderá a estimular la consolidación de las variantes.

11 Facilitar el flujo del trabajo y eliminar las altas y bajas del mismo.

Para este fin considérense aspectos tales como:

a) Reducción de tiempo de espera; por ejemplo, considérese un cambio de una nómina quincenal a una semanal en el que debe suministrarse la contabilidad semanal para la distribución de labores.

b) Eliminación de obstrucciones: con el incremento de capacidad para un proceso automático que da la computadora, un procedimiento en el cual se han creado obstrucciones puede a veces integrarse en otro general de modo que el trabajo se aligere y dichas obstrucciones desaparezcan.

c) Adelantar la fecha de corte para los documentos fuente, de manera que los datos entren al sistema de proceso antes de tiempo.

d) Organizar el sistema para que una parte del trabajo se ejecute a la cabeza del proyecto. Por ejemplo, en una nómina semanal en la que las tarjetas de reloj disponibles para la oficina de contabilidad diaria, en lugar de agruparlas para su proceso a fin de semana, acondicionar que las tarjetas del día se procesen en el sistema tan pronto sea posible.

En la preparación de su labor, el diseñador efectúa algunas preguntas.

1 ¿Puede mejorarse el procedimiento de manera que el objetivo básico se perfeccione completamente?
2 ¿Puede simplificarse el procedimiento o reducirse el volumen de trabajo modificando algunos factores externos tales como políticas, estructuras de organización, o las prácticas o desarrollos de otros departamentos?
3 ¿Es necesaria cada operación?
4 ¿Genera alguna otra operación la duplicación o la superposición? ¿Pueden éstas integrarse con otras?
5 ¿Puede llevarse a cabo el trabajo de un modo más simple, rápido y económico?

DISEÑO DEL SISTEMA

I. ORGANIZACION Y DISEÑO DE ARCHIVOS.

TIPOS DE ARCHIVOS:

a) Clasificación por contenido del archivo:

Cuantitativamente.-

Cuando se clasifica de acuerdo con los campos ya sea de cantidades o de importes.

Cualitativamente.-

Cuando se clasifican los campos del archivo tomando en cuenta los campos que identifican al archivo.

Estadísticamente.-

Es aquel tipo especial de información cuantitativa.

b) Clasificación por uso:

Operacional.-
Es aquel que contiene información cuyo uso está delimitado a la operación del sistema. Ej.: sistema operativo.

Transitorio.-
Cuando contiene información de cuyo proceso se obtienen resultados transformados. Ej.: actualizaciones de archivos.

De archivo.-
Se conocen como archivos maestros. Ej.: clientes, artículos, proveedores, etc.

Histórico.-
Es aquel que contiene información de cuyo proceso no se va a obtener ninguna modificación sobre el mismo.

EL ARCHIVO DESDE UN PUNTO DE VISTA DINAMICO.

a) Crecimiento.-
En el momento de diseñar el archivo el analista requiere conocer ciertos porcentajes de crecimiento.

b) Retención y purga de datos.-
Se deben considerar en el registro datos fidedignos y además tener los datos bien actualizados.

c) Dinámica y actividad.-
Dinamismo es el número de veces que el archivo es consultado en un período de tiempo dado.
Actividad.- el cociente de dividir el número de registros operados entre el número total de registros, multiplicar por 100.

$$\% \text{ actividad} = \frac{\text{No. de registros operados}}{\text{No. total de registros}} \times 100$$

Ejemplo: un archivo poco dinámico y muy activo podría ser una nómina, y un archivo muy dinámico y poco activo podría ser cuenta de ahorros.

d) Requerimientos de actualización.-
Puede ser causada por elementos externos al sistema. Ej.: debido a los Release de un sistema operativo.

- Frecuencia y ciclo de actualización.
- Fuentes de transacción.
- Requerimientos de nivel de calidad.

e) Requerimientos de mantenimiento.-
Esto se debe principalmente a problemas operacionales del sistema antes de tiempo.

f) Requerimientos de respuesta.-
Este es un factor muy importante a considerar en el diseño del archivo ya que del tiempo de respuesta requerido depende que tipo de organización le demos al archivo.

DISEÑO DEL ARCHIVO.

a) Organización del registro.

1.- Formas
Unico de longitud fija
Unico de longitud variable
Múltiple de longitud fija
Múltiple de longitud variable
Múltiple de número variable

2.- Algunas consideraciones aplicables a organización de registros son:

Número variable de campos

$$\text{Actividad por campo} = \frac{\text{Frec. salida en reportes}}{\text{No. reportes x No.total campos en reg.}} \quad 100$$

Tipo de dispositivo

Relación con otros sistemas

Requerimientos de acceso

b) Agrupación de registros en archivos.

Además del obvio criterio de optimizar el flujo del proceso, otros factores para determinar los límites de un archivo son:

- Diferencias distintivas en la eficiencia del medio de almacenamiento disponible.

- Grado de actividad del archivo.-
Archivo de mucha actividad en cintas.
Archivo de poca actividad en discos.

- Elementos de uso común por diferentes sistemas.-
Se puede agrupar los registros dependiendo del número de sistemas en el cual se use el mismo campo..

c) Medio disponible para archivo.-

Hay que tomar en cuenta para la agrupación u organización de los requisitos las unidades disponibles en la instalación.

d) Método de acceso (ordenamiento).-

Esto depende del tipo de unidades instaladas y del sistema operativo existente.

e) Diseño de archivos redundantes.-

Es aquel que es idéntico a otro pero organizado en forma diferente.
Las razones para su existencia:

- Diferentes secuencias del archivo pueden requerirse para salida voluminosa.
- Duplicación total o parcial de archivos por diferentes áreas geográficas.
- Con archivos jerárquicos, puede haber duplicación entre niveles.
- El uso de pequeños archivos que son sub-archivos de otros más grandes puede mejorar la eficiencia de operación del sistema.

f) Reestructuración de archivos.

Razones para la reestructuración.-
Pueden ser mal diseño de los registros, expansión de los volúmenes, cambio de unidad de almacenamiento, lo cual origina también modificar el sistema.

- Cambios de longitud, número o posición de los campos dentro de un registro.
- Combinación de registros unitarios para formar compuestos o lo opuesto.- Archivos de diferentes aplicaciones que se unen para formar un reporte.
- Cambio de la clave (índice)secuencial del registro.
- Establecimiento de nuevas cadenas entre campos en el índice de un dispositivo de almacenamiento masivo.
- Creación de nuevos tipos de registros; ejem.: cambiar uno de longitud variable en dos o tres de longitud fija.

g) Características de los datos.

1.- Datos en formatos (empacados) o en forma textual, y si la longitud de los campos es variable o fija.
2.- Los registros deben ser fácilmente divisibles si estos son muy largos.

3.- Desarrollo de métodos de direccionado o acceso a archivos directos para minimizar la competencia para espacio de almacenamiento.- Encontrar métodos de direccionamiento o randomización para que haya menos sinónimos.

h) Consideraciones sobre el equipo.

La posibilidad de agrupar (block) registros depende:

- Consistencia en la longitud de los grupos entre distintos dispositivos de almacenamiento.-
  Registros estandard.
- Del impacto de la longitud de los grupos en la memoria primaria.-
  Dependiendo de la memoria como agrupar los registros..
- Asignación de archivos a dispositivos particulares de entrada/salida dependiendo del uso de los datos, tiempos de búsqueda y tiempos de latencia en dispositivos de almacenamiento masivo.-
  Latencia = tiempo que se tarda en encontrar un registro en acceso directo.
- Posibilidad de dividir los registros para organizar los archivos de acuerdo a los mecanismos del acceso en dispositivos de almacenamiento masivo.-
  Dividir el registro en tal forma que la información cualitativa de discos y cuantitativa en cintas.
- Previsión de continuar la operación del sistema en caso de falla del dispositivo.

CONTROL DE ARCHIVOS.

Control es un factor vital en diseño de archivos cuando estos se prueban por recuperación, retención y seguridad.

a) Recuperación.

Los puntos de chequeo y reiniciación obligan a un doble chequeo.

Totales o cifras de control. Son un método planeado de mantener vigilancia sobre los datos o registros en proceso.

Regeneración. Esta capacidad debe estar incorporada en el sistema para regenerar la información desde el previo nivel de datos.

b) Retención.

El período de tiempo que una información debe ser retenida en un archivo específico depende de muchos factores: proceso, políticas, requerimientos del sistema, medio de almacenamiento, etc.

Estos factores al menos, deben tomarse en cuenta para fijar tiempos de retención para cada archivo.

c) Seguridad.

A medida que las bases de datos son más grandes, las consideraciones sobre seguridad son cada vez más importantes y especial atención debe darse a:

- Autorización para acceso.
- Autorización para cambio.
- Autorización para almacenamiento.
- Seguridad física.

II. SEGMENTACION DEL SISTEMA.

a) La interrelación "usuario-sistema" debe tomarse en cuenta para formar el criterio en el diseño del sistema.

Algunos de los criterios a considerarse son:

- Simplicidad de los procedimientos para el usuario.-
  Para que exista cooperación y eficiencia en el sistema que le corresponde al usuario se deben diseñar procedimientos ágiles,y fácil llenado de formas, archivo de los mismos, etc.

- Simplicidad en el manejo del sistema.-
  Se deben diseñar sistemas o sub-sistemas fáciles de manejar y mientras más óptimo sea el flujo y el personal involucrado se evitarán errores además de hacer más rápido el sistema.

- Simplicidad de los requerimientos de validación (control de calidad).-
  Optimizar sistemas de control y fáciles de operar. La auditoría interdepartamental es uno de los procedimientos de validación más importantes y los sistemas deben diseñarse pensando en esta posibilidad (sistemas abiertos).

b) Segmentación por funciones.

Una clara definición de cada una de las funciones del sistema facilita la segmentación del mismo, así como pruebas, modificaciones y mantenimiento mejorando de esta forma la eficiencia de operación.

c) Segmentación por archivos.

La segmentación en archivos debe ser tal, que maximice la utilización de dispositivos en que los archivos están almacenados sin maximizar la lógica detallada requerida para procesarlos.-
La utilización de los archivos existentes usualmente proporciona ayuda para la definición de las funciones del sistema. Es necesario encontrar la secuencia óptima para reducir el número de accesos a dos archivos.

d) Segmentación por documentos.

La producción de reportes, formatos, y otros documentos debe ordenarse de tal forma que ninguña parte del sistema se convierta lógicamente compleja.- Normalmente la orden se controla tomando en cuenta la accesabilidad de los datos requeridos de tal forma que optimice el acceso a los archivos.

e) Segmentación por proceso.

Para la mayoría de los sistema, la segmentación del nivel proceso es relativamente importante, y puede dejarse al criterio del programador.- Es importante especificar al programador cual es la lógica del proceso, cuales son los cálculos que se necesitan hacer, y asegurarse que lo entienda perfectamente..

f) Segmentación por equipo.-
Las razones para segmentar un sistema por equipos se justifica normalmente por tiempo, costo y disponibilidad de recursos. (tiempo del equipo y costo).

La segmentación de un sistema puede requerirse para:

- La aplicación de programas operacionales existentes así como sub-programas.
- Uso común de paquetes de uso general.
- La asignación de más de un programador.
- La instalación independiente de cada programa en el sistema.
- Prueba de programas y sistemas.
- Modificación y mantenimiento de sistemas.

g) La segmentación requiere una clara definición de los elementos en los que el sistema será dividido. La agrupación puede basarse en una o más de las siguientes consideraciones:

- Agrupación de funciones dependientes o relacionadas.
- Funciones aisladas únicas.
- Agrupación de procesos de acuerdo a la actividad de archivos.
- Agrupación de procesos de acuerdo con la producción de documentos.
- Enfasado cuando procesos dependientes o relacionados están agrupados.
- Enfasado cunado procesos independientes son aislados.
- Simple división de acuerdo al tamaño de la memoria.
- Simple división de acuerdo a un estimado del tiempo de corrida.

III. PERFIL DEL SISTEMA.

El perfil del sistema es un enunciado formal del criterio que regira el diseño del mismo; esto es importante porque en si es la base sobre la cual futuros mantenimientos del sistema serán hechos.-
El perfil del sistema es detallar los objetivos a mecanizar, es decir, el criterio a seguir que serán las reglas para el diseño del sistema.

a) Definir la función primaria del sistema; como ejemplo: Administración de archivos, procesamiento de datos, recuperación de información, etc.- Como función primaria es encontrar las bases y las condiciones de diseño, características especiales, políticas a seguir, restricciones.

b) Determine cual de los siguientes factores son más importantes en influenciar el diseño, y defina el significado de estos factores en términos del sistema que se está evaluando.

Tiempo, costo, complejidad de procedimientos, control de calidad, volumenes de E/S u objetivos del usuario.

IV. SISTEMAS ADMINISTRATIVOS DE ENTRADA/SALIDA.

Estos son subsistemas manuales que sirven para manejar y controlar el flujo de datos y documentos que entran y salen del sistema primario. La primerísima consideración al diseñar tales subsistemas son: organización y personal.

A mayor número de fuentes diferentes de datos o documentos de salida, mayor complejidad del subsistema. El diseño óptimo de tal subsistema deberá minimizar:

- El número de procedimientos.
- El grado de complejidad.
- La probabilidad de error.
- El número de organizaciones y personas involucradas.
- Ineficiencia.
- La necesidad de juicios individuales.
- El tiempo involucrado.

a) Subsistema de entrada.

El flujo se inicia en el punto de captura de los datos y termina con la entrega de estos datos al sistema primario.

Los puntos a considerarse son:
- Fuente de los datos.
- Identificación de documentos fuente
- Registro.
- Recolección de datos (data collection)
- Perforación (o transformación a otras formas sensibles por máquinas).
- Entrega al sistema primario.
- Edición y validación.

b) Subsistema de salida.

El flujo se inicia con la salida de los documentos o archivos del sistema primario y termina en el momento de distribución o almacenamiento.

Los procedimientos para manejar documentos son normalmente complejos y requieren una cuidadosa planeación.

Los puntos a considerarse son:
- Identificación.
- Membretado.
- Desempapelar y encuadernar.
- Distribuir.
- Empacar y enviar por correo.
- Seguridad de control y calidad.

Consideraciones sobre el equipo.

Identificación de requerimiento de equipo.

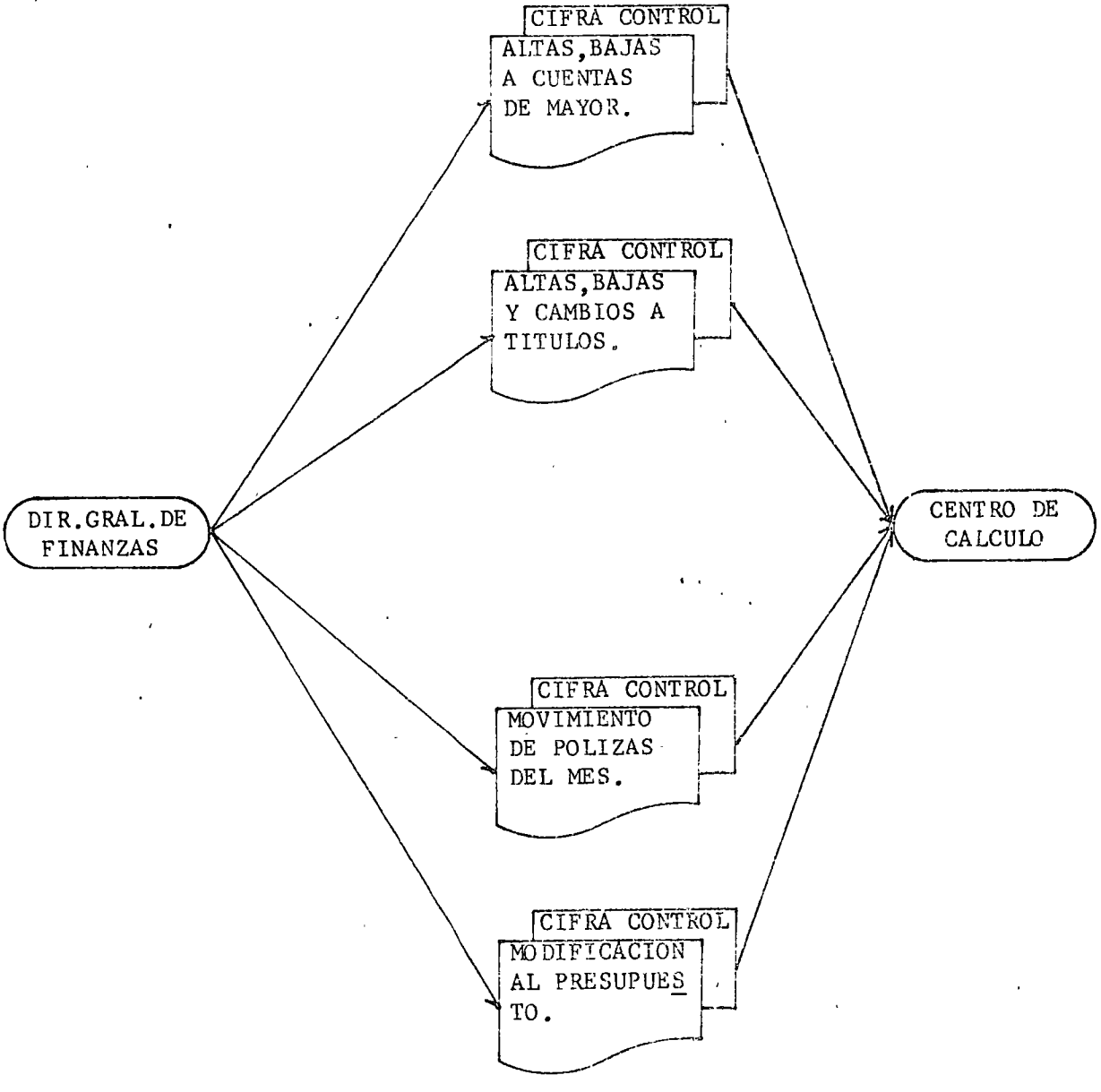
Reconciliación de equipo.

Selección de equipo en base a:

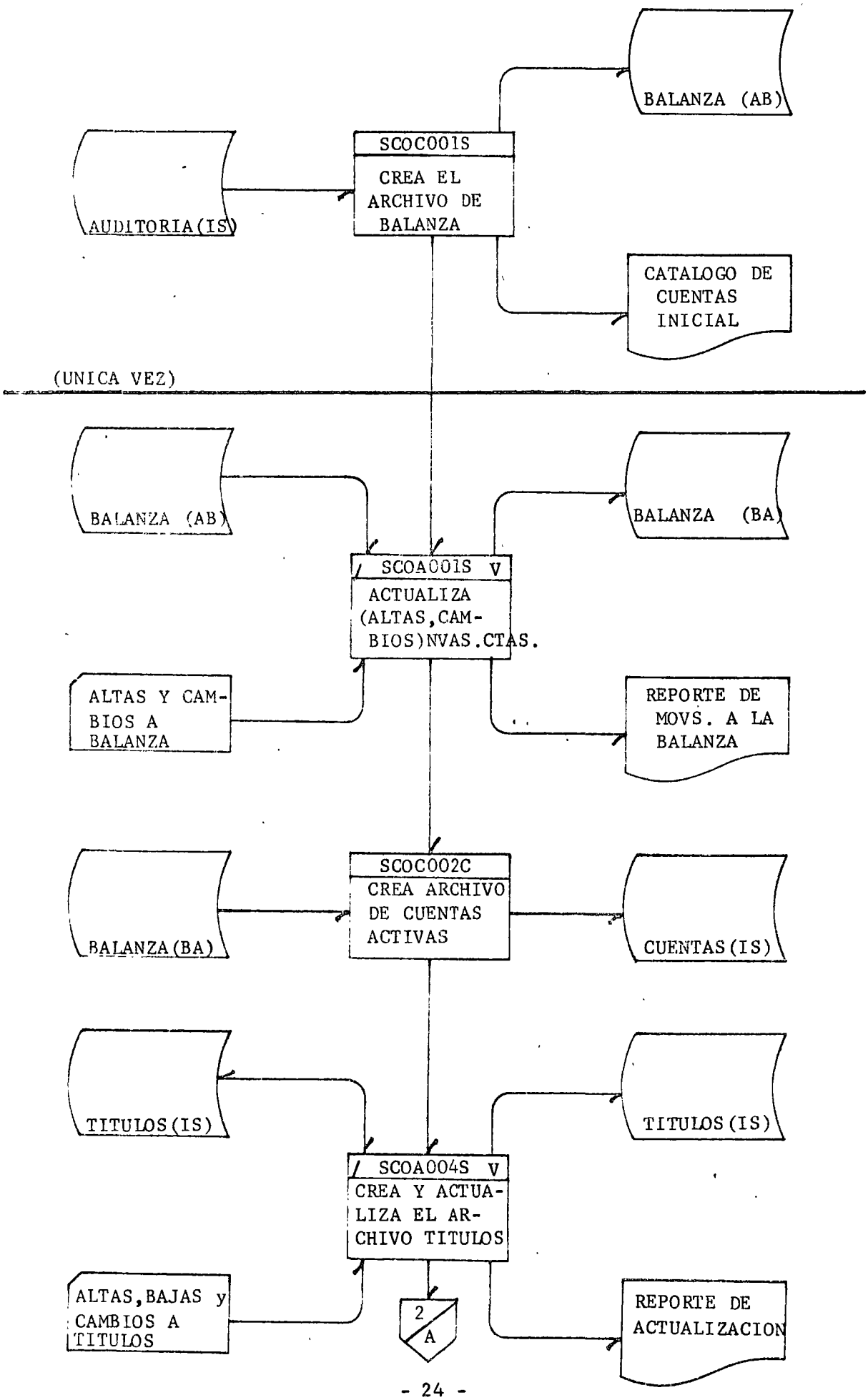
Equipo auxiliar X=Rox, microfilms, Moore Business.

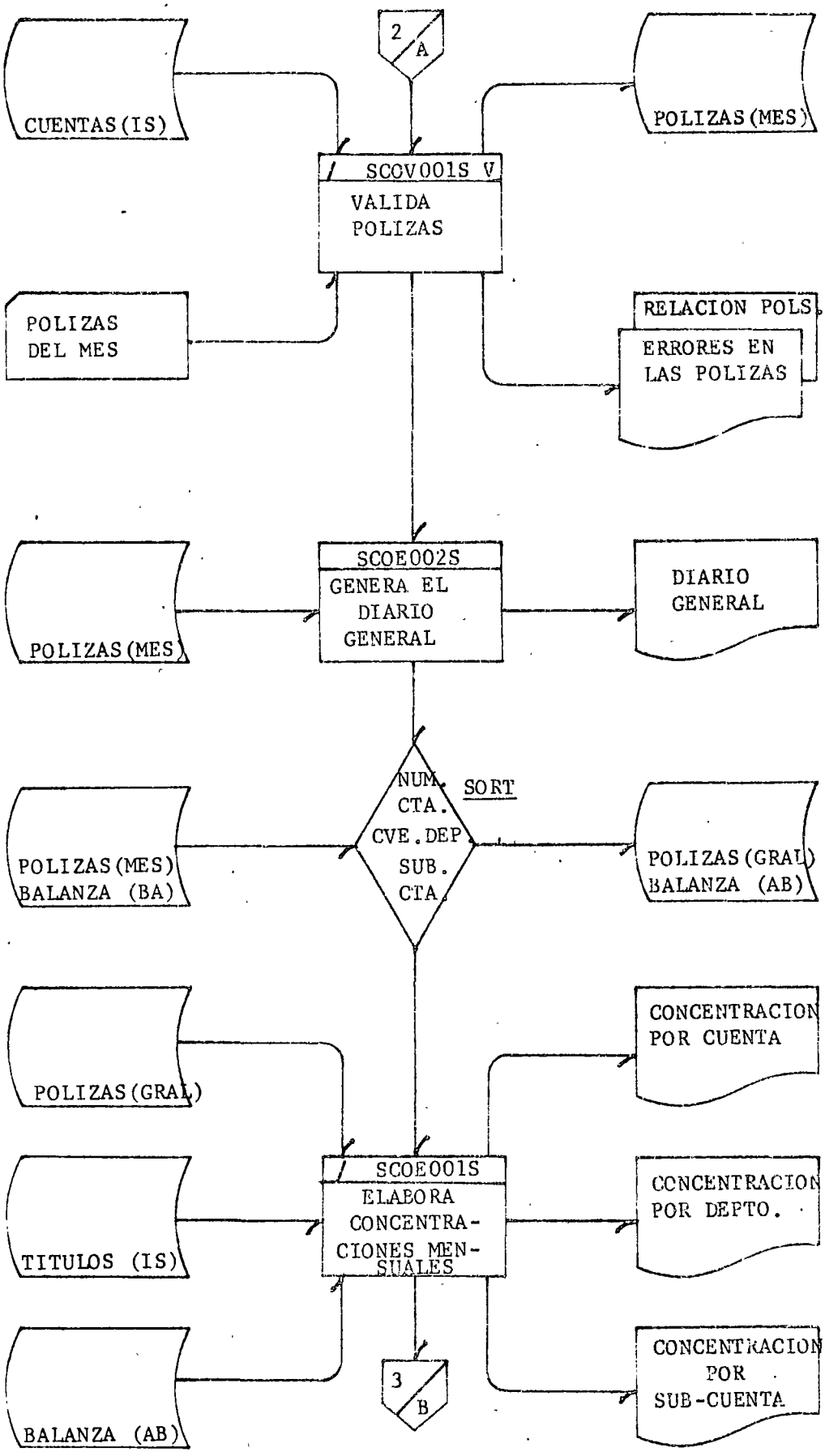
- Impacto en los sistemas diseñados.
- Justificación inicial de costo.
- Maximización de eficiencia.
- Maximización de control de calidad.
- Impacto en el comportamiento de los sistemas.
- Costo de operación por sistema.
- Disponibilidad de equipo adicional.
- Requerimientos de personal calificado.
- Requerimientos de entrenamiento.

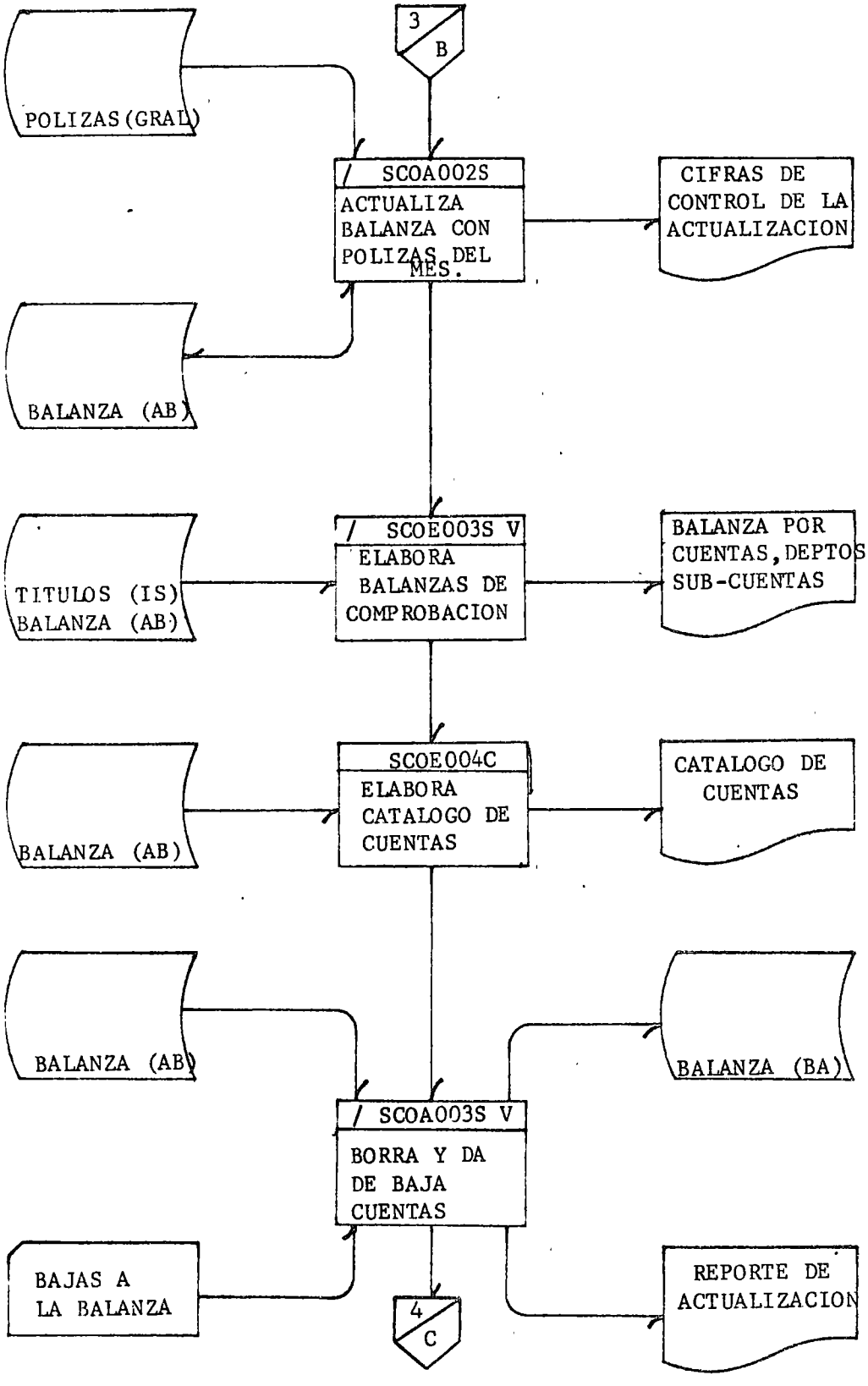
FLUJO DEL SISTEMA

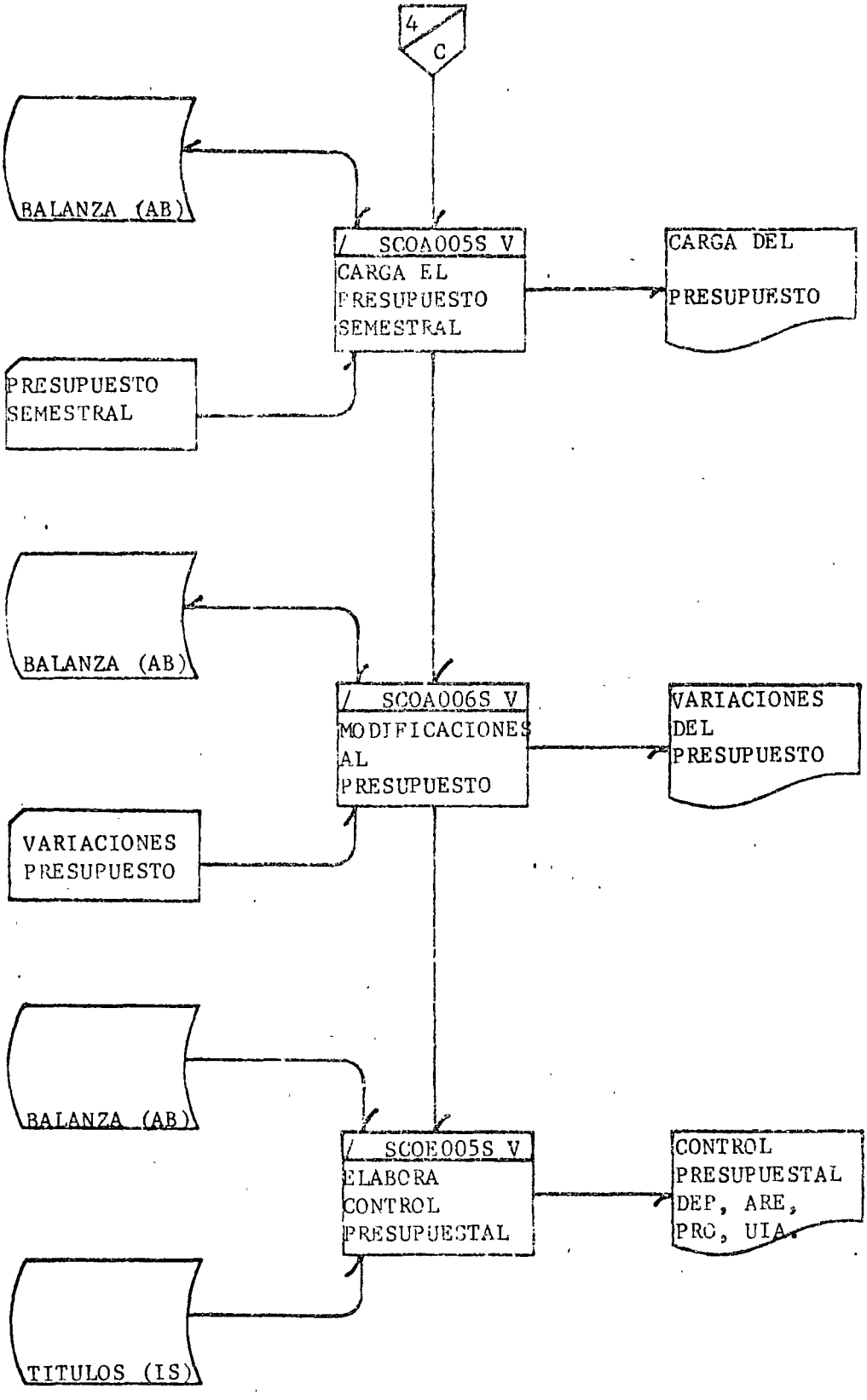


FLUJO DE CONTABILIDAD




2
A
CUENTAS(IS)
POLIZAS(MES)
SCOV001S V
VALIDA
POLIZAS
POLIZAS
DEL MES
RELACION POLS.
ERRORES EN
LAS POLIZAS
SCOE002S
GENERA EL
DIARIO
GENERAL
POLIZAS(MES)
DIARIO
GENERAL
NUM. CTA.
CVE. DEP.
SUB. CTA.
SORT
POLIZAS(MES)
BALANZA (BA)
POLIZAS(GRAL)
BALANZA (AB)
POLIZAS(GRAL)
CONCENTRACION
POR CUENTA
SCOE001S
ELABORA
CONCENTRA-
CIONES MEN-
SUALES
TITULOS (IS)
CONCENTRACION
POR DEPTO.
3
B
BALANZA (AB)
CONCENTRACION
POR
SUB-CUENTA


3
B
POLIZAS (GRAL)
/ SCOA002S
ACTUALIZA BALANZA CON POLIZAS DEL MES.
CIFRAS DE CONTROL DE LA ACTUALIZACION
BALANZA (AB)
/ SCOE003S V
ELABORA BALANZAS DE COMPROBACION
TITULOS (IS)
BALANZA (AB)
BALANZA POR CUENTAS, DEPTOS SUB-CUENTAS
SCOE004C
ELABORA CATALOGO DE CUENTAS
BALANZA (AB)
CATALOGO DE CUENTAS
BALANZA (AB)
BALANZA (BA)
/ SCOA003S V
BORRA Y DA DE BAJA CUENTAS
BAJAS A LA BALANZA
REPORTE DE ACTUALIZACION
4
C


4
C
BALANZA (AB)
SCOA005S V
CARGA EL PRESUPUESTO SEMESTRAL
CARGA DEL PRESUPUESTO
PRESUPUESTO SEMESTRAL
BALANZA (AB)
SCOA006S V
MODIFICACIONES AL PRESUPUESTO
VARIACIONES DEL PRESUPUESTO
VARIACIONES PRESUPUESTO
BALANZA (AB)
SCOE005S V
ELABORA CONTROL PRESUPUESTAL
CONTROL PRESUPUESTAL DEP, ARE, PRO, UIA.
TITULOS (IS)

I

INTRODUCCION A LA PROGRAMACION

a) QUE ES UN PROGRAMA.

Todos los pasos necesarios y las correspondientes instrucciones para dirigir al sistema hacía la resolución de un problema o para llevar a cabo una aplicación de proceso.

b) ESTRUCTURA LOGICA DE UN PROGRAMA.

La computadora sólo es capaz de obedecer instrucciones simples, mismas que va ejecutando una por una, sin percatarse en ningún momento, en forma global, del problema que en un tiempo dado esté resolviendo.

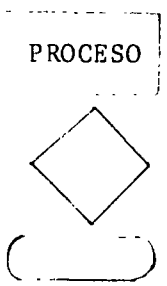
El hombre es el encargado de ensamblar estas instrucciones simples en forma tal que tengan un significado.

Para esto se cuenta con instrucciones básicas, las demás son sólo derivaciones de éstas:

a) Aritméticas (+ - x /)
b) Lógicas (compara, y si (condición) entonces - - - - -)
c) Operativas (mueve: instrucción que permitirá llevar información de un lugar a otro.
salta: cambiar la secuencia normal de ejecución de las instrucciones.
lee: usada para traer información de un archivo manejado por un dispositivo periférico usado dentro del programa.
escribe:

d) Control (alto: indicar donde debe detenerse el proceso de un programa, inicio.)

Diagrama de Flujo.-
Es un método para ayudar al programador a visualizar sobre papel sus ideas sobre como organizar la secuencia de pasos o eventos necesarios para resolver un problema.



= haga algo

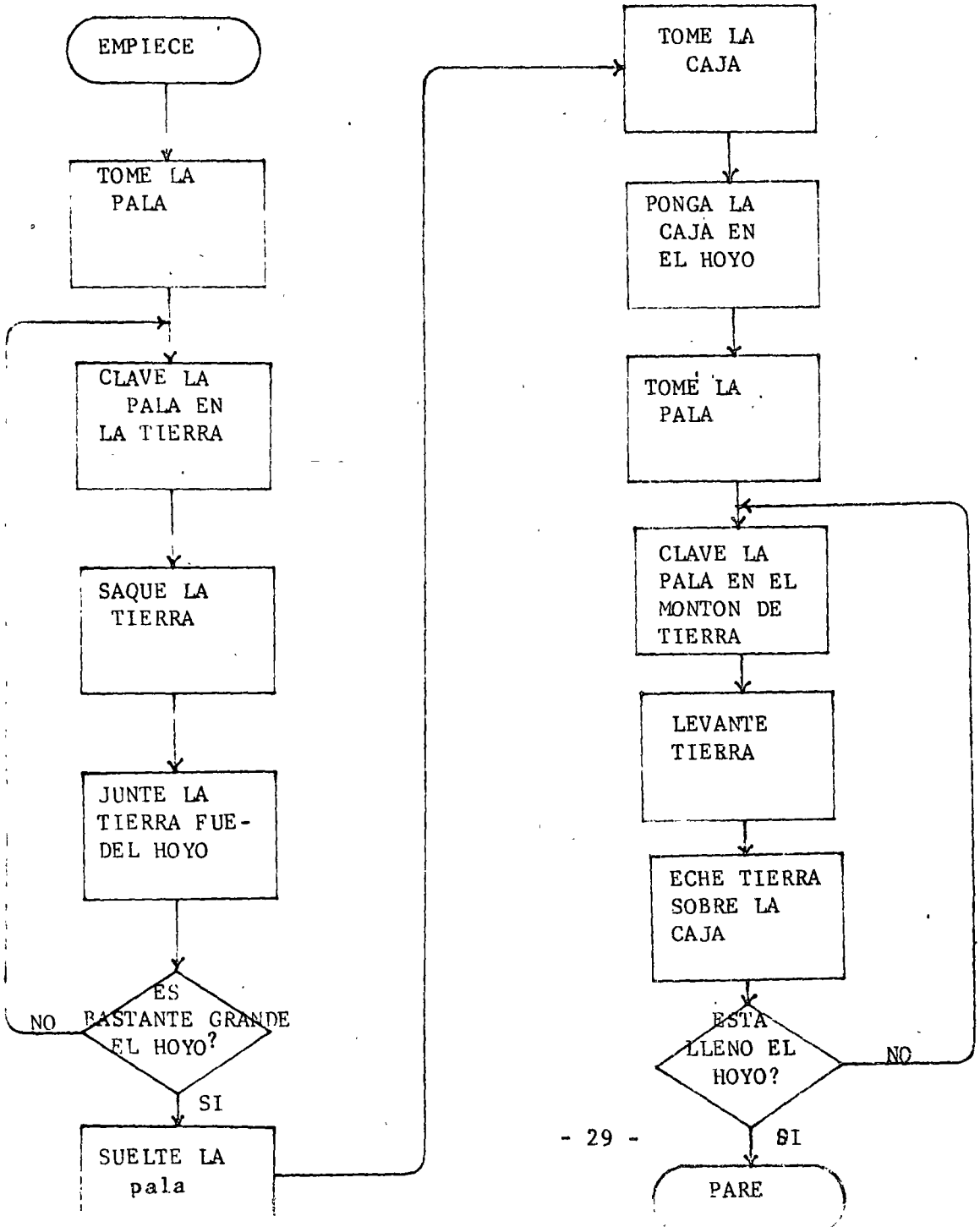
= decisión respuesta si/no

= inicie
pare

Problema:

Cavar un hoyo y enterrar un cofre con un tesoro.
Lista de etapas para un ser humano:
1) Tomar una pala.
2) Cavar un hoyo suficientemente amplio.
3) Colocar el cofre en el hoyo.
4) Llenar el hoyo.

Recordando que a un computador debemos decirle todo, veremos en un diagrama de flujo como quedan las cuatro etapas anteriores (suficientes para un ser humano) se han convertido en trece. El diagrama de flujo representa la secuencia global de eventos según la cual puede ser resuelto el problema e indica el nivel de detalle requerido para instruir a un computador.



3.3 Lenguaje Máquina.

Llamamos lenguaje de máquina al grupo de instrucciones que son identificadas directamente por la computadora y le permiten realizar un proceso.

El programa que se almacena en la memoria principal y que permite operar a la computadora, está integrado por instrucciones legibles a éste; es decir, está integrado por instrucciones del lenguaje de máquina.

El lenguaje de máquina se compone de instruccciones que la computadora identifica fácilmente y que le permiten procesar los datos que han de manejarse con el propósito de obtener información.

Programar en lenguaje de máquina resulta sumamente complicado ya que es necesario utilizar como método de programación el proceso que el computador lleva a cabo, paso a paso e instrucción por instrucción. Además, es necesario señalar en forma explícita:

- Las direcciones de memoria donde debe cargarse cada una de las instrucciones del programa.
- Las direcciones de memoria donde deben reservarse registros para operaciones de entrada/salida.
- Las direcciones donde deben almacenarse los datos constantes y variables que serán usados por el programa.

3.4 Superlenguajes.

Al traductor de superlenguajes se le llama compilador. Un compilador nos permitirá pasar un programa escrito en superlenguaje a lenguaje de máquina. Superlenguajes: COBOL, FORTRAN, BASIC, ALGOL y PL1.

COBOL.- "Common Business Oriented Language" es el lenguaje más usado comercialmente y casi todos los fabricantes de computadoras incluyen como soporte de la máquina un compilador de Cobol.

FORTRAN.- Lenguaje orientado a problemas científicos de carácter matemático; debe ser usado exclusivamente para resolver problemas científicos y de investigación.

BASIC.- Lenguaje conversacional que permite a personas con poca experiencia en programación realizar consultas a través de la computadora, Muy usado también hoy día en teleproceso.

ALGOL y PL1.- Superlenguajes de menor uso; no son manejados tan universalmente como el Cobol y Fortran; por otra parte, sólo algunos proveedores cuentan con estos lenguajes como soporte para sus máquinas. El PL1 es una mezcla de Cobol y Fortran, con ello se hace un lenguaje muy poderoso.

V.- CONVERSION DE NUMEROS DE BASE DECIMAL A OTRAS BASES DE NUMEROS O CONVERSION A Y/O DE BINARIO, OCTAL, HEXADECIMAL.

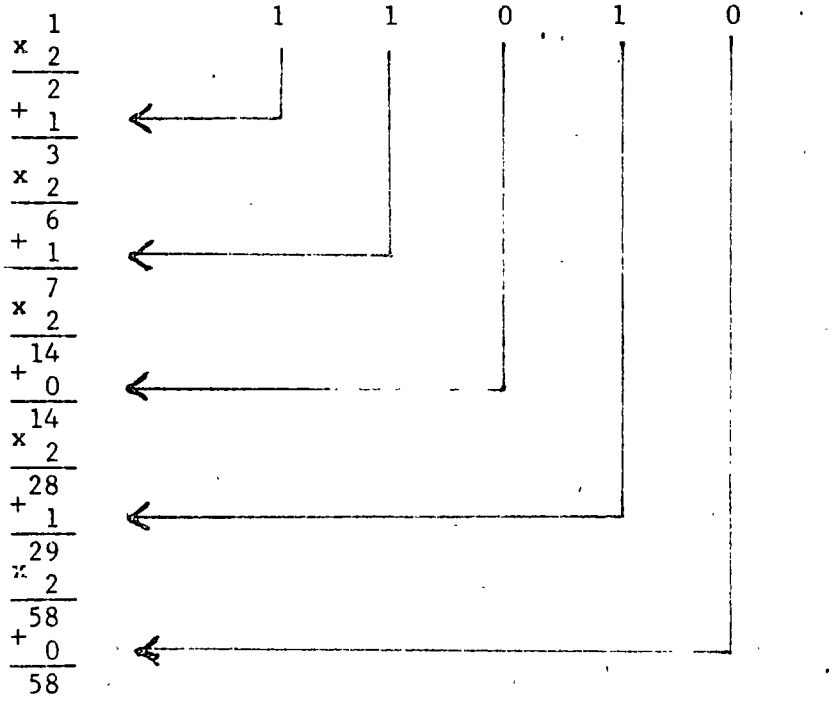
En el sistema binario, los únicos números aceptables son 0 y 1. Como "bi" significa dos, ese sistema se llama binario.
Conversión del sistema decimal al binario, consiste en dividir sucesivamente la cantidad decimal por dos. El resto de cada división sucesiva, leyendo hacia arriba, constituye el equivalente binario.

2	85	sobrante
2	42	1
2	21	0
2	10	1
2	5	0
2	2	1
2	1	0
2	0	1

Leer hacia arriba

Por lo tanto, 85 en decimal equivale a: 1010101 en binario

Conversión del sistema binario al decimal, sigue exactamente la rutina contraria, o sea que para convertir un valor, binario a su equivalente decimal, cada dígito binario, a partir de la izquierda, se multiplica por dos, y su producto se suma al número, hasta que se convierte toda la cantidad binaria. Por ejemplo, para convertir la cantidad binaria 111010, procederemos así:



Por lo tanto, 111010 en binario, equivale a 58 en decimal

En el sistema octal sólo hay los dígitos de 0 a 7. Los números 8 y 9 nunca se usan.
Un número en clave octal puede convertirse a decimales, multiplicando cada dígito octal por 8 (comenzando con el de la extrema ezquierda) y sumando su producto al dígito de la derecha, hasta que se incluye todo el número. Por ejemplo, 721 en octal, se convierte a decimales, como sigue:

```
                7    2    1
Multiplicar   x 8
              ----
              56
Sumar         + 2
              ----
              58
Multiplicar   x 8
              ----
              464
Sumar         + 1
              ----
              465
```

Por lo tanto, 721 en octal es igual a 465 en decimales

Un valor decimal se convierte a octal dividiéndolo sucesivamente entre 8. El sobrante de cada división representa un número octal. Por ejemplo, el valor decimal de 614 se convierte a octal del modo siguiente:

```
8 |  614     sobrante
8 |   76        6
8 |    9        4
8 |    1        1
8 |    0        1    1    1    4    6   (equivalente octal)
```

Los números en clave octal pueden convertirse fácilmente a binarios asignando un número fijo de tres dígitos binarios a cada dígito octal. Por ejemplo, el número octal 721 se convierte a binario del modo siguiente:

```
Octal                  7      2      1
equivalente binario   (111)  (010)  (001)
```

El sistema hexadecimal para manejar esos largos números, porque cada dígito hexadecimal representa cuatro bits binarios.
Hexadecimal usa una base de 16; como el sistema decimal sólo proporciona diez dígitos para representar los diez primeros valores del sistema hexadecimal (0-9), los seis valores hexadecimales restantes se representan arbitrariamente con las seis primeras letras del alfabeto A, B, C, D, E y F.
La conversión de hexadecimal a binario se hace reemplazando cada dígito hexadecimal con la serie equivalente de cuatro dígitos binarios. Por ejemplo, el número hexadecimal E 7 A se convierte a binario del modo siguiente:

```
Hexadecimal            E       7       A
equivalente binario   (1110)  (0111)  (1010)
```

Un número binario se convierte a hexadecimal, separando los dígitos binarios (comenzando con el de la extrema derecha), en series fijas de cuatro dígitos. Si la última serie tiene menos de cuatro dígitos, entonces se añaden ceros a la izquierda. Por ejemplo, el número binario 010111001011 se convierte a hexadecimal del modo siguiente:

binario	\0101/	\1100/	\1011/
equivalente hexadecimal	5	C	B

Un número decimal se convierte a hexadecimal dividiéndolo sucesivamente entre 16. El sobrante de cada división representa un número hexadecimal. Los sobrantes de 10 a 15 deben convertirse a su número hexadecimal de A a F. Por ejemplo, el valor decimal 1 9 7 0 se convierte a hexadecimal del modo siguiente:

		sobrante
16	1970	
16	123	2
16	7	(11) B
16	0	7

→ 7 B 2 (equivalente hexadecimal de 1970)

La conversión de hexadecimal a decimal se efectúa sencillamente con una serie de multiplicaciones y sumas.

1.- Multiplicar el dígito hexadecimal más significativo (el de la extrema izquierda) por 16.
2.- Sumar el siguiente dígito hexadecimal más significativo al producto, y multiplicar la suma por 16.
3.- Continuar el procedimiento de suma y multiplicación, hasta que se ha sumado al último producto el dígito hexadecimal menos significativo (el de la extrema derecha).

Con el resultado hexadecimal del ejemplo anterior (7 B 2), procederemos como sigue:

	7	B	2	Hexadecimal
Multiplicar	x 16			
	112			
Sumar	+ 11	←		
	123			
Multiplicar	x 16			
	1,968			
Sumar	+ 2		←	
	1,970			

Así, pues, 7 B 2 equivale a un valor decimal de 1970.

PROGRAMACION DEL SISTEMA

A. La etapa de programación constituye la implementación del diseño del sistema. Esta etapa se inicia después de que han sido totalmente definidos y especificados los programas que componen al sistema. En este punto tanto los diseños de entrada como los de salida deben estar ya completamente definidos.

El analista debe tener en cuenta, para la implementación del sistema, que los programas pueden clasificarse en tres categorías:

1.- Programas utilizados para un aplicación especifica.-
Son aquellos programas que se desarrollan para una cierta corrida y que realizan una sola función (la mayoría de los programas cae dentro de esta categoría.

2.- Programas utilizados como función de soporte a una aplicación.-
Son aquellos programas de uso similar en varios pasos o etapas de un sistema o aplicación (por ejemplo: un cierto listado que se use tres veces dentro de un sistema).

3.- Programas utilizados en función a un soporte general.-
Son aquellos programas de uso muy general que pueden ser utilizados por distintas aplicaciones de la empresa.

NOTA:
Cuando en una instalación se cuenta con programas que caigan en las categorías 2 y 3 se ahorra esfuerzo de programación, se obtiene gran modularidad y los sistemas serán fáciles de mantener.

B. ESPECIFICACION DEL PROGRAMA.

La mayor o menor extensión de la especificación de un programa depende de la complejidad del programa en cuestión.

Se siguen varios métodos para especificar un programa:

1.- Narrativo.-
Especificar un programa en forma narrativa consiste en describir el trabajo mediante palabras como si fuera un reporte o memorándum..

2.- Tabla de decisiones.-
Cuando existe un situación compleja de condiciones y acciones la tabla de decisiones o una forma tabular es el método adecuado para especificar un programa.

3.- Otros.-
Para programas sencillos en ocasiones la simple presentación de muestras o ejemplos, es suficiente especificación del programa deseado; por ejemplo: copia de un listado.

NOTA:
En muchos casos el método adecuado será la combinación de los tres métodos señalados.

C. DIAGRAMACION (FLOWCHARTING).

El aspecto de más controversia en la programación es el nivel de diagramación requerido en un programa.-
La recomendación en términos generales consiste en hacer diagramas generales, sin demasiado detalle, sin embargo, la simplicidad de los diagramas estará en forma directa a la experiencia del programador.

a) Niveles de Flow Chart.

- Diagramas de presentación.-
  Son aquellos que se preparan con símbolos no estandard como para que los entiendan los ejecutivos.

- Diagrama a nivel sistema.-
  Son los que se conocen como diagramas de flujo y deben dibujarse con símbolos estandard.

- Diagramas de programa (macrolevel)
  Diagramas de bloque a un nivel general.

- Diagramas de programa (microlevel)
  Diagramas de bloque detallados.

b) Convención de símbolos y dirección.

Los "flowchart" a cualquier nivel deben elaborarse utilizando los símbolos standard usados en procesamiento de datos; así como las direcciones de flu convencionales. Estos estandares son importantes para mejor comprensión del programa por personas ajenas a aquel que hizo el diagrama. Eso ayuda para el mantenimiento de programas.

c) Conectores.

Es conveniente utilizar tanto para los conectores de salida como para los de entrada nombres que permitan reconocer fácilmente una "referencia cruzada" entre el flowchart del programa y su codificación.

D, CODIFICACION.

Después de haberse elaborado el diagrama del programa y de haberse hecho la prueba de escritorio, el programador debe codificar el programa en el lenguaje adecuado.

La selección del lenguaje de programación utilizado, es responsabilidad del jefe o supervisor de programación, y debe ser asesorado por el analista del sistema, la elección debe hacerse en base a los siguientes puntos:

- Tipo de aplicación.
- Facilidad de mantenimiento del programa.
- Velocidad de depuración.
- Eficiencia del programa objeto.
- Velocidad de codificación.
- Necesidad de asesoramiento.

a) Simplicidad.-
La codificación debe ser lo más clara y sencilla posible, debiendo utilizar en la codificación instrucciones claras y rutinas sencillas sin repetición.

b) Comentarios.-
Recalcar en que son necesarios para las personas que no tienen idea de lo que hace el programa.

c) Etiquetas.

- Identificación de programas .-
Desarrollar una técnica para la identificación de programas.

- Símbolos internos.

d) Altos programados.-
Recurrir lo menos posible a ellos, pero cuando son necesarios documentarlos adecuadamente.

e) Comunicación con el operador.-
Derivado de situaciones anormales, se debe preparar mensajes que permitan al operador actuar en un momento dado.

f) Calidad de la codificación.-
Cuidadosa y limpia.

g) Construcción de un programa.-
Existen tres fases:
1.- Poner en claro las especificaciones del programa.
2.- Dibujar el diagrama de bloque.
3.- Codificación.

E. DATOS DE PRUEBA.

Los datos de prueba para cada uno de los programas deben ser elaborados de tal manera que permitan probar el funcionamiento correcto del programa, y deben ser de dos tipos:

- Cualitativos.-
  Que prueben todas las condiciones.

- Cuantitativos.-
  Datos en suficiente volumen para probar todas estas condiciones.

F. DEPURACION DEL PROGRAMA.

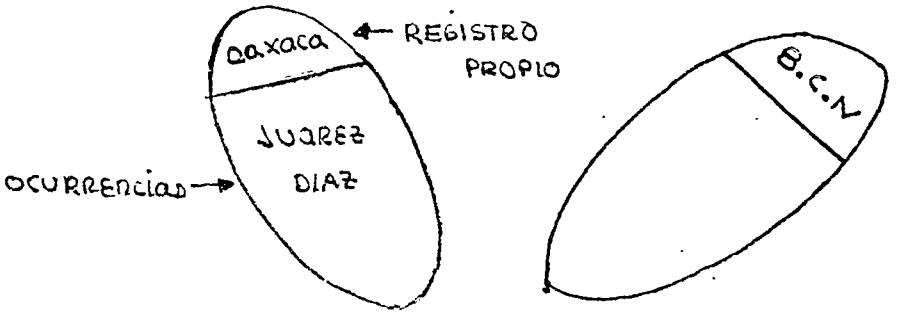
Un conjunto de ocurrencias es una ocurrencia de un tipo de registro propio con cero o más ocurrencia de tipos de registro miembros.

En cada conjunto de ocurrencias las siguientes relaciones existen:

- Dado un registro propio, es posible procesar todos los registros miembros asociados a ese conjunto de ocurrencias.
- Dado un registro miembro, es posible procesar el registro propio asociado a ese conjunto de ocurrencias.
- Dado un registro miembro, es posible procesar todos los demás registros miembros asociados a ese conjunto de ocurrencias.

Cualquier implementación que satisfaga estas tres reglas, será una implementación válida de el concepto de tipos de conjunto.

Imaginemos los siguientes conjuntos de ocurrencias.



Puede suceder que un conjunto de ocurrencias tenga un registro propio pero no registros miembros. A este conjunto le llamaremos vacío.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

SISTEMAS DE INFORMACION GERENCIAL

SISTEMAS BATCH Y EN LINEA

DR. VICTOR GUERRA

AGOSTO, 1978.

SISTEMAS BATCH VS. EN LINEA

Características trabajo "Batch" en paquete:

- No se puede accesar los archivos centrales mientras corre el trabajo "batch"

- Generalmente los procesos de validación se hacen por separado de los procesos de actualización.

- Costo de operación bajo { en líneas de transmisión / en CPU

- Obtención de resultados en varias horas. Los sistemas se corren una vez al día o a la semana, etc.

- Es ideal para iniciarse en el área de sistemas o para sistemas "pequeños"

- Poco impresionante.

Características trabajo en línea:

- Los archivos están siendo modificados al momento de accesar en dato

- La validación y edición se realiza en el momento.

- Costo de operación alto
  - en líneas de transmisión
  - equipo de telecomunicación
  - en procesador I/O
  - manutención del sistema

- Obtención inmediata de resultados

- Es costeable en sistemas muy grandes como: cuentas bancarias, cuentas de crédito, aviación.

- Muy impresionante (se abusa mucho)

Captura Directa

- Consiste en capturar la información en el mismo lugar donde se generó; eliminando perforistas o intermediarios. La captura se realiza a través de terminales o en papelería especial (tarjetas pre-perforadas o caracteres ópticos)

- Ventajas
  a) eliminación de costos relativos a perforistas
  b) menos errores dada la verificación inmediata
  c) velocidad de captura.

- Desventajas
  a) inversión inicial alta
  b) personal entrenado y responsable
  c) procesos administrativos confiables y eficientes.

- Este es el método ideal de captura dado que se realiza en la misma "ventanilla", con el personal responsable y generalmente con la fuente de la información unidos.

- Conforme los dispositivos son más sofisticados será más fácil realizar captura directa.

Captura por contrato

- Ventajas

a) es recomendable cuando se realizan trabajos largos por una sola vez.

b) muchas veces es más económica que haciéndola en casa.

c) se pueden exigir niveles altos de confiabilidad.

d) hay que tener buenos mecanismos de control.

e) excelente para organizaciones pequeñas.

- Desventajas

a) No es útil cuando hay poco tiempo disponible para la captura.

b) Puede ser caro para grandes volúmenes de información.

c) Seguridad.

TABLA 1

INSTALACION	CHICA	MEDIANA	GRANDE	GIGANTE
# de personas	1-10	5-25	5-100	100-1000
Equipo (tipo OBM)	360/20 a Sistema 3	360/40 a 370/145	360/50 a 370/158	varios procesadores. 360/65 a 370/168
Ambiente de operación	Propósitos Especiales	Uso general, Servicios Administrativos	Uso general, Telecomunicación, cálculos sofisticados.	Uso general, Muchas telecomunicaciones, cálculos sofisticados
Procesos (principalmente)	BATCH	BATCH-LINEA	BATCH-LINEA	LINEA
Costo de operación en dólares	60,000	100,000	300,000	1.5 Millones

EQUIPOS PARA TRABAJOS "BATCH" (EN PAQUETE)

Cualquier equipo con memoria local sirve para preparar trabajos BATCH

- Terminal con lectora e impresora de papel, cassette o diskette.

  Terminales 3735, 3740, 3770 IBM

- Tarjetas perforadas.

- Cinta magnética.

EQUIPO PARA TRABAJO EN LINEA

Cualquier terminal (económica)

- Teleimpresora:Decwriter, Diablo 2,000 - 5,000 U.S. Dlls. (más lenta pero con escritura de los mensajes)

- Terminal de Video CRT
  (rápidas) 3270 IBM o semi TC-Burr
  1,500 - 5,000 U.S. Dlls.

Estas terminales pueden ser con memoria local dependiendo si se necesita edición, validación o algún tipo de preproceso antes de enviar al computador central.

- Perforadora sin memoria

  a) 125 - 200 U.S. Dlls. mes.

  b) Descomposturas periódicas.

  c) Alto costo de las tarjetas.

  d) Baja velocidad de captura.

  e) Necesidad de verificar.

  f) Desperdicio de tarjetas.

  g) IBM 029...

- Perforadora con memoria

  a) Más cara que la sin memoria.

  b) Descomposturas periódicas.

  c) Costo de las tarjetas.

  d) Baja velocidad de captura

  e) Necesidad de verificar.

  f) No se desperdician tarjetas.

  g) Univac

- Lectora a cinta a disco

  a) varias estaciones con un solo procesador.

  b) posibilidad de formatear, verificar y validar.

  c) Menos costosa que las perforadoras.

  d) Menor procentaje de errores

  e) Más velocidad.

  f) Costeable cuando son más de 8 perforadoras.

- Lectoras ópticas

a) Hay de varios tipos y costos 100,000 - 2,000 U.S. Dlls.

b) Gran velocidad de captura.

c) Mínimo de errores.

d) Algunos reconocen caracteres escritos a mano, la mayoría reconocen sólo marcas negras.

ORGANIZACION Y CONTROL DE LA FUNCION DE ADQUISICION DE DATOS

L.P.M.I.D.U.S.

- Es el proceso de obtener información leíble por la computadora.

- Tradicionalmente se realiza con perforadoras de tarjetas:

  1.- Enviar todas las formas a perforar.

  2.- Verificar y correr programas de validación.

  3.- Correr los programas de aplicación,

Hoy esta forma de proceder no es la mejor.

EQUIPOS DE CAPTURA

* Perforadoras sin y con memoria.
* Lectoras a cinta magnética o disco,
* Lectoras ópticas.
* Teleimpresoras o Video-terminales en línea.

Operación de una instalación con perforadoras

$2.50 a $3.00 por tarjeta

Perforista	60%	$7,000 al mes 21 días por mes 7 horas diarias 8,000 caracteres por hora
Renta de la perforadora	22%	125 U.S. Dlls perf. 100 U.S. Dlls verif.
CPU	12%	200 t.p.m. $1,000 hr de CPU
Tarjetas	8%	$50 por 1,000 tarj.
	100%	

Datos numéricos	Alfanuméricos
10-20% más velocidad	0%
20-30% más errores	1 a 2% p.car.
Graficación	.05%

Instalación sin perforadoras

30 a 40% más productivos.

Administración del Proceso de Captura de Datos

- Además de haber seleccionado el equipo más eficiente es indispensable administrar con igual eficiencia al proceso en todo momento. Hay 5 tareas principales:

  1.- Preparación del documento.

  2.- Conversión del documento (perforación).

  3.- Detección de errores.

  4.- Correccion de los errores.

  5.- Reiniciación del proceso.

- Sistemas en línea:

  * El usuario en su terminal realiza las 5 tareas.

  * La administración se torna pasiva: mantener "vivo" al sistema.

  * Si el sistema se cae (muere) hay que reinicializarlo.

  * Hay que tener un equipo que vigile y repare los equipos de telecomunicación

- Sistemas batch en línea

  * Es similar al anterior pero no se detectan todos los errores dado que no se actualizan los archivos en el momento de la captura.

  * Es más conveniente tener un sistema por separado para la captura que estar conectados al CPU.

  * Ejemplos: CADE de UNIVAC, CYBER-DATA de CDC
    IBM 3740.

- Sistemas fuera de línea

  * Existe una organización universal

Administrador del sistema

Administrador de operación

Administrador de analistas

Supervisor de preparación de datos	Supervisor de control de los datos	Supervisor de operación
Supervisor de turno	encargados del control	operadores senior
Perforistas		Operadores Junior

* Problemas más comunes:

a) Familiarizar a los perforistas con los datos.

b) Producir listados de error que se puedan perforar directamente una vez corregidos.

c) Horarios definidos.

d) Hay que instaurar mecanismos de control de la información: varias copias del documento (3 ó 4).

- Otros factores importantes

* Tacto en el manejo de los operadores; hacerlos participar en la problemática.

* Selección de personas con características especiales para el trabajo.

* Entrenamiento

a) en casa o por el proveedor.
b) conocimiento de la organización de la compañía.
c) reglas laborales.
d) localización de los materiales de trabajo.
e) mantenimiento básico del equipo.
f) corrección y detección de errores.

DIRECTORIO DE ALUMNOS DEL CURSO : "SISTEMAS DE INFORMACION GERENCIAL", DEL 4 AL 26 DE AGOSTO DE 1978.

1. ING. OSCAR AGUILAR CAMACHO
PACHUCA No. 56
COL. VALLE CEYLAN
TLANEPANTLA, MEX.
TEL. 390-19-37

S. A R. H.
JEFE DE DEPTO.
REFORMA No. 69
MEXICO, D. F.
TEL. 546-59-85

2. SR. JOSE MANUEL AVILA FLORES
NORTE 79-B No. 198-4
COL. ELECTRICISTAS
MEXICO 16, D. F.
TEL. 561-86-24

ORGANIZACION MEXICANA DE CONSTRUCCIONES, S. A.
GERENTE DE COMPUTACION
INSURGENTES SUR No. 1650 - 10 º PISO
COL. FLORIDA
TEL. 534-47-05

3. SR. JOSE LUIS BECERRA LOPEZ
M. OCARRANZA No. 107
COL. MIXCOAC
MEXICO 19, D. F.
TEL. 593-52-22

UNIVERSIDAD INTERCONTINENTAL
PROFESOR
INSURGENTES SUR No. 4135
TEL. 573-85-44

4. C.P. ARELI BETANCOURT OLVERA
OLIVOS No. 34
FRACC. JARDINES DE ATIZAPAN
ATIZAPAN, EDO. DE MEXICO
TEL. 91-594-21600

TELEFONOS DE MEXICO, S. A.
AUDITOR INTERNO
PARQUE VIA No. 198 OFNA. 125 "SHIRLEY"
COL. CUAUHTEMOEC
TEL. 535-26-68

5. SR. OSCAR CARMONA CRUZ
CAMPIÑA No. 154
COL. PASTORES
EDO. DE MEXICO
TEL. 560-67-17

PETROLEOS MEXICANOS
COORDINADOR DE PROYECTOS
MARINA NACIONAL No. 329
COL. ANAHUAC
MEXICO 17, D. F.

6. ING. HECTOR CASTRO BAUTISTA
AV. CUAUHTEMOC No. 1026-A
COL. DEL VALLE
MEXICO 12, D. F.
TEL. 575-23-46

INSTITUTO MEXICANO DEL PETROLEO
JEFE DEL DEPTO. DEL SISTEMA DE INFORMACION
DE SUBD. PROYECTO
AV. CIEN METROS No. 152
COL. INDUSTRIAL VALLEJO
MEXICO 14, D. F.
TEL. 567-66-00 EXT. 2421

7. SR. PORFIRIO N. CORDOVA SANCHEZ
ORIENTE 172 No. 115
COL. MOCTEZUMA
MEXICO 9, D. F.
TEL. 762-02-52

OFICINA DE EVALUACION DE PROYECTOS
Y PROG. D.D.F.
TECNICO EVALUADOR
PINO SUAREZ No. 15
CENTRO
MEXICO 1, D. F.
TEL. 522-64-38

8. ACT. SERGIO DEL VILLAR MARTINEZ
AV. UNIVERSIDAD No. 1953 EDIF. 34-304
COL. COPILCO
MEXICO 21, D. F.
TEL. 550-32-16

FACULTAD DE MEDICINA - UNAM
ANALISTA DE SISTEMAS
CIUDAD UNIVERSITARIA
MEXICO 20, D. F.
TEL. 548-99-48

9. SR. JESUS N. FIGUEROA CERVERA
AV. DE LAS GRANJAS No. 86
COL. SECOTR NAVAL
MEXICO 16, D. F.
TEL. 561-11-14

SECRETARIA DE PROGRAMACION Y PRESUPUESTO
JEFE DE LA UNIDAD DE CONTROL DE CONTRATOS
IZAZAGA No. 38 P.B.
MEXICO 1, D. F.
TEL. 521-75-44

10. SR. JUAN ANTONIO FRANCO DIAZ
MARTHA No. 70
COL. GPE. TEPEYAC
MEXICO 14, D. F.
TEL. 517-79-87

SECRETARIA DE LA REFORMA AGRARIA
AUXILIAR DE JEFE DE PROCESAMIENTO DE DATOS
F.S.T. DE MIER No. 127- P.B.
COL. OBRERA
MEXICO 8, D. F.
TEL. 588-21-67

11. SRITA. GPE. BARBARA GARCIA HDEZ.
CALLE DEL LAGO No. 7
FRACC. AMPLIACION LOS FRESNOS
NAUCALPAN, EDO. DE MEXICO

BANCO DE MEXICO, S. A.
PROGRAMADOR
CONDESA No. 6 - 3 º PISO
CENTRO
MEXICO 1, D. F.
TEL. 585-42-99

12. ING. J. MARIO GARCIA LUGO
TLAXCALA No. 5
COL. FIDEICOMISO
CD. LAZARO CARDENAS, MICH.

SERVICIOS PORTUARIOS DE LAZARO
CARDENAS, S. A. DE C. V.
SUB-JEFE AREA OPERACIONAL
CALZ. AL PUERTO S/N
LAZARO CARDENAS, MICH.
TEL. 2-03-33 EXT. 1103 - 1107

13. SR. XAVIER HARO SOLORZANO
M. CERVANTES SAAVEDRA No. 647-9
COL. IRRIGACION
MEXICO 10, D. F.

SARH. COMISION DE AGUAS DEL VALLE DE MEXICO
JEFE DE OFICINA
BALDERAS No. 55 - 2 º PISO
CENTRO
MEXICO 1, D. F.
TEL. 510-02-94

14. SRITA. GUADALUPE ISLAS GUZMAN
TLAXCALA No. 112 - 18
COL. ROMA SUR
MEXICO 7, D. F.
TEL. 564-34-33

CENTRO LATINO AMERICANO DE TECNOLOGIA
EDUCACION PARA LA SALUD
ANALISTA JUNIOR-DEPTO. SERVS. GENERALES
PRESIDENTE CARRANZA No. 162
COL. COYOACAN
MEXICO 21, D. F.
TEL. 554-86-55

15. SR. MIGUEL ANGEL LOPEZ SANCHEZ
PLUTARCO ELIAS CALLES No. 1976
COL. PRADO ERMITA
MEXICO 13, D. F.
TEL. 539-68-29

COMISION DE ENERGENTICOS- SPFI.
ASESOR TECNICO
RIO RHIN No. 22 - 1º PISO
COL. CUAUHTEMOC
MEXICO 5, D. F.
TEL. 592-10-67

16. SR. JOSE T. LOPEZ YAÑEZ
AV. INDUSTRIA No. 28
COL. MOCTEZUMA
MEXICO 9, D. F.
TEL. 762-49-31 - 522-55-85

TELEFONOS DE MEXICO
AUDITOR INTERNO
PARQUE VIA No. 198
COL. CUAUHTEMOC
MEXICO 5, D. F.
TEL. 518-82-20 EXT. 5626

17. ING. JULIO CESAR MARGAIN COMPEAN

PETROLEOS MEXICANOS

18. SR. ENRIQUE F. MARTINEZ TOSCANO
EDIF. F-301
COL. UNIDAD HABITACIONAL SANTIAGO
2a. SECCION
MEXICO 13, D. F.

SECRETARIA DE PROGRAMACION Y PRESUPUESTO
JEFE DE UNIDAD "A" DE INSPECCION DE OBRAS
IZAZAGA No. 38 - 2 º PISO
CENTRO
MEXICO 1, D. F.
TEL. 521-52-27

19. SR. FRANCISCO J. MUÑGUIA Y NOCEDAL
REFORMA No. 48
COL. ATLANTIDA COY.
MEXICO 21, D. F.
TEL. 549-14-77

COLEGIO DE BACHILLERES
SUBDIRECTOR DE PROGRAMACION Y ESTADISTICA
AV. CUAUHTEMOC No. 1236 - 8 º PISO
COL. STA. CRUZ ATOYAC
MEXICO 13, D. F.
TEL. 559-55-22 EXT. 135

20. SR. JUAN ORTIZ ANGUIANO
MANCHURIA No. 11
COL. ROMERO RUBIO
MEXICO 9, D. F.

SISTEMAS ELECTRONICOS DE CONTROL-INDUSTRIAL
JEFE DEPTO. DE DISEÑO Y SERVICIO TECNICO
MANCHURIA No. 14
MEXICO 9, D. F.
TEL. 760-34-54

21. SR. ALFONSO QUIROZ CHAVOLLA
XOCHICALCO No. 51 - 103
COL. NARVARTE
MEXICO 12, D. F.
TEL. 546-63-93

S. A. R. H.
JEFE DEPARTAMENTO
REFORMA No. 35 - MEZANINE
MEXICO 1, D. F.
TEL. 530-05-49

22. SR. JOSE FERNANDO RIVERA RIOS
ISIDRO FABELA No. 85
COL. JACARANDAS
MEXICO 13, D. F.
TEL. 518-05-00 EXT. 538

BANCO DE MEXICO, S. A.
OPERADOR DE SISTEMAS
5 DE MAÑO No. 2
MEXICO 1, D. F.

23. SR. JOSE ARTURO ROJAS OLVERA
CEIBA No. 34
TLALNEPANTLA, EDO. DE MEXICO
TEL. 545-65-70

ORGANIZACION DANDO, S. C.
CONTRALOR DE FILIALES
LAGO ALBERTO No. 43-B

24. SR. JOSE ANTONIO SALMON T.
REAL DE LOS REYES 77 ALAMO - 2
COL. COYOACAN
MEXICO 21, D. F.
TEL. 544-21-56

COLEGIO DE BACHILLERES
ANALISTA
AV. CUAUHTEMOC No. 1236 - 8º PISO
COL. STA. CRUZ ATOYAC
MEXICO 13, D. F.
TEL. 559-55-22 EXT. 132

25. SR. JORGE SANCHEZ VAZQUEZ
COPILCO 76, B2-204
COL. SAN ANGEL
MEXICO 20, D. F.
TEL. 548-57-42

SEC. DE PROGRAMACION Y PRESUPUESTO
JEFE DE LA UNIDAD DE INDICES DE COSTOS
J.M. IZAZAGA No. 38 - 1º PISO
MEXICO 1, D. F.
TEL. 510-90-65

26. SR. PORFIRIO SILVA PEREZ
1º RET. DE RIFLEROS S/N L.P. 5a.
COL. HABIT. EJTO. DE O.
MEXICO 9, D. F.
TEL. 558-81-96

C.E.C., D.E.S.F.I., UNAM
JEFE DE DEPTO. DE SERVICIOS DE APOYO
TACUBA No. 5
MEXICO 1, D. F.
TEL. 512-31-23

27. SR. LUIS TORRES GARCIA
NORTE 11-A No. 24
COL. NUEVA VALLEJO
MEXICO 14, D. F.
TEL. 567-04-17

INSTITUTO DE INVESTIGACIONES ELECTRICAS
JEFE DE SERVICIOS DE INFORMACION A LA IND.
LEIBNITZ No. 14 - 3 º PISO
COL. ANZURES
MEXICO 5, D. F.
TEL. 514-66-36

28. SR. CANDELARIO TREJO FLORES
PISCO No. 568
COL. LINDAVISTA
MEXICO 14, D. F.

PETROLEOS MEXICANOS
COORDINADOR DE PROYECTOS
MARINA NACIONAL No. 329
COL. ANAHUAC
MEXICO 17, D. F.
TEL. 531-66-92 EXT. 3745 - 3590 - 545-74-60

29. SR. JOSE URIBE SANCHEZ
TIANQUISTENCO No. 217
COL.S.J.I. DE LA CRUZ
TOLUCA, MEX.
TEL. 599-44

UNIVERSIDAD AUTONOMA DEL EDO. DE MEXICO
DIRECTOR DE CICALI
CONSTITUYENTES No. 100
TOLUCA MEXICO
TEL. 7-77-77-

30. SR. RAUL VARELA GOMEZ
GAMMA No. 101
COL. ROMERO DE TERREROS
MEXICO 21, D. F.
TEL. 554-03-23

U.N.A.M.
ANALISTA DE SITEMAS
CIUDAD UNIVERSITARIA
MEXICO 20, D. F.

31. SR. EUSEBIO VELAZQUEZ HDEZ.
AZAHARES No. 26
FRACC. VILLA DE LAS FLORES
EDO. DE MEXICO
TEL. 91-591-4-01-91

S. A. R. H.
ANALISTA
REFORMA No. 35 - MEZANINE
MEXICO 1, D. F.
TEL. 546-59-28

32. SR. PEDRO VERJAN VARGAS
VIRGINIA No. 164
COL. NATIVITAS
MEXICO 13, D. F.
TEL. 532-27-32

C. A. V. MAL
INGENIERO
BALDERAS No. 55 - 2º PISO
MEXICO 1, D. F.
TEL. 510-02-94

33. ING. LUIS JIMENEZ ESCOBAR
AV. DEL ROSAL No. 290
COL. MOLINO DE ROSAS
MEXICO 19, D. F.

INSTITUTO DE INGENIERIA - UNAM
SECRETARIO TECNICO
CIUDAD UNIVERSITARIA
MEXICO 20, D. F.
TEL. 550-52-15 EXT. 3649

'gm.