UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

ARAGON

PERSPECTIVAS DE SOLUCIONES EN RED PARA LA DINAMICA ORGANIZACIONAL: MODELO CLIENTE/SERVIDOR

T E S I S

PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A

GABRIELA RAYA PEREGRINA

ASESOR:

ING. LAURA SANDOVAL MONTAÑO

SAN JUAN DE ARAGON, EDO. DE MEXICO 1999

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGON

PERSPECTIVAS DE SOLUCIONES EN RED PARA LA DINÁMICA ORGANIZACIONAL: MODELO CLIENTE/SERVIDOR

TESIS

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

PRESENTA

GABRIELA RAYA PEREGRINA

"DEDICATORIAS"

A MI MAMÁ, PAPÁ Y HERMANOS

ROSA MARÍA PEREGRINA CARDENAS

JOSE ALEJANDRO RAYA GAMIÑO

DIANA ANGELICA RAYA PEREGRINA

JOSE ALEJANDRO RAYA PEREGRINA

LAURA AURORA RAYA PEREGRINA

TANIA ADRIANA RAYA PEREGRINA

Y A MIS DOS SOBRINOS

ARIADNA BERENICE TREJO RAYA

ALEJANDRO TREJO RAYA

PORQUE MI FAMILIA ES LO MEJOR QUE TENGO EN LA VIDA.

GRACIAS

A MI ASESORA DE TESIS

LAURA SANDOVAL MONTAÑO

POR SER UNA EXCELENTE PERSONA, QUE ME TUVO TANTA PACIENCIA, ADEMÁS DE BRINDARME SU APOYO, SU TIEMPO Y SUS CONOCIMIENTOS.

GRACIAS

MI AGRADECIMETO A LA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

A MI ESCUELA

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

Y A TODOS MIS

PROFESORES

GRACIAS

INDICE

PERSPECTIVAS DE SOLUCIONES EN RED PARA LA DINÁMICA ORGANIZACIONAL: MODELO CLIENTE/SERVIDOR

INTRODUCCION 1
CAPÍTULO I ANTECEDENTES 2
I.1 Surgimiento del modelo Cliente/Servidor 4
I.1.1 Procesamiento Anfitrión 4
I.1.2 Procesamiento Maestro-Esclavo 4
I.1.3 Procesamiento Cliente/Servidor 5
I.1.3.1 Definición 5
I.1.3.2 Procesamiento 6
I.2 Conceptos básicos 13
I.2.1 Redes de área local y redes de área amplia 14
I.2.2 Topologías de red 15
I.2.3 Medios físicos de transmisión 19
I.2.4 Métodos de codificación, transmisión y control de acceso 20
I.2.4.1 Codificación 20
I.2.4.2 Transmisión 21
I.2.4.3 Control de acceso 22
I.2.5 Capas 24
I.2.6 Protocolos 31

CAPÍTULO II MODELO CLIENTE/SERVIDOR. 37
II.1 Bases de Datos Distribuidas 37
II.1.1 Transparencia y sincronización 45
II.1.2 Requerimientos de soporte 46
II.1.2.1 Restauración 48
II.1.2.2 Optimización 49
II.1.2.3 Diccionario/Directorio de Datos 50
II.1.2.3.1 DD/D para la optimización de consultas 53
II.1.2.3.2 Fragmentación 54
II.1.2.3.3 Replicación 56
II.2 Procesamiento cooperativo 58
II.2.1 Aspectos del procesamiento cooperativo 59
II.2.2 Downsizing en el procesamiento cooperativo 62
II.2.3 Estructura del procesamiento cooperativo 63
II.2.3.1 Presentación lógica 68
II.2.3.2 Procesamiento lógico de negocios 70
II.2.3.3 Administración lógica de datos 71
II.2.4 Técnicas del procesamiento cooperativo 74
II.2.5 Distribución de los elementos del sistema 76

INDICE

II.3 Sistemas Cliente/Servidor	79
II.3.1 Plataformas Cliente/Servidor	82
II.3.1.1 Computadoras Personales	83
II.3.1.2 RISC y otras Workstations UNIX	84
II.3.1.3 Minicomputadoras	84
II.3.1.4 Mainframes	85
II.3.2 Bases de Datos Cliente/Servidor	88
II.3.2.1 Modelos de DBMSs	88
II.3.2.2 Arquitectura de los sistemas DBMS	90
II.3.2.3 Lenguajes de programación para aplicaciones de bases de datos	92
II.3.2.4 Sistemas Administradores de Bases de Datos (DBMSs)	92
II.3.3 Aplicaciones de usuario (front-ends)	95
CAPÍTULO III METODOLOGÍA EN EL DESARROLLO DE SISTEMAS BAJO ARQUITECTURA CLIENTE/SERVIDOR	103
III.1 Panorama General de las Metodologías	103
III.1.1 Metodología de William H. Inmon	103
III.1.2 Metodología de Larry T. Vaughn	114
III.1.3 Metodología de David Vaskevitch	120
III.1.4 Metodología de Paul E. Renaud	128
III.1.5 Metodología de Paul Kavanagh	133
III.2 Selección de la metodología	141
CAPÍTULO IV ANÁLISIS Y DISEÑO DEL " SISTEMA DE CONTROL DE GESTIÓN"	145
Reingeniería de Procesos	
Desarrollo de Sistemas	
Arquitectura	
CAPITULO V CONCLUSIONES	217
BIBLIOGRAFIA	226

INTRODUCCIÓN

El objetivo de este trabajo es identificar y puntualizar los elementos y conceptos clave para que **"de acuerdo a la complejidad y necesidades de una organización, determinar la conveniencia de migrar o, en caso de que no exista algún sistema, implantar un sistema cliente/servidor. Y así cubrir los requerimientos de software y hardware para el control y manipulación adecuado de sus datos".**

Este trabajo se ha dividido en cinco capítulos, en el primer capítulo **ANTECEDENTES** se hace una revisión de conceptos que son necesarios para introducirse en lo que es el modelo Cliente/Servidor. Abordando el tema de cómo ha surgido el concepto de cliente/servidor, así como conceptos básicos de redes y telecomunicaciones.

En el segundo capítulo **MODELO CLIENTE/SERVIDOR** se expone de forma detallada los diferentes aspectos que integran este modelo, abordando temas como: Bases de Datos Distribuidas, Procesamiento Cooperativo, plataformas, Sistemas Administradores de Bases de datos y Aplicaciones.

En el tercer capítulo **METODOLOGÍA EN EL DESARROLLO DE SISTEMAS BAJO ARQUITECTURA CLIENTE/SERVIDOR** se analizan las metodologías desarrolladas por algunos expertos (William H. Inmon, Larry T. Vaughn, David Vaskevitch, Paul E. Renaud y Paul Kavanagh) en el análisis y diseño de sistemas cliente/servidor, y con base en esto seleccionar la metodología a utilizar para nuestro caso de estudio.

En el cuarto capítulo **ANÁLISIS Y DISEÑO DEL "SISTEMA DE CONTROL DE GESTION"**, se desarrolla la solución para nuestro caso de estudio, aplicando la metodología seleccionada. Enfatizando lo fundamental que resulta la aplicación de una metodología en el desarrollo de cualquier tipo de sistema. De esta forma se propone el *modelo de datos* que permitirá cubrir los requerimientos de datos de la organización. Y de acuerdo a la plataforma disponible y el nivel de utilización de la información, por las diferentes áreas, se presenta un esquema de distribución de la base de datos.

El quinto capítulo **CONCLUSIONES**, recapitula los aspectos fundamentales del desarrollo de sistemas cliente/servidor y se evalúan los resultados obtenidos del diseño del **SISTEMA DE CONTROL DE GESTION**.

CAPÍTULO I

Antecedentes

I. ANTECEDENTES

Para comprender lo que es el modelo cliente/servidor es necesario analizar la evolución de los sistemas de cómputo. En dicha evolución podemos identificar los diferentes tipos de procesamiento, que son: Procesamiento Anfitrión, Procesamiento Maestro-Esclavo y Procesamiento Cliente/Servidor. Los cuales han sido asociados con un tipo de tecnología específica. Que a lo largo del tiempo han ido cubriendo las necesidades organizacionales, y como consecuencia de la evolución de estas organizaciones, se ha requerido buscar nuevas tecnologías y arquitecturas que den solución a las necesidades de crecimiento e integración.

El mayor atractivo del ambiente cliente/servidor es que el costo de adquisición y operación es absorbido por cada departamento que lo utilice. Y su utilización se presenta como una necesidad, cuya implantación proporciona el control del destino de la organización, esto es que se cubren las necesidades de trabajar cerradamente y cooperativamente con otros departamentos, que igualmente tienen control de sus propios procesos.

En los últimos tiempos ha tenido gran popularidad el modelo de sistemas Cliente/Servidor (C/S), teniendo como base de dicho éxito sus características más sobresalientes como son: gran reducción de costos, mayor accesibilidad a los datos, incremento significativo de la productividad, etc.

Pero se ha visto en la práctica que cuando las empresas se deciden a migrar a C/S se han presentado una serie de problemas que en un momento dado hacen dudar si en realidad C/S ha de proporcionar las ventajas que tanto se mencionan. Se han presentado problemas tales como: altos costos, problemas de implantación, el nivel de seguridad y otra serie de problemas. Al enfrentar la teoría con las implantaciones reales, se pone en tela de juicio si en realidad el modelo C/S tiene grandes ventajas o se ha exagerado por ser algo novedoso.

Pero antes de hacer un juicio es necesario enfatizar que se deben tomar en cuenta dos razones fundamentales cuando se toma la decisión de migrar o implantar un sistema basado en el modelo Cliente/Servidor: 1) cuando se necesita dar poder a la gente, para lo cual es necesario proporcionar sistemas que sean fáciles de usar y requieran de poco entrenamiento y soporte. Así como la gran necesidad de acceso a datos corporativos para la toma de decisiones de una manera más rápida; y 2) dar poder a la organización.

Por ejemplo si cada departamento decide por separado el camino a seguir para el proceso y manipulación de datos, los cuales son los mismos que utilizan otros departamentos, el resultado es un caos institucional. Por lo que debe haber un grado de consistencia y uniformidad de los mismos datos que se manejan en diferentes departamentos de la organización. El costo del procesamiento cliente/servidor facilita a la organización los aspectos antes mencionados, pues al formar parte de una comunidad que comparte la misma información dejan de ser preocupantes los aspectos de consistencia y uniformidad. Pero un caso muy peligroso, es cuando el resultado de la aceptación de un sistema cliente/servidor es la construcción de muchas islas de procesamiento. Porque consecuentemente, la falta de

integración da muchos resultados negativos, tomando en cuenta que no se toman todas las ventajas que proporciona este sistema.

Ahora, en cuanto a lo que es realidad y ficción a las características del modelo Cliente/Servidor tenemos que:

- Es falso que C/S tenga una pronunciada curva de aprendizaje y requiera de mucho entrenamiento. Se menciona que se necesitan aproximadamente seis meses para tener la habilidad y estar familiarizado con las herramientas. Y las estadísticas rebelan que los desarrolladores son un 33% más productivos. Los desarrolladores deben enfocarse a manejar con destreza el desarrollo rápido de aplicaciones y el diseño de Interfases Gráficas de Usuario (GUI pro sus siglas en inglés- Graphic User Interfase).
- Es falso que C/S sea una tecnología inestable e inmadura. Existen herramientas que son bastante confiables, pero como todo hay herramientas que son deficientes, lo cual depende mucho de la empresa que proporcione las herramientas.
- Es falso que los sistemas C/S no proporcionen la eficiencia que pueden proporcionar los sistemas basados en el procesamiento anfitrión. Lo que puede dar esta apariencia es cuando el sistema tiene un excesivo acceso a los servidores, o falta de una adecuada interfase gráfica de usuario. Pero en comparación con un procesamiento anfitrión, en los sistemas C/S se tiene más control de la eficiencia debido a los servidores dedicados.
- Es falso que los costos de mantenimiento sean elevados. Se debe tomar en cuenta que cuando un sistema no cubre los requerimientos del usuario, el 80% del mantenimiento se invierte en todas las adaptaciones necesarias. En cambio, si se diseña y desarrolla de forma adecuada no tiene por qué haber una elevación excesiva de los costos de mantenimiento.

En primer lugar, tomando en cuenta los puntos anteriores, cuando se diseña un sistemas Cliente/Servidor es muy importante tener una especial atención en los requerimientos de los usuarios. Y en segundo lugar, los sistemas C/S tienen grandes posibilidades de rehusar los recursos, lo cual tiene una gran repercusión en los costos de mantenimiento. Que es muy significativo a lo largo de los años. Y algo primordial que se debe tener muy presente cuando se quiere desarrollar una aplicación, es que el empleo adecuado y eficiente de buenas técnicas de diseño librarán casi cualquier deficiencia de las herramientas y de la arquitectura en que se esté desarrollando.

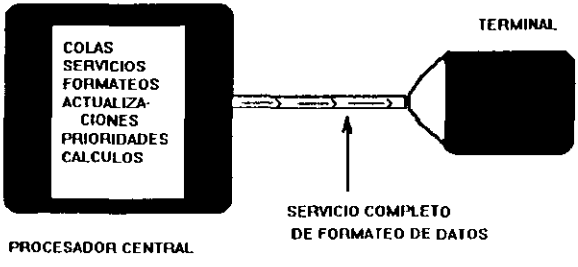


Figura I.1 La clásica arquitectura centralizada

I.1 Surgimiento del modelo Cliente/Servidor

I.1.1 Ambiente de procesamiento Anfitrión

En el ambiente de Procesamiento Anfitrión, conocido como sistemas multiusuario. Este tipo de arquitectura centralizada no tiene la capacidad de procesar algún tipo de aplicación en forma distribuida. Ya que todo el procesamiento se lleva a cabo en una unidad central, y se tiene acceso a la información mediante terminales tontas, las cuales realizan un mínimo de trabajo. Pues las terminales actúan como dispositivos de Entrada-Salida y sin ninguna capacidad de cómputo asociada, dejando al anfitrión (Unidad Central) toda la carga de proceso y control. Dicha terminal recibe de forma completa la presentación y los datos totalmente con formato. En la Figura I.1 podemos apreciar de forma gráfica y sencilla el procesamiento que se da en este tipo de arquitectura.

Con esta arquitectura se dio solución a las necesidades organizacionales, y sigue siendo la solución a cierto tipo de necesidades, teniendo implícitas sus ventajas y desventajas. Algunas de las ventajas de los sistemas multiusuarios son:

- Unidades de disco de gran capacidad y alto nivel de desempeño, hacen posible el mantenimiento y manipulación de archivos muy grandes.
- Alta velocidad de procesamiento.
- Uso de terminales tontas, relativamente económicas, para tener acceso al sistema.
- Eficiente manejo de los procedimientos operacionales , tales como respaldo de datos.

Sus desventajas más notables son:

- Si el sistema se paraliza no se puede realizar ninguna operación.
- Altos costos de mantenimiento.
- Limitaciones en el número de usuarios.

I.1.2 Ambiente de procesamiento Maestro-Esclavo

El procesamiento Maestro-Esclavo tiene un alto nivel de procesamiento distribuido de aplicaciones. Como el propio nombre lo indica, en este tipo de sistemas la computadora esclava se enlaza a la computadora maestra y realiza sus funciones de procesamiento de aplicaciones de forma correcta sólo cuando es dirigida por la computadora maestra. En este ambiente el procesamiento de aplicaciones se da en un proceso distribuido aunque tiende a ser en forma unidireccional de la computadora maestra hacia las esclavas. Típicamente, la computadora esclava tiene la capacidad limitada de procesamiento local de aplicaciones, tal como validación de campos en pantalla, edición o procesamiento de teclas de función. (Un ejemplo del ambiente de procesamiento maestro-esclavo es una computadora mainframe Anfitrión, tal como una IBM 30XX, usadas como controladores de grupo y terminales inteligentes). Pues se descarga el trabajo de Entrada-Salida a los equipos esclavos, pudiendo dedicar entonces el equipo maestro al procesamiento propiamente dicho.

Ejemplos más comunes de este tipo de procesamiento se da en Redes de Area Local (Local Area Network- LAN por sus siglas en ingles). Todas las computadoras de la red están enlazadas a un sistema de dispositivos para poder compartir los recursos. Los archivos de un disco duro y las impresoras son ejemplos comunes de recursos necesarios de compartir. En la terminología LAN a los dispositivos encargados de controlar la compartición de recursos se les llama SERVIDORES, tales como los servidores de archivos y servidores de impresión. Este término es adecuado ya que cuando es requerida la utilización de un recurso, dicha petición es atendida por el servidor correspondiente.

El procesamiento de los datos se da en cada una de las Computadoras Personales (Personal Computer- por sus siglas en inglés PC) y sólo ciertas funciones son distribuidas. Un ejemplo es cuando un archivo se manda a una PC porque hubo una solicitud de lectura. O cuando se hace la solicitud de modificación, el archivo es bloqueado en el servidor para que puedan hacerce las modificaciones sin que otro usuario trate de modificar la misma información. Este tipo de procesamiento es el más ampliamente utilizado e implantado. Se puede representar en forma sencilla la arquitectura Maestro-Esclavo como se muestra en la Figura I.2.

Compartición de recursos, tales como servidores de archivos y de impresión. Con cierto grado de procesamiento distribuido.

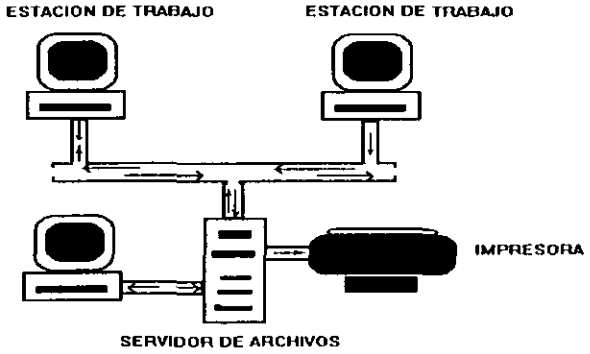


Figura I.2 Arquitectura Maestro-Esclavo ejemplificada en una LAN.

I.1.3 Ambiente de procesamiento Cliente/Servidor

I.1.3.1 Definición

En principio se tiene que definir el concepto central "ARQUITECTURA CLIENTE/SERVIDOR", y para tal objetivo hay que comprender inicialmente qué significa ARQUITECTURA en este contexto. Arquitectura se define como "el conjunto de definiciones, reglas y términos, que se emplean para construir un producto".

Por lo que la arquitectura cliente/servidor la podemos definir como un PROCESAMIENTO COOPERATIVO a una sola aplicación que se divide en varias tareas comprendiendo como tareas el hecho de que se va a ejecutar dicha aplicación en dos o más plataformas funcionando

de forma separada e independiente. El concepto de "plataforma" se refiere a las características del hardware y software del sistema operativo, en la cual se ejecuta alguna aplicación.

Una definición más completa define a la arquitectura cliente/servidor como "un tipo de cómputo DISTRIBUIDO y cómputo COOPERATIVO". Se agrega a dicha definición un AMBIENTE DE CÓMPUTO DISTRIBUIDO (DCE por sus signos en inglés -Distributed Computing Environment). El DCE comprende varios aspectos importantes que son los siguientes: Sistema de Archivos Distribuidos, Servicios de Directorio, Llamados de Procedimientos Remotos, Servicios de Enlace y Servicios de Tiempo. Conceptos que serán expuestos oportunamente.

I.1.3.2. Procesamiento

El modelo de procesamiento Cliente/Servidor es una extensión natural del procesamiento basado en compartición de dispositivos (Procesamiento Maestro-Esclavo). Las Redes de Area Local se han desarrollado en gran medida y se ha incrementado el número de estaciones de trabajo que soporta, la evolución de sistemas de compartición de dispositivos, también se han desarrollado en capacidad y poder.

En este modelo, el procesamiento de la aplicación es dividida (no siempre es necesario) entre el cliente y el servidor. El procesamiento es iniciado y particularmente controlado por los requerimientos de servicios del cliente pero no como en la forma de procesamiento maestro-esclavo. Tanto el cliente como el servidor cooperan para ejecutar exitosamente una aplicación. Servidores de Bases de Datos tales como Sybase, Oracle, SQL Server de Microsoft, son ejemplos de ambientes de procesamiento cliente/servidor.

Básicamente el procesamiento cliente/servidor requiere los siguientes elementos:

- Una robusta y eficiente comunicación entre los clientes y los servidores.
- La interacción cooperativa entre el cliente y el servidor es iniciada por el cliente.
- Procesamiento distribuido entre el cliente y el servidor.
- Control forzoso del servidor sobre los servicios o datos que puede requerir el cliente.

De forma simple, la arquitectura C/S puede ser representada como se muestra en la figura I.3, donde dos computadoras personales (Pc's) o estaciones de trabajo (Workstations), llamadas nodos, forman parte de una red más grande. Pero lo relevante de esto es que las dos PC's guardan una relación especial. La PC-B sostiene y administra los datos que pueden ser accedidos por la PC-A a través del control de la PC-B. La PC-A es asignada para ser el "cliente" y la PC-B para ser el "servidor". En otras palabras la PC-B atiende o sirve las necesidades de la PC-A. Esta simple configuración crea el aspecto de las aplicaciones que son construidas en la arquitectura C/S. Pero desde un punto de vista técnico, hay muchas variaciones de la simple arquitectura mostrada en la figura I.3.

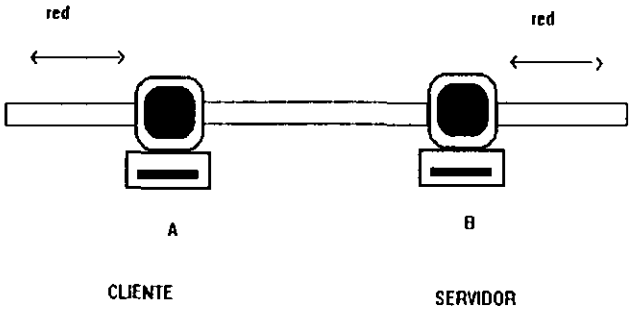


Figura I.3 Una simple perspectiva del ambiente cliente/servidor.

De la misma forma la red puede estar conectada a otras redes, como se puede ejemplificar en la figura I.4.

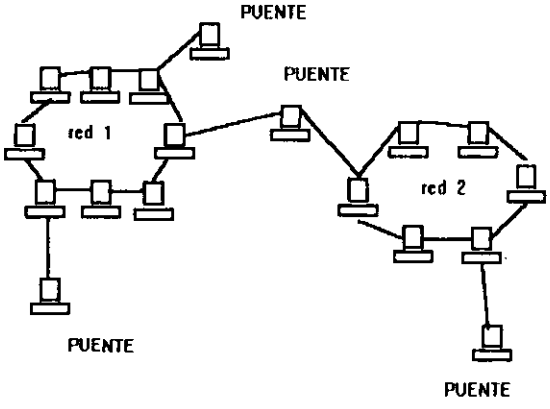


Figura I.4 Enlace entre diferentes redes.

Por lo anterior se puede ver que hay grandes posibilidades de enlace entre nodos y redes en el ambiente C/S. Pero el mejor aspecto de la arquitectura C/S no se centra en la configuración física de los nodos y las redes. El mejor aspecto de la arquitectura C/S se centra alrededor del procesamiento que se da en dicha arquitectura, el almacenamiento de datos y el flujo a través de la red. Poner atención en la tecnología que es utilizada en el ambiente C/S es interesante, porque ciertamente las necesidades de diseño/desarrollo están íntimamente relacionadas con la tecnología en la que se ejecuta este ambiente. Pero el propio enfoque de desarrollo puede no ser la tecnología, sino la aplicación a ser construida y las grandes estructuras de datos y procesos.

Un simple arreglo de importantes componentes de software se encuentra en cada nodo de la red, los cuales se muestran en la figura I.5. Dicha figura muestra que cada nodo tiene su Sistema Operativo, Programas de Aplicación, Datos Internos y Datos Externos. Los datos externos pueden ser una amplia variedad de medios, tales como cartuchos, unidades de disco, etc. Cada nodo es conectado con otros nodos por medio de la red. La comunicación entre nodos es lograda por medio de capas de protocolos para paquetes de datos. Algunos nodos tienen componentes de software más poderosos de uno o más de estos aspectos, en comparación con otros nodos. Pero todos los nodos comparten estas características.

Una de las cuestiones interesantes es cómo se hace que la arquitectura C/S sea diferente de la clásica arquitectura centralizada mainframe. Ante todo, la arquitectura centralizada mainframe existió mucho tiempo antes de que se hiciera el procesamiento C/S.

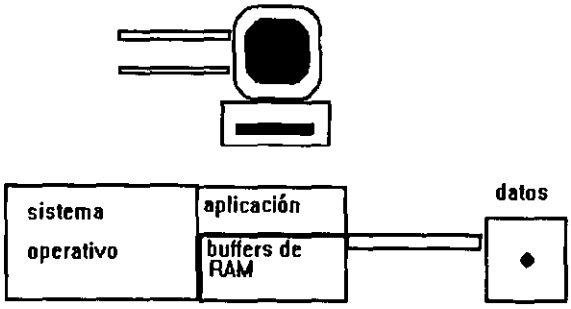


Figura I.5. Los componentes de cada nodo en la red.

Como se muestra en la figura I.6, a diferencia del ambiente mainframe, los datos que llegan al nodo en la arquitectura C/S, pueden ser recalculados, reformateados, recanalizados por una variedad de medios, y combinados con otros datos, sin que tengan que regresar necesariamente al servidor.

Algunas veces el ambiente C/S es llamado ambiente "inteligente" porque después de entregar los datos al nodo cliente, pueden ocurrir muchos procesos. En el ambiente mainframe, si algún proceso puede ocurrir en la terminal de datos, éste será muy pequeño. Por esta razón el ambiente mainframe es llamado ambiente "tonto".

En el ambiente Mainframe se tiene un mayor procesamiento, ya que el control se pasa completamente al procesador central.

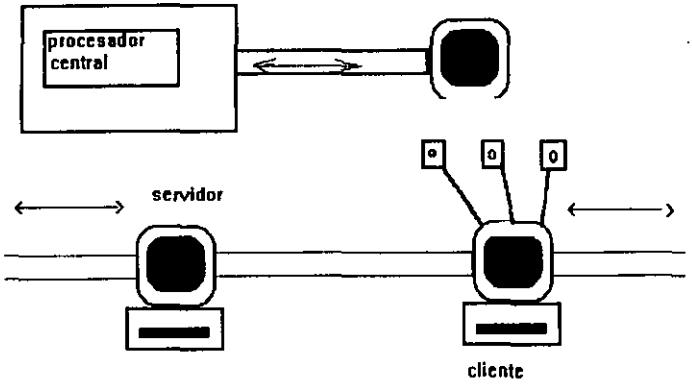


Figura 1.6 Una vez que llegan los datos al cliente, éstos pueden ser procesados muchas veces sin que tenga que regresar el control al servidor.

Costos

Un segundo factor en la arquitectura C/S son los costos. La escala de costos en el procesamiento C/S es baja en comparación con los costos asociados con grandes procesadores mainframe. Las diferencias de costo pueden ser los siguientes:

- Costos del procesador
- Costos de la red
- Costos de desarrollo
- Costos de operación
- Costos de software de sistemas
- Costos de mantenimiento.

Sólo el último de los costos -mantenimiento- normalmente permanece constante en el ambiente C/S.

Uso de la aplicación

El primer factor en esta arquitectura es saber qué camino se tomará para la construcción de la aplicación y cómo va a ser usada. El uso del procesamiento C/S, puede ser clasificado tanto como uso operacional o uso del Sistema de Soporte de Decisiones (Decision Support Systems-DSS por sus siglas en inglés). La diferencia en uso es tal vez no sólo la más profunda, sino el factor más importante en el desarrollo de aplicaciones, pero también el más mal entendido, ya que la mayoría de estos sistemas no cumplen con este factor.

Procesamiento operacional
El procesamiento operacional es un proceso que afecta día a día, detalladamente, todas las decisiones de negocios que se realizan.

Algunas de las características sobresalientes del procesamiento operacional son las siguientes:

- El procesamiento operacional opera con valores de datos actuales - datos que se generan precisamente en el momento de usarlos.
- El desarrollo operacional esta clásicamente hecho para aplicaciones de tiempo.
- El procesamiento operacional es un manejador de requerimientos. Esto significa que la atención de requerimientos tiene una alta prioridad y la aplicación resultante se corre repetitivamente.
- El procesamiento operacional manipula datos modificando o accediendo basado en un detallado manejo de registro por registro.
- El procesamiento operacional tiene acceso a datos archivados que tiene una alta probabilidad de acceso y un bajo volumen relativamente.
- La cuestión de posesión de datos -quién puede modificar datos- es rígidamente para el ambiente operacional.
- El procesamiento operacional es para el beneficio de una comunidad de oficina permitiendo segundas decisiones.

Procesamiento DSS

El procesamiento DSS es diametralmente diferente del procesamiento operacional. El procesamiento DSS es usado para soportar una comunidad administrativa, y no una comunidad de oficina, DSS tiene las siguientes características:

- El procesamiento DSS opera en datos archivados. Datos archivados una vez propiamente registrados, no es factible de ser modificado. Los datos archivados son fundamentalmente diferentes de los valores que se van generando, porque en el transcurso de este proceso pueden ser modificados.
- El procesamiento DSS esta hecho para integración de datos, esto es opuesto para los datos que se obtienen en una aplicación de tiempo.
- El procesamiento DSS es esencialmente un manejador de datos en comparación con el procesamiento operacional que es un manejador de requerimientos.

- El procesamiento DSS opera básicamente con conjuntos de datos, y no basado en datos de forma detallada.
- El procesamiento DSS tiene acceso a datos archivados que tienen de una posibilidad de acceso que va de medio-baja a baja.
- No puede haber una modificación de datos desde el procesamiento DSS.
- El procesamiento DSS sirve a la comunidad administrativa haciendo largas sesiones de decisiones.

Hay diferencias significativas entre el procesamiento operacional y DSS, comprender las diferencias entre estos tipos de procesamiento son esenciales para un exitoso ambiente cliente/servidor. Cuando el desarrollador no comprende la diferencia (o aún peor cuando el desarrollador, no conoce del tema en cuestión), el resultado será un ambiente C/S que estará muy escaso de armonía.

Tal vez lo más obvio, la más profunda diferencia entre el procesamiento operacional y el DSS en el ambiente C/S está localizado en cómo son usados los recursos de Hardware. El procesamiento operacional y DSS reflejan un amplio patrón de diferencias en la utilización del Hardware. La figura I.7 muestra los patrones de diferencias en la utilización de Hardware.

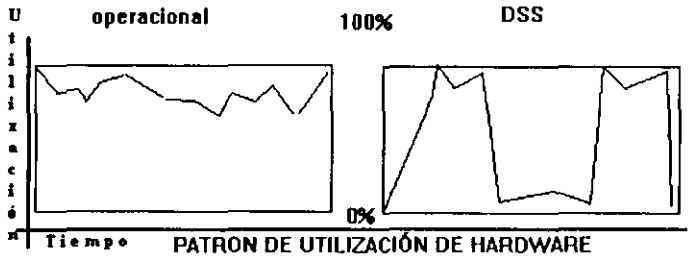


Figura I.7 Patrón de diferencias en la utilización de hardware en el procesamiento Operacional y DSS.

La figura I.7 muestra que el hardware en C/S es usado estáticamente -con menos picos y valles- en el ambiente operacional. Y al fin del día, manejando los status de forma operacional se puede calcular el promedio de utilización de hardware en C/S y tener un número significativo.

El patrón de utilización de hardware para el procesamiento DSS es fundamentalmente diferente. El patrón del procesamiento DSS es un patrón binario. En el procesamiento DSS C/S, el hardware llega a ser intensamente usado o el hardware no se usa en su totalidad. Calculando con un significado aritmético la utilización del hardware C/S en el ambiente DSS da como resultado un número sin significado alguno.

La diferencia en la utilización del hardware en el ambiente operacional y DSS, es insignificante cuando se tiene un nivel bajo de utilización (10%). Pero conforme aumenta la utilización de recursos, las diferencias se van marcando de manera pronunciada.

Diferencias entre procesamiento cliente/servidor DSS y cliente/servidor Operacional

Las diferencias entre procesamiento operacional y DSS tienen profundos efectos sobre el camino inequívoco para construir aplicaciones en el ambiente C/S, ya que puede haber una separación -lógica y física- del procesamiento operacional y DSS. Uniendo los dos tipos de procesamiento, aún inadvertidamente o conscientemente, cuando se tiene una cantidad considerable de procesamiento se tienen desventajas en cuanto a:

- Mantenimiento de embotellamientos
- Rendimiento de embotellamientos
- Aplicaciones complejas

Un buen camino para separar el procesamiento operacional y DSS en el ambiente C/S es aislar el procesamiento operacional en una red (ring) y el procesamiento DSS en otra red (ring).

Los grandes factores que hacen que el ambiente C/S sea atractivo en primer término es la negativa a ignorar el diseño en el cual libremente se une procesamiento operacional y DSS.

Autonomía contra integración

Otro factor importante que forma parte de la arquitectura cliente/servidor es el procesamiento autónomo contra integración (procesamiento cooperativo). De lo mejor a hacer referencia en el ambiente cliente/servidor, es que el procesamiento autónomo que se brinda da independencia a cada nodo, obteniendo como resultado: independencia de procesamiento, independencia de datos, independencia de utilización, etc. Y tenemos como resultado que sobre cada dato que llega al nodo el usuario tiene el control, o en el caso de que los datos pertenezcan a archivos privados residentes en el nodo. Bajo todas esta condiciones necesarias, se tiene un alto grado de libertad en cada nodo. Con lo que hay una imperante necesidad de consistencia y uniformidad de datos y procesos, cuando los datos y/o procesos son corporativos y tienen que residir en el nodo pero que también son necesarios en otro punto de la red.

Una matriz

Una matriz puede ser construida en orden, donde se puedan clasificar los diferentes tipos de aplicaciones cliente/servidor que hay, como se muestra en la figura 1.8.

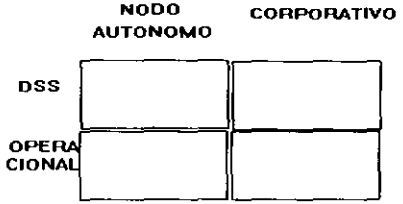


Figura 1.8 La matriz básica cliente/Servidor en la cual todas las aplicaciones pueden ser clasificadas. Haciendo énfasis en que las características de una aplicación difieren ampliamente de las características de otras aplicaciones en otro cuadrante de la matriz.

Un nodo de aplicación autónoma en el ambiente cliente/servidor es aquel que está contenido totalmente en sí mismo. En otras palabras, un nodo de aplicación autónoma no almacena datos o se comunica con otros nodos en la red.

Una aplicación cliente/servidor puede ser construida como DSS/Nodo Autónomo, Operacional/Nodo Autónomo, DSS/Cooperativo y Operacional/Cooperativo.

Estructura de datos
La estructuración de datos es siempre muy importante en cualquier ambiente de procesamiento. Y esto es aún más importante en el ambiente cliente/servidor. La primera gran división de la estructura de datos que es importante se da dentro del mismo nodo - qué datos son públicos (o corporativos), y qué datos son privados para ser claramente delineados. Unir los dos tipos de datos dentro del mismo nodo no es una buena idea.

La segunda división de datos consiste en determinar si un nodo de la red es operacional o DSS. Para que no halla confusión en la forma en que los datos son utilizados.

La tercera estructuración de datos en el ambiente cliente/servidor es la organización de los archivos, los datos DSS son definidos como "depósitos de datos(warehouse)", los depósitos de datos en el ambiente cliente/servidor se definen como:

- Organizados a lo largo de la corporación.
- Integrados a todas las aplicaciones cliente/servidor.
- No volátiles, apareciendo como una colección de imágenes de datos.
- Variantes en tiempo, cuando cada imagen de datos tiene registrado el momento de ejecución.

Los depósitos de datos en el ambiente cliente/servidor pueden ser implementados por una variedad de caminos - sobre varios servidores, o un solo servidor, o aún residir en una red separada o computadora mainframe.

I.2. Conceptos Básicos

Cuando se diseñan redes de área local, la interdependencia entre factores a ser tomados en consideración son: medios de transmisión, métodos de transmisión y control de acceso, topología de la red, capacidad de canal, y los niveles de datos. Todos estos factores afectan la eficiencia y el costo de la red. La toma de decisión con respecto a la topología de la red, métodos de transmisión y métodos de control de acceso se pueden hacer basándose en los requerimientos de procesamiento y costo de una LAN en particular.

I.2.1 Redes de Área Local y Redes de Área Amplia

Primeramente los objetivos que se persiguen con la implantación de una red son los siguientes:

- Proporcionar productividad para la automatización del trabajo en funciones rutinarias.
- Proporcionar manejabilidad de la información y así reducir la duplicación y proveer la accesibilidad.
- Proveer la interacción en la compartición de la información.
- Reducir o controlar los costos en el uso costo-efectivo en los métodos de comunicación.
- Procesamiento cooperativo con un alto nivel de enlace entre computadoras personales y mainframes.
- Proveer la estandarización del uso de computadoras y comunicaciones.

Y de acuerdo a la magnitud de las necesidades se tienen varios tipos de redes: Red de Área Amplia (Wire Area Network-WAN), Red de Área Local (Local Area Network-LAN) y Red Metropolitana (Metropolitan Area Network-MAN).

WAN.- Las redes de área amplia comprenden las redes que se interconectan a grandes distancias geográficas. La comunicación en este tipo de red se realiza vía microondas. La velocidad de transmisión puede sobrepazar los 100 Mbps.

MAN.- Las redes metropolitanas generalmente se utilizan para centralizar información de forma regional. Las redes metropolitanas caen en el rango de 50 Km. La estructura de una red de gran cobertura tiende a ser más irregular, debido a la necesidad de emplear en las líneas computadoras centrales, concentradores y terminales multiplexadas y/o multipunto. Y en el caso de que las líneas de comunicación sean privadas, las organizaciones procuran tener el máximo de utilización, y para conseguirlo se van conectando los diferentes equipos a la línea independientemente del lugar en que se encuentren. Como consecuencia la topología de las redes de área amplia suele ser muy irregular.

LAN.- Las redes de área local, como el mismo nombre los especifica solamente trabajan en áreas que caen en un rango de 1 a 10 Km. Para la implantación de redes se utiliza cable UTP de dos pares con los cuales la velocidad va de 4 a 10 MBps, o también cable UTP 4 pares o fibra óptica con la cual se alcanzan los 100 MBps. El usuario de una red local no tiene que preocuparse tanto de la utilización máxima de los canales, ya que el coste de los mismos es pequeño en comparación con su capacidad de transmisión de bits. Por tanto, la necesidad de esquemas eficaces de multiplexado y distribución no es tan crítico en un entorno local como lo es en una red de área amplia. Por otro lado, como las redes suelen residir en un mismo edificio, la topología tiende a ser mucho más ordenada y estructurada.

Para abordar el problema de la limitación del ancho de banda, han aparecido diversas técnicas:

* Arquitectura de Conmutación Digital de Circuitos RDSI-BE (Red Digital de Servicios Integrados - Bnda Estrecha).

* Arquitecturas de Compartición del Medio
  - ⇒ Adaptación de las arquitecturas existentes:
    - Fast Ethernet o Ethernet a 100 Mbps
    - AnyLAN
    - Ethernet Isócrona
  - ⇒ Nuevas arquitecturas
    - DQDB (Distributed Queue Double Bus)
    - FDDI (Fiber Distributed Data Interface)

* Arquitecturas de Conmutación Rápida de Paquetes
  - Conmutación de Tramas
  - Retransmisión de Tramas (Frame Relay)

* Arquitectura de retransmisión de células (Cell Rellay) o tecnología ATM, Modo de Transferencia Asíncrono, utilizada en la RDSI-BA, Red Digital de Servicios Integrados - Banda Ancha.

I.2.2 Topologías de Red

Un sistema de comunicación es la colección de hardware y software que soporta comunicación entre sistemas y procesos a lo largo de componentes de software en nodos distribuidos. El actual enlace entre nodos comprende una red. Lo cual representa una colección ordenada de capas físicas con nodos interconectados.

Ahora, la configuración de una red suele conocerse como topología de la misma. La topología es la forma (la conexión física) de la red. El término topología es un concepto geométrico con el que se alude al aspecto de una cosa. A la hora de establecer la topología de una red, el diseñador ha de plantearse tres objetivos principales:

- Proporcionar la máxima fiabilidad posible, para garantizar la recepción correcta de todo el tráfico (encaminamiento alternativo).
- Encaminar el tráfico entre la terminal transmisora y receptora a través del camino más económico dentro de la red (aunque si se consideran otros factores más importantes, como la fiabilidad, este camino de coste mínimo puede no ser el más conveniente).
- Proporcionar al usuario final un tiempo de respuesta óptimo y un caudal eficaz máximo.

El segundo objetivo a cumplir a la hora de establecer una topología para la red consiste en proporcionar a los procesos de aplicación que residen en las terminales el camino más económico posible. Para ello es preciso:

- Minimizar la longitud real del canal que une los componentes, lo cual suele implicar el encaminamiento del tráfico a través del menor número posible de componentes intermedios.
- Proporcionar el canal más económico para cada actividad concreta; por ejemplo, transmitir los datos de baja prioridad a través de un enlace de baja velocidad por línea telefónica normal, lo cual es más barato que transmitir esos mismos datos a través de un canal vía satélite de alta velocidad.

El tercer objetivo es obtener un tiempo de respuesta mínimo y un caudal eficaz lo más elevado posible. Para reducir al mínimo el tiempo de respuesta hay que acordar el retardo entre la transmisión y la recepción de los datos de una terminal a otra. En aplicaciones interactivas, por ejemplo, es fundamental conseguir un tiempo de respuesta bajo. El caudal efectivo o eficaz expresan la cantidad máxima de datos de usuario que es posible transmitir en un determinado periodo de tiempo.

TOPOLOGÍAS FISICAS

Una topología física es el actual camino para alambrar la red entre los nodos. Cada uno de estos tipos de red será descrita.

Una conexión punto a punto es una conexión derivada entre dos dispositivos. Estas conexiones son generalmente usadas cuando la eficiencia es lo importante. Esto puede ser usado para implantar una conexión de alta velocidad entre dos servidores de red o sistema anfitrión. Las conexiones punto a punto son conexiones que rara vez son utilizadas para un nivel de tráfico normal o para el servicio de estaciones de trabajo.

En la siguiente sección se discutirán las tres topologías físicas LAN que soportan un eficiente uso de los recursos de la red: topología bus, topología de estrella (star) y topología de anillo (ring). Además de que cada red tiene una topología lógica que define los caminos a seguir por las funciones de red. Una red frecuentemente tiene una topología lógica que es diferente de una topología física.

TOPOLOGÍA DE BUS

La topología bus es la forma simple de una red multinodo. En esta topología todos los nodos de la red se conectan directamente a la misma pieza de cable. Cada nodo de red tiene una dirección asignada, un número que sólo identifica a un nodo. Esta dirección de nodo permite la identificación de mensajes que van dirigidos a un nodo específico.

Un segmento de red es un tramo de cable, por lo general es cable coaxial, encapsulando al final de cada extremo con un terminador. Cuando una estación transmite un mensaje a través de la red, las señales eléctricas viajan en ambas direcciones desde el punto original hasta que llega al final del cable, lo cual es absorbido por el terminador. Como la señal propagada baja del cable, cada estación en el cable puede examinar los datos. Para adherirse a las reglas del

protocolo de red, cada estación sólo recupera el mensaje cuando es destinado a él. Se puede ver la ilustración gráfica de esta topología en la figura I.9.a.

Los protocolos de red Ethernet típicamente corren bajo una topología de bus pero no todas las topologías en bus requieren ejecutarse en ethernet. Otros protocolos, tal como ARCnet, sólo se pueden usar en topologías bus.

Las ventajas de esta topología son:

- La topología de bus usa un mínimo de cable.
- La topología de bus requiere hardware de red que es barato.

Pero la gran desventaja es que cuando el cable se rompe o un nodo funciona mal puede afectar a todo el sistema y paralizarlo.

TOPOLOGÍA DE ESTRELLA

El cableado de una topología en estrella (un sistema central), el cual puede ser un servidor o un alambrado central conectando PC's o estaciones de trabajo. Cada nodo es conectado al sistema central por un cable individual porque cada computadora en la red requiere su propio cable para conectarse al centro de la red, la topología de estrella usa generalmente mucho más cable que las topologías de bus o anillo. Los concentradores centrales de alambrado representan un costo que no es requerido para las redes de bus. En general, una red de estrella representa altos costos en la topología física. Una simple red de estrella se presenta en la figura I.9.b.

A pesar de los altos costos, los beneficios son mejor mostrados en cuanto al diseño de red en esta topología porque cada máquina en una red en estrella es cableada individualmente, al romperse un cable sólo afecta una estación de trabajo. Los concentradores pueden ser excelentes lugares para diagnosticar los dispositivos de la red. Porque todas las señales son ruteadas a través del concentrador, y estas pueden ser monitoreadas desde un lugar central. Los altos costos de la red en estrella son generalmente justificados por la gran fiabilidad que provee.

Las topologías de estrella son empleadas con ARCnet, Token Ring, y una versión de Ethernet (10BASET).

Resumiendo, las ventajas de la topología de estrella son:

- Gran autonomía del nodo, al romperse un cable sólo afecta a la máquina que está conectada a ese cable.
- Localización centralizada para el diagnóstico del equipo de la red.

Las desventajas más notables son las siguientes:

- La topología de red requiere un gran tendido de cable.

- La topología de red requiere componentes como concentradores centrales, los cuales agregan un costo adicional a la red.

TOPOLOGÍA DE ANILLO

La topología en anillo se llama así por el aspecto circular del flujo de datos a diferencia de la topología bus, en la cual la señal se transmite a lo largo de todo el cable, la topología de anillo opera pasando la señal de nodo a nodo a lo largo de todo el anillo. Cada nodo recibe un mensaje que es pasado y, si el mensaje está direccionado a otro nodo, repite la señal al siguiente nodo. Porque esta acción de repetir amplifica y reacondiciona cada mensaje para ser retransmitido al siguiente nodo, las redes en anillo son menos sensitivas, en cuanto a la señal, cuando se incrementa el número de nodos en la red.

Las redes de área local por lo general no se implementan con la topología física de anillo. La señal de anillo (Token Ring), a pesar de su nombre de anillo, normalmente es alambrada como topología en estrella porque la configuración en estrella ofrece ventajas en términos de centralización. La topología en anillo no ofrece un punto central para administrar la red.

Más frecuentemente, las redes de anillo son implementadas sobre grandes áreas geográficas en las cuales una red en estrella podría ser ineficiente. Esta red es implantada por lo general cuando se requiere tolerancia a fallas a lo largo de toda la estructura de la red.

Las redes de anillo proporcionan una tolerancia a fallas, ya que contienen una señal de reserva de la ruta. Si el rompimiento del cable afecta a uno de los segmentos del anillo, la señal puede ser redireccionada utilizando la señal de reserva, la cual usualmente correrá en dirección opuesta a la ruta que tenía. Se da un doble recorrido pero tenemos como resultado que la red sigue funcionando.

En resumen algunas de las ventajas de la topología de anillo son:

- La redundancia en las rutas de envío pueden redireccionar los mensajes cuando un cable se llega a romper.
- La topología de red utiliza el cableado de forma eficiente cuando tiene una amplia área de cobertura.

Las desventajas de una topología de anillo incluyen el no tener un lugar central de monitoreo de la red.

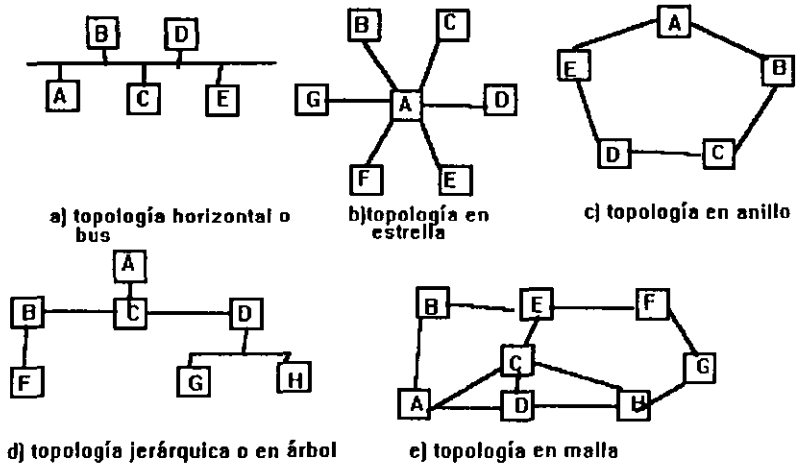


Figura 1.9 Topologías de red.

1.2.3 Medios Físicos de Transmisión

Los medios de transmisión físicos más comunes son:

- Cable telefónico
- Cable coaxial
- Fibra óptica
- Ondas de radio y microondas, las cuales son raramente utilizadas en redes LAN.

El **cable telefónico** es muy utilizado en la implantación de redes de área local ya que es muy común, pues se pueden usar las líneas públicas. La transmisión en este tipo de cable puede llegar hasta 10 Mbps en los mejores casos, ya que normalmente es menos. Este cable es susceptible a la interferencia. Está constituido por dos hilos trenzados recubiertos por una capa aislante.

El **cable coaxial**, normalmente se utiliza en la transmisión de la señal de televisión. Su nivel de transmisión es aproximadamente de 100 Mbps. Este cable esta constituido por un núcleo de cobre (conductor), recubierto por una capa aislante, además tiene una malla o capa conductora; y para protección de todo el cable esta cubierta por una capa aislante. Este es poco susceptible a la interferencia.

La **fibra óptica** está constituida por un núcleo de vidrio recubierto por una cubierta concéntrica de vidrio, y finalmente recubierta por una película protectora. La transmisión en la fibra óptica es por medio de la refracción de ases de luz, a través del núcleo. Comúnmente en un solo cable están incluidas varias fibras. Tiene capacidad para soportar ancho de banda muy

grande. Los niveles de transmisión de datos van de los 565 Mbps hasta los 200,000 Mbps. La fibra óptica no es susceptible a la interferencia.

Cuando se instala una red LAN en la que las terminales y los dispositivos están muy cerca físicamente, el tipo de cable a utilizar es lo de menos; pero cuando se va a instalar una red LAN que conecta grupo de desarrollo y diversos dispositivos o redes WAN, el tipo de cable es una de las características más importantes.

Existen dos tipos de Sistemas de Cableado: Sistema IBM y DECconnect Comunication System. El sistema de cableado IBM, maneja el cable telefónico y la fibra óptica. Los representa en diferentes tipos de acuerdo a necesidades específicas. El sistema IBM utiliza la topología de estrella y la estructura lógica de anillo (RING).

El sistema de cableado DECconnect Comunication System usa el cable delgado para Ethernet (coaxial), cable estándar Ethernet (coaxial), par trenzado (incluye 4), cable telefónico, cable de video (coaxial). Existe un módulo que puede concentrar cualquiera de estos tipos de cable. Recomendando centralizar las terminales de cada sección de la red (físicas). Este sistema utiliza la topología de BUS.

I.2.4 Métodos de Codificación, Transmisión y Control de Acceso

I.2.4.1 Codificación

Las estaciones de comunicación pueden usar una variedad de esquemas de codificación para representar valores binarios para transmitirse sobre un canal de comunicación. Con un simple esquema de codificación, usado durante muchos años en los circuitos telegráficos, la presencia de un pulso indicaba el valor 1 y la no presencia de un pulso representaba un valor cero. Una técnica que se sigue usando en comunicaciones de datos de baja velocidad sobre un canal ordinario de telecomunicaciones, lo cual es definido en un estándar llamado RS-232-C, que es publicado por la Asociación Industrial de la Electrónica (EIA). Con la transmisión en RS-232-C, un voltaje negativo en la línea representa el valor 1 y un voltaje positivo un valor 0. Las redes de área local por lo general usan esquemas de codificación más sofisticados. Los códigos más comúnmente usados en las redes locales son el código Manchester y Manchester Diferencial.

Los métodos de codificación más comunes en la industria son los siguientes, que son ilustrados en la figura I.10:

- No Retorno a Cero
- Retorno a Cero
- Manchester
- Manchester Diferencial
- AMI Bipolar
- Diferencial Complementado a Cero

Todas estas señales presentan una o varias de las siguientes características:

- Código unipolar. La señal nunca toma valores negativos o nunca valores positivos (es decir su signo algebraico no cambia: 0 voltios para el 1, y tres voltios para el cero, como un ejemplo).
- Código polar. La señal toma valores positivos y negativos (los signos opuestos identifican los estados lógicos: +3 y -3 voltios).
- Código bipolar. La señal varia entre tres niveles (ejemplo: +3, 0 y -3 voltios).
- Código con inversión alternada del uno (AMI). Representa los unos mediante pulsos cuya polaridad va alternando de un uno al siguiente.

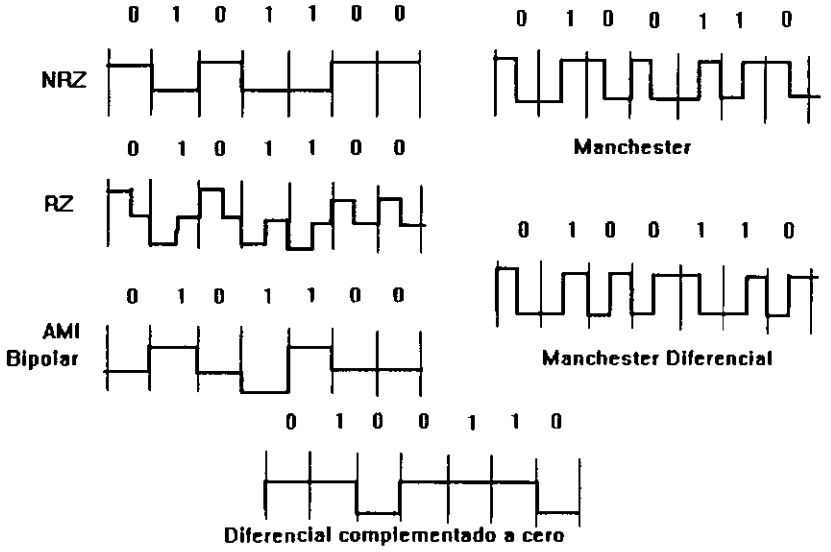


Figura I.10 Técnicas de codificación.

I.2.4.2 Transmisión

Existen dos técnicas de transmitir sobre un canal de comunicación: Banda Base, que se utiliza para la transmisión digital, y Ancho de Banda, que es para la transmisión analógica.

TRANSMISIÓN BANDA BASE: Se transmite la información mediante pulsos eléctricos discretos o aces de luz, a través del medio físico. Al viajar la señal por el cable puede sufrir errores por la interferencia que puede ejercerse sobre el medio físico de transmisión. Por lo cual se utilizan repetidores de señal. Para así regenerar la señal para que llegue hasta el receptor. La transmisión de Banda Base puede utilizar la **Multiplexación por División de Tiempo**, que consiste en la transmisión por turnos de diferentes transmisores (TDM). Lo que

determina el modo de acceder al medio físico de transmisión son los **métodos de control de acceso**.

TRANSMISIÓN ANCHO DE BANDA: Este tipo de transmisión analógica, transmite ondas electromagnéticas que son continuas y no discretas. Las ondas electromagnéticas tienen tres características: Amplitud, frecuencia y fase. La amplitud se refiere en el caso de las ondas electromagnéticas al nivel de voltaje aplicado en la señal, cuando es fibra óptica se refiere a la intensidad de la luz. La frecuencia es el nivel de oscilaciones o ciclos por segundo; la fase se describe con base en grados, que se refiere a dónde comienza un ciclo. Un ciclo constituye los 360°. Para que una señal sea transmitida debe ser modulada, lo cual quiere decir que se le superpone una señal que podrá variar en alguna de sus tres características, para poder transportar a la señal original.

I.2.4.3 Control de Acceso

Los protocolos de comunicaciones se conocen como protocolos de línea (enlace o canal) o Control de Enlace de Datos (DLC-Data Link Control) reciben este nombre porque gobiernan el flujo de tráfico entre estaciones a través de un canal físico de comunicaciones. Al gestionar un canal de comunicaciones, los protocolos de enlace de datos siguen varias etapas perfectamente ordenadas:

- Establecimiento del enlace. Una vez que un circuito de datos ha conseguido una conexión física con el otro circuito de datos remoto, el control de enlace de datos dialoga con el control de enlace de datos remoto para asegurarse de que ambos sistemas están preparados para intercambiar datos de usuario.

- Transferencia de información. Ambas máquinas intercambian datos a través del enlace. El control de enlace de datos comprueba todos los datos por si existe algún error en la transmisión y envía validaciones de los mismos a la máquina que transmite.

- Terminación del enlace. El control de enlace de datos renuncia al control del enlace (canal), lo cual significa que no pueden transmitirse más datos hasta que se restablezca el enlace. Por lo general un control de enlace de datos mantiene activo el enlace siempre que la comunidad de usuarios desee enviar datos a través del mismo.

Al hablar de los métodos de control de acceso, nos referimos a los nodos que acceden a una red y los medios de la red que se comparten en una forma de organización. En general, varios métodos de control de transmisión pueden ser clasificados de la siguiente forma:

- Control centralizado: Una estación controla toda la red y da a otras estaciones permiso para transmitir.
- Control aleatorio: Se le permite a una estación transmitir cuando se le proporciona un permiso específico.

- Control distribuido: Se da el derecho de transmitir a una sola estación a un tiempo. El derecho de transmisión se pasa de una estación a todos las demás estaciones que cooperan en el control de acceso a la red.

CONTROL CENTRALIZADO
· Sondeo (Polling)
· Circuito conmutado (Circuit switching)
· Acceso múltiple por división de tiempo (Time-división multiple access - TDMA)
CONTROL ALEATORIO
· Acceso múltiple por senso de portadora con detección de colisiones (Carrier sense multiple access with collision detection - CSMA/CD)
· Anillo ranurado (Slotted ring)
· Inserción de registro (Register insertion)
CONTROL DISTRIBUIDO
· Paso de testigo: testigo de anillo, testigo horizontal (Token ring, token bus)
· Acceso múltiple por senso de portadora con eliminación de colisiones (Carrier sense multiple acces with collision avoidance - CSMA/CA)

Figura 1.11 Técnicas del Control de Acceso.

De las técnicas de la figura I.11, tres son fundamentalmente importantes porque están basadas en el estandar IEEE para redes de área local y por lo tanto básicos para la implantación de una red LAN.

- Acceso múltiple por senso de portadora con detección de colisiones (CSMA/CD-carrier sense multiple access with collision detection). Este es el protocolo utilizado por internet.
- Token ring. Este es el básico para las redes LAN con arquitectura IBM.
- Token bus. Este es la base para arquitectura basada en protocolos de automatización de fábricas (MAP-Manufacturing Automation Protocolo) de General Motors.

El análisis matemático a dado la posibilidad de determinar de varios tipos de métodos de control de acceso cuáles son los más eficientes. La elección de opciones dependen de factores tales como estaciones a servir, longitud del mensaje y distribución del mensaje, la velocidad del canal y el tiempo de propagación en cierto radio para el tiempo de transmisión de un mensaje.

CONTROL DE ACCESO CENTRALIZADO

Las técnicas de control centralizado son más comunmente asociadas con redes de área amplia que con redes de área local, las cuales son LAN's en que el control es centralizado en un simple dispositivo de red. El control de transmisión centralizado proporciona facilidad en la administración y coordinación de la red y requiere la interfaz en una sola estación de red. Al mismo tiempo, el control de transmisión centralizado, por definición, da un solo punto de fallo

y un embotellamiento potencial en la red. El control centralizado puede emplear los siguientes métodos de control de acceso:

- Sondeo: Una estación maestra manda una notificación a las demás estaciones (secundarias) indicando que se le permite a cierta estación transmitir. Los conflictos de acceso a los medios son eliminados hasta que la estación de sondeo puede transmitir.

- Circuito conmutado: Este puede ser utilizado con éxito en una LAN implementada con control centralizado basada en una topología de estrella. Hay una estación central que recibe los requerimientos de transmisión de una estación secundaria y establece una conexión entre el transmisor y el que intenta recibir. El circuito conmutado es ampliamente usado en telefonía, especialmente en el tendido de líneas privadas (PBX-Private branch exchanges).

- Acceso múltiple por división de tiempo (Time división múltiple access-TDMA): Este proporciona un tiempo específico al canal para cada estación en la red. La estación está autorizada para transmitir a lo largo del canal durante un tiempo determinado. El ciclo de tiempo es inicializado y sincronizado por una estación maestra. TDMA puede usarse exitosamente en una topología bus.

CONTROL DE ACCESO DISTRIBUIDO.

Con los métodos de control de acceso que utilizan control de transmisión distribuida, todas las estaciones en la red cooperan en la realización del trabajo para controlar el acceso al medio de transmisión.

El paso de testigo en anillo (Token Ring) es el método mas ampliamente usado en el control de acceso distribuido y el más frecuentemente usado en redes con topología anular (Por ejemplo, IBM's Token Ring). El testigo puede ser usado en una topología de bus o de árbol. El método de testigo en bus es similar al testigo en anillo.

El paso de testigo es un mecanismo más complicado que el esquema CSMA. Por lo tanto es más cara su implantación y se tiene mayor control en la red para poder asegurar su correcta operación. Cuando el tráfico de la red es elevado, entonces, el paso de testigo llega a significar una garantía de que todas las estaciones de la red se les ha permitido un acceso igual a la red.

I.2.5 CAPAS

Los modelos OSI y TCP/IP se guían por estándares, la comunidad que desarrolla estándares para OSI y TCP/IP comparten las mismas prácticas. Por ejemplo, ambos tienen un avance tecnológico con comités y procesos de consenso usando alguna forma de procedimiento parlamentario. Ambos tiene una infraestructura jerárquica para coordinar el trabajo y creación

(modificación) de reglas de conducta. A ambos se les usa internacionalmente. En otros aspectos, tienen diferencias substanciales, especialmente con respecto a su imagen.

ESTANDAR OSI

A finales de los 70'S y principios de los 80'S, el primer estándar OSI fue desarrollado por el Comité Técnico 97 (TC97). En el caso de todos los comités de estandarización de la Organización Internacional de Estandarización (International Organization Standarization - ISO), los miembros de TC97 estuvo compuesto por los organismos nacionales de estandarización de cada país que decidió participar.

ESTÁNDARES DE INTERNET

El desarrollo de estándares para Internet se remonta a las investigaciones realizadas por los protocolos DARPA. En 1983, DARPA reestructura la ICCB y forma un Comité Administrativo Central llamado Tarjeta de Actividades Internet (Internet Activities Board - IAB). La IAB coordinó el diseño, ingeniería y aspectos cotidianos de operatividad en Internet, el cual permanece formalmente descrito como "una organización libre internacional de colaboración autónoma, redes interconectadas, soporte de comunicación anfitrión-anfitrión con una adherencia voluntaria a los protocolos abiertos y definición de procedimientos para estándares internet".

ARQUITECTURAS

Una arquitectura es un modelo abstracto de alguna parte del mundo real, en este caso, un modelo de organización y comportamiento de redes interconectadas, donde se comunican sistemas de cómputo y aplicaciones. Por que esto es una abstracción, es una forma útil para describir conceptos y relaciones en una forma clara y concisa, sin saturar de descripciones con referencias a características de sistemas o aplicaciones específicas.

La utilidad de la descripción de una arquitectura depende del poder de la abstracción (con éxito la arquitectura expresa importantes conceptos y relaciones, y sugiere unas nuevas) y de su relevancia (utilidad que provee la arquitectura para el desarrollo de redes y sistemas reales).

Para los diseñadores y constructores de redes, y las aplicaciones distribuidas que éstas usan, el desarrollo de la descripción de una arquitectura de un ambiente en el cual muchos de los componentes individuales de su diseño e implantación interactuarán con los servidores tienen dos propósitos esenciales. Primero, crear un marco conceptual de forma global en el cual las relaciones entre componentes individuales puedan ser estudiados y explorados con un mismo nivel de abstracción. Este marco generalmente alienta de forma general la solución de problemas colocando cada componente en un contexto abstracto que ilumina estas interacciones con otros componentes. Segundo, estos servidores como la base de descripciones formales de las características de componentes individuales, llegando a una "especificación funcional de forma global" para los ambientes distribuidos. Con alguna especificación

funcional, se establecen puntos de referencias comunes para el desenvolvimiento del diseño e implantación que sigue a esto.

Los sistemas abiertos son "abiertos" en virtud de su mutua adherencia de uno o más estándares de sistemas, el cual especifica estos aspectos para el comportamiento de un sistema abierto que son directamente relevantes en relación con la habilidad para comunicarse con otros sistemas abiertos. Hay muchas especificaciones y estándares de sistemas abiertos, cada uno de los cuales pertenece a una arquitectura particular (OSI o TCP/IP). Una arquitectura es típicamente descrita por un modelo de referencia, el cual expresa los principios de organización de la arquitectura (los puntos de referencia) y da un marco (modelo) dentro del cual varios servicios y protocolos, y las relaciones entre ellos, pueden ser definidas. Así, para OSI se tiene el "modelo de referencia OSI"; para TCP/IP el modelo de arquitectura internet o simplemente la arquitectura Internet.

El modelo de referencia OSI describe principios generales de redes de sistemas abiertos y especifica las prescripciones para sistemas abiertos siguiendo la arquitectura OSI, el modelo de referencia OSI recolecta "reglas universales" sobre los sistemas abiertos; la interconexión entre redes implica muchos tipos de redes diferentes en el mundo real; por lo que las direcciones no deben ser ambiguas.

El modelo de referencia OSI específicamente se abstiene de definir las características de un sistema abierto, tal como el manejo de buffers o el paso de información de un proceso a otro, que no son relevantes par la interacción de sistemas abiertos.

La "Guerra de Arquitecturas" entre OSI y TCP/IP involucra primeramente los aspectos del código de construcción de sistemas abiertos. Las diferencias entre los modelos de arquitectura OSI e Internet es relativo a variaciones a los temas que son comunes en todos los sistemas abiertos: capas, servicios, protocolos, y otros conceptos generalmente aplicables para la creación de un sistema abierto.

CAPAS

Muchas funciones pueden estar por encima de los medios de transmisión para soportar una eficiente comunicación entre sistemas de cómputo. Por ejemplo, a menudo es necesario asegurar que la información que se manda de una computadora a otra llegue en orden, sin corrupción, y libre de pérdida o duplicación. Si las dos computadoras están físicamente enlazadas en diferentes medios de transmisión (una en una red de área local Ethernet y otra pública, red de paquetes-conmutados), es necesario definir una función que selecciona la ruta y el envío de datos hacia diferentes puntos del origen al destino. Funciones adicionales que codifican y preservan la semántica y el contexto de la información, esto es tomando en cuenta que se ejecutará una aplicación distribuida en donde se están comunicando varios sistemas de cómputo (red de servicio de archivos, correo electrónico o un directorio de servicios). Como un ejemplo, considerar una lista de funciones que pueden ser útiles para la comunicación de computadoras (sistemas abiertos reales) para realizar la transferencia de archivos de un sistema a otro:

- Acceso a los servicios locales de transferencia de archivos.
- Identificación de la aplicación (y computadora destino) hacia la cual el archivo va a ser transferido.
- Establecer las comunicaciones entre el par de aplicaciones que pueden estar involucradas en la transferencia del archivo.
- Determinar una representación común de la información a ser transferida, incluyendo tanto la estructura del archivo y de los datos.
- Acceder al almacenamiento local para contener el archivo a ser transferido.
- Seleccionar el servicio de red y/o los medios de transmisión mediante los cuales el contenido del archivo va a ser enviado (ruteo y envío).
- Segmentación y reensamble de datos.
- Señalamiento del nivel-físico y la transmisión de bits.
- Seguridad de la transportación del contenido del archivo de la fuente al destino, resultando un duplicado exacto del archivo original en el destino.

Si se quisiera que un protocolo realizara estas funciones podría ser ineficiente, inflexible y completamente desordenado. El principio de "Capas" resuelve este problema con colecciones de funciones con asignaciones de relación y manejo. Por ejemplo, las funciones asociadas con la fiabilidad del transporte de datos (detección y corrección de pérdidas, desorden, duplicado, o corrupción de paquetes) lógicamente forman una asignación; las que se encargan de rutear y enviar y las que se encargan de la representación de datos. En el proceso, la asignación de funciones son organizadas jerárquicamente. La representación de datos, típicamente se presenta como una aplicación específica o funciones de "usuario final", contando con una alta fiabilidad de entrega o funciones de "transporte final". Tanto el modelo de referencia OSI y el modelo de arquitectura internet utilizan la asignación de capas de relación y manejo.

LAS SIETE CAPAS DE OSI

Las capas de aplicación, presentación y sesión del modelo OSI son referidas conjuntamente como capas superiores. Éstas proporcionan servicios al usuario final: funciones de aplicación que habilitan la compartición y manipulación de la información. El resto de las capas (transporte, red, enlace de datos, y físico) se refieren en forma conjunta como capas inferiores. Éstas proporcionan servicio de transporte de datos de sistema-a-sistema, organizando los recursos de comunicación que existen en un mundo real para transportar información de algún sistema fuente a algún sistema destino.

Las capas superiores son orientadas a la aplicación; se enfocan a los procesos de la aplicación que están relacionados con los "usuarios finales". Estos operan como si estuvieran conectados directamente de igual a igual con los servicios de transporte, sin tomar en cuenta el camino que toma su comunicación.
Las capas inferiores están orientadas a la comunicación; y su trabajo se enfoca a soportar el trabajo de las capas superiores que son las que ordenan los datos a ser transportados fuera de la computadora anfitrión o sistema; hacia otro sistema a través del arbitrario, complejo y heterogéneo mundo real formado de cables, fibras, redes, puentes y ruteadores.

CAPA	NIVEL	DESCRIPCIÓN
Aplicación	CAPAS SUPERIORES	Provee servicios generales relacionados con aplicaciones: Transferencia de archivos (ftp). Login remoto (rlogin, telnet). Correo electrónico (mail). Acceso a basees de datos, etc.
Presentación		Formato de datos: Establece una sintaxis y semántica de la información transmitida. Se define la estructura de los datos a transmitir (v.g. define los campos de un registro: nombre, dirección, teléfono, etc). Define el código a usar para representar una cadena de caracteres (ASCII, EBCDIC, etc). Compresión de datos. Criptografía.
Sesión		Coordina la interacción en la sesión (diálogo) de los usuarios: Permite a los usuarios en diferentes máquinas establecer una sesión. Una sesión puede ser usada para efectuar un login a un sistema de tiempo compartido remoto, para transferir un archivo entre 2 máquinas, etc. Controla el diálogo (quién habla, cuánto tiempo, half duplex o full duplex). Función de sincronización.
Transporte	CAPAS INFERIORES	Provee una transmisión de datos confiable punto a punto: Establece conexiones punto a punto sin errores para el envío de mensajes. Permite multiplexar una conexión punto a punto entre diferentes procesos del usuario (puntos extremos de una conexión). Provee la función de difusión de mensajes (broadcast) a múltiples destinos. Control de Flujo.
Red		Enruta unidades de información: Divide los mensajes de la capa de transporte en paquetes y los ensambla al final. Utiliza el nivel de enlace para el envío de paquetes: un paquete es encapsulado en una trama. Enrutamiento de paquetes. Envía los paquetes de nodo a nodo usando ya sea un circuito virtual o como datagramas. Control de Congestión.
Enlace de datos		Provee intercambio de datos entre dispositivos en el mismi medio: Estructura el flujo de bits bajo un formato predeterminado llamado trama. Para formar una trama el nivel de enlace agrega una secuencia especial de bits al principio y al final del flujo inicial de bits. Transfiere tramas de una manera confiable libre de errores (utiliza reconocimiemtos y retransmisión de tramas). Provee control de flujo.
Físico		Transmisión del flujo de bits a través del medio. No existe estructura alguna. Maneja voltajes y pulsos eléctricos. Especifica cables, conectores y componentes de interfaz con el medio de transmisión.

figura 1.12 Modelo OSI.

La **CAPA DE APLICACIÓN** da acceso a los servicios de OSI. Esta capa es donde reside la aplicación distribuida y en la cual se tiene acceso al ambiente de la red.

La **CAPA DE PRESENTACIÓN**, a esta capa le concierne la representación (sintaxis) de los datos y sus cambios, permitiendo obtener su significado (semántica). Esto para definir una forma común o canónica para la representación y manipulación de la información de la aplicación. Algunas computadoras usan como codificador de caracteres nativo al EBCDIC, y otras usan el ASCII, y así las diferentes computadoras y sistemas operativos almacenan información en memoria o archivos de disco de diferentes formas. Las funciones de la capa de presentación permiten a la aplicación representar los datos de forma independiente, para proporcionar un lenguaje universal en el cual se pueda describir una estructura de datos abstracta. Lo que comúnmente es llamado "lenguaje de programación de red", la capa de presentación permite a la aplicación modificar la información estructurada. Esta también define el camino a través del cual los elementos que están en el lenguaje de red son transmitidos de un sistema a otro.

La **CAPA DE SESIÓN** proporciona mecanismos para organizar y sincronizar los cambios de datos entre los procesos de la aplicación. La capa de sesión permite esta función a la aplicación mediante el envío y recepción de datos (puntos de sincronización), proporcionando caminos a la aplicación para controlar la dirección del flujo de la información (administración de turno), coordinando múltiples cambios independientes (actividades) dentro de un contexto de sesión simple, y proporciona a las aplicaciones la información acerca de cada ocurrencia de errores y los pasos que hay que seguir para resincronizar parte o todo el diálogo afectado.

La **CAPA DE TRANSPORTE** proporciona la "transferencia transparente de los datos desde un sistema-fuente de un sistema abierto hacia un sistema-final " (ISO/IEC 7498: 1993). La capa de transporte es responsable de la creación y mantenimiento de la conexión básica sistema-a-sistema (end-to-end) entre la comunicación de sistemas abiertos resultando que los bits entregados al receptor son los mismos bits enviados por el transmisor: en el mismo orden y sin modificación, pérdida o duplicación.

La **CAPA DE RED** proporciona "una ruta entre las entidades de transporte, relevando a las capas superiores de dar con el camino en el cual los datos serán transferidos de un (end) sistema abierto a otro". (ISO/IEC 7498:1993). La capa de red determina la ruta que van a tomar los datos de su fuente original hasta su destino final y envía los datos sobre la ruta. Esta da un servicio que es independiente de los medios de transmisión, conteniendo las funciones necesarias para ir de la fuente al destino.

La **CAPA DE ENLACE DE DATOS** "proporciona el control de la capa física, y detecta y/o corrige los posibles errores cuando pueden ocurrir" (ISO/IEC 7498: 1993). La capa de enlace de datos cubre de forma particular los métodos de acceso físico siendo que dichas funciones son necesarias para recuperar errores que pueden ser introducidos durante la transmisión (ruido y otras formas de señales de interferencia).

La **CAPA FÍSICA** proporciona "elementos mecánicos, eléctricos, funcionales y procedurales; para activar una conexión física para la transmisión de bits" (ISO/IEC 7498:1993), la función de la capa física en la arquitectura de una red es transformar los bits de un sistema de cómputo en señales electromagnéticas (o equivalentes) para un medio de transmisión en particular (cable, fibra, etc.).

LAS CINCO CAPAS DE INTERNET

La arquitectura TCP/IP se caracteriza por la compresión de los servicios en el nivel de aplicación y en el nivel de red. La distinción entre capas superiores (actualmente, en la arquitectura internet, con una sola capa superior) y capas inferiores sigue la misma lógica para TCP/IP como para OSI.

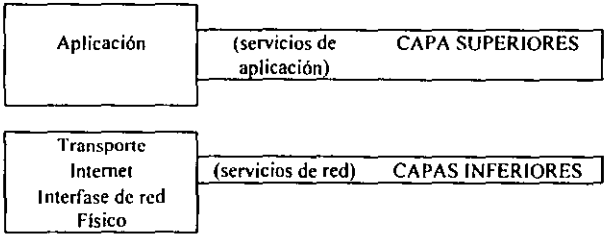


Figura I.13 Modelo Internet.

Los servicios de internet en el nivel de aplicación son asignados a los programas de aplicación que operan a lo largo de internet basados en TCP/IP. Todas las funciones de usuario final, las cuales son divididas en tres capas en OSI, son incorporadas en la arquitectura TCP/IP dentro de cada programa de aplicación individualmente. Las aplicaciones TCP/IP, son diseñadas para operar directamente sobre una única interfaz de transporte.

Los servicios de internet en los niveles de red corresponden más o menos directamente con los servicios proporcionados por las capas inferiores de OSI. La capa de transporte de OSI proporciona muchos más servicios de comunicación sistema-a-sistema en comparación con la capa de transporte TCP/IP. La capa de interfaz de red en la arquitectura TCP/IP corresponde a una combinación de la capa de enlace de datos de OSI y las funciones específicas de red de la capa de red de OSI. La capa física, en ambas cumple la misma función.

La única capa superior en la arquitectura internet refleja la introducción deliberada de la forma en que son organizadas las aplicaciones distribuidas. A este respecto, OSI proporciona un modelo más adecuado. De otra forma, la correspondencia entre las capas inferiores en la arquitectura internet y las redes del mundo real es mucho más clara y adecuada que en el caso de OSI. Pero ha sido sugerido que OSI es el mejor modelo para aplicaciones distribuidas y que TCP/IP es el mejor modelo para las redes que soportan este tipo de comunicaciones.

Los principios aplicados para el desarrollo del modelo de referencia son muy similares a la arquitectura de TCP/IP. Pero desafortunadamente no utilizan la misma terminología, OSI

utiliza definiciones que en muchos casos suelen ser confusas, ya que dichos términos no están relacionados con la ingeniería de redes ya que se utiliza una terminología muy formal. En cambio TCP/IP utiliza términos y conceptos que son muy familiares, apegándose a la descripción de las redes reales.

En la arquitectura OSI, los servicios que proporciona una capa, son el resultado de un conjunto de actividades de un sistema de cómputo integrado con OSI. Cada capa representa un subsistema, y dichos subsistemas representan las funciones de cada una de las capas. Las actividades de los elementos de un subsistema son llamados ENTIDADES.

I.2.6 PROTOCOLOS.

En esta sección se verán los protocolos de comunicación para el transporte de datos, pero es importante distinguir los diferentes tipos de protocolos con que se cuenta. Al hablar de protocolos nos referimos al termino **MIDDLEWARE**, lo que podemos definir como la conectividad que permite a las aplicaciones una comunicación transparente con otros programas o procesos, independientemente de su ubicación. El elemento principal de la conectividad es el Sistema Operativo de Red (NOS Network Operating System). El Sistema Operativo de Red proporciona los servicios de ruteo, distribución, mensajes y servicios de administración de la red para los archivos y la impresión. En realidad el Sistema Operativo de Red de apoya en los protocolos de comunicación para proporcionar los servicios específicos. Los protocolos son divididos en tres grupos: protocolos del medio, transporte y cliente/servidor. Los protocolos del medio determinan el tipo de conexión física que se usa en una red (algunos ejemplos de protocolos del medio son Ethernet, Token Ring, Fiber Distributed Data Interface (FDDI), coaxial y par tenzado (twisted-pair). Los protocolos de transporte proporcionan los mecanismos para mover paquetes de datos del cliente al servidor (algunos ejemplos de protocolos de transporte son el IPX/SPX de Novell, AppleTalk de Apple, Transmission Control Protocol/Internet Protocol (TCP/IP), Open Systems Interconnection (OSI)). Cuando la conexión física ha sido establecida y ha sido elegido el protocolo de transporte, entonces el protocolo cliente/servidor es requerido antes de que el usuario pueda acceder a los servicios de red. Un protocolo cliente/servidor determina la forma en la cual los clientes requieren información y servicios de un servidor y también cómo los servidores responden a estos requerimientos (algunos ejemplos de protocolos cliente/servidor son NetBIOS, RPC, Advanced Program-to-Program Communication (APPC), Names Pipes, Sockets, Transport Level Interface (TLI) and Sequenced Packet Exchange (SPX)) (Ver la figura I.14).

Los protocolos son acuerdos acerca de la forma en que se comunican entre sí dos estaciones y los dispositivos de comunicaciones, y pueden incluir regulaciones concretas que recomienden u obliguen a aplicar una técnica o convenio determinado.

Los protocolos de comunicaciones de datos son usados para coordinar el intercambio de información entre diferentes dispositivos de red. Éstos establecen el mecanismo mediante el cual cada dispositivo reconoce completamente el significado de la información de otro dispositivo. En el mundo de las comunicaciones, hay un número de protocolos en uso, los que

cuentan con varias estructuras básicas que comprenden diferentes aspectos de la comunicación de datos.

TIPOS DE PROTOCOLOS	
PROTOCOLOS DE MEDIO	* Ethernet * Token Ring * Fiber Distributed Data Interface (FDDI) * Coaxial * Par trenzado
PROTOCOLOS DE TRANSPORTE	* IPX/SPX de Novell * AppleTalk de Apple * TCP/IP -Trnamission Control Protocol/Internet Protocol
PROTOCOLOS CLIENTE/SERVIDOR	* NetBIOS * RPC Llamada a Procedimientos Remotos * Advanced Program-to-Program Communication (APPC) * Names Pipes * Sockets * Transport Level Interface (TLI) * Sequence Packet Exchange (SPX)

Figura 1.14 Tipos de Protocolos

Veremos cinco de los protocolos más ampliamente utilizados dentro del contexto del modelo de referencia OSI y las alternativas en la comunicación de protocolos sobre una red.

Bajo la perspectiva de pequeños grupos de trabajo u oficinas departamentales, las redes locales han llegado a ser la plataforma de mayor integración de cómputo. De tener tan solo veinte usuarios en la red ahora se puede expandir a quinientos usuarios en una interred. Con el aumento de esta nueva demanda, los protocolos de red han llegado a ser muy poderosos y flexibles. Los siguientes cinco protocolos de red soportan los grupos de trabajo y el modelo de integración, de acuerdo a diferentes grados:

- **XEROX NETWORK SYSTEMS (XNS)**
- **NOVELL IPX/SPX**
- **NETBIOS**
- **APPLETALK**
- **INTERNET TCP/IP**

IPX/SPX

Los protocolos de red Novell están basados en el protocolo Xerox Network System el cual fue desarrollado por Xerox Corporation's Palo Alto Research Center (PARC). La estructura de capas, interacción del protocolo y direccionamiento de la red corresponde idénticamente a XNS.

Los primeros protocolos Novell fueron el Internetwork Packet Exchange (IPX) y Sequenced Packet Exchange (SPX). SPX es confiable, protocolo orientado a conexión, mientras que IPX tiene un esquema de datos del protocolo poco confiable.

Los paquetes de la red no sólo son dirigidos a un dispositivo en particular, sino a la ejecución de un proceso particular en un dispositivo. Los dispositivos son identificados en forma de direcciones, los cuales son asociados con una subred o red local en particular. El objetivo del proceso en el dispositivo es ser identificado por un número de enlace. Novell adoptó la estructura de direccionamiento entre redes de XNS en la cual una dirección completa es dada por el número de la red, el número del anfitrión y el número de enlace en el anfitrión. El número de red es de 32 bits, el número del anfitrión es de 48 bits y el enlace es de 16 bits.

IPX

EL Internetwork Packet Exchange (IPX) es la capa de red del protocolo Novell. IPX proporciona poca conectividad, es poco confiable, y proporciona servicio de datos para estaciones de trabajo y servidores. IPX hace el mejor intento para distribuir un paquete a su destino, pero no identifica si en realidad el paquete ha llegado a su destino. IPX depende de la capa superior del protocolo, tal como SPX o NCP, para proporcionar fiabilidad y secuencia en los servicios de bloques de datos.

Un paquete en IPX consiste de una cabeza (30 bytes) y una sección de datos. Como IPX no proporciona alguna facilidad para la fragmentación de paquetes, las implantaciones de IPX aseguran que los paquetes que son enviados sean lo bastante pequeños para poder ser transmitidos por algún medio físico. IPX requiere que todos los enlaces físicos que sean habilitados para su uso tengan una longitud de 576 bytes. Varias implantaciones refinan un poco este proceso detectando cuando los paquetes son enviados directamente a su destino sobre un solo enlace físico. Si este enlace físico puede manejar paquetes mas grandes que 576 bytes, se usarán paquetes más grandes.

SPX

El Sequenced Packet Exchange (SPX) es la capa de transporte del protocolo Novell. Esta fue derivada de Xerox Sequenced Packet Protocol (SPP).

SPX proporciona fiabilidad, orientada a la conexión, servicio virtual de circuito entre estaciones de red. SPX hace uso del servicio de datos de IPX para dar la secuencia a bloques de datos. Esto es realizado para implementar un sistema que requiere que cada paquete a ser

enviado pueda ser reconocido. También proporciona flujo de control entre las estaciones de la red y asegura que no sean duplicadas al ser liberadas por un proceso remoto.

SPX reduce el número de tiempos que se dan en las retransmisiones con el fin de disminuir el congestionamiento en la red. Normalmente las retransmisiones ocurren después de que la estación ha mandado a la espera con un tiempo fuera para un reconocimiento del paquete que se perdió, daño o truncó. SPX utiliza un algoritmo de cronometraje de incremento-heurístico para estimar un tiempo de retransmisión acertado. Y también usa información histórica para determinar el tiempo inicial, y entonces incrementar el tiempo en un cincuenta por ciento si un tiempo fuera ocurre. El proceso continúa hasta que el valor máximo de tiempo fuera es rebasado o hasta que se regresa el reconocimiento dentro del tiempo y las retransmisiones no son requeridas. En el segundo caso, el tiempo fuera se establece en un valor que se considere más acertado respecto a las condiciones de la red.

SPX agrega 12 bytes a la cabeza del paquete de IPX, esto para transportar la información de control de la conexión. Adicionando los bytes mencionados al total que tenía originalmente tenemos como resultado un total de 42 bytes. El tamaño máximo de un paquete de SPX es el mismo que para un paquete de IPX.

NETBIOS

El Network Basic Input/Output System (NetBios) es un programa interfaz de aplicación de alto nivel (API) el cual fue diseñado para permitir a los programadores construir aplicaciones de red en redes con PC del tipo IBM. Este protocolo fue desarrollado por Sytek, y fue implantado originalmente en tarjetas adaptadoras de red para PC IBM. NetBios fue introducido por IBM en 1984 y adoptado por Microsoft para utilizar con productos de red MS-Net. Después IBM proporcionó un emulador que habilita a NetBios para trabajar con tarjetas de interfaz de red que utilicen Token Ring.

NetBios no es en realidad un protocolo; es una interfaz que proporciona aplicaciones de red con asignación de comandos para establecer seciones de comunicación, envío y recepción de datos, y nombres de objetos de la red.

Las mejores compañías en cuanto a redes, incluyendo IBM , Novell, Microsoft y Banyan, tienen soporte para la interfaz NetBios ya sea que la utilicen directamente o mediante emuladores, pero es utilizado en sus respectivos protocolos. Novell creó el soporte de NetBios utilizando el protocolo IPX. Respecto a la implantación, la interfaz NetBios para aplicaciones distribuidas sigue siendo consistente.

Muchas aplicaciones de red han sido escritas usando la interfaz NetBios. Y continuará siendo un medio popular para el desarrollo de aplicaciones distribuidas.

Viéndose desde la perspectiva del modelo de referencia OSI, NetBios nos da una interfaz para la capa de sesión. A este nivel, NetBios es capaz de proporcionar fiabililidad, transferencia de datos en bloque orientados a conexión y con un sistema de etiquetas o nombres para identificar

las estaciones en la red. Mientras NetBios proporcione un servicio de datos entre conexiones poco fiable, y no proporcione un servicio de ruteo, la construcción de interredes será muy difícil.

APPLETALK

Apple Computer diseñó su protocolo de comunicaciones, llamado AppleTalk, entre 1983 y 1984. La meta fue conectar las computadoras personales Macintosh con impresoras, servidores de impresión, servidores de archivos, ruteadores y puentes hacia sistemas de computadoras que fueron construidas por otros fabricantes. El primer uso común de AppleTalk fue la conexión de una computadora gráfica a una impresora láser Apple LaserWriter.

Cada Macintosh e impresora Apple LaserWriter incluye hardware nativo que soporta la arquitectura de red AppleTalk. Adicionalmente, el software del sistema incluye un extenso soporte de red. Esta combinación de soporte en hardware y software ayuda a que AppleTalk sea una de las soluciones de red más comunes para computadoras personales de todo uso.

AppleTalk fue diseñado desde bases firmes, para ser abierto, con una importante arquitectura de red que soporta nuevas tecnologías físicas en la red y nuevas fases de protocolo. Esto permite un amplio número de conexiones de computadoras y periféricos sobre una potencial área geográfica para enlazar áreas locales en interredes.

AppleTalk es diseñada para soportar redes punto-a-punto. La filosofía de diseño de la computadora Macintosh también se extiende hacia el diseño de redes. El modelo de usuario que interactúa en una red han sido lo más transparente posible para que las operaciones y paradigmas estándar sean extendidos en el acceso de los recursos a través de la interred. Esto significa que los usuarios pueden acceder a volúmenes remotos desde sus escritorios y hacer uso de archivos.

La instalación de nodos en la red fue diseñado para soportar un modelo "plug-and-play", en el cual los usuarios acceden al enlace físico, en donde la configuración es administrada automáticamente por el sistema de software.

La fase 2 de AppleTalk fue introducida en junio de 1989 con un aumento de la compatibilidad que existía en AppleTalk en la fase 1. AppleTalk en su fase 2 proporciona un mayor soporte, en redes amplias para poder habilitar un mayor número de nodos (estaciones de trabajo, impresoras y servidores) en la red. En su fase 1 AppleTalk soportaba un máximo de 254 nodos en una sola red, la fase 2 habilita a múltiples números de red para ser asociados con un solo número de red, habilitando 253 nodos por número de red. En su fase 2 proporciona soporte para LocalTalk, EtherTalk y TokenTalk. LocalTalk es la especificación física y de enlace de datos para proteger el esquema de la familia de par de cable trenzado de Apple. Ethertalk y TokenTalk son implantación de Apple para Ethernet y Token Ring. AppleTalk Internet Router también fue modificado para habilitar la conexión de 8 redes AppleTalk en una mezcla de LocalTalk, EtherTalk y TokenTalk. (Cada red individual debe tener un solo tipo de enlace físico).

ANTECEDENTES

Los diseñadores han utilizado AppleTalk para proporcionar una gran variedad de aplicaciones distribuidas, con soporte de Apple Macintosh, IBM PS/2 y computadoras PC compatibles, estaciones de trabajo y ejecuciones en UNIX. Estas aplicaciones incluyen compartición de archivos, sondeo de impresoras, compartición de impresoras y mensajes electrónicos. Además de que los gateways son habilitados para enlazar redes AppleTalk con computadoras DEC, mini y mainframe.

TCP/IP (TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL)

Como ya tuvimos oportunidad de estudiar un poco lo que es TCP/IP en el punto I.2.5, sabemos que por la gran trascendencia que ha tenido en los últimos tiempos, se le ha llegado ha considerar como un modelo de referencia. Y no tan sólo como un protocolo más que se tiene disponible.

CAPÍTULO II

Modelo Cliente/Servidor

II MODELO CLIENTE/SERVIDOR

II.1 Bases de Datos Distribuidas

Un ambiente de bases de datos distribuidas es una BASE DE DATOS que tiene los elementos de datos bajo un control múltiple, nodos enlazados mediante un medio de comunicación que operan independientemente. Un NODO es definido como uno o mas CPU´s que comparten el mismo medio de almacenamiento bajo el control de un mismo sistema operativo.

Un sistema operativo puede tener capacidad multiusuario, capacidad multitarea o ambos. La capacidad multiusuario implica la capacidad multitarea, pero multitarea no implica necesariamente un ambiente multiusuario. Capacidad multiusuario significa que múltiples usuarios pueden correr una aplicación al mismo tiempo. Capacidad multitarea significa que un usuario puede ejecutar múltiples tareas en el mismo nodo. Entonces, el nodo consiste de un solo procesador, el cual ejecuta una sola tarea a un tiempo, pero da la apariencia de ejecutar todas las tareas a un mismo tiempo. Un nodo puede ser una microcomputadora, una estación de trabajo, una minicomputadora, una mainframe, o cualquier otro sistema que tenga uno o mas procesadores. Un nodo puede ser inteligente, inteligente significa que tiene un procesador y memoria de almacenamiento. Por lo anterior, una terminal tonta no es un nodo.

Ambientes Potenciales

Cuando se hace referencia a un ambiente de bases de datos distribuidas, nos podemos referir a diferentes caminos de representación de datos: en plataformas de hardware heterogéneas, bajo diferentes sistemas operativos, posiblemente bajo el control de diferentes DBM´s, o tal vez con diferentes aplicaciones de software. Y es también posible que la organización en sí sea heterogénea, teniendo como resultado natural un ambiente de procesamiento descentralizado.

Una base de datos distribuida es una colección de sitios de procesamiento-almacenamiento autónomos, teniendo distancia de por medio entre uno y otros, conectado a través de una red de comunicación de datos. Una red de comunicación de datos es una colección de facilidades de transmisión, procesos de red, y además proporciona el movimiento de los datos a través de terminales y procesos de información.

BASES DE DATOS DISTRIBUIDAS

- Una colección de sitios autónomos con una capacidad razonable de almacenamiento
- Sitios conectados mediante una RED DE COMUNICACIÓN DE DATOS
- Los datos deben de cumplir con:
  - TRANSPARENCIA en la ubicación de los datos
  - SINCRONIZACIÓN para preservar la integridad de la base de datos
- Los datos pueden ser (una combinación de):
  - DISTRIBUIDOS.- algunos datos en un sitio y otros datos en otros sitios
  - PARTICIONADOS.- datos relacionados lógicamente pueden ser físicamente divididos en varios nodos
  - REPLICADOS.- múltiples copias de datos que se mantienen en varios sitios
- El control es distribuido
  - OPERACIÓN CONTINUA a pesar de un nodo o un fallo de comunicaciones

MODELO CLIENTE/SERVIDOR

La principal ventaja de los sistemas distribuidos de base de datos es la capacidad de compartir y acceder a la información de una forma fiable y eficaz.

Utilización compartida de los datos y distribución del control.- Si varias localidades diferentes están conectadas entre sí, entonces un usuario de una localidad puede acceder a datos disponibles en otra localidad. La ventaja principal de compartir los datos por medio de la distribución es que cada localidad puede controlar hasta cierto punto los datos almacenados localmente. En un sistema centralizado, el administrador de la base de datos central controla la base de datos. En un sistema distribuido existe un administrador global de la base de datos que se encarga de todo el sistema. Parte de estas responsabilidades se delegan al administrador de la base de datos de cada localidad. Dependiendo del diseño del sistema distribuido, cada administrador local podrá tener un grado de autonomía diferente, que se conoce como autonomía local. La posibilidad de contar con autonomía local es en muchos casos una ventaja importante de las bases de datos distribuidas.

Fiabilidad y disponibilidad.- Si se produce un fallo en una localidad de un sistema distribuido, es posible que las demás localidades puedan seguir trabajando. El sistema debe detectar cuándo falla una localidad y tomar las medidas necesarias para recuperarse del fallo. Aunque la recuperación de fallos es más compleja en sistemas distribuidos que en los centralizados, la capacidad que tiene el sistema para seguir trabajando, a pesar del fallo de una localidad, da como resultado una mayor disponibilidad.

Agilización del procesamiento de consultas.- Si una consulta comprende datos de varias localidades, puede ser posible dividir la consulta en varias subconsultas que se ejecuten en paralelo en distintas localidades.

La desventaja principal de los sistemas distribuidos de base de datos es la mayor complejidad que se requiere para garantizar una coordinación adecuada entre localidades.

El aumento de complejidad se refleja en:

- Coste de desarrollo de software. Es más difícil estructurar un sistema de base de datos distribuida y, por lo tanto, su costo es mayor.
- Mayor posibilidad de errores. Puesto que las localidades del sistema distribuido operan en paralelo, es más difícil garantizar que los algoritmos sean correctos. Existe la posibilidad de errores extremadamente sutiles.
- Mayor tiempo extra de procesamiento. El intercambio de mensajes y los cálculos adicionales que se requieren para coordinar las localidades son una forma de tiempo extra que no existe en los sistemas centralizados.

Una definición de Sistemas de Cómputo Distribuido es referida como " un número de elementos de procesamiento autónomo (no necesariamente homogéneos) que son interconectados por medio de una red de cómputo y que cooperan en la ejecución de sus tareas asignadas". El "elemento de procesamiento" en esta definición se refiere como un dispositivo de cómputo que puede ejecutar un programa por sí mismo".

Los sistemas de cómputo distribuidos pueden ser clasificados respecto a un número de criterios. Algunos de estos criterios son los siguientes (Bochmann, 1983): grado de acoplamiento, estructura de interconexión, interdependencia de componentes, y sincronización entre componentes. ***Grado de acoplamiento*** se refiere a la medida en que se determina cómo los elementos de procesamiento son interconectados. Esto puede ser medido en razón de la cantidad de datos intercambiados con la cantidad de procesamiento local realizado para la ejecución de las tareas. En la ***estructura de interconexión***, los elementos de procesamiento pueden depender bastante uno del otro en la ejecución de las tareas, o la interdependencia puede ser tan poca como el paso de mensajes para comenzar la ejecución y reporte de resultados finales. La ***Sincronización*** entre elementos de procesamiento puede ser mantenida de forma síncrona o asíncrona. Se debe notar que algunos de estos criterios no son completamente independientes. Por ejemplo, si la sincronización entre elementos es síncrona, se puede esperar que los elementos de procesamiento son fuertemente interdependientes, posiblemente trabajan en un fuerte acoplamiento.

Una ***Base de Datos Distribuida*** se define como una colección de múltiples bases de datos interrelacionadas lógicamente distribuidas a través de una red de cómputo. Un ***Sistema Administrador de Base de Datos distribuida*** es definido como el sistema de software que permite la administración del DDBS y proporciona la distribución transparente a los usuarios. Los dos términos importantes en estas definiciones son "interrelacionados lógicamente" y "distribuidos sobre una red de cómputo".

Por lo tanto, un DDBS no es una colección de archivos que puede ser almacenada individualmente en cada nodo de una red de cómputo. Para formar un DDBS, los archivos no sólo pueden ser lógicamente relacionados, existe una estructura entre los archivos, y el acceso puede ser mediante una interfaz común. La distribución física de los datos es muy importante. Esto crea problemas que no se encuentran cuando la base de datos reside en la misma computadora. Se debe notar que la distribución física no necesariamente implica que los sistemas de cómputo son geográficamente dispersos, ya que pueden estar en un mismo piso.

La distribución de datos y aplicaciones promete ventajas potenciales. Y dichas ventajas también pueden ser consideradas como los objetivos de un Sistema de Base de Datos Distribuido (DDBS Distributed Database System).

A continuación se enumeran algunas **ventajas** y **desventajas** que son proporcionadas por los Sistemas Distribuidos:

Ventajas

Autonomía local: Desde que los datos son distribuidos, un grupo de usuarios que comúnmente comparten tales datos pueden necesitar disponer de dichos datos de forma local. Esto permite asignar y reforzar localmente la consideración de las políticas en el uso de los datos. Hay estudios que indican que la habilidad para compartir la autoridad y responsabilidad de la administración de la información es una de las más grandes razones por la que las organizaciones empresariales consideran los sistemas de información distribuidos.

Mejoramiento de la eficiencia: Con el uso regular de los datos por los usuarios y dado el paralelismo inherente en los sistemas distribuidos, puede ser posible el proporcionar la eficiencia en el acceso a los datos. Por otro lado, en cada sitio se mantiene sólo una porción de la base de datos. Y la recuperación de los datos, mediante una transacción, que puede estar almacenada en un cierto número de sitios, siendo posible que se ejecute la transacción en paralelo.

Mejora en la fiabilidad/disponibilidad: Si los datos son replicados, por lo tanto existen en más de un sitio, la inhabilitación de alguno de estos sitios, o la falla de un enlace de comunicación hace alguno de estos sitios inaccesible, no necesariamente ocurre que los datos sean inaccesibles. Este tipo de fallas no provocan la inaccesibilidad total al sistema. Aún cuando ciertos datos puedan ser inaccesibles, el DDBS puede aún así proporcionar algunos servicios.

Economía: Esto es posible si se observa desde dos perspectivas. La primera es en términos de costos de comunicación. Si la base de datos es geográficamente dispersa y las aplicaciones se ejecutan necesitando una fuerte interacción de los datos dispersos, esto puede ser mucho más económico mediante la partición de aplicaciones y haciendo en lo posible que el procesamiento sea de forma local. El segundo punto de vista es el costo mucho menor al utilizar sistemas de computadoras pequeñas con un poder equivalente al de alguna máquina grande.

Expandibilidad: En un ambiente distribuido, es muy fácil adaptarse al incremento en el tamaño de la base de datos. La expansión es requerida cuando se agrega procesamiento y poder de almacenamiento a la red. Obviamente, no puede ser posible obtener un incremento lineal en "poder", ya que esto también depende del nivel de distribución.

Compartición: Las organizaciones que operan geográficamente distribuidas almacenan sus datos en una forma distribuida. Por lo que, si el sistema de información no es distribuido, es prácticamente imposible compartir los datos y recursos. Un sistema de base de datos distribuida hace por lo tanto factible esta compartición.

Desventajas

Falta de experiencia: El propósito general de los sistemas de base de datos distribuidas aún no es muy comúnmente utilizado. Por lo regular, se cuentan con sistemas prototipo o sistemas que son hechos a la medida para alguna aplicación. Esto tiene serias consecuencias porque las soluciones que han sido propuestas para varios problemas no han sido examinadas en el ambiente operativo actual.

Complejidad: Los problemas en DDBS son inherentemente más complicado que al administrar una base de datos centralizada, ya que esto no sólo incluye los problemas encontrados en un ambiente centralizado, sino también son agregados nuevos problemas sin resolver.

Costo: Los sistemas distribuidos requieren hardware adicional (mecanismos de comunicación), esto implica un incremento en el costo de hardware. La tendencia de decremento en los costos de hardware no hacen de esto un factor significativo. Un aspecto más importante en los costos es el factor del software que hay que agregar y el costo que implica, además de las comunicaciones que puedan ser necesarias para resolver los problemas técnicos. El desarrollo de técnicas de ingeniería de software pueden ayudar en este aspecto.

Quizá el componente de costo más importante se debe a la duplicación de esfuerzos (personal). Cuando las computadoras se localizan en diferentes sitios, llega a ser necesario emplear más gente para dar mantenimiento al equipo de cada sitio. Esto da un incremento de personal para las operaciones del procesamiento de datos. Entonces, algo muy importante ha considerar es si es rentable el aumentar la eficiencia del sistema de información de la organización tomando en cuenta el incremento en el costo que trae consigo el contratar todo el personal necesario.

Distribución del control: La distribución crea problemas de sincronización y coordinación. El control distribuido puede muy fácilmente llegar a ser una desventaja si no se toma con cuidado la adopción de políticas del negocio de forma adecuada respecto a este asunto.

Seguridad: Uno de los mayores beneficios de las bases de datos centralizadas es el control que proporciona sobre el acceso a los datos. La seguridad puede llegar a controlarse fácilmente en un lugar central, con las reglas de control que mantiene el DBMS. En una base de datos distribuida, una red está involucrada, la cual es un medio en el que se implantan los requerimientos de seguridad. También es bueno saber que hay serios problemas para mantener una seguridad adecuada sobre las computadoras de la red. Estos problemas de seguridad en los sistemas de bases de datos distribuidas son por naturaleza más complicados que en los sistemas centralizados.

Dificultad de cambio: Muchas organizaciones han invertido grandes recursos en sus sistemas de bases de datos, las cuales no son distribuidas. Desafortunadamente no existen herramientas o metodologías que ayuden a los usuarios de las bases de datos centralizadas a convertirlas a una DDBS. Investigaciones en bases de datos heterogéneas e integración de bases de datos esperan superar estas dificultades.

Áreas de problema

Existen varios problemas técnicos que tienen que ser resueltos antes de que los beneficios potenciales de un DDBMS pueda ser implantado.

Diseño de bases de datos distribuidas: La cuestión es establecer la forma en que la base de datos y las aplicaciones van a ser distribuidos a lo largo de la red, y ser colocados para su ejecución. Hay dos alternativas básicas para la colocación de los datos: particionar (o no replicado) y replicar. En el esquema de partición la base de datos es dividida en un cierto número de particiones cada una de las cuales es colocada en un sitio diferente. Los diseños replicados pueden ser *completamente replicados* (también llamados completamente duplicados) cuando la base de datos completa es almacenada en cada sitio, o *replicada*

parcialmente (o parcialmente duplicada) cuando cada partición de la base de datos es almacenada en más de un sitio, pero no en todos los sitios. El segundo diseño fundamental utilizado es la fragmentación, la separación de la base de datos en particiones llamadas *fragmentos*, con la distribución óptima de los mismos.

Las investigaciones en esta área involucran principalmente la programación matemática para minimizar los costos combinados de la base de datos, el procesamiento de transacciones y la comunicación. En general es un problema muy difícil, y las soluciones propuestas están basadas heurísticamente.

Procesamiento distribuido de ***query***: El procesamiento de ***query*** es inseparable del diseño de algoritmos que analizan los ***queries*** y los convierten en una serie de operaciones de manipulación de datos. El problema está en cómo se decide la estrategia para ejecutar cada ***query*** sobre la red por el camino de menor costo. Los factores a ser considerados son la distribución de los datos, costo de comunicaciones, y la falta de disponibilidad en la información local. El objetivo es la optimización cuando el paralelismo inherente es utilizado para proporcionar la eficiencia de ejecución de la transacción que está sujeta a la restricción antes mencionada.

Administración distribuida de directorios: Un directorio contiene información (tal como descripciones y localizaciones) acerca de los elementos de datos en la base de datos. Un directorio puede ser global para todo el DBMS o local para cada sitio. Esto puede ser centralizado en algún sitio; y puede ser una sola o múltiples copias.

Control de concurrencia distribuido: El control de concurrencia involucra el acceso sincronizado a la base de datos distribuida, tal que la integridad de la base de datos se mantenga. Este es sin duda uno de los problemas más ampliamente estudiados en el campo de los DBMSs. El problema del control de concurrencia en un contexto distribuido es bastante diferente que en un marco centralizado. No solo existe la preocupación acerca de la integridad de una sola base de datos, sino también acerca de la consistencia de múltiples copias de la base de datos. La condición que requiere cualquier unidad de datos para todos los valores en múltiples copias, es tener la cobertura del mismo valor, lo cual es llamado "consistencia mutua".

Las alternativas de solución a este problema se clasifican de forma general en dos clases, la primera es la PESIMISTA, sincroniza la ejecución de los requerimientos de usuario antes de comenzar la ejecución, y la OPTIMISTA, ejecuta los requerimientos y entonces checa si la ejecución compromete la consistencia de la base de datos.

Administración distribuida de puntos muertos: El problema de puntos muertos en DBMSs es similar en naturaleza a los encontrados en sistemas operativos. La competencia entre los usuarios para acceder a la asignación de recursos (datos , en este caso) puede resultar en un punto muerto si el mecanismo de sincronización se basa en el bloqueo. Las alternativas de prevención, evasión y detección/recuperación también son aplicadas a los DDBSs.

Fiabilidad de los DBMSs distribuidos: Anteriormente se mencionó que una de las ventajas potenciales de los sistemas distribuidos es el obtener fiabilidad y disponibilidad. Pero esto no es algo que se da de forma automática. Y es importante ya que la medida de la consistencia de la base de datos depende de los mecanismos que son proporcionados, tal como la detección de fallas y su recuperación.

Un DDBS implica que cuando una falla ocurre y varios sitios llegan a ser inoperables o inaccesibles, la base de datos de los sitios en operación permanecen consistentes y con los datos accesibles.

Cuando el sistema de cómputo o la red se recupera de las fallas, el DDBS debe ser apto para recuperarse y regresar el acceso a los datos en los sitios de falla. Esto puede ser especialmente difícil en el caso de que se particione la red (cuando los sitios son divididos en dos o más grupos sin comunicación entre ellos).

Soporte del sistema operativo: La implantación de sistemas de bases de datos distribuidas bajo un sistema operativo convencional sufre de embotellamientos lo que repercute en la eficiencia. El soporte proporcionado por sistemas operativos para la operación de la base de datos no corresponde propiamente a los requerimientos del software administrador de la base de datos. El principal problema relacionado con el sistema operativo se da en los sistemas de un solo procesador los cuales administran en memoria el sistema de archivos y los métodos de acceso, la recuperación de caídas y la administración de procesos. En ambientes distribuidos existe el problema adicional de tener que tratar con múltiples capas de software de red. El problema en esta área es encontrar soluciones a la dicotomía de proporcionar un soporte adecuado y simple, para la operación de bases de datos distribuidas, tal que el sistema operativo proporcione un soporte general para otras aplicaciones.

Bases de datos heterogéneas: Cuando no hay homogeneidad entre los diferentes sitios de la base de datos en términos de la forma en que los datos están estructurados lógicamente (modelo de datos) o en términos de los mecanismos proporcionados para su acceso (lenguaje de datos), esto llega a ser necesario para proporcionar un mecanismo de traslado entre sistemas de la base de datos. El término heterogéneo por lo regular se introduce si se construye un DBMS distribuido de un cierto número de DBMSs centralizados. En este caso el problema es mucho más general que heterogéneo. De hecho estos sistemas pueden ser llamados Sistemas de Bases de Datos Múltiple, que pueden ser consideradas de forma complementaria en los DBMSs distribuidos.

Relación entre problemas: Todos estos problemas no están aislados unos de otros. Cada problema puede ser afectado por la solución encontrada a otros problemas. La relación entre los diferentes componentes se muestra en la siguiente figura II.1.

El diseño de bases de datos distribuidas afecta muchas áreas. Esto afecta la administración del directorio, porque la definición de fragmentos y su colocación determina el contenido del directorio (o directorios) tanto como la estrategia que puede ser empleada para administrarlo. La misma información (la estructura de los fragmentos y la colocación) es utilizada por el

procesador de ***query*** para determinar la estrategia de evaluación del ***query***. Por otro lado, los patrones de acceso y uso están determinados por el procesador de ***query***, los cuales son usados como entradas para los algoritmos de distribución de datos y fragmentación. Similarmente la colocación del directorio y contenido tiene influencia sobre el procesamiento del ***query***.

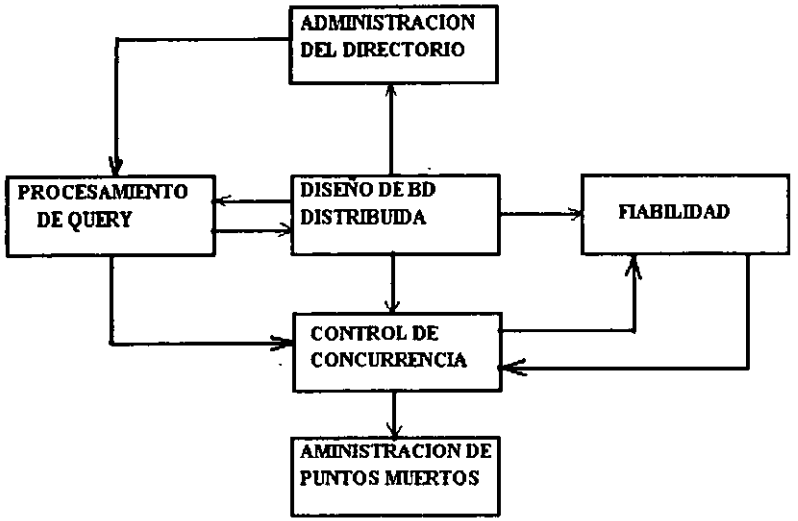


Figura II.1 Relación entre los temas investigados.

Cuando los fragmentos replicados son distribuidos afectan la estrategia del control de concurrencia que puede ser empleada. Algunos algoritmos de control de concurrencia no pueden ser fácilmente utilizados con bases de datos replicadas. Similarmente, los patrones de acceso y uso de la base de datos pueden influenciar los algoritmos de control de concurrencia.

Existe una fuerte relación entre los problemas de control de concurrencia, problemas de administración de puntos muertos y fiabilidad. Todos estos son llamados en su conjunto "problemas de administración de transacciones". Los algoritmos de control de concurrencia que son empleados determinarán si o no son requeridas las facilidades de administrar los puntos muertos de forma separada.

Mecanismos de fiabilidad son implantados dentro de los algoritmos de control de concurrencia. Por lo tanto, la relación entre estos dos aspectos se explica obviamente. También puede ser mencionado que los mecanismos de fiabilidad que son considerados tienen un efecto sobre la selección del algoritmo de control de concurrencia. Técnicas que proporcionan fiabilidad también hacen uso de la información de localización de los datos, desde la existencia de copias replicadas del servidor de datos, como una garantía para mantener una operación fiable.

Dos de los problemas no son ilustrados en la figura, sistema operativo y bases de datos heterogéneas, y no es porque no tengan relación con los demás aspectos discutidos. El tipo de sistema operativo utilizado y las características de soporte es lo que determina que el sistema operativo puede tener gran influencia sobre qué estrategias de solución pueden ser aplicadas en alguna de las otras áreas de problema. Similarmente, la naturaleza de todos estos problemas cambia considerablemente cuando el ambiente es heterogéneo. Los mismos temas tienen que ser tratados de forma diferente cuando la arquitectura de la máquina, el sistema operativo y el software del administrador de la base de datos local varía de sitio a sitio.

II.1.1 Transparencia y sincronización.

Transparencia significa que cuando un usuario accede a los datos no necesitan conocer dónde están almacenados, en qué formato están almacenados, y cómo van a ser accedidos. Ellos sólo necesitan una simple consulta, y no necesitan saber los detalles de presentación.

La transparencia de la red se relaciona, en algún modo, a la autonomía local. La transparencia de la red es el grado hasta el cual los usuarios del sistema pueden ignorar los detalles del sistema distribuido. La autonomía local es el grado hasta el cual el diseñador o administrador de una localidad puede ser independiente del resto del sistema distribuido.

Los temas de transparencia y autonomía son considerados desde los siguientes puntos de vista:

- Nombre de los datos
- Repetición de los datos
- Fragmentación de los datos
- Localización de los fragmentos y copias

En un ambiente de bases de datos distribuidas, algunos datos tienen que ser replicados por varias razones. Una de las principales razones es para proporcionar una buena eficiencia al acceder los datos localmente, en lugar de tener la transparencia de datos localizados en varios nodos. Otra razón para almacenar los mismos datos en varios lugares, puede ser disponer de los mismos datos en múltiples nodos por si algún nodo que contiene datos no funciona. La replicación de datos minimiza la dependencia de uno y sólo un nodo. Pero viéndolo desde otra perspectiva el inconveniente que se presenta cuando se replican los datos, es cuando se modifican, lo que representa el principal problema de sincronización.

Los datos pueden ser distribuidos lo cual significa que algunos datos pueden estar en un sitio y otros datos en otros sitios. Los datos pueden ser particionados, particionar significa que datos relacionados lógicamente pueden ser divididos en varios lugares. Esto también puede ser replicado. Múltiples copias pueden ser almacenadas en varios lugares. Partición y replicación deben ser utilizadas sólo si la organización lo amerita.
Una base de datos distribuida primero tiene que soportar la arquitectura del negocio, la arquitectura de los datos y la arquitectura de la red. Si el control organizacional es distribuido, esto posibilita a proporcionar operaciones continuas a pesar de la falla de algunos nodos y de algunas conexiones de comunicación.

La replicación, copias exactas de la base de datos para el uso local. Estas copias son algunas veces llamadas "snapshots". Los Snapshots son de sólo lectura, los cuales intentan proporcionar una copia estática de la base de datos. Los snapshots no se pueden modificar cuando la base de datos es modificada. Por lo que éstos pueden ser actualizados periódicamente recopiando las relaciones base. Los snapshots pueden ser de gran ayuda cuando se requiere dar una rápida respuesta a cierto tipo de aplicaciones que no necesitan que la información sea exacta. Un snapshot es similar a ejecutar una consulta SQL. Otras veces la partición llamada fragmentación, se refiere a la práctica de particionar o separar relaciones de datos (tablas) con base en el contenido relevante de la información. La PARTICIÓN HORIZONTAL permite que una base de datos sea particionada entre varios administradores, teniendo que cada administrador sólo recibirá los datos referentes a lo que emplean bajo su supervisión. La PARTICIÓN VERTICAL permite a los usuarios tener una visión lógica de la base de datos para su área de responsabilidad, tal como datos financieros o de mercadotecnia.

Una de las principales misiones del ambiente de bases de datos distribuidas es proporcionar el balance entre no almacenar los mismos datos redundantemente en varios nodos, y al mismo tiempo, también proporcionar una adecuada eficiencia para satisfacer individualmente.

Cuando se habla de una base de datos distribuida se implica que la sincronización de los datos se hará automáticamente mediante software. Ciertos tipos de organizaciones son estructuralmente opuestas a un ambiente de bases de datos distribuidas. Y dichos negocios tienen un control centralizado, por lo que no pueden ser inmersos en un ambiente de bases de datos distribuidas.

La principal causa de moverse a un ambiente de bases de datos distribuidas es la " IMAGEN DE UN SOLO SISTEMA". Esta es la meta que debe ser lograda, por lo que los usuarios no deben de saber dónde están almacenados los datos, cómo son accedidos, y cuál es su formato.

Para lograr la imagen de un solo sistema, los dos criterios importantes son: SITIOS AUTÓNOMOS y UN IMPACTO MÍNIMO EN LA EFICIENCIA. Si se logra que la eficiencia sea mejor que antes de la implantación de la base de datos distribuida, considerando esto cuando todas las fases del nuevo sistema sean integradas y trabajen de forma completa, y aún así se mantiene la eficiencia, se tendrán mas posibilidades de éxito.

II.1.2 Requerimientos de soporte

Uno de los requerimientos de soporte en el ambiente de bases de datos distribuidas, es la localización que se les asignará a los datos. Un sistema de DIRECTORIO/DICCIONARIO DE DATOS (DD/DS - DATA DICTIONARY/DIRECTORY SYSTEM- por sus siglas en inglés) es un sistema que es diseñado para tener un soporte adecuado con la centralización lógica de los datos acerca de los datos (metadatos). Y esto no es sólo para almacenar los metadatos (en el diccionario de datos), eso también es para proporcionar la información de cruces de referencias sobre los metadatos (en el directorio). El diccionario proporciona información acerca de qué es cada dato y lo que significa (lógicamente); el directorio proporciona

información de dónde puede ser localizado físicamente y cómo puede ser accedido. Esto es algo que se tiene automáticamente, ya que es una función que soporta el administrador de datos (DA - DATA ADMINISTRATOR, por sus siglas en inglés).

El ambiente de bases de datos distribuidas no es la solución al problema de mantenimiento de la integridad de los datos, y ésta puede ser aún peor. Ésta es una cuestión de ruteo y definición de los elementos de datos. En un ambiente de bases de datos distribuidas, muchos procesos acceden a la base de datos, y como consecuencia, uno de los requerimientos de mayor soporte del ambiente de bases de datos distribuidas es la consistencia en la definición de los niveles de seguridad.

Implantar niveles de seguridad y autorización definidos, puede llegar a ser una tarea desafiante. La seguridad es una de las cuestiones en las que se debe tener mayor control en los ambientes de bases de datos distribuidas.

REQUERIMIENTOS DE SOPORTE

- El significado de la localización de los datos
- Métodos de definición y ruteo de los elementos de datos
- Control para asegurar consistencia en la definición de los niveles en la base de datos mientras se permite el acceso concurrente de múltiples procesos
- Definición de niveles de seguridad y chequeo de autorización en la base de datos distribuida

Soporte de procesamiento

Para presentar una sola imagen del sistema, algunos aspectos de cómputo tienen que ser manejados de forma transparente, tales aspectos son: transparencia de los ***queries***, transparencia de modificación, transparencia de los ***views*** y transparencia en la ejecución.

Transparencia de ***query*** significa que un usuario puede ser capaz de acceder un ***query*** de la misma forma no importando desde qué lugar se esté ejecutando. Transparencia de modificación significa lo mismo que transparencia en ***query***, pero la diferencia reside en que en lugar de manejar ***queries***, el usuario desea modificar las tablas.

La modificación y borrado requieren un mecanismo de bloqueo. Algunos DBMSs no permiten a un usuario leer los registros que están siendo modificados por otro usuario. Algunos otros permiten a los usuarios ver los datos de los archivos en modificación, pero presentando los datos anteriores. Los mecanismos que no permiten ver el registro de la base de datos mientras está siendo modificada o borrada se llaman "bloqueos" (locking). El bloqueo se puede hacer a toda la base de datos, a páginas de memoria que tienen ciertos registros o campos de registros de la base de datos. Esto es llamado "granularidad " de bloqueo.

Un "bloqueo muerto" es una situación en la que dos o más unidades compiten por los mismos recursos, y ninguno puede proceder porque cada unidad está esperando que las otras desistan de los recursos que solicitan. Cuando un bloqueo muerto es detectado, algunos DBMSs abortan la transacción. En suma, un bloqueo puede ser un requerimiento para modificar o borrar. Ésta puede ser de "no lectura y sólo modificación" o también puede ser "múltiple lectura y una sola modificación". No lectura y sólo modificación significa que cuando alguien

está modificando un registro de una base de datos, nadie puede leerlo. Algunas veces se le ha llamado "modificación exclusiva". Múltiple lectura y una sola modificación significa que mientras alguien esta modificando un registro, varias personas pueden leerlo. Existen muchas variaciones de bloqueos y recuperación de bloqueo.

II.1.2.1 Restauración

En un ambiente de bases de datos distribuidas se debe tener un método para restaurar la base de datos dentro de un estado de consistencia ante todo tipo de fallas. La consistencia es muy importante, es la medida de la integridad y la eficiencia de los datos.

Una transacción es una secuencia de pasos que constituye una buena definición en la actividad de negocios. Una transacción puede ser en forma específica una consulta (***query***), o ser el resultado de crear, borrar o modificar registros en la base de datos.

Otra definición de transacción es la siguiente " Una transacción es una unidad de programa cuya ejecución conserva la consistencia de una base de datos. Una transacción debe ejecutarse de una forma atómica. Es decir, se ejecutan completamente todas las instrucciones de la transacción, o no se ejecuta ninguna".

Cuando se tiene un sistema de base de datos distribuido, es mucho más difícil garantizar la propiedad de atomicidad de una transacción. Esto se debe a que es posible que participen varias localidades en la ejecución de una transacción. El fallo de una de estas localidades o el fallo de la línea de comunicación entre ellas, puede dar como resultado un cálculo erróneo.

La función del gestor de transacciones de un sistema de bases de datos distribuido es asegurar que la ejecución de las distintas transacciones de un sistema distribuido conserven la atomicidad. Cada localidad cuenta con su propio gestor de transacciones. Los distintos gestores de transacciones cooperan para ejecutar las transacciones globales. Para comprender cómo puede estructurarse un gestor de este tipo, se define un modelo abstracto para un sistema de transacciones. Cada localidad del sistema contiene los siguientes subsistemas :

- Gestor de transacciones, cuya función es gestionar la ejecución de aquellas transacciones (o subtransacciones) que accedan a datos almacenados en esa localidad. Se observa que cada transacción puede ser bien una transacción local (es decir, que se ejecuta sólo en una localidad), o parte de una transacción global (es decir, que se ejecuta en varias localidades).

- Coordinador de transacciones, cuya función es la de coordinar la ejecución de varias transacciones (tanto local como global) iniciadas en la localidad.

La estructura de un gestor de transacciones es similar en muchos aspectos a la que se utiliza en el caso de un sistema centralizado. Cada gestor de transacciones se encarga de:

- Mantener una bitácora para la recuperación.

- Participar en un esquema de control de concurrencia apropiado para coordinar la ejecución en paralelo de las transacciones que se ejecuten en esa localidad.

El subsistema coordinador de transacciones no se necesita en el sistema centralizado, ya que las transacciones tienen acceso a datos de una sola localidad. Como su nombre indica, un coordinador de transacciones se encarga de coordinar todas las transacciones que se inicien en esa localidad. Para cada una de estas transacciones, el coordinador debe:

- Iniciar la ejecución de la transacción.
- Dividir la transacción en varias subtransacciones, las cuales se han de distribuir entre las localidades apropiadas para su ejecución.
- Coordinar la terminación de la transacción, ya sea que quede ejecutada o abortada en todas las localidades.

En el ambiente de procesamiento de transacciones, una transacción puede ser abortada en algún nodo, alguno de los componentes de la base de datos distribuida puede fallar, o alguno de los nodos físicos puede funcionar mal. Para lo cual, se debe de tener la capacidad de restaurar el hardware y los medios de comunicación (en alguna porción de la base de datos distribuida). La capacidad de restaurar una base de datos depende del estado de consistencia, por lo tanto un sistema administrador de bases de datos distribuidas (DDBMS-DISTRIBUTED DATABASED MANAGEMENT SYSTEM por sus siglas en inglés) debe suministrar algunas funciones; tales como componentes de comunicación y herramientas de bases de datos. Desafortunadamente, la mayoría de los DDBMS son especializados en el desarrollo de Software, pero no son lo suficientemente especializados en redes y telecomunicaciones. Una organización debe estar segura de contar con gente experta en ello.

EL MÉTODO DE RESTAURACIÓN DE LA BASE DE DATOS DEBE SER CONSISTENTE ANTE CUALQUIER FALLA, INCLUYENDO LOS SIGUIENTES:

- El proceso en la transacción de algún nodo
- Alguno de los componentes en la base de datos distribuida
- Alguno de los nodos
- El hardware de alguno de los nodos independientes
- Alguna porción del medio de comunicación
- El trabajo de comunicación entre los nodos por medio de las transacciones
- Pérdida de mensajes
- Fragmentación de la red

Para que un sistema sea robusto, es necesario que detecte cualquiera de los fallos anteriores, que reconfigure el sistema de manera que pueda reanudar el proceso, y que se recupere una vez que haya sido reparada la línea de comunicación afectada.

II.1.2.2 Optimización

La clave para mantener la imagen de un solo sistema consiste en proporcionar un tiempo de respuesta óptimo, para lo cual es necesario implantar técnicas de optimización. Por lo tanto, se asume que se está haciendo referencia a una base de datos relacional, y que los productos

DDBMS están basados en la tecnología relacional. También se asume que muchas relaciones bajo un DDBMS necesitan ser implantadas; y adicionalmente se asume que muchas relaciones son replicadas y particionadas, por lo que algunas relaciones tienen que ser almacenadas separadamente.

Los resultados pueden ser obtenidos mediante la combinación de relaciones localizadas en varios lugares. Pueden ser muchos lugares de los cuales los datos son extraídos. La localización es determinada por las necesidades, y la localización puede cambiar en el transcurso del tiempo.

Todas las copias deben contener los mismos valores para que el DDBMS pueda acceder a alguna copia. Esto es realizado con un algoritmo de optimización. Los algoritmos de optimización pueden determinar la ruta que debe ser tomada para la transmisión, dependiendo del tráfico en la línea de comunicación, la cual sigue la lógica de los mecanismos de conmutación de los sistemas de ruteo de llamadas telefónicas. Si alguna copia cercana llega a ser inaccesible por alguna razón, ésta puede ser acertadamente modificada hasta que nuevamente es accesible. Esto se debe a que las modificaciones son hechas en los datos de un lugar, y los mismos datos son almacenados en otros lugares, pero estas modificaciones no son realizadas instantáneamente. Los datos serán actualizados hasta que el sistema lo realice.

Una pregunta interesante es: ¿Las modificaciones se deben hacer inmediatamente, o las modificaciones pueden esperar al final del día o el fin de semana? Lo cual significa que dependiendo de las necesidades se pueden tomar diferentes estrategias de modificación. El DDBMS no puede saber mágicamente las estrategias de modificación, por lo que se debe hacer el análisis e implantar la estrategia almacenándola en el DDBMS.

Para tener bajo control las modificaciones, es necesario no tener más copias (de los datos) de las que son necesarias. El número adecuado de copias ha de ser determinado por los diseñadores y los implantadores de la base de datos.

OPTIMIZACIÓN EN BASES DE DATOS DISTRIBUIDAS

- Sitios de unión
- Orden de unión
- Métodos de unión
- Métodos de sincronización

II.1.2.3 Diccionario/Directorio de Datos

Las dos formas de comunicación del Diccionario/Directorio son la interfaz con la gente y con el software. La interfaz con la gente se traslada a cierto número de reportes impresos. El área que necesita soporte con la ayuda de Diccionario/Directorio de datos, son los Sistemas de Información Ejecutiva (EIS) o Sistema de Soporte de Decisiones (DSS). Otras personas que necesitan la interfaz con DD/D son los administradores de la base de datos (DBAs), los administradores de datos (DAs) y los auditores de procesamiento de datos. Las funciones del administrador de datos es que la política funcione en un lugar centralizado.

Un sistema de Diccionario/Directorio de Datos es diseñado para soportar la centralización lógica de los datos acerca de los datos (metadatos). Esto no sólo tiene la capacidad de almacenar los metadatos (diccionario de datos), también tiene la capacidad de proporcionar información de referencia acerca de los metadatos (directorio). El diccionario proporciona información acerca de qué dato es y qué significa (lógicamente); el directorio proporciona información acerca de dónde están localizados físicamente los datos y cómo pueden ser accedidos. Ésta es una facilidad automatizada que soportan las funciones de administración de datos (DA).

Un Diccionario/Directorio de Datos puede soportar la fragmentación horizontal y vertical, o hasta la replicación de datos. El Diccionario de Datos en si mismo es una base de datos o varias bases de datos que pueden estar constituidas de muchas relaciones. Una relación puede ser almacenada localmente, o también en alguna otra base de datos, y los datos pueden ser visualizados localmente o globalmente. Un Diccionario de Datos puede tener la facilidad de ser tanto local como global.

Un Diccionario de Datos Local almacena información acerca de los datos, aplicaciones, usuarios, medidas de seguridad, etc., localmente; y se tiene la información acerca de los nodo en donde los datos del diccionario local también son almacenados.

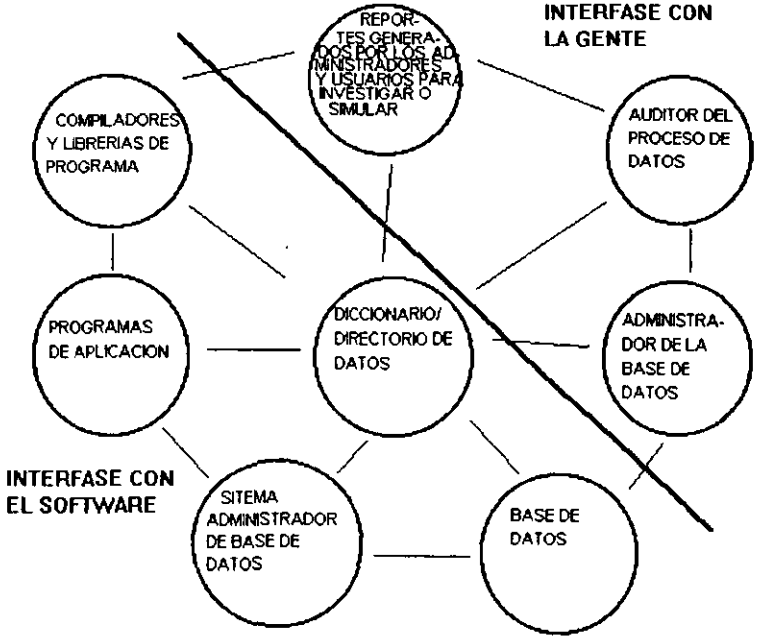


Figura II.2 DIRECTORIO/DICCIONARIO DE DATOS

Un Diccionario de Datos Global almacena información acerca de los datos, aplicaciones, usuarios, medidas de seguridad, etc., pertenecientes a un número de nodos remotos.

La premisa para que la capacidad que tiene un DDBMS de soportar un Diccionario/Directorio de datos resulte, está en que los diseñadores tienen que modelar la base de datos tal que ésta presente la facilidad de almacenar información acerca de los nodos remotos y que se pueda modificar fácilmente dicha información.

Los DDBMS soportan la distribución de datos, lo cual implica el soporte de datos almacenados en tablas que son dispersas, replicadas o particionadas. La combinación de éstas también puede existir. Esto significa que los datos particionados pueden también ser replicados. Varias combinaciones de partición y replicación pueden ser implantadas con la ayuda del DD/D porque el DD/D guarda la localización de los datos.

Un Diccionario es comúnmente usado por la gente. La gente necesita saber acerca de la localización de los datos tanto como la accesibilidad de los mismos. Un directorio de datos es usado por los sistemas para el acceso óptimo a los objetos de datos (Ver figura II.2).

Convenciones de Nombramiento

El problema de los nombres en los sistemas distribuidos consiste en que se permite compartir los datos bajo excesivas restricciones para los nombres que eligen los usuarios. Para sitios autónomos, no es deseable estar dentro de un sistema global de nombres, ya que los usuarios evitan las complicaciones de este tipo en su sistema. De otra forma, agregando una base de datos definida en algún sitio para que previamente la red establezca las reglas, evitando así el problema de renombrar.

Se tienen algunos tipos de convenciones de nombres. Y las convenciones de nombres usualmente permanecen como guías y no siempre son implementadas religiosamente. Una sugerencia al esquema de nombres puede ser:

USER_ID USER_SITE.OBJECT_NAME BIRT_SITE

Los usuarios solo proporcionan OBJECT_NAME, y el sistema proporciona el resto de los componentes, USER_ID es conocido por el sistema que tiene "clave activa" para entrar al sistema. USE_SITE es conocido por el sistema porque los usuarios están usando el sistema en cierto sitio. BIRTH_SITE puede ser agregado también como un objeto, para que dicho OBJETO proporcione el tiempo en que fue creado por primera vez.

CATÁLOGO es un término utilizado en los DBMS relacionales. Un catálogo es un directorio de todos los datos en la base de datos. En bases de datos distribuidas, los catálogos pueden contener la localización de cada fragmento de la base de datos. Un amplio sistema de directorio puede ser llamado catálogo. Para un acceso rápido al contenido del catálogo, alguna parte del catálogo puede ser almacenado en el buffer, lo cual es parte de la memoria real.

El sistema puede localizar los objetos con la ayuda de un amplio sistema de nombres. Un DBMS puede implantar esta capacidad para relaciones locales (accesibles desde un solo sitio) y globales (accesible desde varios sitios). El nombre de todas las relaciones globales son

almacenadas en todos los sitios. La creación de relaciones globales involucra la emisión de estos nombres (y localizaciones) a todos los sitios de la red. Por lo cual, un catálogo global puede restringir sitios autónomos y complicar el desarrollo del sistema.

En un ambiente distribuido, el catálogo identifica cada sitio guardando y manteniendo información sobre cada objeto de la base de datos, incluyendo réplicas o fragmentos almacenados en cada sitio. El catálogo es el sitio base donde se guarda la información de los objetos indicando donde están almacenados los objetos, y se modifica si un objeto es movido. La catalogación de los objetos se hace de una manera totalmente distribuida para preservar los sitios autónomos. Un objeto puede ser localizado por el sistema desde el Sistema Amplio de Nombramientos, y la centralización no será necesaria.

Para introducir un objeto en el catálogo, las siguientes asignaciones deben ser introducidas: el nombre que se le dará al objeto en el sistema, tipo de objeto, formato del objeto, rutas de acceso disponibles para acceder al objeto, mapeo en el caso de que dicho objeto tenga niveles más bajos de objetos, y varias estadísticas que asistan la optimización de los ***queries***.

II.1.2.3.1 DD/D para la optimización de consultas

EL DD/D puede contener información del número de opciones para la optimización de consulta. La rapidez en el enlace de las comunicaciones puede habilitar al diccionario para determinar cuáles enlaces de comunicación se usarán. La cantidad de datos en cada base de datos ayudará para determinar a cuál base de datos ir. El área de diseñadores de la base de datos podría hacer una inspección, además de tener un inventario de las aplicaciones existentes que actúan sobre la base de los datos, con lo cual se puede determinar el patrón de uso. No todos los datos son igualmente creados. En algunos ambientes los datos más recientes son los datos que se usan más frecuentemente, y la vieja regla 80-20 se cumple (veinte por ciento de los datos se usan el 80 por ciento del tiempo).

Los datos más frecuentemente usados pueden ser replicados. Comenzar a clasificar los datos de la organización y hacer cinco clases de datos. Cinco no es el número mágico, y una organización puede tener dos o tres clases. Para determinar cuáles datos serán replicados es necesario responder a las siguientes preguntas: ¿Cuál es la frecuencia de uso de los datos replicados? ¿Cuál es la rapidez de cada máquina en la red? Estos son algunos de los criterios que pueden ser almacenados en el DD/D, junto con los elementos de datos, entidades y procesos. Los algoritmos de optimización pueden esquematizar, basándose en esta información, la selección de estrategias en la toma de decisiones.

Los DDBMS proporcionan administración de transacciones, administración de entregas, detección de puntos muertos (antes de que estos ocurran), recuperación del sistema, y nombramiento de objetos.

Los DDBMS también proporcionan administración de catálogos, y el chequeo de autorización tiene que ser tratado más seriamente en un ambiente de DDBMS. Pero la seguridad y el chequeo de autorización requerirá algo de tiempo, lo cual puede llegar a ser costoso, y

probablemente resultará en la degradación de la eficiencia. El llegar a determinar cuánta es la seguridad y el chequeo de autorización suficiente para tener una adecuada eficiencia, puede llegar a ser un acto de malabarismo.

OPTIMIZACIÓN DE CONSULTAS CON LA AYUDA DEL DD/D

LO QUE EL DD/D DEBE CONTENER PARA LA OPTIMIZACIÓN DE CONSULTAS :

- Rapidez en los enlaces de comunicación
- Cantidad de datos en cada sitio de la base de datos
- Cuáles datos son replicados y en dónde
- Frecuencia de uso de los datos replicados
- Rapidez de cada máquina en la red

UN DDBMS DEBE PROPORCIONAR:

- Administración de transacciones
- Procesamiento de entregas
- Detección de puntos muertos
- Recuperación del sistema
- Nombramiento de objetos
- Administración de catálogos
- Chequeo de la autorización

II.1.2.3.2 Fragmentación

El estado de la red tiene que ser transparente, y no se necesita un programa de aplicación para conocer qué enlaces en el ambiente distribuido están disponibles en un momento. En un ambiente distribuido, el tener transparencia y la imagen de un solo sistema es de suma importancia ya que hasta las aplicaciones de la base de datos y las redes están involucradas. La imagen de un solo sistema significa proporcionar la impresión de que todos los datos se localizan en un solo lugar. La imagen de un solo sistema puede ser implantado a todos los niveles, tal efecto evita a los usuarios la carga de escoger la ruta disponible que sea más óptima para acceder a los datos. Cierto número de estrategias pueden ser implantadas para minimizar el impacto de los problemas en la red. Enlaces redundantes, replicación de datos y respaldo de los archivos del sistema, todo contribuye para la efectividad del sistema. Si un enlace falla, la redundancia de enlaces proporciona alternativas de uso posibles. Si por alguna razón ciertos datos no se pueden acceder, los datos almacenados de forma redundante pueden ser usados, en el mismo nodo o en otros nodos. Y en casos extremos, si el almacenamiento redundante de datos no es accesible, se pueden restaurar los archivos de respaldo.

Pero los DDBMS también pueden considerar factores humanos, en el caso de alguna degradación en la eficiencia del sistema el DDBMS podría informar al usuario. Esta cuestión puede ser una parte de los algoritmos de optimización.

Los factores humanos son más importantes porque los usuarios no tienen la misma formación que los administradores del sistema, lo cual no facilita responder las razones por las cuales hay retraso en el acceso a la información.

SOPORTE EN LA DISTRIBUCIÓN DE DATOS

ADMINISTRACIÓN DE UNA BASE DE DATOS DISTRIBUIDA:

- Fragmentación horizontal
- Fragmentación vertical
- Replicación
- Diccionario de datos global
- Diccionario de datos local
- Directorio de datos global
- Directorio de datos local

Una de las facilidades más requeridas en un DDBMS es la capacidad de distribución de datos a lo largo de la red. La distribución de datos está basada en la premisa de que no todos los datos son creados de igual forma. Esto significa que no todos los datos son utilizados con la misma frecuencia. Puede haber muchas diferentes opciones para separar una unidad de datos, este proceso es conocido como fragmentación. La fragmentación de datos es fácil con relaciones. Una relación es una tabla de dos dimensiones con renglones y columnas. Los datos pueden ser físicamente almacenados como una relación o varias relaciones. Las relaciones pueden consistir de millares de registros de datos, y cada relación puede contener 30, 40 ó 50 campos. Los campos son columnas y los registros son renglones.

Fragmentación horizontal

La fragmentación horizontal está basada en tener almacenados registros completos de la base de datos en un solo lugar. Lo que significa que algunos renglones son almacenados en un sitio, y algunos otros pueden ser almacenados en otro lugar físico de acuerdo al mismo criterio de separación.

Fragmentación vertical

Asumiendo que fueran 50 campos en una relación. De esos 50 campos 4, 5 ó 6 campos son utilizados con mayor frecuencia que todos los demás campos. Para separar una relación de forma vertical algunos campos tienen que ser almacenados en un lugar y algunos en otros lugares diferentes.

Al fragmentar los datos verticalmente, la fragmentación tiene que llevarse a cabo sin pérdida de información. Si una relación es separada de forma horizontal, ésta no es sensible a la pérdida de datos porque los registros completos son almacenados en cada sitio. Pero la fragmentación vertical es más vulnerable porque la separación de la relación con algunos campos en un lugar y algunos otros campos en otros lugares diferentes, de acuerdo al mismo criterio de separación de columnas. Y la relación original tiene que ser obtenida de los fragmentos verticales.

La fragmentación vertical es más difícil de administrar que la fragmentación horizontal. Éste es el porque más DDBMS soportan la fragmentación horizontal y no la vertical.

II.1.2.3.3 Replicación

Copias de las tablas pueden estar almacenadas de forma redundante, y la garantía de que sean iguales en cualquier momento, depende del ambiente en que se encuentren. Éste es el mayor problema de implantación debido a la sincronización que requiere. Para evitar los problemas de sincronización, los datos pueden ser almacenados en un solo lugar, pero la eficiencia puede ser un problema significativo cuando hay necesidad de acceder a los mismos datos desde varios lugares. Para una mayor eficiencia los datos tienen que ser replicados, pero la replicación trae consigo el problema de la sincronización. Los protocolos de entrega **dos-fases** o **tres-fases** soportados por los DDBMSs, son necesarios para proporcionar sincronización en la modificación de datos.

Entre los más comunes y más ampliamente usados está el compromiso de dos-fases. Otra alternativa es el protocolo de compromiso de tres-fases, el cual impide ciertas desventajas del compromiso de dos-fases, pero añade complejidad y tiempo extra.

Entrega dos-fases es un protocolo que es usado para asegurar la entrega uniforme de una transacción o abortarla entre dos o más participantes (administradores de transacciones y administradores de bases de datos). El protocolo consiste de dos fases: la primera, el alcanzar una decisión común; y la segunda, la implantación de dicha decisión. Un participante es definido como coordinador y administra la ejecución del protocolo "entrega dos-fases".

Para garantizar la atomicidad, es preciso que todas las localidades en las que se haya ejecutado la transacción T coincidan con el resultado final de la ejecución. T debe quedar ejecutada o abortada en todas las localidades. Para garantizar esta propiedad, el coordinador de transacciones encargado de T debe de ejecutar un **protocolo de compromiso**.

Compromiso de dos-fases

Sea T una transacción que se inició en la localidad L, y sea C el coordinador de transacciones de esa localidad. Cuando T termina de ejecutarse, es decir, cuando todas las localidades en las que se ejecutó T informan a C que T llegó a su término, C inicia el protocolo de compromiso de dos-fases. En la fase 1: C añade el registro <preparar T> a la bitácora y la graba en memoria estable. Una vez hecho esto envía un mensaje de <preparar T> a todas las localidades en las que se ejecutó T. Al recibir el mensaje, el gestor de transacciones de cada una de esas localidades determina si está dispuesto a ejecutar la parte de T que le correspondió. Si no está dispuesto, éste añade un registro <no T> a la bitácora y a continuación enviará un mensaje abortar T a C. Si la respuesta es afirmativa agregará un registro <T lista> a la bitácora y grabará todos los registros de bitácora que corresponden a T en memoria estable. Una vez hecho esto, responderá a C con el mensaje <T lista>. Y en la fase 2: Una vez que todas las localidades hayan respondido al mensaje preparar T enviado a C, o después de un intervalo de tiempo, previamente especificado, C puede determinar si la transacción T puede ejecutarse o abortarse. Si C recibió el mensaje <T lista> de todas las localidades que participan, la transacción T puede ejecutarse. En caso contrario, la transacción T debe abortarse. Según haya sido el veredicto, se agregará un registro <ejecutar T> o <abortar T> a la bitácora y se grabará

en memoria estable. En este punto, el destino de la transacción se habrá sellado. A continuación, el coordinador enviará un mensaje <ejecutar T> o <abortar T> a todas las localidades participantes. Al recibir este mensaje, cada una de las localidades lo registra en la bitácora.

Compromiso de tres-fases

El protocolo de compromiso de tres-fases está diseñado para impedir la posibilidad de bloqueo en un caso restringido de los fallos posibles. El protocolo requiere que:

- No debe ocurrir una fragmentación de la red.
- . Debe haber al menos una localidad funcionando en cualquier punto.
- En cualquier punto, como máximo un número K de participantes puede caer simultáneamente (siendo K un parámetro que indica la resistencia del protocolo a fallos en localidades).

El protocolo alcanza esta propiedad de no-bloqueo añadiendo una fase extra, en la cual se toma una decisión preliminar sobre el destino de T. Como resultado de esta decisión, se pone en conocimiento de las localidades participantes cierta información, que permite tomar una decisión a pesar del fallo del coordinador.

De la misma forma, sea T una transacción iniciada en la localidad L y C el coordinador de transacciones en L. En la fase 1: esta fase es idéntica a la fase 1 del protocolo de compromiso de dos-fases. En la fase dos: Si C recibe un mensaje abortar T de una localidad participante, o si C no recibe respuesta dentro de un intervalo previamente especificado de una localidad participante, entonces C decide abortar T. La decisión de abortar está implantada de la misma forma que en el protocolo de compromiso de dos-fases. Si C recibe un mensaje <T lista> de cada localidad participante tomará la decisión preliminar de <preejecutar T>. La diferencia entre preejecutar y ejecutar radica en que T puede ser todavía abortado eventualmente. La decisión de preejecutar permite al coordinador informar a cada localidad participante que todas las localidades participantes están <listas>. C añade un registro <preejecutar T> a la bitácora y lo graba en un almacenamiento estable. De acuerdo a esto, C envía un mensaje preejecutar T a todas las localidades participantes. Cuando una localidad recibe un mensaje del coordinador (ya sea abortar T o preejecutar T), lo registra en su bitácora, grabando esta información en almacenamiento estable, y envía un mensaje de reconocimiento de T al coordinador. Y en la fase 3: esta fase se ejecuta sólo si la decisión tomada en la fase 2 fue de preejecutar. Después de que los mensajes de preejecutar T se han enviado a todas las localidades participantes, el coordinador debe esperar hasta que reciba al menos un número K de mensajes de reconocimiento a T. Siguiendo este proceso el coordinador toma una decisión de compromiso. Añade un registro <ejecutar T> en su bitácora y lo graba en un almacenamiento estable. De acuerdo a esto, C envía un mensaje ejecutar T a todas las localidades participantes. Cuando una localidad recibe el mensaje, lo registra en su bitácora.

Puesto que la fase 3 siempre induce una decisión de compromiso, parece que no es muy útil. El papel de la tercera fase toma importancia por la forma en que el protocolo de compromiso

de tres-fases maneja los fallos. A pesar de la posibilidad de bloqueo, el protocolo de compromiso de dos-fases es utilizado más comúnmente. La probabilidad de que en la práctica ocurra un bloqueo es normalmente lo suficientemente baja para que no esté justificado el coste extra que supone el compromiso de tres-fases. Además, el compromiso de tres-fases es muy vulnerable a la hora de enlazar fallos. Esta desventaja puede ser salvada con protocolos de nivel de red, pero de esta forma se aumenta el tiempo extra.

II.2 Procesamiento cooperativo

En un ambiente de procesamiento cooperativo, porciones de una aplicación se ejecutan en diferentes computadoras, y los datos pueden estar en diferentes instalaciones. Una de la arquitecturas utilizadas para implantar el procesamiento cooperativo es la arquitectura cliente/servidor. En la arquitectura cliente/servidor, las aplicaciones de usuario se ejecutan en uno o más computadoras cliente y trabajan cooperativamente con aplicaciones especializadas ejecutándose en uno o más nodos servidores.

El **Procesamiento Cooperativo** es una arquitectura en la cual dos o más computadoras comparten el procesamiento de un programa. Y esto con la posibilidad de tener recursos distribuidos; programas, archivos y bases de datos; a lo largo de la red. El procesamiento cooperativo puede permitir el acceso transparente a través de algún sistema tal que los usuarios no tienen que saber si los recursos son locales o remotos.

Algunos beneficios del procesamiento cooperativo son los siguientes:

- Lo que ve el usuario en la pantalla es el sistema, pero limitado a lo que le interesa.

- El uso de interfaces de usuario amigables tales como Windows, Open Look, New Wave, Motif o Presentation Manager, permiten al usuario de microcomputadora o workstation poder aprender a utilizar las aplicaciones fácilmente, y tener menos posibilidades de cometer errores.

- El procesamiento cooperativo puede reducir la cantidad de procesamiento ejecutado en el anfitrión o servidor.

- El trabajo es distribuido entre clientes y servidores. Usualmente los servidores son máquinas grandes y los clientes son máquinas pequeñas.

- El costo por unidad de trabajo realizado en el servidor es por lo regular más elevado que en el cliente.

II.2.1 Aspectos del procesamiento cooperativo

Existen varias alternativas para compartir un programa entre dos o más computadoras. Este incluye el **procesamiento front-end**, **procesamiento punto-a-punto** (peer-to-peer), **llamada a procedimientos remotos** (remote procedure call), y **arquitectura cliente/servidor**.
Procesamiento front-end

Un programa de PC puede ser escrito para ejecutarse en una aplicación anfitrión existente sin cambiar ningún código en la aplicación anfitrión. Esto se hace escribiendo código para PC que envía llamadas a la Interfaz del Programa de Aplicación residente (API).

En el procesamiento front-end, la aplicación anfitrión es ejecutada antes, mandando imágenes de pantalla para que se simule una terminal tonta. El programa de PC lee las imágenes de pantalla anfitrión para enviar llamadas al API. El programa de PC es el que se utiliza como soporte de la interfaz de usuario y edita los datos de entrada del usuario. Cuando el usuario a concluido el trabajo con el registro, el programa de PC mueve los datos desde la PC al anfitrión.

Procesamiento punto-a-punto

En una aplicación de procesamiento punto-a-punto, típicamente dos procesadores comparten la carga y ejecución de un programa. La mayor parte del tiempo uno de los procesadores sostiene la tarea de presentar la interfaz de usuario, mientras el otro procesador sostiene un mayor procesamiento intensivo o tareas intensivas de entrada/salida, tales como mantenimiento de la base de datos. El procesamiento punto-a-punto también es utilizado para tareas intensivas para el procesador, el cual es optimizado para dicho trabajo, ejemplos de esto son la arquitectura de procesamiento paralelo o coprocesador matemático.

Dependiendo del ambiente y la naturaleza del punto, el procesamiento punto-a-punto es relacionado con Llamas de Procedimientos Remotos (RPC), o la arquitectura cliente/servidor.

Llamada de procedimientos remotos (RPC)

En el mundo de redes UNIX y TCP/IP, las RPC son utilizadas para implantar el procesamiento punto-a-punto. Las RPC permiten a un programa o a una computadora llamar a una subrutina que se ejecuta en otra computadora de la red. Las herramientas RPC proporcionadas por la red o por el sistema operativo cuidan del trabajo de comunicación requerido para pasar la llamada del procedimiento a la computadora que ejecutará la subrutina.

La diferencia primaria entre la aplicación RPC y la más convencional aplicación punto-a-punto está en la sincronización. En una convencional aplicación punto-apunto, dos programas se ejecutan en computadoras separadas. Cuando uno de estos programas requiere datos o procesamiento de la otra computadora, se abre una sesión de comunicación y se lleva a cabo una conversación con la otra computadora. Y en algún punto durante la conversación, el programa que inició el enlace puede terminar y ejecutar algún otro proceso mientras el

programa llama eficientemente las funciones asignadas. La implantación de una aplicación punto-a-punto va estrechamente unido a una forma de procesamiento paralelo.

Con las llamadas RPC, si se da una llamada a un procedimiento local, el programa de llamada espera hasta que la rutina se ejecute. Cuando se reciben los datos o cuando el código regresa los resultados indicados del trabajo de la subrutina, el programa de llamada continúa procesando.

Arquitectura cliente/servidor

Los sistemas con capacidad de cómputo para bases de datos distribuidas no requieren necesariamente procesamiento cooperativo como un prerequisito. Sin embargo, el procesamiento cooperativo requiere algún tipo de ambiente de base de datos distribuida o por lo menos un acceso distribuido a los datos. Actualmente, las organizaciones saben si necesitan un acceso distribuido a los datos. Si una organización intenta implantar el procesamiento cooperativo sin proporcionar acceso a los datos desde diferentes plataformas, las ventajas proporcionadas por el procesamiento cooperativo pueden ser limitadas. Como en el procesamiento cooperativo se da el procesamiento distribuido, en el caso de convertir aplicaciones existentes al procesamiento cooperativo puede consumir mucho tiempo, porque un rediseño será obligatorio.

El procesamiento cooperativo significa desarrollar y ejecutar la aplicación en una plataforma óptima. Como podemos observar en la figura II.3. Para implantar una base de datos distribuida, el acceso distribuido a los datos, y el procesamiento cooperativo; la red tiene que existir de antemano.

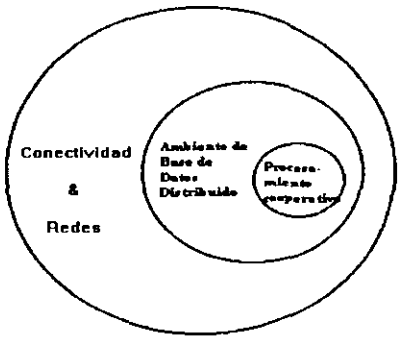


Figura II.3 El procesamiento cooperativo es una subasignación de un ambiente de base de datos distribuido, el cual es una subasignación de conectividad y redes.

En un ambiente de procesamiento cooperativo automatizado, el administrador del procesamiento cooperativo determina en cuáles plataformas una aplicación específica se puede ejecutar, como lo podemos ver en la figura II.4. La Interfaz del Programa de Aplicación (API) es utilizada para proporcionar el mismo "ver y sentir" para el usuario. No sólo pueden ser tres

plataformas para llevar a cabo el procesamiento cooperativo, pueden ser dos, cuatro o más plataformas.

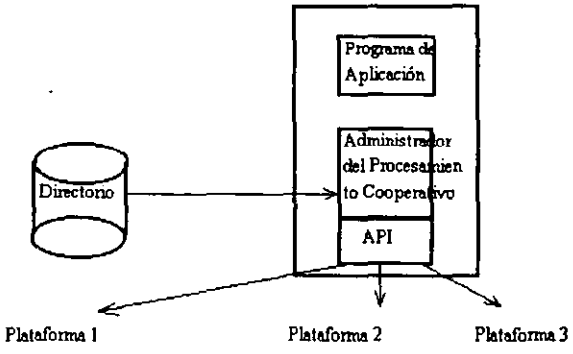


Figura II.4 Procesamiento Cooperativo

En el ambiente de bases de datos distribuidas el programa de aplicación es entregado al DBMS distribuido. El diccionario de datos "inteligente" determina dónde están los datos. Inteligencia significa tener la información acerca de dónde están almacenados los datos, cómo se acceden, quién puede utilizarlos, cuáles aplicaciones los utilizan, etc. El directorio inteligente proporciona información referente al acceso de los datos y las aplicaciones. El administrador cooperativo determina en cuáles plataformas una aplicación específica puede ejecutarse. Un ejemplo esquemático lo vemos en la figura II.5.

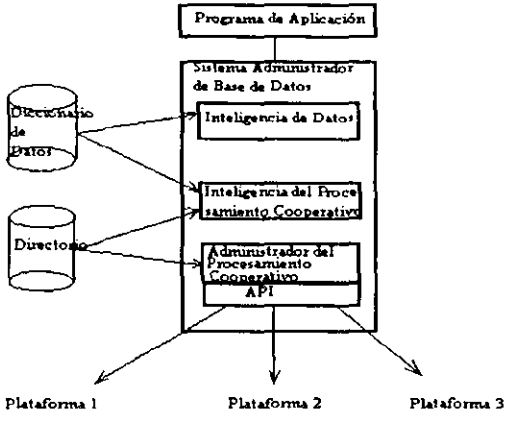


Figura II.5 Administración de Bases de Datos Distribuidas

El procesamiento cooperativo es implantado a través de múltiples niveles de computadoras, permitiendo una sección para las aplicaciones a ser direccionadas para el procesamiento; en la plataforma que mejor conviene financieramente, operacionalmente y organizacionalmente. El

procesamiento cooperativo hace referencia a un tipo de operación multicomputacional para obtener la manera efectiva y eficiente en la cual los procesos deben ser ejecutados en las mismas. Por ejemplo, la edición de alguna entrada puede realizarse en una microcomputadora en lugar de mandar esto a la mainframe. Por razones financieras existe una gran iniciativa de la gente al "downsize", y muchas compañías desean desarrollar aplicaciones "downsizing". Para lo cual, sería ideal si para las mainframes pudiera implantarse, pero un obstáculo para esto es el costo de la conversión.

En el procesamiento cooperativo, una mainframe puede ser un "super servidor" o un "switchbox". La mainframe es algunas veces referida como un servidor de archivos gigante porque puede soportar bases de datos enormes. Un servidor de archivos gigante también puede consistir de una asignación de minicomputadoras en "cluster".

II.2.2 Downsizing en el Procesamiento Cooperativo

Cuando se hace referencia al procesamiento cooperativo, en realidad se está hablando del downsizing. Downsizing es cuando muchas computadoras pequeñas reemplazan una o más computadoras grandes. Estas computadoras son comúnmente microcomputadoras. Pero obviamente una mainframe no puede ser reemplazada por una microcomputadora. Un cierto número de microcomputadoras son necesarias para reemplazar una mainframe, además es necesario que dichas microcomputadoras estén en red para poder tener grupos dinámicos.

El downsizing con la ayuda de las redes LAN, y la conexión de estas redes LAN a una red de área amplia (WAN), es cuando el poder del procesamiento distribuido toma lugar en las transacciones del negocio. En muchas organizaciones, el procesamiento cooperativo puede ser un prerrequisito para el downsizing porque muchas de las tareas de cómputo de "misión crítica" son hechas en computadoras mainframe. Si alguno de estos sistemas puede ser movido a microcomputadoras, hay una necesidad de coexistencia entre las mainframe y la microcomputadoras. Primero, de tener sólo terminales tontas, ahora la PC emula una terminal de mainframe, y de aquí la habilidad de las mainframe y PC de trabajar con los procesadores en paralelo. Ahora el procesamiento cooperativo permite a los dos procesadores ejecutar simultáneamente la misma aplicación, por lo que cada máquina realiza lo que hace mejor.

Downsizing típicamente involucra la descarga de procesamiento de una computadora anfitrión a microcomputadoras. Por lo que hay una variedad de razones para que se dé el downsizing. En primera, las PCs son más capaces para algunos trabajos, tal como proporcionar la interfaz de usuario. Moviendo otras tareas a la PC, tal como la validación y edición de datos, reduce la necesidad de comunicación de la PC con la mainframe, cuando los recursos que se encuentran en la mainframe también están disponibles en el disco duro de la PC.

En una definición más precisa "Downsizing es el proceso de migrar una aplicación completa, o algunas funciones de la aplicación, desde la mainframe centralizada a la red de pequeños sistemas descentralizados. Esto es implantado con tecnología de vanguardia, lo que resulta en reducción de costos, un acceso de usuario mejorado y gran flexibilidad.

Con el Downsizing, las organizaciones pueden mover las aplicaciones de grandes computadoras a pequeños sistemas. El downsizing implica una gran inversión, ya que es necesario comprar el hardware y software necesario. Y las organizaciones están poco dispuestas a gastar grandes recursos, aunque dicha inversión se vea recuperada o recompensada en pocos años.

Downsizing es el delegar tareas al procesamiento local en sitios que están hacia abajo en la escala organizacional. Los sistemas que son sometidos al downsizing reflejan de forma más exacta la estructura organizacional a comparación de los sistemas basados en mainframes. Una corporación es identificada como una colección de negocios individuales, en donde las decisiones son evaluadas desde la perspectiva del negocio. El downsizing se puede implantar utilizando la arquitectura cliente/servidor. Donde porciones de la aplicación se ejecutan en diferentes computadoras.

II.2.3 Estructura del Procesamiento Cooperativo

El procesamiento cooperativo es el fundamento y la fuerza que mueve a la arquitectura cliente/servidor. La característica distintiva de una aplicación con procesamiento cooperativo es el grado de interacción entre varios componentes de la aplicación (o fragmentos de la aplicación). En la arquitectura cliente/servidor estas interacciones son entre los requerimientos del cliente y las reacciones del servidor a estos requerimientos. Una orden para comprender estas interacciones, dejando ver los componentes generales de una aplicación. Una aplicación típica consiste de los siguientes componentes (Figura II.6):

- Procesamiento Lógico de la Presentación: esta es una parte del código de la aplicación que interactúa con un dispositivo que es la terminal del usuario final. La presentación lógica es ejecutada en tareas tales como el formato de la pantalla, lectura, y escritura de la información de pantalla, administración de pantallas, teclado y mouse.

- Procesamiento Lógico de Negocios: esta es una parte del código de la aplicación que utiliza las entradas de datos (de la pantalla y/o base de datos) para realizar las tareas del negocio. Típicamente este código es escrito por el usuario en alguno de los lenguajes de tercera generación que son soportados, o en lenguajes de cuarta generación de alto nivel (escrito por el usuario o producido por un generador de código).

Funciones de Administración de Datos:

- Procesamiento Lógico de Datos: esta es una parte del código de la aplicación que manipula los datos junto con la aplicación (Lenguaje de Manipulación de Datos - Data Management Language - DML). Los datos son administrados con un Sistema Administrador de la Base de Datos (DBMS). La manipulación de datos en un DBMS relacional es utilizado con el uso de algún dialecto del SQL (Structured Query Language). Los lenguajes de manipulación de datos SQL por lo regular son embebidos en lenguajes de tercera o cuarta generación.

- Procesamiento de la Base de Datos: esto es el actual procesamiento de la base de datos que es ejecutado por un DBMS. Idealmente, el procesamiento del DBMS es transparente para los negocios lógicos de una aplicación. Por lo que el procesamiento de datos es una parte esencial de las interacciones del procesamiento cooperativo, y puede ser considerado como un componente del procesamiento cooperativo de la aplicación.

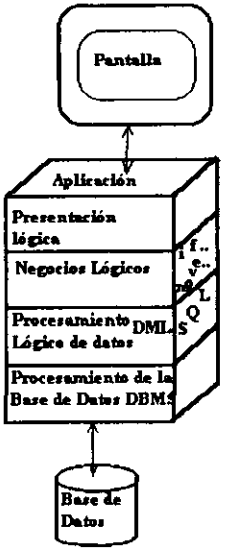


Figura: II.6 Componentes típicos de una aplicación.

La arquitectura cliente/servidor emplea el procesamiento cooperativo distribuido para:

- El procesamiento distribuido de los componentes de la aplicación entre clientes (típicamente la presentación y alguna parte de los negocios lógicos) y servidores (típicamente alguna parte de los negocios lógicos, la base de datos lógica y el DBMS).

- Soporte en la cohesión de las interacciones en un modo cooperativo entre los clientes y los servidores.

Una de las cuestiones de diseño de los sistemas cliente/servidor que siempre es cuestionada es el CÓMO se distribuyen los componentes de la aplicación entre clientes y servidores. En una arquitectura multinivel esta cuestión se extiende a la colocación de los componentes en los diferentes niveles. Mientras no exista un método universal para una apropiada distribución de los componentes de la aplicación, los siguientes puntos son algunas recomendaciones generales que dan los expertos (Figura II.7):

- En general el componente de la presentación lógica con las facilidades de entrada/salida de pantalla es colocada en el sistema cliente, y estos clientes son típicamente colocados en el nivel más bajo de un ambiente multinivel (PC's y Workstations).

- Dado el poder de las estaciones de trabajo clientes, y el hecho de que la presentación lógica reside en los sistemas cliente, hace factible colocar en el sistema cliente alguna parte de los negocios lógicos. Lo anterior se puede dar, para la parte de la lógica de la aplicación que se relacione con ediciones en pantalla, y tal vez algunas piezas de código que son específicas para un cliente en particular.

- Si el procesamiento lógico de la base de datos está embebido en los negocios lógicos, y si los clientes mantienen una baja interacción, entonces el procesamiento lógico de la base de datos (manipulación local de datos) puede ser colocado en el sistema cliente.

- Dado que los clientes están conectados a una LAN con un propósito común de grupo, y asumiendo que los grupos de trabajo comparten la base de datos, comparten fragmentos del procesamiento lógico de negocios y del procesamiento lógico de la base de datos y el propio DBMS, entonces estos mismos elementos pueden ser colocados en el servidor.

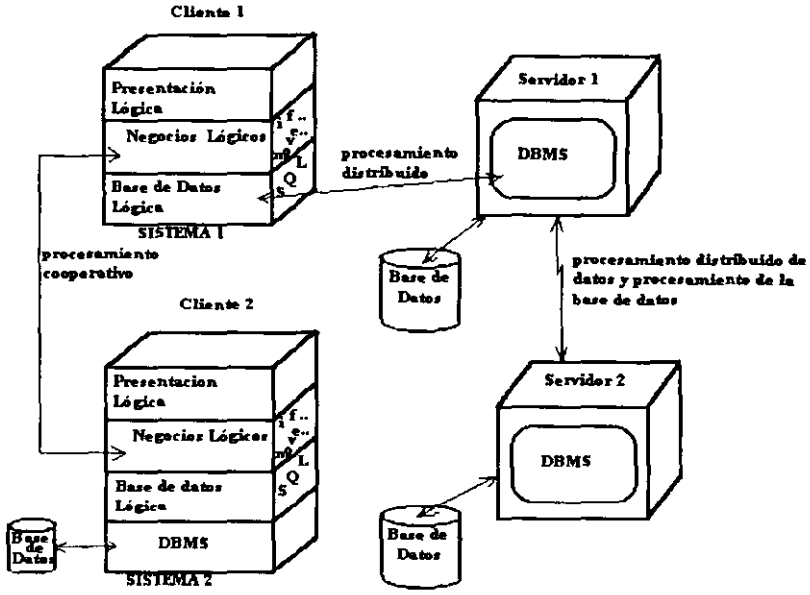


Figura II.7 Procesamiento Cliente/Servidor, Procesamiento Distribuido y Procesamiento Cooperativo.

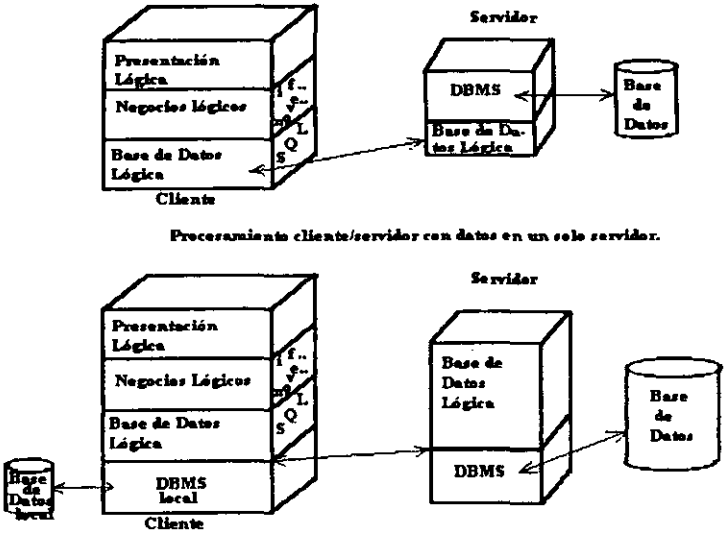


Figura II.8 Procesamiento Cliente/Servidor.

Principios similares pueden ser utilizados para decidir la colocación del componente de la aplicación cliente/servidor en un ambiente cooperativo multinivel. La colocación puede ser decidida por (Figura II.8):

- La cantidad de datos relevantes en una aplicación.
- El número de usuarios activos que ejecutan la aplicación utilizando los mismos datos.
- El número de interacciones entre varios componentes de la aplicación.
- Las características técnicas de la plataforma seleccionada para los clientes y los servidores.

Conceptualmente la arquitectura cliente/servidor puede ser definida como un caso especial de procesamiento cooperativo donde una aplicación es dividida (no necesariamente en porciones iguales) entre un sistema cliente y un sistema servidor. Tanto el componente cliente como el servidor están involucrados en el procesamiento de la operación, dado que los componentes de software interactúan unos con otros para ejecutar cooperativamente los funciones de la aplicación. Los componentes de hardware del cliente y del servidor son especializados para facilitar la cooperación de los componentes de software.

Para expander la definición de Arquitectura Cliente/Servidor, ésta no solo incluye el procesamiento cooperativo entre los componentes de software, sino también interaciones

cooperativas entre los componentes de hardware. Por lo que el término cliente/servidor obtiene dos significados, el primero, el procesamiento cooperativo entre los componentes de software del cliente y servidor; y el segundo significado es la relación entre el hardware del sistema cliente y el servidor.

Cuando los componentes de hardware son relegados, el procesamiento cooperativo entre los componentes de software es a veces llamado computación servidor-requeridor. Contrastando este modelo con el modelo de computo cliente/servidor, donde las especialización del hardware juega un papel importante. Ahora, por todo lo anterior se considera que el "Procesamiento Cooperativo" es el fundamento y la fuerza que mueve a la arquitectura cliente/servidor.

Algo que es muy importante notar, es que la separación de la aplicación lógica en los componentes de la aplicación, antes mencionados, no es sencilla; y el límite de la división entre cada uno de los componentes no es siempre claramente definido. Por lo que los desarrolladores de aplicaciones hacen uso de técnicas apropiadas de programación estructurada y de la ingeniería de software para ayudarse a la separación de estos componentes.

Ahora, dependiendo de la estructura de la aplicación particular son definidos los siguientes estilos de procesamiento cooperativo (figura II.9):

- Presentación Distribuida (Distributed Presentation - DP)
- Presentación Remota (Remote Presentation - RP)
- Negocios Lógicos Distribuidos (distributed Business Logic - DBL)
- Administración de Datos Distribuidos (Distributed Data Management DDM)
- Administración de Datos Remota (Remote Data Management - RDM)

Esta clasificación se ocupa de los estilos elementales del procesamiento cooperativo. Y sólo ciertas combinaciones en los estilos del procesamiento cooperativo son posibles o mejor dicho deseables. Se hace notar que la clasificación antes mencionada es considerada desde el punto de vista de la aplicación (las funciones del negocio). Esta es en realidad la razón principal de que la aplicación exista.

Por lo tanto el procesamiento cooperativo es apto para ser dividido de la siguiente forma:

- Poner de forma separada la presentación lógica, la administración de datos lógica y/o los negocios lógicos.
- Combinar y/o distribuir parte de la aplicación a través de múltiples sistemas.

En este contexto, los negocios lógicos no pueden ser considerados remotos. En realidad no pueden ser remotos pero pueden ser distribuidos entre dos o más plataformas. Cada uno de los estilos de procesamiento cooperativo definido anteriormente pueden ser implementados en

una arquitectura c/s. El modelo de cómputo c/s responde a cada estilo de procesamiento cooperativo que puede ser soportado por una o más técnicas del procesamiento cooperativo.

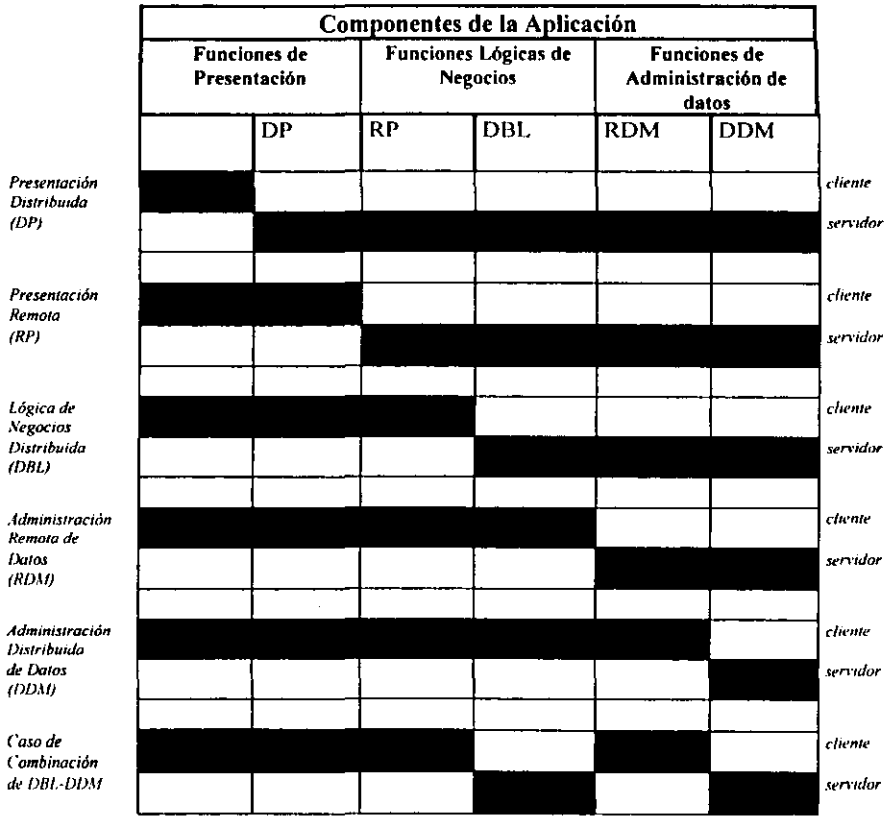


Figura II.9 Componentes de la Aplicación.

En las siguientes subsecciones veremos más a fondo los tipos de distribución más utilizados para cada uno de los elementos que componen la aplicación.

II.2.3.1 Presentación Lógica

Las funciones de presentación están directamente relacionadas con las entradas/salidas del usuario final. Las funciones de presentación lógica están diseñadas para interactuar con dispositivos del usuario final, tales como terminales basadas en carácter o workstations con capacidades gráficas. Las funciones de presentación efectúan tareas como el formateo de pantalla, administración del diálogo, lectura y escritura de la información de pantalla, administración de ventanas, teclado y mouse. Las funciones de presentación avanzadas

pueden soportar de forma general la edición de entradas, tipos de datos y validación de rango en los datos, edición de campos, contexto sensitivo a la ayuda, transcripción de sesiones, mensajes y control de acceso. En una arquitectura cliente/servidor las funciones de presentación tienden a ser distribuidas y por lo general se ejecutan en el sistema cliente. El alcance y capacidades de las funciones de presentación son enfocadas para la especialización del sistema cliente. El sistema **X Windows** es un ejemplo de la implementación de las funciones de presentación. Desde el punto de vista del procesamiento cooperativo existen dos estilos de distribuir las funciones de presentación de un sistema: **Presentación distribuida** y **Presentación Remota.**

Presentación Distribuida.

La Presentación Distribuida (PD) es cuando dividimos las funciones de presentación entre dos o más nodos de la red. Un modelo típico de presentación distribuida consiste de dos componentes el front-end y el back-end. El front-end reside en el nodo cliente y dicho componente se encarga de una parte de la interfaz de usuario que consiste del desplegado de pantalla, la interfaz gráfica de usuario, administración de ventanas, color, letras, mouse y teclado. El componente de back-end de la presentación reside en un nodo diferente, junto con el resto de la aplicación y está localizado en el servidor, como lo vemos ilustrado en la figura II.10. Dichos componentes comparten las funciones de presentación para ejecutar una tarea en común. Un ejemplo de PD es el Administrador de Presentación OS/2.

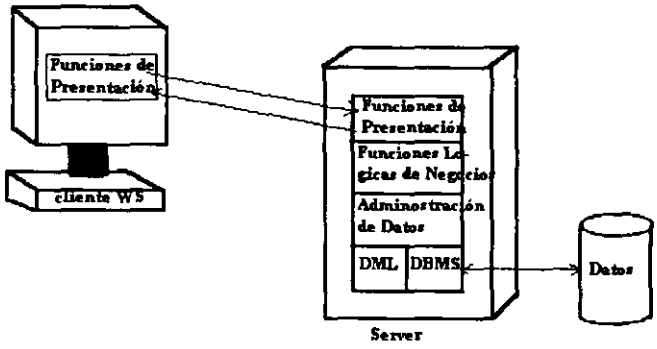


Figura II.10 Presentación Distribuida.

Presentación Remota

La Presentación Remota es cuando dividimos las funciones de presentación del resto de la aplicación y las colocamos en el nodo cliente, como lo vemos ilustrado en la figura II.11. Este

estilo de procesamiento cooperativo es por lo regular para aplicaciones poco convencionales donde las interacciones del usuario final son predeterminadas.

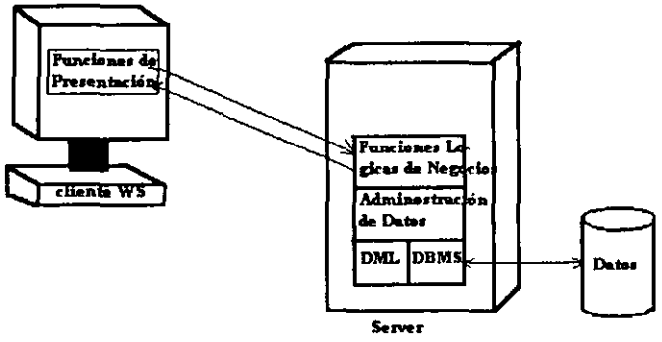


Figura II.11 Presentación Remota.

II.2.3.2 Procesamiento Lógico de Negocios

Las Funciones Lógicas de la Aplicación son diseñadas para satisfacer las necesidades del negocio y es la razón de que las aplicaciones existan. Las Funciones Lógicas del Negocio no pueden ser separadas del resto de la aplicación y colocarse de forma remota. Este caso sólo puede ser considerado cuando la red es pequeña, tiene baja actividad y acceso limitado de las aplicaciones.

Pero si nos vamos al otro extremo y se coloca toda la aplicación en un solo nodo, esto puede traer como consecuencia los siguientes problemas:

- Puede resultar en un gran embotellamiento cuando el volumen de transacciones es alto.
- El sistema se vuelve menos confiable porque se crea un solo punto de falla.
- Como los recursos de cómputo son distribuidos la utilización de la aplicación será muy utilizada, por lo que se dará un incremento importante en la carga de trabajo en el procesamiento de transacciones.

Por los puntos anteriores es planteada como una mejor opción la distribución de las funciones de negocios lógicos.

Funciones Distribuidas

Los Negocios Lógicos Distribuidos se dan cuando se dividen sus funciones entre dos o más nodos. Y la forma en que son divididas las funciones es entre el front-end y el back-end. El front-end es colocado en el sistema cliente y cumplirá la tarea de iniciar las interacciones, mientras que el back-end se coloca en el sistema servidor y tendrá como tarea la reacción a los requerimientos, como lo muestra la figura II.12.

En el nodo cliente tendremos las funciones de presentación adheridas a las funciones de negocios lógicos que les corresponde estar en el nodo. Y la otra parte de las funciones de negocios es colocada en el sistema servidor junto con el DBMS. Este tipo de procesamiento cooperativo es considerado como uno de los más difíciles de diseñar y desarrollar. Por lo regular, para la implementación de este tipo de procesamiento cooperativo es requerido un Sistema Administrador de Procesamiento de Transacciones Distribuidas (Distributed Transaction Processing -DTP- Management System- TPM). El cual garantice la integridad de los datos, la recuperación y la consistencia en una transacción dentro del ambiente complejo del Procesamiento de Transacciones Distribuidas.

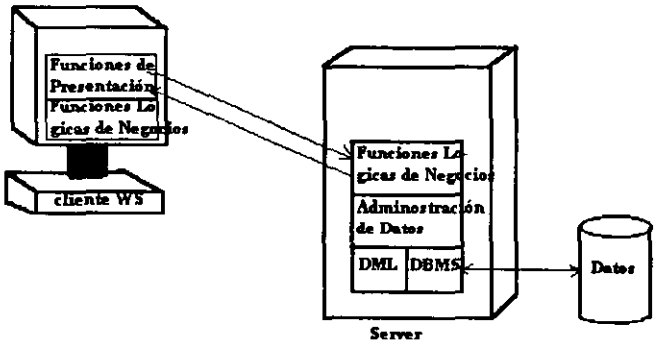


Figura II.12 Funciones de Negocios Lógicos Distribuidos.

II.2.3.3 Administración Lógica de Datos

De una forma general, la arquitectura de un ambiente distribuido cooperativo es caracterizado por los siguientes puntos:

- Los datos se distribuyen entre varios nodos del sistema.
- El procesamiento es distribuido entre varios nodos y es administrado por un Administrador de Transacciones Distribuidas (control de red, coordinación de procesos, sincronización y horario).
- El acceso distribuido a los datos es proporcionado por funciones multicapa como la aplicación, el lenguaje de la interfaz de usuario, controladores de entrada/salida de datos, diccionario de datos y directorio (catálogo).
- El sistema administrador de datos se distribuye junto con los datos (base de datos y sistema administrador de archivos).

El ambiente que satisface estos requerimientos es ilustrado en la figura II.13, el cual representa el punto de vista conceptual de "The American National Standards Institute (ANSI)" para el desarrollo de arquitecturas distribuidas. Los componentes del modelo ANSI también se pueden encontrar en el ambiente cliente/servidor, donde los datos pueden ser distribuidos entre varios servidores, y los datos locales pueden residir en las estaciones de trabajo clientes, aunque la terminología cliente/servidor es bastante diferente a la arquitectura ANSI.

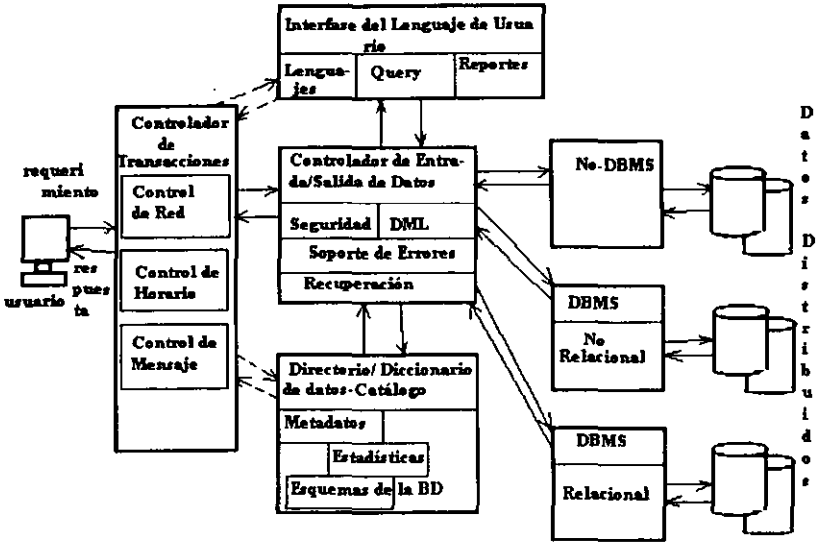


Figura II.13 Arquitectura Distribuida ANSI.

Los dos diferentes estilos de Administración de datos en un ambiente distribuido cooperativo pueden ser definidos como: Administración de Datos Remoto y Administración de Datos Distribuidos. Un estilo particular dependerá en si los datos son distribuidos entre diferentes nodos o son remotos (relativo a la aplicación lógica), y qué porción de la administración de datos lógica es distribuida (relativo a la aplicación de negocios lógicos).

Administración de Datos Remoto

Lo primero que hay que tomar en cuenta es que la Administración de Datos Lógica está compuesta de dos elementos importantes: El Procesamiento de Datos Lógico y el Procesamiento de la Base de Datos. El Procesamiento de Datos Lógico es una parte de código de la aplicación que manipula los datos conjuntamente con la aplicación. Y el Procesamiento de la Base de Datos que está relacionado directamente con el procesamiento necesario para responder a los requerimientos formulados a la base de datos. El Procesamiento de la Base de Datos puede residir junto con la base de datos, mientras que el Procesamiento de Datos Lógico (funciones del Lenguaje de Manipulación de datos-DML) puede ser colocado con el DBMS o estar embebido en la aplicación de negocios lógica.

Cuando los datos, el Procesamiento de Datos Lógico y el DBMS residen en un solo sistema es cuando se hace referencia a una Administración de Datos Remoto. Y como se ha referido en los estilos anteriores, tenemos dos componentes que son el front-end y el back-end. El front-end contiene una porción de negocios lógicos de la aplicación, mientras que el back-end

contiene la base de datos y la ejecución del DBMS. El RDM representa un ejemplo de una arquitectura cliente/servidor simple, la cual es ejemplificada con la figura II.14.
Si el RDM se implanta con una base de datos relacional, se podrá implementar dicha arquitectura con la técnica de Interacciones SQL Cliente/Servidor; o sino es relacional se podrían utilizar las RPC´s.

Ahora, la desventaja de este estilo de procesamiento cooperativo es la problemática que se presenta con los embotellamientos, la sobrecarga en los enlaces de comunicaciones, la creación de un solo punto de falla y el no tomar todas las ventajas de los recursos disponibles. Por lo anterior, la implantación del RDM es recomendable para aplicaciones que se caractericen por su bajo volumen, utilización infrecuente, queries de procesamiento dinámico (que se utilizan en Aplicaciones de Soporte de Decisiones) y cuya aplicación produzca resultados de bajo volumen.

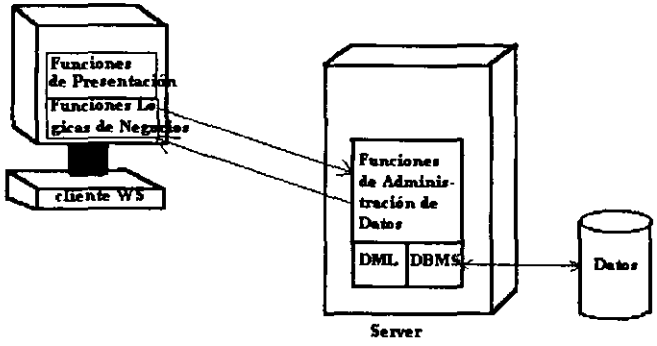


Figura II.14 Administración de Datos Remota.

Administración de Datos distribuidos

El DDM se introduce para eliminar los problemas producidos por el RDM. El DDM nos plantea la distribución de la base de datos entre varios nodos, lo cual proporciona un alto grado de fiabilidad en el sistema, ya que al tener la posibilidad de múltiples copias de los datos más críticos de la organización, con lo que se elimina la posibilidad de un solo punto de falla. Cuando los datos son distribuidos en varios nodos, también tenemos la alternativa de distribuir completamente o de forma parcial el Procesamiento de Datos Lógicos, acompañando cada porción de datos. Y de la misma forma una porción del Procesamiento de la Base de Datos puede residir en cada nodo que contenga una porción de la base de datos, como se ilustra en la figura II.15. Este estilo de procesamiento cooperativo se caracteriza por las siguientes características: 1) los datos y el DBMS son distribuidos entre múltiples nodos, incluyendo el nodo donde se encuentra la aplicación lógica; 2) las Funciones de Administración de Datos son distribuidos entre el front-end (procesamiento de datos lógico - DML) y el back-end (funciones de la base de datos - DBMS).

Esta arquitectura puede reducir el tráfico de la red, considerando que los requerimientos de acceso a los datos son enviados desde la aplicación de negocios lógico al procesamiento de datos lógico en el mismo nodo. El procesamiento de datos lógico realiza un chequeo inicial de la sintaxis, compila y determina la localización de los datos requeridos. Si los datos locales son utilizados, los requerimientos del DML son satisfechos sin tener que acceder a un servidor de base de datos remoto. Por lo que el DDM se adapta idealmente para ser implementado en una arquitectura cliente/servidor.

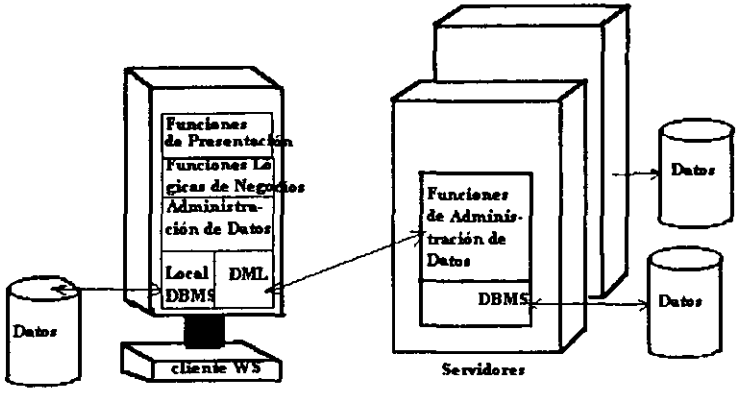


Figura II.15 Administración de Datos Distribuido.

II.2.4 Técnicas del Procesamiento Cooperativo

Teóricamente, hay tres tipos básicos de técnicas de comunicación para el procesamiento cooperativo que pueden ser utilizadas en la arquitectura cliente servidor:

- Pipes
- Llamadas a Procedimientos Remotos (RPC-Remote Procedure Call)
- Interacciones SQL cliente /servidor.

Pipes representa un mecanismo orientado a la conexión que pasa datos de un proceso a otro. Pipes es ampliamente usado en sistemas basados en UNIX. Primeramente, el proceso puede ser en diferentes máquinas, y aún ejecutarse bajo diferentes sistemas operativos.

Varias implementaciones de pipe pueden soportar uno o varios mecanismos de transporte concurrentes. Los detalles de los mecanismos de transporte soportados son ocultos para los usuarios y pipes impone a los usuarios algo mínimo en cuanto al protocolo y a restricciones de formateo. Básicamente, pipes proporciona facilidades para marcar los límites entre mensajes discretos, para determinar la identidad de quien envía, y para realizar la verificación de la recepción del mensaje. La implementación pipe varía desde una arquitectura muy simple hasta una compleja como el Advanced Program-to-Program Comunication (APPC), SNA de IBM y NFS de Suns; son ejemplos de mecanismos pipe.

Llamada de Procedimiento Remoto (Remote Procedure Call - RPC). Un RPC es un mecanismo mediante el cual un proceso puede ejecutar otro proceso (subrutina) que reside en un sistema diferente, por lo general remoto, el cual posiblemente corre bajo un sistema operativo diferente. Algunos parámetros necesarios para la subrutina son pasados entre el proceso original y la subrutina. Los detalles de los mecanismos de transporte utilizados por la RPC son ocultos para los usuarios. En específico una herramienta RPC puede soportar uno o varios mecanismos de transporte diferentes. El principal requerimiento para la implantación exitosa de RPC es la habilidad de la llamada para encontrar el servidor donde reside la subrutina. Un camino para lograr la búsqueda es identificar el nombre de la subrutina buscada en algún servidor de base de datos, y verificar que en realidad corresponde el nombre. Los accesos a la base de datos pueden ser modificadas tal que el servidor objetivo para la subrutina pueda ser asignado dinámicamente. Esta asignación puede cambiar de una invocación de RPC a otra. Las RPC imponen restricciones de formato a los usuarios. Algunos ejemplos de RPC son RPC de OSF, Netwise de Sun, OS/2 Remote Program Link (RPL), y RPC de Sybase.

Interacciones SQL Cliente/Servidor es un mecanismo para pasar los requerimientos SQL (Structured Query Language) y datos asociados de un proceso (usualmente el cliente) a otro proceso (servidor). SQL Cliente/Servidor es un caso especial de interacciones cliente/servidor, adecuado para Aplicaciones de Bases de Datos Relacionales Distribuidas. En este caso, el servidor es un servidor de bases de datos relacionales. La mayoría de los productos cliente/servidor a la fecha son basados en interacciones SQL cliente/servidor.

Como con Pipes y RPC, muchos mecanismos de transporte diferentes son soportados. Y mientras los detalles de los mecanismos de transporte están ocultos a los desarrolladores de aplicaciones, Interacciones SQL Cliente/Servidor impone a los usuarios varios protocolos y restricciones de formato. La sintaxis SQL, funcionalidad y formateo de datos, son la razón para estas restricciones.

Pipes, con un mecanismo orientado a la conexión, puede resolver una gran asignación de problemas de comunicación en el procesamiento cooperativo. RPC puede resolver sólo una parte de los problemas que resuelve Pipes. Por lo cual, la RPC es un mecanismo menos orientado a la conexión. Las conexiones RPC sólo existen en el lapso que dura la llamada. Las Interacciones SQL Cliente/Servidor representan un mecanismo orientado a la conexión similar a Pipes.

La desventaja de las interacciones SQL Cliente/Servidor es que por naturaleza está orientada a los datos. La arquitectura cliente/servidor que sólo implementa el mecanismo de interacciones SQL está limitado a aplicaciones de bases de datos relacionales (RDBMS).

Los requerimientos de la aplicación fuera del ámbito de una aplicación RDBMS no puede ser satisfecha mediante comunicaciones SQL Cliente/Servidor. Por lo que los sistemas abiertos cliente/servidor deben ser diseñados para tomar ventaja de varias técnicas de comunicaciones del procesamiento cooperativo que son requeridas para los negocios lógicos de la aplicación.

II.2.5 Distribución de los Elementos del Sistema

Con todo lo que hemos analizado en la sección anterior y en la presente, necesitamos constituir un solo sistema que forme una ***UNIDAD***. Lo que quiere decir que la interacción entre todos los elementos darán la ***IMAGEN DE UN SOLO SISTEMA.***

Éste es nuestro objetivo, conjuntar todo lo necesario de una forma adecuada, y así ***DISTRIBUIR*** los diferentes aspectos del sistema de una forma óptima, conforme a las particularidades que nos presenta una ***ORGANIZACIÓN*** en especial. Con lo visto en "Base de datos Distribuidas" y "Procesamiento Cooperativo" hemos abarcado los puntos necesarios para poder vislumbrar o clarificar las respuestas de las siguientes interrogantes:

¿QUÉ ES LA ESPECIALIZACIÓN DE LAS FUNCIONES?

¿CUÁL ES EL PAPEL QUE JUEGAN EL CLIENTE Y EL SERVIDOR EN LA ARQUITECTURA CLIENTE/SERVIDOR Y POR QUÉ DECIMOS QUE SE DA UN PROCESAMIENTO COOPERATIVO?

¿QUÉ PODEMOS DISTRIBUIR EN UNA ARQUITECTURA CLIENTE/SERVIDOR?

¿CÓMO PODEMOS CONTROLAR Y DISTRIBUIR LOS DATOS?

¿CON BASE EN QUÉ SE DETERMINA LA DISTRIBUCIÓN DEL SOFTWARE?

¿POR QUÉ SE PUEDEN DELEGAR PARTE DE LAS FUNCIONES DE ADMINISTRACIÓN DE FORMA LOCAL?

¿CUÁLES SON LOS ELEMENTOS DE UNA APLICACIÓN CLIENTE/SERVIDOR Y CÓMO PODEMOS DISTRIBUIRLOS?

Las respuestas a esto se entrelazan y se influyen una a otras, y esto se debe a que todas en su conjunto forman UNA ARQUITECTURA CLIENTE/SERVIDOR. A continuación se expondrá brevemente lo que responde a todas estas preguntas, las cuales de alguna forma ya se han visto más a fondo en secciones anteriores.

El aspecto clave o el punto de partida de la Arquitectura cliente/servidor es precisamente el determinar de forma clara cuáles son las tareas de cada uno de los elementos que conforman el sistema. Esta especialización afecta tanto hardware como al software. Pero esta especialización no es determinada como una regla inquebrantable, sino que pueden variar las funciones de cada elemento de un sistema a otro. Ya que esta distribución de tareas es de acuerdo a la "Arquitectura del Negocio" en que se esté

implantando la arquitectura cliente/servidor. Un ejemplo claro de la especialización de funciones lo tenemos en el hardware de una estación de trabajo, la cual tiene el poder de manejar una Interfaz Gráfica de Usuario y la Administración de ventanas entre otras cosas, contando con monitores de altísima resolución y poder de procesamiento. Con lo anterior, podemos decir que la especialización de funciones la podemos determinar de la manera que más convenga para la eficiencia óptima de nuestro sistema.

Entonces, tenemos la especialización de funciones que precisamente se dará entre el cliente y el servidor. El cliente es el elemento del sistema que tiene la FUNCIÓN DE INICIAR la comunicación con el servidor mediante un REQUERIMIENTO. Y de la misma forma, el servidor tiene la FUNCIÓN DE RESPONDER a los requerimientos del cliente. Y así estos dos elementos interactúan conjuntamente, para realizar y llevar a un buen término la satisfacción de los requerimientos, y esto es lo que llamamos una transacción. Entonces, el aspecto que caracteriza al modelo cliente/servidor es precisamente el procesamiento cooperativo entre los elementos del sistema, para así lograr de forma eficiente el objetivo de la empresa. Además de utilizar de la mejor forma posible todos los recursos de hardware y software disponibles.

También debemos tener claro cuáles son los principales elementos que están formando nuestro sistema y de qué forma interactúan unos con otros. De una u otra forma, nuestro sistema al estar inmerso en una arquitectura de este tipo, tiene la posibilidad de distribución. Los elementos o aspectos que podemos distribuir son los siguientes:

DATOS
CONTROL (Administración y software)
FUNCIONES (Elementos que constituyen la aplicación)
PROCESAMIENTO

Los DATOS son el elemento fundamental, es el motivo por el cual se desarrolla un sistema. El cual integra todos los datos de una organización y así obtiene la información necesaria para que funcione el negocio.

Algunos departamentos o áreas de la organización generan los datos, y otras hacen uso de estos datos. Pero de alguna forma todas las áreas necesitan información de una área ajena. Entonces, el objetivo es lograr que toda la información de la empresa tenga un ORIGEN ÚNICO. Lo que nos lleva a que no exista duplicidad ni redundancia que pueda provocar inconsistencias y falta de confiabilidad en la información. Ahora, como ya hemos visto en la sección de "bases de Datos Distribuidas ", tenemos varias formas o estrategias de distribuir nuestros datos a través de la red con que cuenta la empresa. Para tal objetivo, nuestra información puede

ser REPLICADA o PARTICIONADA en los nodos clientes. El uso de estas dos opciones va de acuerdo a las necesidades específicas de la empresa. Además también puede incluirse el uso de los *SNAPSHOTS*.

Pero al distribuir los datos también va implícito la distribución del control, por lo que la administración ya no se realiza desde un lugar centralizado, aunque sí existe una coordinación central. En cambio, en cada lugar donde exista una porción de los datos, también tendrán parte del software necesario que ayuda a la administración local. Lo anterior da un cierto grado de independencia al nodo cliente respecto al resto de la red. El software, es una porción del DDBMS que se coordina con el DDBMS principal que se encuentra en el servidor; el cual cuenta con un Diccionario/Directorio de Datos que tiene toda la información respecto a la estrategia de distribución. Y en consecuencia como ya lo mencioné, las funciones de procesamiento se dan de un modo distribuido entre los clientes y los servidores.

El medio por el que almacenamos y explotamos los datos es precisamente a través de una APLICACIÓN. Y dicha aplicación también la podemos distribuir dividiendo los elementos que la constituyen de acuerdo a sus funciones. Principalmente está constituida por tres componentes: 1) ***LA PRESENTACIÓN***, 2) ***PROCESAMIENTO DE NEGOCIOS LÓGICO*** Y 3) ***LA ADMINISTRACIÓN DE DATOS LÓGICA***.

La PRESENTACIÓN es la interfaz con el usuario, la forma en que interactúa el sistema con el mundo. La presentación puede ser DISTRIBUIDA o REMOTA, si la distribuimos el front-end de la aplicación se localiza en el nodo cliente y el back-end en el servidor. Y si la consideramos de forma remota toda la presentación se ubica en el nodo cliente.

EL PROCESAMIENTO DE NEGOCIOS LÓGICO puede ser DISTRIBUIDO o estará junto con el resto de la aplicación. Pero no puede ser remota, ya que sus funciones no pueden ser aisladas de la Presentación ni de la Administración de Datos Lógica. Ya que su función es ejecutar los procesos que requiere el negocio, mediante los datos proporcionados al sistema a través del nodo cliente.

La ADMINISTRACIÓN DE DATOS LÓGICA está constituida de dos elementos: ***EL PROCESAMIENTO DE DATOS LÓGICO*** y ***EL PROCESAMIENTO DE LA BASE DE DATOS***. El PROCESAMIENTO DE DATOS LÓGICO es el lenguaje mediante el cual la Presentación de la aplicación se comunica con la base de datos. Lo que es llamado el ***Lenguaje de Manipulación de Datos*** (DML -

Data Management Lenguage), que en las bases de datos relacionales es el SQL. El SQL puede estar por separado o estar embebido en el Procesamiento de Negocios Lógico.

El PROCESAMIENTO DE LA BASE DE DATOS es la otra parte que es sumamente importante, ya que está constituida por el DBMS. Que es el que controla todos los accesos a la base de datos, la sincronización de accesos concurrentes; y también lo que implica el mantener la INTEGRIDAD DE LA BASE DE DATOS entre otras tareas relevantes.

Las Funciones de Administración de Datos pueden ser DISTRIBUIDAS y REMOTAS. La decisión de Distribuirlas por lo general va guiada por la premisa de que la base de datos es también distribuida, y en cada nodo donde se encuentre una porción de la base de datos, requerirá el software necesario del DBMS, y así sincronizar las transacciones sobre la base de datos. El caso Remoto sólo se da cuando tenemos una base de datos que sólo tendrá su localización en el servidor, por lo que no es necesaria la distribución del software del DBMS en los nodos cliente.

Lo anterior, es en forma muy breve lo que hemos abarcado hasta el momento. Pero un punto que también es básico en la Arquitectura Cliente/Servidor es la ***PLATAFORMA*** en la cual se implantará EL SISTEMA. La selección del software de desarrollo y el hardware apropiado es una tarea que no se puede olvidar. Por lo que la siguiente sección se enfocará a dar una rápida revisión sobre lo que implica la selección de la plataforma para una ARQUITECTURA CLIENTE/SERVIDOR.

II.3 Sistemas Cliente/Servidor

Un sistema cliente/servidor es implantado dependiendo de las plataformas del front-ent y back-end en que se ejecutará, y el grado en que el procesamiento se dividirá entre dos. Para mostrar una clasificación de las categorías en que se dividen los sistemas cliente/servidor se muestra la siguiente tabla II.16, no todos los diferentes sistemas cliente/servidor caerán específicamente en alguna categoría, pero podemos tener un panorama general.

Como ya se ha mencionado, la principal ventaja de un sistema cliente/servidor radica en que se divide el procesamiento de la base de datos entre el sistema cliente y el servidor. Esta división de trabajo también reduce la carga de las estaciones de trabajo que se conectan a la red. Esto se debe a que el servidor en lugar de enviar archivos completos a través de la red, reduce el tráfico debido a que el servidor sólo manda *queries* como respuesta a los requerimientos. Otro beneficio de la separación del cliente y el servidor, es que se pueden tener estaciones de trabajo independientes; usuarios que no están limitados a un tipo de sistema o plataforma. Y una de las mejores ventajas de los sistemas cliente/servidor es la preservación de la integridad de datos. La mayoría de los servidores de bases de datos ejecutan DBMSs que están basados en el modelo relacional, y los usuarios están exentos de accesos a los datos desde DBMSs externos.

Ahora analizando las desventajas, la más notable es que en los sistemas cliente/servidor se incrementa el costo administrativo y el soporte de personal, quienes mantendrán el servidor de la base de datos. También hay un incremento en el costo del hardware, además del incremento en el nivel de complejidad del sistema. La ventaja de la independencia de las aplicaciones c/s tienen sus inconvenientes, al tener múltiples front-ends se incrementa el nivel de las necesidades de soporte de programación, porque más código y variedad de programas deben de ser desarrollados y mantenidos. Hacer un cambio en la estructura de la base de datos también tiene un efecto sobre los diferentes front-ends. También se tienen limitaciones de soporte por la interconectividad entre diferentes sistemas DBMS cliente/servidor.

Antes de tomar la decisión de implantar un sistema de base de datos cliente/servidor, los expertos en esta materia recomiendan se analicen ciertos aspectos fundamentales para la toma de decisiones. En primera, pensar si se tiene o se puede contratar asesoría especializada para llevar a cabo el proyecto, lo que es crítico debido a que se tiene que decidir si se capacita a la gente que se tiene o contratar asesoría externa, esto puede implicar tiempo y dinero. Este punto no se puede omitir porque es la base de un buen sistema.

Por lo anterior, y considerando todo lo que puede implicar a la empresa. Es preciso plantearse si en realidad se necesita un sistema c/s. Algunos puntos de referencia que son recomendados por los expertos, son los siguientes:

- Si se necesita un acceso multiusuario, ¿Cuántos usuarios intervienen? si la respuesta es menos de 20. Entonces, contar con una LAN y un DBMS multiusuario basado en PCs es más que suficiente. En caso de no contar con una LAN, se puede considerar un pequeño sistema operativo multiusuario (ejemplos de éste serían DOS multiusuario, OS/2 multiusuario o UNIX) que se ejecute en una PC de alto poder o en una RISC Workstation/Minicomputadora. Y las PCs con que se cuente pueden ser las terminales, y si en el futuro se necesita alguna expansión se pueden incorporar otras PCs.

- Ahora en el caso de que se tengan más de 20 usuarios, se debe determinar el tipo de acceso que tendrán los usuarios. Si la mayoría de los usuarios solo consultarán la información y una mínima parte de los usuario modificara la base de datos, entonces un tradicional DBMS multiusuario puede proporcionar una aceptable eficiencia para un rango de 32 a 40 usuarios. Pero si todos los usuarios necesitan derechos para modificar los datos, entonces será necesario el poder multiusuario y la integridad de la base de datos c/s. El límite de 20 usuarios es algo arbitrario, ya que se han realizado estudios que demuestran que la eficiencia de los diferentes DBMSs basados en PC varía dependiendo del número de usuarios.

Tabla II.16 Categorías de sistemas y plataformas Cliente/Servidor ("Guide to client/server database", Joe Salemi).

CATEGORÍA	DESCRIPCIÓN	COMENTARIOS
Clase 1: Procesamiento Completamente Distribuido	♦ Los datos residen en sistemas y/o plataformas múltiples. ♦ El acceso de los usuarios es transparente (los usuarios se conectan a un servidor , el cual accede a otros sistemas). ♦ Eficiencia del servidor en todas las funciones del DBMS y en el procesamiento. ♦ Los usuarios no pueden acceder a los datos si están fuera del DBMS que se ejecuta en el servidor. ♦ Múltiples frons-ends proporcionan *queries*, modificación de datos y servicio de reportes.	Implantación muy limitada para los datos.
Clase 2: Completamente cliente/servidor	♦ Los datos residen en uno o más servidores. ♦ El usuario o la aplicación hace conexiones explícitas a cada servidor. ♦ Eficiencia del servidor para todos los procesos del DBMS. ♦ Los usuarios sólo pueden acceder los datos mediante el DBMS que se ejecuta en el servidor. ♦ Múltiples frons-ends proporcionan *queries*, modificación de datos y servicio de reportes.	Es el tipo de sistema cliente/servidor más común.
Clase 3: Cliente/Servidor con compuertas	♦ Sistemas de ***gateway*** y aplicaciones para crear un puente entre los usuarios de la aplicación front-end y el DBMS que se ejecuta en el sistema que no es cliente/servidor. ♦ Sistemas de ***gateway*** para trasladar *queries*, datos modificados, etc; mediante procedimientos y llamadas al sistema de base de datos que puede procesar. ♦ El ***gateway*** soporta múltiples front-ends.	Comúnmente usado entre sistemas basados en PC y DBMSs que se ejecutan en una Mainframe o Minicomputadora, o como un enlace entre un sistema de clase 2 y una Mainframe o Minicomputadora.
Clase 4: Cliente/Servidor limitado	♦ El servidor proporciona algunas funciones del DBMS, comúnmente el almacenamiento de datos y funciones de indexación. ♦ El servidor no siempre previene de usuarios que acceden los datos desde DBMSs que se ejecutan fuera del servidor. ♦ Gran parte del procesamiento se lleva a cabo en el sistema cliente. ♦ Soporta múltiples front-ends, no comprendiéndose de la misma forma que en los sistemas de la clase 2.	Los sistemas de este tipo agregan funcionalidad al servidor para trabajar con estándares de bases de datos basados en PC, usando comúnmente archivos de formato dBase DBF.
Clase 5: Cliente/Servidor propietario	♦ Requiere una plataforma de hardware y sistema operativo propietario. ♦ Los datos sólo pueden ser accedidos mediante el software del front-end proporcionado por el distribuidor del DBMS.	Se utilizó a principios de los 80's; recientemente se ha ido desarrollando al incorporarse a sistemas abiertos.

Pero ahora si se toma la decisión de implantar un sistema cliente/servidor, se tienen que ver las posibilidades de aprovechamiento de la plataforma con que se cuenta o en caso necesario reemplazar lo existente.

Ahora, si se va ha implantar un sistema con la arquitectura cliente/servidor, es necesario que los siguientes aspectos sean cubiertos:

- Una interfaz gráfica de usuario amigable debe residir en el cliente.
- Una porción significativa , o toda la aplicación lógica reside en el cliente.
- El sistema cliente/servidor cuenta con capacidades de red.
- El cliente y el servidor son distinguibles uno del otro, sin embargo ellos pueden interactuar estrechamente.
- El cliente y el servidor pueden trabajar en plataformas de cómputo diferentes, pero también pueden trabajar bajo la misma plataforma de cómputo.
- El servidor es capaz de servir múltiples clientes de forma concurrente.
- Por lo general el cliente es el que inicia las acciones. Pero alguna acción en particular puede ser programada mediante un *trigger* en el servidor de la base de datos.
- El servidor de la base de datos puede proporcionar protección de datos, seguridad, restauración, recuperación y capacidades SQL.
- Si el hardware de la plataforma necesita ser modificado en el cliente o en el servidor, no es necesario modificar ambos.

II.3.1 Plataformas cliente/servidor

La ***plataforma*** es la combinación de hardware y software en que se ejecuta el DBMS cliente/servidor. Son cuatro las categorías de plataformas: PCs, UNIX (usualmente RISC) Workstation, Minicomputadoras y Mainframes, mientras que la plataforma mas común para el computo cliente/servidor es la PC. Cada una de las plataformas tienen sus ventajas y sus desventajas.

El Sistema Operativo (OS - Operating System) es el software primario que actúa como interfaz entre el hardware y las aplicaciones de software que se ejecutan en el hardware. La característica primaria del Sistema Operativo necesaria para un DBMS c/s es que sea **multitarea** (multitasking), lo que significa ejecutar múltiples tareas concurrentemente. Los sistemas operativos multitarea pueden ser **preemptive** o **nonpreemptive**. En un sistema preemptive, los controles del sistema operativo asigna tiempo del CPU para cada tarea, en contraste en un sistema nonpreemptive los controles de la aplicación son los que asignan los tiempos al CPU.

Observando otro aspecto un sistema operativo multiusuario no ofrece ninguna ventaja o desventaja para un DBMS C/S. Un nuevo concepto para los sistemas operativos multitarea es que sea **multithreading** lo que significa la capacidad que da un aplicación multitarea consigo misma, un ejemplo de esto es cuando un solo usuario en un DBMS puede comenzar la ejecución de un nuevo proceso o tarea como un reporte complejo, mientras el mismo usuario está ejecutando un *query*. Multithreading tiene implicaciones significativas para el diseño de

complejos DBMSs C/S, ya que da a la aplicación un enorme control sobre nuevas tareas que comienzan o terminan. Las aplicaciones pueden ser diseñadas con inteligencia para identificar cuáles procesos o tareas tienen alta prioridad y se les tiene que dar más tiempo del CPU.

II.3.1.1 Computadoras personales

Hasta años recientes las PCs han llegado ha ser una plataforma aceptable para bases de datos cliente/servidor. El advenimiento de los sistemas de alto poder con 32 bits en los procesadores 80386, 80486 y pentium; discos duros con rangos de gigabytes y los sistemas operativos multitarea dan a las PCs la capacidad para competir con workstations RISC y minicomputadoras que han sido la plataforma tradicional para los DBMSs.

Aunque un sistema basado en un CPU 80386 puede funcionar como servidor de base de datos, el sistema base recomendado es un 80486 a 66 Mhz. Con un mínimo de 8 Mb, aunque se recomiendan de 12 a 16 Mb para tener una eficiencia adecuada. Además uno de los componentes mas críticos para la eficiencia del sistema es el disco duro ya que involucra la actividad del DBMS en cuanto a la lectura y la escritura de los datos, en este aspecto se tiene la ventaja que se puede expandir la capacidad mediante la tarjeta SCSI (SCSI small computer system interface) que permite hasta siete discos de hasta 1.2 G.

Otra de las decisiones que se tienen que tomar es el tipo de bus interno (sistema de conexión para agregar tarjetas) que puede tener el servidor. El bus ISA (Industry Standard Architecture) se basa originalmente en un bus de 16 bits que fue diseñado por IBM para su sistema PC-AT. Como los procesadores de 32 bits llegaron a ser más comunes, la industria de las PCs vieron la necesidad de un bus de 32 bits, desafortunadamente este no se basa en un estándar. IBM propuso la arquitectura MICRO CHANNEL (MCA) como estándar y la implantó en su línea de computadoras PS/2. Pero aunque MCA tiene una técnica excelente tiene dos problemas: no es compatible con ISA y no ha tenido el alcance que esperaba IBM, ya que no ha ido más hallá de ser implantada en equipos IBM.

En contraparte otras empresas propusieron una alternativa como estándar EISA (Extended Industry Standard Architecture). El diseño del bus EISA permite a los usuarios que continúen usando ISA como tarjetas agregadas, mientras soporta EISA con 32 bits cuando lo necesita. Los sistemas EISA han tenido gran difusión, y dan a los usuarios opciones. Cuando un bus es por completo de 32 bits tiene gran velocidad tanto en los procesos del disco y en los accesos a la red, eliminando así serios problemas de tráfico en el servidor.

Los Sistemas Operativos recomendados para sistemas DBMS C/S son OS/2, Windows NT o Netware. Algunos distribuidores de los DBMS C/S proporcionan el MS/PC-DOS o Windows 3.1 para su software, pero si se tiene el propósito de desarrollar una aplicación, no se recomienda utilizarlos, ya que no proporcionan todos los servicios necesarios al DBMS C/S, por lo tanto no son adecuados para esta tarea.

Windows NT es un sistema operativo de 32 bits, preemptive, multitarea y multithreading. El sistema operativo OS/2 2.0 de 32 bits incrementa el nivel de memoria real a 32 Mb y con

soporte para memoria virtual, multitarea y multithreading. El sistema operativo Netware versión 3.2 o más proporciona acceso hasta por 4 G de RAM, por restricciones de hardware esto se limita bastante, y tiene la habilidad de ejecutar las aplicaciones en el servidor como Módulos Cargables (NLMs - Netware Loadable Modules), además de proporcionar capacidades multitarea nonpreemptive.

También existen algunas versiones de UNIX que se ejecutan en 80386, 80486 y pentium; que pueden ser utilizados como sistemas operativos para bases de datos cliente/servidor basadas en PCs. Aunque esto no es muy común, ya que UNIX está orientado a una cierta plataforma de hardware, pero puede darse esto debido a que algún tipo de software C/S sólo se puede ejecutar en versiones UNIX del sistema operativo.

II.3.1.2 RISC y otras Workstation UNIX

Las Workstation basadas en los procesadores RISC (Reduced Instruction Set Computing) son por lo general utilizadas para aplicaciones científicas o de ingeniería, los CPUs RISC son por lo general más rápidos y más poderosos que los de INTEL Pentium. En un CPU RISC se obtiene un aumento de eficiencia al reducir la cantidad de microcódigo en el mismo chip; menos código significa que el CPU puede ejecutar las mismas operaciones internas mas rápidamente. Hay numerosos chips RISC tales como Sun SPARC, DEC's Alpha, MIPS y Motorola 88000. Y el Sistema operativo usual para Workstation RISC es UNIX.

La línea que divide una RISC de alto poder o Workstation UNIX de una minicomputadora puede llegar a ser muy borrosa. Para hacer dicha distinción más simple, las Workstation incluyen algún sistema multitarea de un solo usuario basado en UNIX que puede ser utilizado como servidor a través de una conexión de red. En contraste, las minicomputadoras son sistemas multiusuario que soportan tanto conexiones a red como terminales conectadas directamente.

Las Workstations se parecen y operan como PCs, pueden ser de tipo torre y tienen conectado directamente el teclado, el mouse y el monitor (usualmente a color y alta resolución). Y con un mínimo de 8 Mb en RAM y 300 Mb en disco duro. Todas las workstation RISC tienen como sistema operativo una variación de UNIX, o también Microsoft's Windows NT. Pero la desventaja es que las bases de datos y aplicaciones tienen que ser diseñadas y compiladas para ejecutarse en un CPU particular.

II.3.1.3 Minicomputadoras

Minicomputadoras y mainframes son los ambientes tradicionales en los cuales se desarrollan aplicaciones de bases de datos, y las minicomputadoras por lo general son optimizadas para aplicaciones multiusuario. En años recientes, varios métodos han sido desarrollados para conectar una mini a una red LAN basada en PCs, por lo que ahora es posible utilizar la minicomputadora tanto como servidor de archivos y como servidor de bases de datos.

Un sistema c/s también reduce la carga de trabajo en la minicomputadora, porque mueve una parte del procesamiento al sistema front-end, lo cual deja a la mini más soporte de usuarios sin expansión de hardware. Las minicomputadoras usualmente están basadas en CPUs propietarios que son generalmente más poderosos que las workstations y con equipo más grande. Y en tamaño el equipo pasa de pequeños sistemas tipo torre a cajas que parecen enormes refrigeradores. Usualmente las minis tienen un número de puertos seriales para conectar terminales tontas y comúnmente incluyen tarjetas de red. Estos equipos soportan mucha más RAM (típicamente de 128 a 256 Mb) de la que es común en PCs y Workstations, lo cual permite tener aplicaciones que pueden soportar el acceso simultáneo de cientos de usuarios.

Las minicomputadoras también soporta múltiples CPUs en la misma caja, la cual agrega tanto poder de procesamiento (high-end) y redundancia en el sistema en caso de falla. Las minis también soportan sistemas de disco duro de alta velocidad que está dentro del rango de cientos a miles de gigabytes. Las minicomputadoras pueden estar en *clustered*, lo cual significa que todas las máquinas son enlazadas mediante conexiones de alta velocidad, todas las máquinas en el *cluster* comparten los mismos discos. El *clustering* deja a los usuarios expander la capacidad de las computadoras (número de usuarios que soporta) sin tener que comprar más drives de disco o mover datos entre diferentes máquinas. Las minicomputadoras más comunes son hechas por Digital Equipment Corporation (DEC) y Hewlet Packard (HP).

Las minicomputadoras utilizan sistemas operativos propietarios, contando con UNIX como un sistema operativo opcional. Mientras los DBMS c/s son disponibles para todos estos sistemas operativos, ejecutar UNIX en la minicomputadora tiene la ventaja de que se tienen amplias opciones para el software de aplicaciones y los DBMSs, además de que existe más personal de soporte de dónde escoger.

Antes de decidir usar una mini como servidor de LAN, se debe estar seguro que el sistema operativo de la minicomputadora en particular, soporte el acceso a las aplicaciones que tú necesitas a través del sistema operativo. Algunos sistemas operativos de minicomputadoras no soportan la clase 2 de aplicaciones c/s y requieren algún tipo de sistema *gateway* para proporcionar a la PC acceso al DBMS que se ejecuta en la mini. El *gateway* puede ser una combinación adicional de software y hardware o puede ser una aplicación de software que se ejecuta en la mini y sirve de interfaz entre los front-ends basados en PCs y el DBMS en el anfitrión.

II.3.1.4 Mainframes

La mainframe es la computadora de propósito general más poderosa, ésta soporta múltiples procesadores de alta velocidad, enormes cantidades de espacio en disco duro y de cientos a miles de usuarios simultáneos. Las mainframes ofrecen mayor seguridad que cualquier computadora disponible, en términos de seguridad de datos y redundancia de hardware. También las mainframe son las computadoras más caras que existen, en términos de hardware, software, ambientes de soporte y personal de soporte. Todas las mainframe pueden soportar

cientos y aún miles de usuarios accediendo múltiples aplicaciones a través de terminales o conexiones de red.
A diferencia de otras computadoras, una *mainframe* no está contenida en una sola caja - ésta por lo general consiste de un cierto número de diferentes subsistemas que se ocupan de diferentes tareas, todas se enlazan a través de cable de cobre de alta velocidad y/o cables de fibra óptica. Dichos subsistemas incluyen los CPUs, módulos de RAM, sistemas de comunicaciones, y drives de disco y cinta. La computadora es accedida a través de terminales o PCs con emuladores de terminal, los cuales son conectados a "controles de terminal" (subsistemas especializados que se ocupan de las comunicaciones de conexiones de red de las terminales). El acceso se realiza a través del "procesador de front-end (FEP) " - un subsistema de hardware que no tiene que ser confundido con las aplicaciones de front-end utilizadas para acceder una base de datos c/s. Este procesador se ocupa de la comunicación entre las terminales remotas y el anfitrión central. Las conexiones de red también son realizadas a través de controles agregados que van dentro del procesador de front-end.

Generalmente las *mainframes* tienen 256 Mb de RAM; el subsistema de drives de disco son de cientos de gigabytes. Estas computadoras son usualmente utilizadas para aplicaciones especializadas tales como cálculos de tiempo, donde se necesitan billones de cálculos por segundo. Las supercomputadoras, raramente se utilizan como sistemas de bases de datos.

Los sistemas operativos de *mainframe* son muy modularizados, con diferentes subsistemas que se encargan de las diferentes asignaciones del CPU, comunicaciones de los sistemas de almacenamiento de disco y cinta, y las interacciones de usuario con la computadora.

Las *mainframe* desarrolladas por IBM fueron las originales "computadora anfitrión" con un software de sistema operativo multitarea y multiusuario, y dichas computadoras ejecutan los sistemas operativos más sofisticados. Las interrelaciones entre los diversos niveles de aplicaciones de software del sistema y las aplicaciones de usuario agregan al sistema *mainframe* un enorme coste en el personal de soporte.

Después de una rápida revisión de las diferentes plataformas y de la generalización de las clases de sistemas c/s. A continuación, se muestra el diagrama II.17 que esquematiza el camino a seguir en la toma de decisiones en cuanto a la plataforma y tipo de sistema c/s que será más adecuado a las necesidades y situación particular de cada organización.

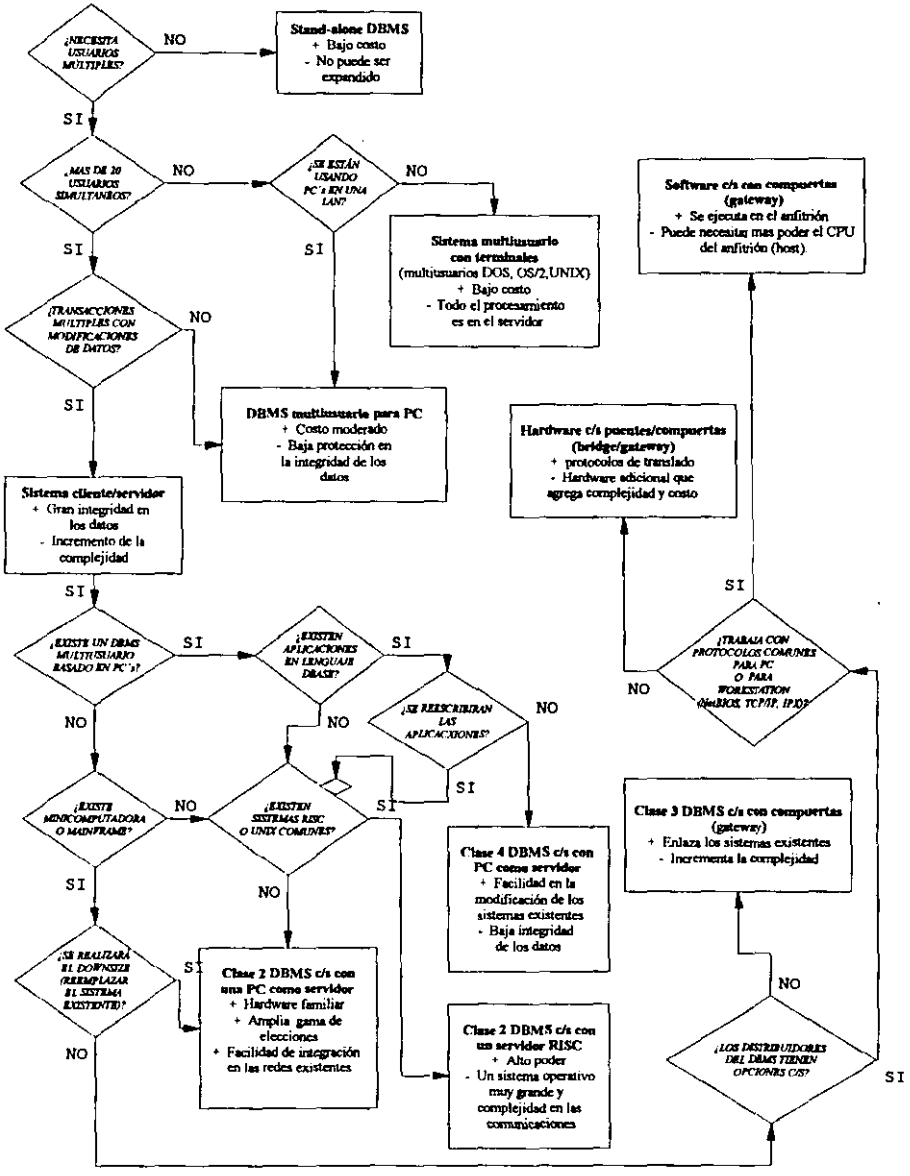


Figura II.17 Guía para base de datos cliente/servidor (**"Make Your Best Connection with** the Database Server Decision Tree", segunda edición, ZIFF-DAVIS PRESS).

II.3.2 Bases de Datos Cliente/Servidor

Un Sistema de Bases de Datos consiste de dos partes: El Sistema Administrador de Bases de Datos (DBMS), el cual es el programa que organiza y mantiene la información, y la Aplicación de la Base de Datos, un programa que nos permite recuperar, ver y modificar la información almacenada por el DBMS.

Una Base de Datos Cliente/Servidor incrementa el poder de procesamiento de la Base de Datos por la separación del DBMS de la aplicación de la base de datos. La aplicación se ejecuta en una o más estaciones de usuario (las cuales son usualmente PCs) y se comunica por la red con uno o más DBMSs ejecutándose en otras computadoras. Los Sistemas de Bases de Datos C/S hacen un mejor uso de todo el potencial de las computadoras, lo cual implica más complejidad.

Un Sistema de Bases de Datos debe proporcionar los siguientes servicios:

- Proporcionar un método de DEFINICIÓN DE DATOS para definir y almacenar el universo de datos.
- El MANTENIMIENTO DE DATOS mantiene el universo de datos usando un registro para cada unidad en el universo con campos que contienen información particular que describen cada unidad.
- Proporciona servicios de MANIPULACIÓN DE DATOS que permite al usuario insertar, modificar, borrar y ordenar en la base de datos.
- Proporciona opcionalmente algunos métodos de DESPLEGADO DE DATOS para el usuario.
- La INTEGRIDAD DE DATOS proporciona uno o más métodos que aseguren que los datos sean correctos.

II.3.2.1 Modelos de DBMSs

Los Sistemas Administradores de Bases de Datos pueden ser agrupados en cuatro diferentes tipos, o más comúnmente llamados modelos: SISTEMA ADMINISTRADOR DE ARCHIVOS, SISTEMA DE BASE DE DATOS JERÁRQUICA, SISTEMA DE BASE DE DATOS DE RED y MODELO DE BASE DE DATOS RELACIONAL. Existe un nuevo tipo de DBMS llamado BASE DE DATOS ORIENTADA A OBJETOS, pero todavía no existe una estandarización.

Es importante notar que históricamente, el modelo Relacional fue la primera descripción de un modelo de base de datos que precede al desarrollo de un DBMS como tal. Los otros tres modelos fueron después definidos de facto para describir los sistemas de base de datos que habían sido utilizados por varios años. La evolución actual de estos sistemas de bases de datos han surgido en el orden de Administración de Archivos a Jerárquico a Red y a Relacional.

Sistema Administrador de Archivos

El Sistema Administrador de Archivos (FMS File Management System - por sus siglas en inglés) es un modelo de datos fácil de comprender, y este sólo describe cómo los datos son almacenados en el disco. En el modelo FMS, cada campo o unidad de datos son almacenados secuencialmente en el disco dentro de un gran archivo.

El sistema administrador de archivos fue el primer método usado para almacenar datos en una base de datos computarizada, y su simplicidad es su única ventaja. Un ejemplo de los productos DBMS construidos con este modelo son el Reflex de Borland. La mayor desventaja del FMS es que no permite la facilidad de cambios en la estructura de la base de datos. La necesidad de describir las relaciones UNO-A-MUCHOS entre diferentes registros, crea la necesidad de otro modelo de Base de Datos, el modelo de bases de datos Jerárquica.

Sistema de Bases de Datos Jerárquicas

El siguiente desarrollo lógico en modelos de Base de Datos es el Sistema de Bases de Datos Jerárquica (HDS). En este modelo, los datos son organizados en estructuras de árbol que se originan de la raíz. Cada clase de datos es localizada en diferentes niveles a lo largo de una rama particular que parte de la raíz. La estructura de datos de cada nivel de clase es llamada nodo.

Las ventajas sobre el modelo FMS es que aquí se pueden definir relaciones UNO-A-MUCHOS. Y también la rapidez para la localización de los datos. La desventaja más sobresaliente del Sistema de Bases de Datos Jerárquica es que no proporciona los medios para representar relaciones MUCHOS-A-MUCHOS entre los registros.

Sistema de Bases de Datos de Red

Los DBMSs basados en el modelo de Red son también conocidos como base de datos CODASYL. Notando que el nombre RED no tiene nada que ver con el medio físico en el que se ejecuta la base de datos. Conceptualmente el modelo de red describe bases de datos en las que se pueden representar relaciones MUCHOS-A-MUCHOS. Las relaciones entre diferentes unidades de datos son comúnmente referidos como "asignaciones" para distinguir de las relaciones padre-hijo en el modelo jerárquico.

Un NDS (NDS - Network Database System - por sus siglas en inglés) depende de su línea de dirección o puntero cíclico para mapear las relaciones entre diferentes unidades de datos. La flexibilidad del modelo NDS está en que muestra las relaciones MUCHOS-A-MUCHOS. Pero las interrelaciones entre las diferentes asignaciones puede llegar a ser extremadamente compleja y difícil en el mapeo. Y para realizar cambios en las diferentes asignaciones se requiere que el programador cree una nueva estructura.

Modelo de Bases de Datos Relacionales

En 1969, el Dr. E. F. Cood hizo su primera publicación en la que define el modelo relacional de base de datos basada en conceptos matemáticos de asignaciones relacionales. El Modelo de Bases de Datos Relacionales (RDM) ha sido constantemente refinado desde entonces, en 1985 el Dr. Cood da las "doce reglas" para bases de datos relacionales, y en 1990 edita un libro en el que define la segunda versión (RV/2) del modelo relacional que contiene 333 reglas que son subasignaciones y expansiones del modelo original.

El Modelo Relacional abandona los conceptos de relaciones padre-hijo entre diferentes unidades de datos. En contraste los datos son organizados en asignaciones de lógica matemática sobre una estructura tabular. En un RDM, cada campo de datos es una columna en la tabla y cada registro es un renglón en la tabla. Cambiar la estructura de la base de datos es algo simple como agregar o borrar columnas de la tabla, lo cual no afecta a las otras tablas de ninguna forma. Nuevas tablas pueden ser "vistas creadas" o "selecciones de tablas existentes", y las viejas tablas pueden ser removidas. No se tiene que reconstruir por completo la estructura de la base de datos para hacer los cambios, también representa un incremento en la preservación de la integridad en los datos. El punto clave en el diseño de bases de datos relacionales está en la definición de las tablas.

La primera meta del modelo de bases de datos relacionales es la preservación de la integridad de los datos. Para ser considerada verdaderamente relacional, un DBMS debe prevenir por completo los accesos a los datos, por lo que DBMS es el encargado de manejar las consultas a la información. Mientras que el modelo relacional no nos dice nada sobre cómo están almacenados los datos en el disco, la preservación de la integridad implica que los datos deben ser almacenados en un formato que prevenga posibles accesos desde un DBMS que no es el que creó la base de datos.

Hasta hace poco, el tener un DBMS relacional implicaba una base de datos que se ejecutaba en una mainframe o minicomputadora. Lo que generó el fenómeno de que existieran DBMSs que sólo cumplían con una parte de las reglas del modelo relacional (particularmente DBMSs diseñados para PCs). Esas bases de datos, conocidas como bases de datos semirrelacionales, eliminan una porción del modelo para incrementar la rapidez o también para permitir la facilidad de acceso del programador a los datos. Hasta 1987, con el incremento del poder de las PC basadas en los CPUs de intel 80486 y pentium, los DBMS diseñados para microcomputadoras se adhieren más a las reglas del modelo relacional.

II.3.2.2 Arquitectura de los DBMS

Como ya lo traté anteriormente (en el primer capítulo) en los tipos de procesamiento tenemos tipos de sistemas de cómputo en los cuales se ejecutan las bases de datos, las cuales pueden caer dentro de las cuatro categorías o plataformas: Centralizada, PC, Cliente/servidor y Distribuida. Las que difieren en cuanto al tipo de procesamiento. La arquitectura del DBMS no determina necesariamente el tipo de sistema de cómputo en que tiene que ejecutarse la base

de datos, pero lo que sí es que ciertas arquitecturas son más adecuadas para algunas plataformas específicas.

En los sistemas centralizados, todos los programas se ejecutan en la computadora anfitrión, incluyendo el DBMS, las aplicaciones que acceden a la base de datos, y las facilidades de comunicación para mandar y recibir datos desde las terminales de usuario.

En los sistemas de cómputo personales cuando el DBMS es ejecutado en la PC, la PC actúa tanto como computadora anfitrión y terminal. En comparación con los grandes sistemas, las funciones del DBMS y las funciones de la aplicación de la base de datos son combinadas en una sola aplicación. Las aplicaciones de bases de datos en una PC sostienen la entrada de usuario, salida de pantalla y acceso a los datos del disco. Combinando estas diferentes funciones dentro de una unidad, tenemos que el DBMS es poderoso, flexible y rápido; pero tiene un costo en cuanto a que decrece la integridad y seguridad en los datos. La mayoría de los DBMSs basados en PC son diseñados bajo el modelo relacional, considerando el factor de que el DBMS no está separado de la aplicación de la base de datos tenemos que muchos de los principios relacionales no son implantados. La más notable omisión es respecto a la integridad de datos. Las bases de datos para PC permiten el acceso directo a los archivos de datos fuera del DBMS que los creó. Esta situación en que puede haber cambios en los archivos de datos, y que violan las reglas con las cuales la aplicación de la base de datos asegura la integridad. Tal violación puede causar que los archivos de datos lleguen a ser inaccesibles por el DBMS. Por esta razón, las bases de datos para PC basadas en el modelo relacional son más adecuadamente descritas como semirrelacionales. Algunos de los más comunes DBMSs para PC semirrelacionales son: Microrim's R:Base, Borland's dBase IV y algunos clones como Microsoft's FoxPro, Borland's Paradox, DataEase Internationals y Revelation Technologies's Advanced Revelation.

En una base de datos Cliente/Servidor se divide el procesamiento de la base de datos en dos sistemas: el cliente PC el cual ejecuta la aplicación de la base de datos y el servidor de la base de datos, el cual ejecuta todo o parte del DBMS. Los servidores de archivos LAN proporcionan la compartición de recursos, tales como espacio en disco para las aplicaciones e impresoras. El servidor de la base de datos puede ejecutarse en la misma PC como servidor de archivos, o (lo más común) en una computadora dedicada. La aplicación de la base de datos en el cliente PC, referido como el **front-end** del sistema, maneja todo el procesamiento de usuario en pantalla y de entrada /salida. El **back-end** del sistema en el servidor de la base de datos maneja el procesamiento y el acceso a disco. La ventaja inmediata de los sistemas Cliente/Servidor es que como divide el procesamiento en dos sistemas reduce en gran medida el tráfico en la red. Y mientras el cliente se ejecuta en una PC, el servidor de la base de datos puede ejecutarse en otra PC o hasta en una Mainframe.

Bajo un sistema de procesamiento distribuido, si el requerimiento del usuario local determina que no tiene los datos, entonces ingresa a la red para obtener la información de otro sistema anfitrión. Obteniendo así respuesta a los requerimientos de usuario, sin importar que tenga que requerir los datos de diferentes lugares.

II.3.2.3 Lenguajes de Programación para Aplicaciones de Bases de Datos

Una aplicación de base de datos es simplemente un programa de computadora que permite a los usuarios introducir, cambiar, borrar y obtener reportes de los datos de la base de datos. Y en los últimos años han surgido herramientas de acceso a la base de datos orientadas al usuario, lo que simplifica el proceso de uso del DBMS y elimina la complejidad de la programación acostumbrada.

Los lenguajes de programación de computadoras estándar, tales como Pascal, Cobol, Basic y C, son lenguajes procedurales. Estos lenguajes pueden ser utilizados para crear aplicaciones de bases de datos a través de la Interfaz del Programa de Aplicacion (API), la cual consiste de asignaciones estándar de funciones (o llamadas) que proporciona el lenguaje para dar acceso a los datos en el DBMS. Las funciones del API están usualmente contenidas en "librerías" que son incluidas en las aplicaciones cuando éstas son compiladas. Algunos tipos de archivos (tal como los DBF usados por dBase) comunes, con los que es posible crear aplicaciones de bases de datos para acceder a los datos sin tener que utilizar una librería API. Son lenguajes de alto nivel, los cuales también pueden ser usados para crear aplicaciones que no son bases de datos, son generalmente referidos como "lenguajes de tercera generación" (3GLs).

El SQL es más propiamente descrito como un **sublenguaje**, el cual no contiene facilidades para mantener la pantalla de usuario o las entradas/salidas. Su gran propósito es dar un método estándar para acceder la base de datos, a pesar de que el resto de la aplicación de la base de datos se tenga que escribir. Esto es diseñado para *queries* interactivos de una base de datos (y referido como SQL dinámico) o como parte de una aplicación escrita en un lenguaje procedural (y referido como SQL embebido).

Otro tipo de lenguaje que es usado con bases de datos en un "Lenguaje Macro" (o Script). Los lenguajes macro no son completamente lenguajes de programación, son una lista de instrucciones que son introducidas manualmente en una aplicación para automatizar ciertas tareas. Altamente específico para aplicaciones particulares, los lenguajes macro se encuentran comúnmente en el paquete DBMS o en **front-ends** de servidores de bases de datos.

II.3.2.4 Sistemas Administradores de Bases de Datos (DBMSs)

Existen algunos puntos importantes a decidir si se utilizará un **sistema de base de datos cliente/servidor en un servidor de PC**, los cuales son mencionados a continuación:

- **Hardware**: Asegurarse de que el hardware es compatible con el sistema operativo bajo el que se va a ejecutar el DBMS.
- **Compatibilidad con las redes que existen**: Esto se puede facilitar si se elige un sistema operativo que conozca el personal de soporte, en lugar de adquirir un DBMS que requiera un sistema operativo que sea nuevo para el personal, lo cual implicará un costo en capacitación y en tiempo.

- **Soporte**: Tomar muy en cuenta que el proveedor proponga un buen plan de soporte o asesoría del producto. Considerando "que el mejor producto en el mundo no te dará nada bueno si no se puede obtener un soporte apropiado cuando se necesita".
- **Monitoreo de eficiencia**: El monitoreo de la base de datos es el camino para detectar si hay un retraso en respuesta que es debido al incremento temporal del requerimiento de datos o por una degradación permanente en la eficiencia. También controla algún tipo de registro histórico con la que se puede checar la eficiencia de la base de datos en periodos de días, semanas o meses.
- **Software de front-end**: Los distribuidores de DBMS cliente/servidor también pueden proporcionar una lista de front-ends que soporta su servidor. Por que se debe estar seguro del software que se utilizará para el desarrollo de la aplicación.

A continuación se mencionan algunos DBMSs basados en PCs que se encuentran en el mercado:

- MICROSOFT SQL SERVER 4.21
- SYBASE SQL SERVER FOR NETWARE 10.0
- GUPTA SQLBASE N5.2 AND 6.0
- IBM DB2/2
- ORACLE7 SERVER
- WATCOM SQL 4.0
- XDB-ENTERPRISE SERVER
- INGRES SERVER FOR OS/2
- EXTENDBASE FOR NETWARE 386
- ADVANTAGE XBASE SERVER
- QUADBASE -SQL SERVER

Ejecutar un **sistema c/s en un sistema basado en Unix** presenta un panorama completamente diferente a los sistemas basados en PC. Los siguientes puntos son importantes y es recomendable tomarlos en cuenta al utilizar una workstation o mini basada en Unix, como servidor de archivos:

- **Hardware**: Si el equipo tiene más de cinco años, se puede considerar un reemplazo total o mover el servidor de la base de datos a diferentes plataformas. Pero el número de usuarios para los que necesita tener soporte puede impedir el movimiento a una plataforma más pequeña, entonces sólo restaría una nueva versión del sistema existente.
- **Compatibilidad con las redes existentes**: Esta es una de las áreas mas críticas y en la cual se cometen más errores. Primeramente se debe estar seguro que el hardware que se va a utilizar tenga soporte para la topología de red de la LAN en que se va a ejecutar. Pero también hay puentes que se pueden utilizar en estos casos, pero esto puede ser una complicación innecesaria. El proceso es mucho más sencillo cuando los sistemas comparten la misma topología.
- **Soporte y entrenamiento**: Como este tipo de sistemas tiene tiempo de existir cuentan con un amplio soporte de operaciones con lo cual tiene lo necesario para satisfacer las necesidades de los clientes.

- **Monitoreo de la eficiencia**: Por el tiempo de existencia de estos sistemas existen numerosos monitoreos de la eficiencia los cuales complementan los que existen en el sistema operativo.
- **Software de front-end**: Este es un factor mucho más crítico en este tipo de sistemas. Muchos front-ends basados en PC trabajan bien con las versiones de los DBMS pero no pueden trabajar con versiones basadas en UNIX. Muchos front-ends requieren un sistema ***gateway*** adicional para comunicarse con versiones que no son para PC, las cuales agregan costo extra de software y hardware al precio del sistema. Por lo que se debe estar seguro que los front-ends a utilizar sean soportados por el software del DBMS.

A continuación se mencionan algunos DBMSs basados en UNIX que se encuentran en el mercado:

- INGRES FOR UNIX AND VAX/VMS
- ORACLE7 SERVER FOR UNIX AND VAX/VMS
- SYBASE SQL SERVER 10.0 FOR UNIX AND VAX/VMS
- GUPTA SQLBASE FOR UNIX
- INFORMIX-ONLINE 6.0
- INTERBASE 4.0

En el caso de **sistemas de minicomputadora**, si se decide permanecer en el sistema propietario y se quiere mover el procesamiento de la base de datos a un sistema abierto, el aspecto fundamental a considerar es el protocolo de red. TCP/IP es el más común y es soportado por la mayoría de los sistemas, también por los sistemas basados en PC y UNIX. Por lo cual se tiene que pagar un precio extra por el hardware y software necesario para agregar TCP/IP al sistema propietario existente. También no se debe olvidar que el protocolo TCP/IP usualmente utiliza más RAM en el sistema cliente a diferencia de otros protocolos, pero esto es un problema mínimo con sistemas basados en Windows. Un ***gateway*** puede ser utilizado en los casos de traslado entre un protocolo de LAN nativo y el TCP/IP, tomando en cuenta que esto agrega un nivel de complejidad al sistema.

A continuación se mencionan algunos DBMSs basados en sistemas de minicomputadora propietarios que se encuentran en el mercado:

- RDB/VMS VERSION 6.0 (DEC)
- ALLBASE/SQL (HP)
- SQL/400 (IBM)

Las *Mainframe* utilizan el protocolo Arquitectura de Sistemas en Red (SNA System Network Architecture) de IBM para comunicarse con terminales dedicadas, con PCs que actúan como terminales dedicadas mediante modems, o con tarjetas emuladoras de terminal. SNA no está diseñado para soportar la compleja comunicación de dos sentidos necesaria para aplicaciones verdaderamente cliente/servidor.

Una solución son los sistemas de *gateway* mencionados anteriormente. Un sistema *gateway* es generalmente un nodo en la LAN que consiste de software especializado que se comunica con la *mainframe* a través de la tarjeta emuladora de terminal. El *gateway* es por lo regular un sistema dedicado, debido a que proporciona el procesamiento necesario para mantener el traslado entre diferentes protocolos y códigos. El cliente se comunica directamente con el *gateway* o manda el requerimiento al servidor de la base de datos local, el cual pasa el requerimiento al *gateway*. Las comunicaciones del cliente utilizan el protocolo de LAN normal, y el *gateway* traslada los requerimientos y los manda a la *mainframe* en el protocolo SNA. Una aplicación de software es comúnmente ejecutada en la *mainframe* que es parte del sistema de *gateway*; el software sostiene la interfase entre los datos en el DBMS y los comandos SQL que pasan a través del *gateway*. Algo que hay que tomar muy en cuenta es que los *gateway* son caros y agregan una gran complejidad a las comunicaciones entre el cliente y la base de datos.

A continuación se mencionan algunos DBMSs basados en sistemas de mainframe que se encuentran en el mercado:

- DB2 (IBM)
- SQL/DS (IBM)
- ORACLE7 MAINFRAME VERSIONS

II.3.3 Aplicaciones de usuario (front-ends)

Los paquetes de desarrollo de front-ends pueden ser divididos en cuatro categorías basándose en sus funciones primarias: **add-ons para productos existentes, programas de reporteador/query, herramientas de desarrollo de aplicaciones y herramientas de análisis e integración de datos.** Los módulos add-on son para aplicaciones existentes de PC tales como dBase o Lotus 1-2-3, para *queries* del servidor de base de datos. Las herramientas Query/Reportes facilitan a los no programadores a crear *queries* para reportes de la información residente en el back-end. Las herramientas para el desarrollo de aplicaciones, son utilizadas por programadores, son diseñadas para facilitar el proceso de creación de aplicaciones front-end clientes. Finalmente, herramientas de análisis e integración de datos son diseñadas para administradores y ejecutivos quienes necesitan examinar datos de cierto número de fuentes para tomar complejas decisiones de negocios. Una aplicación front-end puede ser cualquier software o programa de aplicación cliente para una compañía o usuario.

La secuencia de eventos que toman lugar cuando los usuarios acceden al servidor de la base de datos puede ser generalizado en seis pasos que se ilustran en la figura II.18. Por simplicidad el término *query* representa alguna acción que pueda tomar el usuario sobre la base de datos, tales como modificar datos, insertar nuevos datos, borrar datos o requerimiento de la base de datos.

En primera el usuario crea un *query*. En el siguiente paso la aplicación front-end formatea el *query* en el SQL que se utiliza en el servidor back-end y manda el *query* hacia el servidor. El

servidor verifica que el usuario tenga los derechos de seguridad adecuados. Si lo acepta,

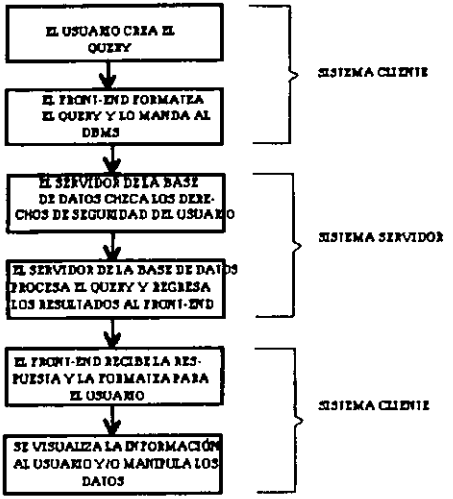


Figura II.18 Secuencia general de eventos que ocurren cuando un usuario accede al servidor de la base de datos

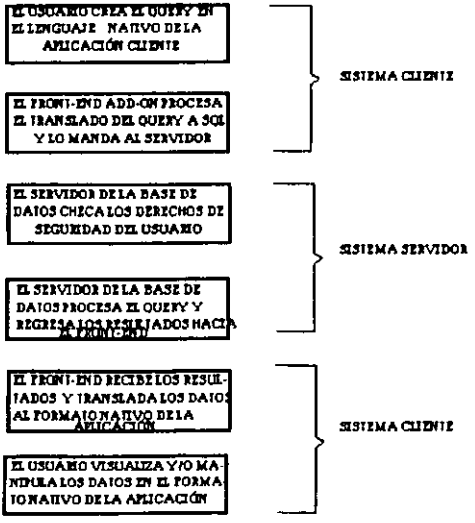


Figura II.19 Cuando el software cliente es un add-on para una aplicación existente, la secuencia de eventos en el procesamiento de un query es un poco más complejo.

procesa el *query* y manda los datos apropiados al front-end. La aplicación cliente recibe los datos de respuesta y los formatea para la presentación al usuario. Finalmente el usuario ve la

repuesta en la pantalla y puede manipular los datos o modificar los *queries* y vuelve a empezar el proceso.

En el caso de front-end add-on, el proceso es usualmente más complicado para el usuario final. Como se muestra en la figura II.19. Primeramente, el módulo add-on traslada el *query* desde el lenguaje nativo cliente a SQL y entonces lo manda al servidor. Cuando los datos son regresados, el add-on procesa el traslado de la respuesta de datos al formato nativo del cliente. Entonces los usuarios visualizan y manipulan los datos como si vinieran los archivos de datos del propio front-end. Toda esta translación es agregada al proceso. Lo cual significa que los add-on requieren más recursos en el sistema cliente, particularmente RAM y poder del CPU adicionalmente.

Existen razones por las cuales los productos de diferentes empresas no son compatibles para el acceso al servidor de la base de datos. Hay dos razones para esto: la variación en el SQL y los protocolos de comunicación.

La proliferación de RDBMSs de varias empresas a resultado en un esfuerzo para crear uno común, la independencia de la base de datos en cuanto a la Interfaz del Programa de Aplicación (API Application Programming Interface). Un API común puede facilitar a los desarrolladores y a las empresas de aplicaciones front-end para tener soporte para algunos servidores de bases de datos, sin tener que escribir drives específicos para cada DBMS. Al mismo tiempo, hay tres versiones de APIs comunes compitiendo, cada uno con sus ventajas y desventajas.

El primer API fue **Microsoft's Open Database Connectivity (ODBC)**. Las aplicaciones de Bases de Datos llaman el driver ODBC, y el driver sostiene las translaciones necesarias para acceder al back-end de la base de datos. El ODBC es el más popular de los API, y rápidamente llegará a ser el estándar. Microsoft sólo proporciona drives ODBC para sus productos de bases de datos; otras empresas de RDBMS tienen que escribir su propia interfaz ODBC para sus productos. Algunos se encuentran renuentes a soportar solo el ODBC, aunque la mayoría de los front-end proporcionan soporte para ODBC.

Information Builder realizó su propio API llamado **EDA/SQL**. EDA/SQL utiliza un sistema de *gateway* para acceder a 50 DBMSs diferentes, incluyendo todas las versiones de SQL Server. El sistema *gateway* es un tipo de servidor de base de datos que no hace almacenamientos actuales de algún dato; en cambio determina el servidor de la base de datos destino, y traslada los argumento en el dialecto apropiado del SQL. También sostiene la necesidad de translación de datos entre la aplicación front-end y el servidor de la base de datos. El *gateway* es completamente independiente de la plataforma del servidor de la base de datos necesaria para ejecutarse, lo que permite la utilización común del protocolo de red para comunicarse.

Borland, IBM, Novell y Wordperfect anunciaron en 1992 el **Integrated Database API (IDAPI)**, un competidor para ODBC. IDAPI soporta bases de datos creadas por Borland's dBase, Paradox e InterBase, tales como IBM's Database Manager, DB2/2, Netware Btrieve,

Netware SQL y Wordperfect. Otras compañías anunciaron soporte para IDAPI, pero pocas lo llevaron a cabo. Y solo Borland, Novell y SYBASE proporciona de alguna forma soporte para IDAPI.

ADD-ONs para productos existentes

La mayoría de los add-on existentes son para bases de datos de PC, considerando que la mayoría de los sistemas cliente/servidor están basados en PCs. Los add-on de bases de datos para PC facilitan el diseño y desarrollo de sistemas cliente/servidor que integran las bases de datos existentes en el nuevo sistema. El costo de desarrollo de aplicaciones puede ser muy bajo, porque los usuarios y programadores están familiarizados con la base de datos existente. Y con un entrenamiento mínimo pueden adaptar sus conocimientos existentes para acceder a los datos del servidor.

Los productos add-on para PC son las más versátiles aplicaciones front-end, primeramente porque tiene por completo los derechos sobre el DBMS. Las bases de datos para PC tienen una interfaz de usuario amigable, presentando el manejo de menú para *queries* y capacidades completas para reportes. Las bases de datos para PC también tienen algo de programación compleja o lenguajes script, los cuales facilitan la creación de aplicaciones complejas que pueden acceder y combinar datos tanto de la base de datos nativa y del servidor de la base de datos. La característica de los front-end de bases de datos basados en PC pueden caer en alguna de las cuatro categorías.

Las bases de datos basadas en PC pueden servir como soluciones permanentes para crear sistemas cliente si se toman en cuenta los problemas que pueden surgir con estos productos, ya que en primera no son diseñados para ser clientes. Los add-on pueden no estar muy bien integrados en el producto, por lo cual puede requerir cambios en como funcionan las aplicaciones, siendo este uno de los problemas por los cuales más empresas incluyen el soporte para ODBC. Las bases de datos en PC pueden servir tanto en términos de soluciones pequeñas mientras nuevos front-ends son desarrollados y los datos son movidos desde una base de datos en PC a un servidor de bases de datos, o como front-ends completos con sus propios derechos.

Los add-on se pueden clasificar dentro de tres subcategorías : módulos externos que son agregados a un producto, módulos que son incluidos en la versión base de un producto y módulos que son incluidos como parte de una versión especial del producto.

A continuación se enlistan algunos productos existentes en el mercado, los cuales son considerados como add-ons:

- **ACCESS (MICROSOFT)**
- **APPROACH (LOTUS DEVELOPMENT CORP.)**
- **DBASE FOR WINDOWS (BORLAND)**
- **PARADOX FOR WINDOWS 5.0 (BORLAND)**
- **DATAEASE SQL (DATAEASE INTERNATIONAL)**
- **SUPERBASE95 (SOFTWARE PUBLISHING CORP.)**

- **Q&A (SYMANTEC)**
- **ADVANCED REVELATION**
- **CLARION**
- **PC/FOCUS AND PM/FOCUS (INFORMATION BUILDERS INCORPORATED)**
- **SPREADSHEET ADD-ONS**

Programas de Query/Reporte

Este es un tipo de herramienta para usuario final, están diseñadas para facilitar al usuario ocasional y no programadores para acceder los datos en una base de datos cliente/servidor. *Queries*, los cuales requieren datos y presentación del resultado en la pantalla; y la creación de reportes, los cuales mandan el resultado a la impresora.

Este tipo de aplicaciones front-end usualmente no tienen un lenguaje de programación propio, pero pueden tener un lenguaje macro o script, con lo cual *queries* comunes pueden ser automatizados para después reusarlos. También permite guardar algún *query* en archivo para que pueda ser reejecutado cuando se necesita o si va a ser utilizado en algún reporte.

Los front-ends para *queries* y reportes varían en sus capacidades. Algunos tienen una excelente interfaz con una mínima capacidad para reportes, mientras otros son más adecuados para hacer reportes. Por tanto dependiendo de las necesidades se elegirá el front-end.

A continuación se enlistan algunos productos existentes en el mercado, los cuales son considerados como Programas de Query/reportes:

- **QUEST (GUPTA)**
- **QUEST REPORTER (GUPTA)**
- **IMPROMPTU**
- **PERSONAL ACCESS**
- **ORACLE CARD**
- **CLEARACCESS**
- **DATAPIVOT AND DATAPRISM**

Herramientas de desarrollo de aplicaciones

Las herramientas para el desarrollo de aplicaciones son utilizados por programadores y desarrolladores de aplicaciones para crear front-ends clientes. Todas las empresas que producen DBMSs también producen sus correspondientes herramientas para que puedan desarrollarse front-ends. O también, producen herramientas front-end para los DBMSs más populares.

El desarrollo de aplicaciones puede llegar a ser una de las áreas más espinosas en el mercado cliente/servidor. Existen firmas de consultoría que se especializan en crear aplicaciones cliente/servidor especializadas para ciertos mercados. Estas aplicaciones son llamadas

aplicaciones de mercado verticales, ya que son creadas para un uso en particular, tal como un control de inventarios.

Ambientes de desarrollo de aplicaciones caen en dos categorías: **herramientas de desarrollo y aplicaciones de análisis e integración**. Las herramientas de desarrollo de aplicaciones es la más común de las dos, y puede ser usada para crear algún tipo de front-end cliente/servidor. Las aplicaciones de análisis e integración son aptas para crear sistemas especializados para examinar datos de varias fuentes y tomar decisiones basadas en el análisis.

Cada empresa creadora de los DBMSs c/s tienen algún tipo de herramienta de programación que permite utilizar lenguajes de tercera generación como por ejemplo el C y el COBOL, para crear aplicaciones front-end clientes. Y muchos tienen también productos de desarrollo de aplicaciones que facilitan la creación de formas para *query* cliente, reportes y menús de usuario; Oracle's SQL *Forms y SQL *ReportWriter e INGRES/Tools son algunos ejemplos. Por lo cual, usualmente estas herramientas son por lo regular restrictivas para crear front-ends de la empresa que produce el DBMS o para empresas que tienen licencias para producir dichas herramientas.

A continuación se enlistan algunos productos existentes en el mercado, los cuales son considerados como Herramientas de desarrollo de aplicaciones:

- **SQLWINDOWS (GUPTA)**
- **POWERBUILDER ENTERPRISE (POWERSOFT CORP.)**
- **DELPHI (BORLAND INTERNATIONAL)**
- **OBJECT/1 (MICRO DATABASE SYSTEMS)**
- **Q&E DATABASE EDITOR AND LIBRARY**
- **FORMFLOW**

Aplicaciones de integración y análisis de datos

Integración y análisis de datos son dos partes de un mismo proceso, primero el usuario hace el *query* de los datos de diferentes fuentes y los combina para después analizar lo que significa.

Los tipos más comunes de aplicaciones de análisis e integración de datos son llamados Sistemas de Información Ejecutiva (EIS Executive Information System). Aplicaciones EIS son diseñados para recoger información de datos en diferentes lugares de una organización y presentar la información a los gerentes de la organización para que los auxilie en la toma de decisiones complejas en el negocio. Por ejemplo, EIS puede combinar datos de la base de datos que impliquen inventario, ventas y personal para auxiliar a los gerentes en la creación de cédulas de vacaciones para que tengan un mínimo impacto en cuanto al funcionamiento en la empresa.

Los sistemas de soporte de decisiones (DSS) son una variación de las aplicaciones EIS. Estos son más especializados y son utilizados para proporcionar datos en tiempo real para un soporte rápido en las decisiones de mercado. Funcionalmente estas son similares a los sistemas

EIS, y los dos términos son utilizados de forma indistinta para describir el mismo tipo de aplicación.

Las aplicaciones de análisis e integración son también utilizadas para estudios estadísticos y científicos, para análisis de mercado, o por la necesidad de combinar y examinar datos de múltiples fuentes. La proliferación de información en el mundo moderno hace difícil distinguir los datos importantes. La integración y análisis de datos es la aplicación más útil que puede haber para los usuarios.

A continuación se enlistan algunos productos existentes en el mercado, los cuales son considerados como Aplicaciones de Análisis e Integración de Datos:

- **FOREST & TREES (TRINZIC CORP.)**
- **LIGHTSHIP AND COMMAND CENTER (PILOT EXECUTIVE SOFTWARE)**
- **GQL - GRAPHIC QUERY LANGUAGE (ANDYNE COMPUTING LTD.)**
- **INTELLIGENT QUERY - IQ (IQ SOFTWARE CORP.)**

Ahora, si se tiene que seleccionar el software de desarrollo del front-end, se tiene que ver con detenimiento cuáles son los problemas que hay que resolver, con lo que se puede presentar la necesidad de utilizar dos o más front-ends dependiendo de las necesidades de los diferente usuarios.

También tenemos que existen ciertos principios que se deben tomar en cuenta como guía en la evaluación de front-ends. Primero, decidir si se usará una aplicación en modo-carácter o GUI. Si se quiere un front-end en modo-carácter, esto está condicionando a que se utilice un add-on para base de datos en PC o escribir tu propia aplicación cliente en un lenguaje de tercera generación, porque la gran mayoría de los front-ends están basados en Windows.

Si se desea programar la aplicación y no se tiene preferencia en que sea en modo-carácter o GUI, se puede utilizar un add-on o herramientas de desarrollo de aplicaciones. Y en el caso que se quieran desarrollar aplicaciones sólo en GUI, se pueden utilizar herramientas de desarrollo de aplicaciones o herramientas de *query*/reporte que tenga un lenguaje script.

Pero si la meta es combinar la base de datos existente con un sistema cliente/servidor, se puede utilizar un front-end add-on o un front-end de análisis e integración. Y si los datos residen en una base de datos en PC, un add-on puede ser la mejor solución.

Las herramientas query/reportes y las herramientas de análisis e integración pueden ser utilizadas por usuarios que no necesitan modificar los datos en el servidor. Estas son herramientas que son mucho mejores para recuperar los datos en lugar de introducirlos. Se puede hacer uso de front-ends add-on o de desarrollo de aplicaciones para los usuarios que constantemente modifican la base de datos.

Finalmente, se puede considerar la utilización de aplicaciones de análisis e integración de datos. Las cuales son el mejor camino para combinar y examinar todos los datos en la organización, y puede ayudar a descubrir relaciones entre las diferentes partes de un negocio de las cuales nunca se había considerado que existieran.

CAPÍTULO III

Metodología en el desarrollo de sistemas bajo arquitectura Cliente/Servidor

III METODOLOGÍA EN EL DESARROLLO DE SISTEMAS BAJO ARQUITECTURA CLIENTE/SERVIDOR

Con el avance de la tecnología se ha visto que han ido evolucionando los sistemas de comunicación y de información. Pero a la par, se ha dado como consecuencia lógica una evolución en las Metodologías de Desarrollo Integral para un sistema de cómputo, y específicamente dentro del modelo cliente/servidor. Todos estos cambios han sido visualizados por diferentes expertos, desde su muy particular punto de vista. Pero lo importante es que todos ellos coinciden en el objetivo de obtener un esquema para el diseño y desarrollo de un sistema cliente/servidor.

En el presente capítulo, el objetivo será mostrar un panorama muy general de las diferentes metodologías propuestas, y a partir de esto, ya teniendo un marco general de referencia, tomar lo que se considere útil y adecuado para resolver la problemática que nos presente el caso específico que se tratará.

Existen diversos autores que presentan su propuesta metodológica, pero en el presente trabajo se tomarán como base cinco, de los cuales se analizarán sus propuestas. Los autores que consideraré son los siguientes: Williams H. Inmon, Larry T. Vaughn, David Vaskevitch, Paul E. Renaud y Paul Kavanagh. Exponiendo de forma breve la metodología propuesta por cada uno de ellos.

Y partiendo de esa base seleccionar la metodología o punto de vista de alguno de estos autores para el análisis y diseño de la problemática que se nos presentará en el ejemplo de diseño que se tratará en el capítulo IV.

III.1 Panorama General de las Metodologías

III.1.1 Metodología de Desarrollo en Cliente/Servidor - W. H. Inmon (A Client/Server Development Metodology)

Inmon plantea que el propósito de una metodología es direccionar el desarrollo bajo un camino racional, puntualizando lo que se necesita realizar, en qué orden y cuánto tiempo tomará la actividad. Pero el entusiasmo por las metodologías ha llevado a la decepción a la hora de la implementación. Y explica el por qué de esta situación, argumentando los siguientes puntos:

- Por lo general las metodologías muestran un flujo lineal de las actividades. Pero de hecho casi todas las metodologías requieren ejecución en términos de iteraciones. En otras palabras, es absolutamente normal el realizar dos o tres pasos anteriores de nueva cuenta. Y las metodologías no muestran nada que se refiera a las actividades iterativas.

- Las metodologías usualmente muestran actividades que ocurren una y sólo una vez. Pero en realidad algunas actividades requieren ser realizadas completamente una sola vez. A diferencia de otras actividades que se hacen repetitivamente para casos diferentes.

METODOLOGÍA

- Las metodologías muestran una prescripción de actividades a ser realizadas. Pero con frecuencia, algunas de las actividades no necesitan hacerse en su totalidad, mientras que otras actividades sí lo requieren; situación que no es mostrada como parte de la metodología.

- Con frecuencia las metodologías no distinguen el tamaño del sistema que se desarrollará bajo la metodología. Algunos sistemas son tan pequeños que la rigurosidad de la metodología no tiene sentido. Algunos otros sistemas rebasan a la metodología por su tamaño y complejidad. Mientras que otros se ajustan perfectamente a la capacidad de la metodología.

- Con frecuencia las metodologías mezclan los asuntos concernientes a la administración del proyecto con las actividades de diseño y desarrollo que se tienen que realizar. Usualmente la administración del proyecto se puede mantener separada de los asuntos metodológicos.

- Con frecuencia las metodologías no hacen la distinción entre Procesamiento Operacional y DSS (Decision Support Systems - Sistema de Soporte de Decisiones). Mientras que el ciclo de vida de desarrollo del Procesamiento Operacional y DSS son diametralmente opuestos. Se obtienen logros si se hace una clara distinción entre Procesamiento Operacional y DSS.

- Por lo regular las metodologías no incluyen puntos de chequeo y fases de alto en caso de falla.

- Por lo regular las metodologías son vendidas como solución y no como una herramienta. Cuando una metodología es vendida como una solución, inevitablemente la metodología invita a reemplazar el buen juicio y el sentido común, lo que es siempre un error.

Después de las anteriores puntualizaciones, indica que el " propósito general de aplicar una metodología en el ambiente cliente/servidor será el desarrollar pero en completo entendimiento de los riesgos de la metodología. Pero qué se requiere para mejorar estos resultados se deja completamente al desarrollador".

Como punto fundamental de partida, Inmon propone dos metodologías. La primera para el Procesamiento Operacional y la segunda para DSS. Las metodologías que describe son por completo para un procesamiento de datos corporativo cliente/servidor. Lo más importante para iniciar la discusión de la metodología es el saber distinguir entre el procesamiento operacional y DSS. Para lo cual indica ciertas reglas:

Reglas para distinguir un procesamiento operacional
- servir a una comunidad de oficina
- modificar datos
- se enfoca a valores contenidos en registros individuales
- son programas que operan con pocos registros relativamente
- requiere una rápida respuesta en tiempo

Reglas para distinguir un procesamiento DSS
- sirven a una comunidad gerencial
- accesos, cálculos y ediciones de datos, pero no modificaciones
- se enfoca en valores que operan en muchos registros
- requiere una respuesta relajada en tiempo

Primero se expondrá el Sistema Operacional Cliente/Servidor, el cual se muestra de forma gráfica en la figura III.1. A continuación se irán describiendo cada una de las etapas de la metodología, y para que quede más claro se debe ir haciendo referencia a la figura que le corresponde, y así identificar el orden.

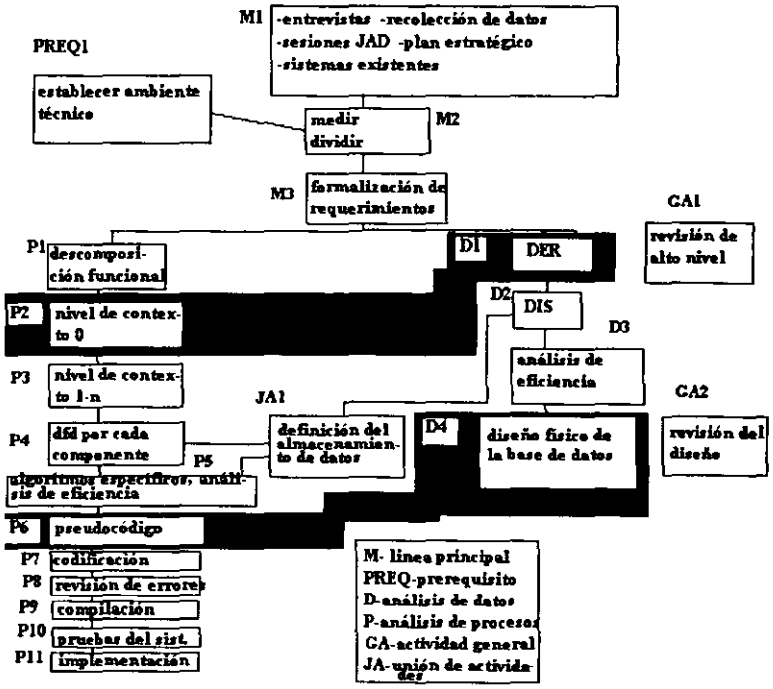


Figura III.1 Desarrollo Operacional.

M1 - Actividades iniciales del proyecto
Esta actividad está constituida de:

- Entrevistas.
- Recolección de datos.
- Sesión JAD (Join Application Design). Son sesiones (grupo Brainstorm) en las que se da un libre flujo de ideas, con personas que están enfocadas a un mismo objetivo. En las cuales se deben de formalizar los requerimientos que colectivamente representan las necesidades del usuario final.

- Analizar el Plan Estratégico del Negocio. La influencia del Plan Estratégico de Negocios puede manifestarse de muchas formas, como puede ser en la identificación de nuevas líneas del negocio, en la descripción de cambios organizacionales; y tantos otros factores que pueden dar forma al sistema que será construido.
- Revisar los requerimientos de los sistemas existentes para profundizar en el nuevo sistema. Para así identificar el impacto y la influencia de los sistemas existentes en los requerimientos del sistema que será desarrollado.

M2 - Medición y División. (Sizing, Phasing)
Determinar la magnitud del problema que se plantea en el sistema, y si es grande es recomendable dividirlo en unidades pequeñas fáciles de administrar.

M3 - Formalización de requerimientos
En esta etapa se debe estar seguro de cumplir con los siguientes puntos:

- Los requerimientos que se han recogido están completos, aunque sigue siendo posible que se puedan agregar más.
- Los requerimientos están organizados.
- Los requerimientos no entran en conflicto unos con otros.
- No se traslapan los requerimientos.
- Los requerimientos Operacionales y DSS han sido separados.

Cumpliendo con los puntos anteriores, se tiene la base para ir hacia un diseño detallado.

PREQ1 - Definición del ambiente técnico
Los siguientes puntos tienen que establecerse:

- La plataforma de Hardware a ser utilizada.
- El sistema operativo que se usará en cada nodo.
- El DBMS.
- EL software de la red.
- El (los) lenguaje(s) de desarrollo.

Además es de gran ayuda establecer de antemano lo siguiente:

- Configuración de los puentes entre las redes.
- Ubicación de nodos.
- Administración del sistema de registro.
- Compatibilidad de protocolos de red.
- Y algunas otras características operativas del ambiente cliente/servidor.

D1 - Diagrama Entidad-Relación

D2 - Asignación de Unidades de Datos (Data Items Sets - DIS)

Esta relacionado con el nivel de detalle. Aquí definimos los atributos, grupos de atributos y llaves. Así como el tipo de dato de cada atributo y las relaciones.

D3 - Análisis de Eficiencia
Este paso se llevará a cabo si el volumen de datos, el volumen de procesamiento, el tráfico sobre la red, el crecimiento en los datos o en el procesamiento pueden representar un aumento significativo de la actividad. Si ninguna de estas situaciones se presenta el paso no se realiza. En caso afirmativo, la desnormalización de los datos se efectúa. Para tal objetivo se puede realizar lo siguiente:

- Introducción de datos derivados.
- Unión de tablas.
- Introducción selectiva de redundancia.
- Creación de arreglos de datos.
- Separación de datos.

Con lo anterior puede presentarse una pequeña pérdida de los beneficios de la normalización de datos, o también puede no afectar.

D4 - Diseño Físico de la Base de Datos
Las actividades que caracterizan este paso son:

- Ordenamiento secuencial de datos.
- Indexación.
- Atributos físicos de datos.
- Designación de llaves.
- Administración de la longitud de las variables de datos.
- Especificación de NULOS y NO NULOS.
- Integridad Referencial.

Con lo anterior obtenemos las especificaciones de la base de datos para el DBMS.

P1 - Descomposición Funcional
En esta etapa se analiza el sistema desde el aspecto funcional. Con lo que obtenemos una descomposición funcional que describe las diferentes actividades a ser ejecutadas, visualizadas desde el más alto nivel hasta el nivel más bajo.

P2 - Nivel de Contexto 0
El nivel de contexto 0 describe la descomposición funcional abstrayéndose al más alto nivel. Corresponde al DER en el modelo de datos.

P3 - Nivel de Contexto 1 - n
En estos niveles se va describiendo con más detalle lo que sucede. Esta etapa corresponde a la asignación de unidades de datos (DIS) en el diseño de datos.

P4 - Diagrama de flujo de datos (Data Flow Diagram - DFD)
Se ilustra de forma gráfica los niveles de contexto.

P5 - Especificación de Algoritmos y Análisis de Eficiencia

P6 - Pseudocódigo

P7 - Codificación

P8 - Revisión
Es la revisión en la que se intenta encontrar errores, procurando que el código esté libre de errores tanto como sea posible.

P9 - Compilación

P10 - Pruebas del sistema (Testing)
Con este paso se tiene listo el código para su ejecución.

P11 - Implementación
Algunas actividades típicas de la implementación son las siguientes:

- Entrenamiento.
- Se cargan los programas en los nodos que se requiere.
- Carga inicial de datos.
- Conversión de datos si se necesita.
- Se establecen utilidades de monitoreo.
- Se hace la documentación.
- Recuperación, establecimiento de procedimientos de restauración.

Con lo que obtendremos la ejecución satisfactoria del sistema.

JA1 - Definición del Almacenamiento de Datos
Presenta la intersección de los datos necesarios para el DFD (modelo de procesos) y la Asignación de Unidades de Datos (modelo de datos). Con lo que obtenemos la definición de los datos almacenados que satisfacen tanto el diseño de procesos y el diseño de datos.

GA1 - Revisión de Alto Nivel
Implica que el diseño de datos y el diseño de procesos estén en sincronía. Para tener el soporte necesario de los datos para el diseño de procesos, y el soporte de procesos necesario para el diseño de datos.

GA2 - Revisión del Diseño
Esta es una de las actividades más importantes en un proceso de desarrollo completo. Porque si tenemos un buen diseño los pequeños cambios o errores no afectarán en forma significativa y serán fáciles de remediar. Y una parte esencial de esta etapa es la visualización del sistema

completo, especialmente en lo que se refiere a la eficiencia del sistema. Para así valorar los puntos fuertes y débiles del diseño, y recomendaciones para mejorar la calidad del sistema.

Hasta aquí es la descripción de cada uno de los puntos del Sistema Operacional, y a continuación se describirá el desarrollo de sistemas DSS.

El desarrollo del DSS en el ambiente cliente/servidor tiene una diferencia esencial con el desarrollo operacional porque en general los requerimientos de procesamiento no son parte del desarrollo de sistemas DSS. En este sentido el desarrollo de sistemas DSS es en verdad un proceso de manipulación de datos. La metodología ha seguir en la fase de desarrollo se muestra en la figura III.2. Alguna de las actividades especificadas en la figura son actividades de UN SOLO TIEMPO, y algunas otras actividades son recurrentes. Principalmente, DSS1, DSS2, DSS3, DSS4 y DSS6 son actividades de un solo tiempo. Todas las demás actividades DSS necesitan ser ejecutadas para cada nueva subárea. En esta etapa se asume que el modelo de datos ya ha sido realizado para poder comenzar el desarrollo del DSS, además de haberse establecido el ambiente tecnológico.

DSS1 - Análisis del Modelo de Datos (UN SOLO TIEMPO)
Se necesita:

- Identificar las subáreas principales.
- Separar los datos primitivos de los derivados.
- Para cada subárea identificar llaves, atributos, grupos de atributos, relación entre grupos de atributos, múltiple ocurrencia de datos y tipo de dato.

Lo anterior es para confirmar que la organización ha construido un modelo de datos sólido.

DSS2 - Análisis Encapsulado (UN SOLO TIEMPO)
En esta etapa es importantísimo considerar las dimensiones del sistema, y evaluarse el tamaño o mejor dicho el volumen de datos. Y en caso de que se tenga un volumen muy grande de datos será necesario dividir el diseño en varios niveles, con lo cual se está determinando la granularidad adecuada para un buen diseño. Si no son demasiados datos no es necesaria esta división.

DSS3 - Valoración Técnica (UN SOLO TIEMPO)
Este paso consiste en determinar cómo serán distribuidos los registros de información del DSS dentro del ambiente cliente/servidor. Para lo cual existen tres elecciones que son las más comunes:

- En un almacenamiento central que se encuentre separado de la red cliente/servidor (por ejemplo una *mainframe* que funcione como servidor.
- Un almacenamiento central pero que sea parte de la red cliente/servidor (un servidor puro).
- La distribución del sistema DSS a través de varios nodos en la red (no es una opción muy popular).

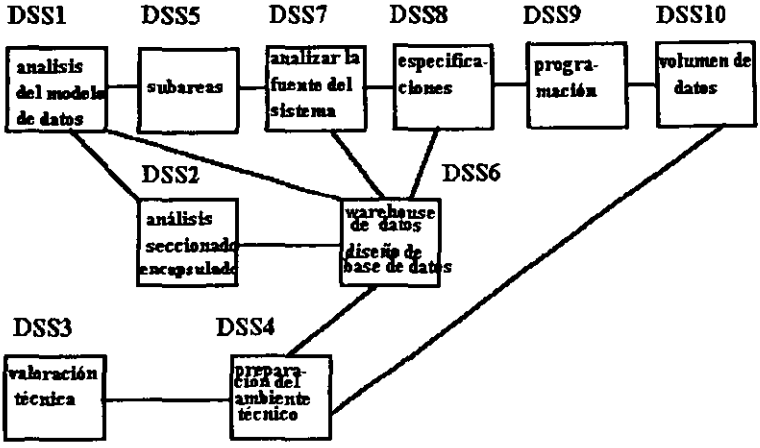


Figura III.2 Desarrollo DSS (Desarrollo Warehouse).

DSS4 - Preparación del Ambiente Técnico (UN SOLO TIEMPO)
Aquí es necesaria la identificación de los elementos técnicos para poder hacer una configuración adecuada. Como por ejemplo: enlaces requeridos, volumen de procesamiento, conflictos de procesamiento, volumen de tráfico, naturaleza del tráfico, etc.

DSS5 - Análisis de Subáreas
Seleccionar la subárea que se considere adecuada para iniciar la implementación, procurando que sea una subárea lo suficientemente grande para que sea significativa y lo suficientemente pequeña para ser implementada. Y así ir con cada una de las subáreas formando todo el sistema DSS. Enfocando el esfuerzo de implementación en términos de las subáreas.

DSS6 - Diseño del Sistema DSS de Registros
El diseño es basado en el modelo de datos, tomando las siguientes consideraciones:

- Considerar los diferentes niveles de granularidad, si es que los hay.
- Orientar los datos a las principales áreas de la corporación.
- Sólo la presencia de datos primitivos.
- La ausencia de datos que no sean DSS.
- Variaciones en tiempo de los registros de datos.
- Desnormalización física de los datos cuando es necesaria (siempre y cuando garantice la eficiencia).
- Creación de datos artificiales cuando las relaciones de datos en el ambiente operacional son trasladados al sistema DSS de registros.

DSS7 - Análisis de la Fuente del Sistema

Identificar la fuente de datos en el ambiente del sistema existente. Con lo que obtendremos el mapeo de los datos del ambiente operacional hacia el ambiente DSS.

DSS8 - Especificaciones
Formalizar la interfaz en términos de las especificaciones del programa, para que puedan ser utilizados los datos obtenidos del ambiente operacional para el sistema DSS de registros.

DSS9 - Programación
Incluye todas las actividades estándar de programación tales como:

- Pseudocódigo de desarrollo.
- Codificación.
- Detección de errores.
- Prueba del sistema.

DSS10 - Funcionamiento
Este paso es la ejecución de los programas DSS previamente desarrollados, teniendo así un sistema en funcionamiento.

La última fase de desarrollo en la arquitectura del ambiente es el uso del sistema DSS de registros con el propósito de análisis. Una vez que los datos en el ambiente del sistema DSS de registro es funcional, su uso puede comenzar.

Se enfatiza que hay varias diferencias esenciales en el desarrollo que se da en este nivel y el desarrollo en otros tipos de ambientes. La primera diferencia principal se da en el desarrollo, y es que el proceso de desarrollo comienza con los datos (Data Warehouse). La segunda diferencia es que los requerimientos no son conocidos hasta que comienza el desarrollo del proceso. La tercera diferencia (la cual es un producto de los dos primeros factores) es que el proceso que se hace es muy iterativo, de un modo muy heurístico. En otro tipo de desarrollo, hay siempre una cierta cantidad de iteración. Pero en el componente DSS de desarrollo ocurre después de que el sistema DSS de registro es desarrollado, en general la naturaleza de la iteración cambia. El proceso de iteración es una parte normal y esencial del proceso de desarrollo analítico, pero aquí se da mucho más que en otro tipo de sistemas.

Los pasos efectuados en los componentes de desarrollo DSS pueden ser divididos en dos categorías: la ocurrencia repetitiva del análisis (algunas veces llamada análisis "departamental" o "funcional"), y el proceso verdaderamente heurístico (o nivel "individual"). La figura III.3 muestra los pasos a efectuarse después de que el sistema DSS de registro es funcional y contiene la información a analizar.

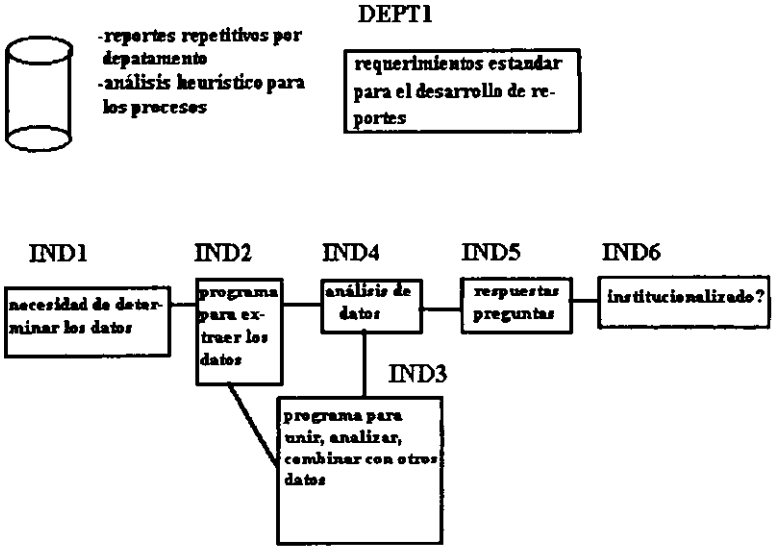


Figura III.3 Desarrollo Heurístico.

DEPT1 - Desarrollo repetitivo estándar
En el proceso analítico repetitivo (usualmente llamado reportes estándar) se maneja el proceso de los requerimientos normales que van surgiendo. Lo cual significa que los siguientes pasos son repetitivos:

M1 - Entrevistas, recolección de datos, JAD, plan estratégico y sistemas existentes.
M2 - Medir y Dividir.
M3 - Formalización de requerimientos.
P1 - Descomposición funcional.
P2 - Nivel de contexto 0.
P3 - Nivel de contexto 1-n.
P4 - DFD para cada componente.
P5 - Especificación de algoritmos, Análisis de Eficiencia.
P6 - Pseudocódigo.
P7 - Codificación.
P8 - Detección de errores.
P9 - Compilación.
P10 - Examinación.
P11 - Implementación.

Adicionalmente, las siguientes partes pueden darse en el tiempo apropiado:

GA1 - Revisión de alto nivel.

GA2 - Revisión del diseño.

IND1 - Necesidad de Determinar los Datos
En este punto, los datos en el DSS son seleccionados por su uso potencial en la satisfacción de los requerimientos de reportes. Mientras que el desarrollador trabaja desde una perspectiva adecuada, se debe comprender que los primeros dos o tres tiempos de esta actividad es iniciada sólo con algunos de los datos necesarios que serán obtenidos. El término de esta actividad es la selección de los datos para el siguiente análisis.

IND2 - Programa para extraer los datos
Una vez que los datos para el proceso de análisis son seleccionados, el siguiente paso es escribir un programa para acceder y extraer los datos. El programa debe escribirse de tal forma que pueda ser modificado muy fácilmente, ya que seguramente se ejecutará y modificará en numerosas ocasiones.

IND3 - Combinar, Unir y Analizar
Una vez que los datos han sido seleccionados, el siguiente paso es prepararlos para el análisis. Lo cual implica:

- Editar los datos.
- Combinar los datos.
- Refinar los datos.

IND4 - Análisis de datos
Una vez que los datos han sido seleccionados y debidamente preparados, la pregunta a contestar es "¿Los resultados obtenidos cubren las necesidades del análisis?" Si el resultado es que no se cubren, ocurrirá otra iteración. Si el resultado es adecuado, entonces el reporte final de preparación a llegado.

IND5 - Respuesta a la pregunta
El reporte final producido es con frecuencia el resultado de muchas iteraciones del proceso. Rara vez el resultado final es la conclusión de una sola iteración del análisis.

IND6 - Institucionalización
Un aspecto final a ser decidido es si el reporte final que se ha creado puede ser institucionalizado. Si hay una necesidad de ejecutar el reporte de manera repetitiva, esto hace posible que el reporte pueda ser asignado como un requerimiento y el reporte se reconstruya como una operación de ocurrencia regular.

Finalmente, Inmon hace notar la importancia del sistema DSS de registro, ya que es el centro de la metodología en su conjunto. Considerando que a través de este sistema se explota la información corporativa.

III.1.2 Metodología - Larry T. Vaughn

Vaughn inicia planteando que a lo largo de los años se han enfocado los esfuerzos en la disciplina del desarrollo de los sistemas de información, lo que ha resultado en diferentes metodologías que han competido entre sí. Todas ellas prometiendo ciclos rápidos de desarrollo con incremento de la calidad en el producto resultante. Desafortunadamente pocas de ellas han cumplido sus promesas. Las razones de estas fallas son muchas y no siempre obvias. Estas metodologías dan un marco de trabajo en el cual hay una secuencia de pasos lineales y rígidos, que están fundamentados en cinco premisas:

- todos los requerimientos pueden ser preespecificados
- los usuarios son expertos en la especificación de sus necesidades
- los usuarios y los desarrolladores tienen una buena visualización
- el equipo del proyecto es capaz de una comunicación no ambigua

Todas estas metodologías están basadas en la premisa de que se necesitan hacer las cosas bien a la primera vez. La naturaleza lineal de estas metodologías se muestra en la figura III.4.

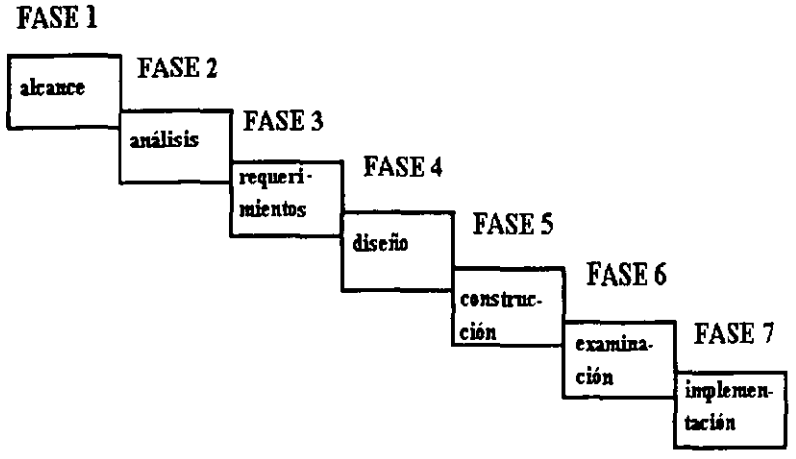


Figura III.4 fases de la metodología tradicional.

Ahora, una metodología que incorpora un manejo de la calidad total y técnicas de mejoramiento continuo, además de aprovechar las ventajas de la tecnología cliente/servidor sigue unos patrones significativamente diferentes. Lo cual es ilustrado en la figura III.5, que muestra que los proyectos son en realidad efectuados en tres etapas, que muestran los diferentes aspectos de la solución.

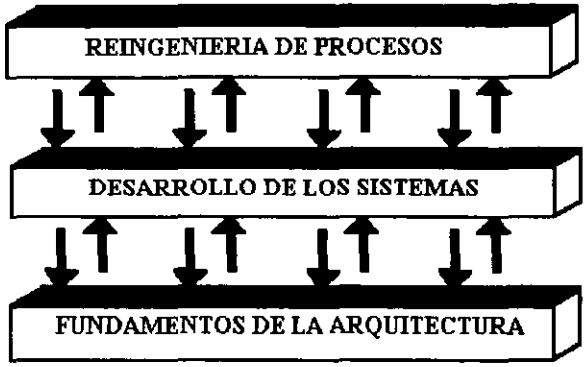


Figura III.5 Una visión más realista.

La etapa de **Reingeniería de Procesos** del proyectos se refiere a la identificación y definición de los problemas a ser direccionados, obteniendo una amplia comprensión de la situación y la identificación de los cambios que pueden ocurrir, definir y hacer prototipo de varias soluciones diferentes hasta que una solución óptima sea determinada, desarrollar la solución óptima, implementar la solución dentro del proceso operacional, y entonces monitorear y marcar el mejoramiento continuo para los nuevos procesos.

La etapa de **Desarrollo de los Sistemas** se refiere al modelado de la información y los flujos de trabajo necesarios para implementar las soluciones. Haciendo prototipos de varias soluciones hasta que una solución óptima es determinada y los requerimientos de la solución son especificados, diseñados e implementados en un sistema con calidad que proporcione el soporte necesario que fue especificado en el prototipo final. La implementación de la solución a los problemas es continuamente cambiada, extendiendo la funcionalidad del sistema resultante con base en el ciclo de mejoramiento continuo del proceso de solución.

La etapa de **Arquitectura** del proyecto se refiere a la identificación y validación de las herramientas ha ser utilizadas en la construcción de la aplicación, seleccionando y contando con el prototipo de la tecnología del cliente, de la red y del servidor a ser utilizados tanto en el prototipo como en la construcción de la solución. Definiendo la manera en la cual los componentes de la aplicación interactuaran, con base en el diseño físico del sistema y la estructura de los datos requeridos para la implementación (diseño de la base de datos física); lo cual determina la distribución de datos y funciones a través de la red, diseñando e implementando todas las interfaces externas del sistema asegurándose que los estándares de la empresa están siendo tomados en cuenta, y preparar la producción final para así iniciar las operaciones. Durante el mejoramiento continuo de la etapa de arquitectura del proyecto se podrá monitorear la eficiencia de la base de datos, optimizar los recursos de la aplicación cuando sea necesario y monitorear si hay algún problema con la capacidad de la red.

Además del fuerte acoplamiento en la concurrencia, cada etapa tiene internamente una serie de pasos cíclicos, son cuatro pasos continuos como se ilustra en la figura III.6. Haciendo referencia a la figura se irán definiendo los pasos de cada etapa.

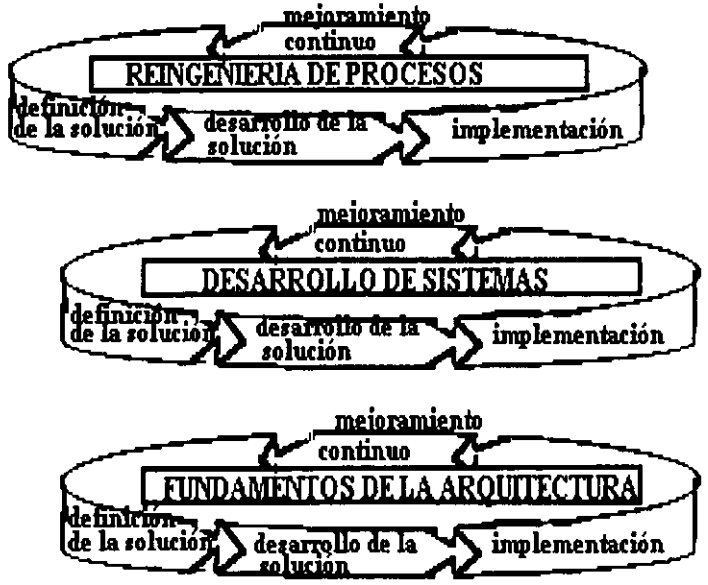


Figura III.6 El ciclo natural

DEFINICIÓN DE LA SOLUCIÓN

Reingeniería de Procesos

Consiste en la identificación del problema y determinación de la solución, lo cual es realizado en tres actividades:

Identificación del problema: Se deben de identificar las causas raíz. Para lo cual deben recoger tantos datos como sea posible que estén relacionados con los problemas existentes, oportunidades potenciales de mejoramiento, para así construir un entendimiento completo del ambiente actual, flujos de trabajo y de procesos. Entonces, el objetivo en este paso es la definición de los problemas, y las oportunidades de mejoramiento.

Determinación de las causas raíz: Cada problema es examinado en términos del ambiente actual y todas las posibles causas del problema son identificadas y verificadas. Las causas son descritas y clasificadas dentro de ocho categorías: procesos, equipo, personal, material,

administración, datos, métricas o el ambiente físico. Entonces, en este paso consiste de una concisa identificación, descripción y priorización de las causas raíz relevantes.

Definición de la solución: para la más prometedora de las alternativas se hace un prototipo, y si el prototipo falla, la lección aprendida es saber qué funciona y qué no, para que otras alternativas sean modificadas. Incorporando así la lección al siguiente prototipo. Este proceso continua hasta que una solución o soluciones son identificadas. Las soluciones resultantes son evaluadas en términos de la facilidad de implementación, efectos negativos, consecuencias si falla la implementación, habilidad en el trabajo necesario para implementar la solución, costo y tiempo de implementación. Entonces, el objetivo de este paso es una descripción completa de la(s) solución(es) ha ser implementada y un plan de acción para su completo desarrollo e implementación.

Desarrollo de Sistema

Esta etapa comienza un poco después de haberse iniciado la Reingeniería de Procesos. Esta constituida de tres actividades básicas:

Obtención del diseño del sistema actual: basicamente es una revisión del soporte del sistema actual que está involucrado dentro del espacio del problema, así como la documentación de las funciones, tecnologías y la estructura de la información que está siendo utilizada, para evaluar el impacto potencial de los procesos bajo estudio.

Determinación de las necesidades de los usuarios: un modelo de los procesos actuales, de los flujos de trabajo y de la información inicial de los principales objetos identificados dentro del espacio del problema son realizados para verificar el entendimiento y así aprovechar el esfuerzo de reingeniería para cuantificar las causas raíz e identificar las soluciones potenciales.

Definiendo los requerimientos funcionales: este paso trata de completar el modelo de información, desarrollando los procesos completamente especificados o modelo de flujo de trabajo para la solución óptima, un prototipo completo de las funciones automatizadas que son determinadas como necesarias para el soporte de la solución óptima, un documento con la descripción completa de los requerimientos de funcionalidad que serán proporcionados por el sistema automatizado y un plan detallado de acción indicando cómo el sistema reúne estos requerimientos que serán desarrollados en coordinación con el plan de implementación de la solución.

Arquitectura

Esta etapa comienza un poco después de la Reingeniería de Procesos y procede en coordinación con ella. Y se divide en tres actividades básicas:

Revisión de la tecnología actual.
Determinar la tecnología apropiada.
Seleccionar las herramientas de desarrollo.

DESARROLLO DE LA SOLUCIÓN

Reingeniería de Procesos

El objetivo es crear desarrollos piloto para el nuevo flujo de trabajo y de los procesos que se requieren para la definición de la solución, preparación de la documentación, programas de entrenamiento para la implementación de la solución y determinar el valor o calidad de las métricas que serán utilizadas para juzgar la efectividad de la solución óptima después de su implementación. Consiste de tres actividades básicas:

Las soluciones piloto.
Documento de las métricas de evaluación.
Preparación del entrenamiento.

Desarrollo del Sistema

Este paso se relaciona con la implementación de la estructura de la base de datos física, la construcción y la examinación del sistema de acuerdo con las especificaciones de arquitectura proporcionadas por la estapa de arquitectura, desarrollo y revisión de las funciones de conversión requeridas para facilitar la migración hacia el nuevo sistema y la ejecución del plan de migración y conversión. Lo anterior se realiza en cuatro actividades básicas:

Construcción de la estructura física de la base de datos.
Construcción de la aplicación.
Prueba de la aplicación.

Arquitectura

Consiste en el diseño físico y especificación de la aplicación y la base de datos; el diseño físico, construcción y examinación de todas las interfaces para los sistemas externos; la instalación, integración y examinación de la tecnología de estaciones de red, servidores o de redes que están siendo utilizadas; y el desarrollo y documentación completa del plan de conversión. Lo cual se realiza en seis actividades básicas:

Completar el diseño de la base de datos física.
Completar la arquitectura de la aplicación.
Desarrollar el plan de conversión.
Desarrollar las interfaces externas.
Preparación para el traspaso de la producción.
Instalación e integración de la tecnología.

IMPLEMENTACIÓN DE LA SOLUCIÓN

Reingeniería de Procesos

En este paso se realiza el entrenamiento de los usuarios, validación de la conversión de datos y la implementación de procesos y los cambios en los flujos de trabajo. Y se divide en dos actividades básicas:

Entrenamiento.
Validación de la conversión.

Desarrollo del sistema

Consiste en la ejecución del plan de migración como fue especificado y la atención inmediata a problemas o cualquier otro inconveniente relacionado con el sistema. Consiste de cuatro actividades básicas:

Instalación de la aplicación.
Iniciación de la conversión.
Desactivación de la aplicación vieja.
Iniciación de la operación.

Arquitectura

En este paso se lleva a cabo el monitoreo de la eficiencia, bajo las condiciones de la operación actual. Y consta de dos actividades básicas:

Monitoreo de la eficiencia de la red.
Monitoreo de la eficiencia de recuperación de los datos.

MEJORAMIENTO CONTINUO

Aunque esta etapa es ignorada o relegada al "programador de mantenimiento" en las metodologías tradicionales, actualmente éste es el periodo más importante en el ciclo de vida de los sistemas. Esto empieza antes de que la aplicación esté en producción y en un uso continuo, dado que se refina y se extiende sucesivamente para la funcionalidad de la aplicación. Y así los usuarios estén más conformes con las capacidades y los cambios de la reingeniería de procesos que se van realizando con la operación diaria.

Reingeniería de Procesos

Se recogen las métricas de eficiencia y se verifica que los resultados esperados son obtenidos. Cuando existe una variación en lo que se esperaba, se determina el por qué de la variación y se inician las medidas correctivas que se requieran. Y entonces probablemente se repita el ciclo para extender la funcionalidad del sistema o para la solución de nuevos problemas.

Desarrollo del Sistema

Se realizan monitoreos rutinarios para la optimización de la eficiencia de la aplicación, por si llega a aparecer algo anormal actuar rápidamente, además de llevar a cabo la reingeniería de procesos en su esfuerzo de mejoramiento continuo. Y también abarca más prototipos, construcción, examinación e implementación de la funcionalidad extendida.

Arquitectura

Se realiza un monitoreo rutinario para optimizar la red y la eficiencia de la base de datos, para actuar rápidamente cuando aparezca algo anormal, además de realizar lo que le corresponde en cuanto a la reingeniería de procesos y al desarrollo de sistemas en el mejoramiento continuo. Con frecuencia también involucra la evaluación, selección e introducción de nuevas tecnologías especialmente en ambientes de manufactura como la generación de nuevos procesos de control y la obtención de datos de las nuevas tecnologías.

Larry Vaughn considera que su metodología, antes expuesta, junto con la tecnología cliente/servidor puede introducir a las modernas organizaciones corporativas hacia la calidad administrativa y al mejoramiento continuo, proporcionando la oportunidad de reestructurar radicalmente la Administración de Sistemas de Información (MIS Management Information Systems). Ya que se debe considerar que no se puede aprovechar por completo los beneficios de la nueva tecnología, si se utilizan metodologías del pasado.

Además de considerar que una metodología es un proceso que debe ser flexible para que se puedan incorporar técnicas existentes y nuevas, además de las herramientas que se requiera. Pues hay que tomar muy en cuenta que la herramientas y las técnicas son una parte importante para la ejecución de alguna metodología.

III.1.3 Una Metodología Cliente/Servidor - David Vaskevitch (A Cliente/Server Methodology)

Vaskevitch apunta que existe la necesidad de utilización de una metodología, independientemente de las múltiples herramientas poderosas que puedan utilizarse en el desarrollo de un sistema. Este punto por lo regular se pasa por alto cuando se construyen sistemas pequeños o medianos. Pero al hacer el desarrollo de un sistema grande es indiscutible la necesidad de utilizar una metodología, y particularmente en los Sistemas Distribuidos de tipo cliente/servidor.

Existen las metodologías tradicionales que carecen de aspectos muy importantes para el desarrollo de aplicaciones gráficas cliente/servidor. Algunos de los puntos más importantes a cuestionar son los siguientes:

- ¿Cómo, cuándo y dónde se efectúa lo relacionado con la distribución de procesos y de datos?

- ¿Cuándo es efectuado el proceso de diseño de la interfaz de usuario y la naturaleza gráfica de la aplicación?
- ¿Cómo se pueden administrar o alterar los procesos para que el código sea menos monolítico; y tener aplicaciones flexibles e interoperables, como piezas reusables?

Para el desarrollo de un nuevo marco de trabajo para diseñar y construir aplicaciones, se tiene que comenzar con un modelo sobre el conjunto de procesos. La figura III.7 muestra las tres capas de la arquitectura de la aplicación, en tres etapas que son los procesos de planeación, diseño y desarrollo. A continuación se verá cada una de éstas:

	Conceptual	Lógico	Físico
Aplicación	Flujo de Trabajo	Secuencia de Formas	Formas
Reglas de Negocio	Flujo de Procesos	Modelo de Objetos	Programas
Base de Datos	Modelo de Datos	Esquema de la Base de Datos	Tablas, Índices

Figura III.7 Planeación, Diseño y Desarrollo de una aplicación.

- Conceptual: En la primera etapa es un análisis para visualizar los requerimientos y desarrollar un primer bosquejo de todo el diseño. Desde el punto de vista de la arquitectura esto es equivalente a contar con un plan base. En tal diseño se considera el flujo de trabajo de oficina a oficina y de persona a persona, sin tomar en cuenta los detalles de las formas o interfaces. Esto es un análisis de alto nivel de abstracción. Después se pasa al nivel de procesos, los cuales son expresados con burbujas que se entrelazan con líneas que indican el flujo. Y respecto a la base de datos, se tiene a un alto nivel que muestra de forma integral la empresa, además de modelos divisionales.

- Lógico: Este diseño toma los detalles de las reglas del negocio y refina el diseño para mostrar de forma adecuada dichas reglas. Las burbujas son exploradas con más detalle en procesos iterativos y diagramas de requerimiento/acción que muestran cómo funcionan las burbujas de procesos. Las formas detalladas son bosquejadas en secuencia, mostrando exactamente la secuencia de pasos requeridos para especificar una tarea completa. Y al nivel de la base de datos, el modelo de alto nivel es transformado en un diagrama entidad-relación clásico que muestra un esquema potencial de la base de datos que cuenta con los aspectos fundamentales respecto a la consistencia y significado de la base de datos.

- Físico: Finalmente, el diseño es trasladado al sistema actual. La primera entrada a esta etapa son las consideraciones de eficiencia. Los flujos de procesos son convertidos en piezas específicas de código. Los programadores diseñan formas individuales, refinan prototipos en detalladas interfaces gráficas, escriben el código de su comportamiento. El diseño de la base de datos es refinada, normalizada si lo necesita y los índices son especificados. Y con herramientas de diseño se genera la estructura de la base de datos.

Ahora, es de suma importancia considerar la forma y el tiempo adecuado para pasar de una etapa a otra. Aunque por lo general cuando se hace un sistema los usuarios y los programadores quieren obtener el sistema físico rápidamente, pero sin embargo sino se hace un diseño adecuado no se tendrá una buena base de datos. Esta tendencia esta respaldada por la existencia de poderosas herramientas que trabajan completamente a nivel físico; construyendo formas, generando código, ejecutando la base de datos y *queries*. Por lo que el procesos de pasar de una etapa a otra es importante (figura III.8).

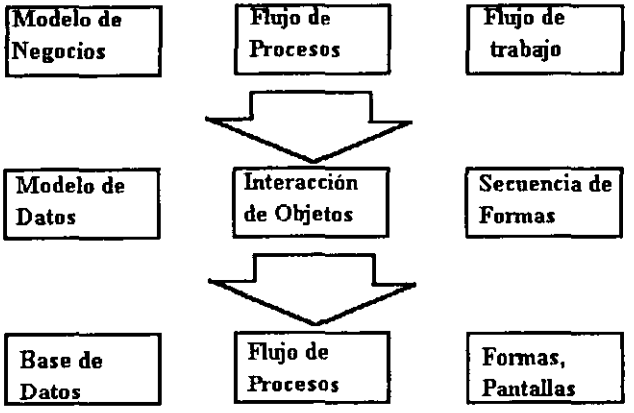


Figura III.8 Movimientos desde el nivel conceptual al lógico y al físico, dentro del diseño de una aplicación.

En el nivel conceptual, a nivel gerencial se tiene la tendencia a enfocarse en el modelado de procesos: girando la reingeniería en torno a este modelo. La gerencia administrativa y los usuarios finales tienen la tendencia a darle mucho peso al modelo de flujo de trabajo. El flujo de trabajo es actualmente un concepto muy popular en los círculos de computación. Los cuales proporcionan un buen modelo conceptual que permite hablar de la aplicación en forma conceptual.

La diferencia fundamental entre el modelo conceptual y el modelo lógico es la necesidad de comenzar a poner atención en los detalles de las reglas del negocio. Un diagrama de procesos lógico es similar a un diagrama de flujo, aunque con la perspectiva mas refinada, refiriéndonos a diagramas de flujo de datos, diagramas orientados a objetos, entre otros. En el contexto de la reingeniería, cada proceso del diagrama de flujo es trabajado por los gerentes en la organización con el conocimiento detallado de cómo los procesos de la compañía trabajan realmente. Por lo que es el momento adecuado en que puede introducirse la reingeniería de procesos.

Las interfaces de usuario tienen su diagrama de flujo equivalente en los diagramas de secuencia de formas. Su importancia es que muestran un panorama general de las opciones que deben tener disponibles los usuarios, sin requerir detalles, obteniendo ventajas en tiempo y claridad conceptual.

En cuanto a la base de datos, el concepto de modelo de base de datos lógica es ampliamente difundido. Llamados diagramas entidad-relación (ERDs), estos modelos muestran todos los elementos llave de la base de datos mientras que ignoran detalles técnicos relacionados con la normalización, índices, formatos de almacenamiento, entre otros.

El modelo conceptual, en sus tres partes, responde las siguiente preguntas:

¿Cómo se visualizará la nueva aplicación?
¿Cómo cambiará el negocio?
¿Cómo se encaminarán los requerimientos de nuevos procesos del negocio?

El modelo lógico es el paso en que se discute la siguiente pregunta:

¿Qué pasos ocurrirán al hacer cada gran procesos descrito conceptualmente?

El modelo lógico proporciona la forma en que trabajará la nueva aplicación. Pero ahora, cómo será llevada a cabo la aplicación, lo que da origen a las siguientes preguntas.

¿Puede ser implementado?
¿Tendrá una eficiencia adecuada?

El nivel físico responde a estas preguntas. El nivel físico es el proceso en donde la aplicación es escrita. La base de datos es definida y construida. Las formas son diseñadas, su respectivo código es escrito, y se lleva a cabo un examen de su utilidad. Los procesos de las reglas del negocio son trasladadas al código utilizando algún lenguaje de programación o alguna otra herramienta. El aspecto físico tiene un componente de diseño y uno de implementación. El proceso de diseño involucra específicamente cómo las reglas del negocio son implementadas en términos de objetos, procesos, módulos y estructuras de datos. En cuanto a la interfaz de usuario, el diseño involucra un extenso prototipo, el uso de herramientas gráficas, y la eventual convergencia en una interfaz final. Y por último en la base de datos, el proceso involucra archivos detallados y diseño de tablas, desarrollo de una estrategia de indexación, entre otros aspectos.

Los dos pasos considerados en el modelo de diseño clásico, como se ve en la figura III.9.a, se acondicionan para soportar el nuevo modelo. El resultado es un nuevo marco de trabajo y un nuevo aproximamiento organizacional.

El primero y más obvio cambio en el nuevo diagrama de procesos, figura III.9.b, es la presencia de cuatro divisiones de diseño que se hacen en paralelo a diferencia de las dos que eran. Comparando el nombre de cada caja, tres son nuevas. La Base de Datos, Procesos y la Interfaz de Usuario son el correspondiente con la Base de Datos, Reglas del Negocio y la capa de Aplicación, respecto a la arquitectura de la aplicación antes presentada. Conceptualmente el diseño se enfoca ampliamente a los procesos, pero en la etapa de diseño lógico y particularmente en la física se enfoca al comportamiento detallado que ocurre dentro de cada proceso.

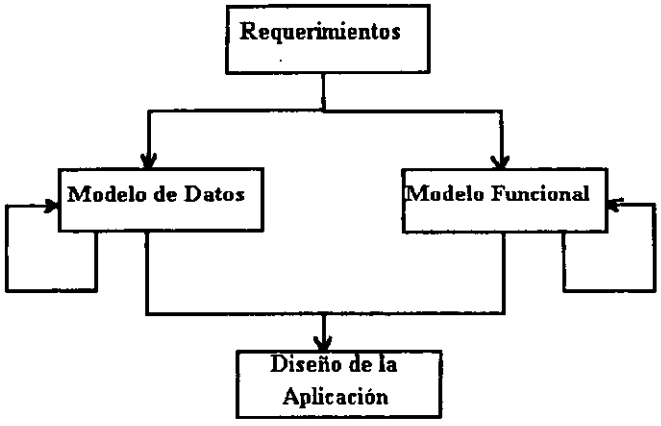


Figura III.9.a El centro de la mayoría de las metodologías de diseño.

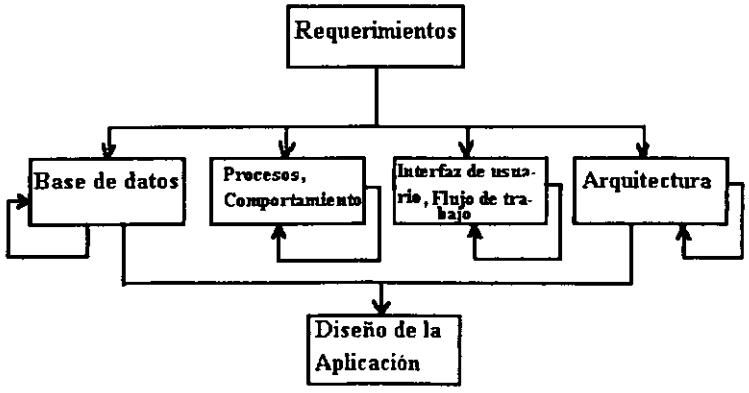


Figura III.9.b Un nuevo diagrama de procesos, con cuatro divisiones del diseño en paralelo en lugar de dos.

El proceso de diseño de la Arquitectura, figura III.10, en paralelo con las otras tres, direccionando las cuestiones de estructura y forma. El diseño de la arquitectura dice finalmente qué hacer dentro de cada capa.

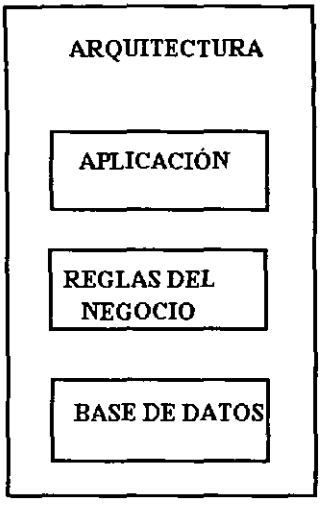


Figura III.10 El rol de la arquitectura en el ensamble de la aplicación.

En cuanto al aspecto del personal requerido para llevar a cabo un proyecto, en los sistemas pequeños una sola persona puede realizar todas las actividades requeridas. En un sistema mediano puede darse una división de acuerdo a las habilidades. Pero en un sistema grande se necesita una verdadera organización en la división de las tareas. Personas con características específicas para fungir como diseñadores y modeladores de la base de datos, diseñadores de procesos, diseñadores de la interfaz de usuario o como arquitectos. Estas cuatro actividades serán llevadas a cabo en forma paralela, llevando una sincronía y una interacción, tal que unas a otras ejercen mutua influencia.

Después de que Vaskevitch hace la comparación entre las antiguas metodologías y la nueva metodología expuesta, la cual cubre las necesidades para realizar un sistema en la actualidad, introduce el concepto de **OBJETO**. Y al respecto puntualizaré lo más relevante que expone el autor en cuanto a este concepto.

Explica, que el truco para entender los objetos está en separar los grandes y los pequeños beneficios, y también reconocer qué es lo que se puede hacer ahora, y lo que tal vez se pueda hacer en un futuro. Y recordar la razón original por la cual se introducen los objetos: encontrar una segunda técnica que complemente la arquitectura de la aplicación para proporcionar una mejor comprensión de los grandes sistemas. Además de visualizar algunos otros beneficios, tales como la reutilización, flexibilidad e interoperabilidad.

METODOLOGÍA

Los objetos a través de la encapsulación, puede proporcionar los siguientes beneficios:

Análisis orientado a objetos	**Si**
Diseño orientado a objetos	**Si**
Programación orientada a objetos	**Tal vez**

Tanto la tecnología cono la filosofía del diseño orientado a objetos pueden ser divididos en tres niveles: análisis, diseño y programación. Los cuales, corresponden aproximadamente a los tres niveles del modelado conceptual, lógico y físico. El acercamiento a la orientación a objetos es posible gracias a estos tres niveles. Por lo que es más factible tener una orientación a objetos hasta cierto nivel, que tener una orientación a objetos por completo.

Los objetos pueden ser utilizados nada más para la construcción del análisis y el diseño. El análisis orientado a objetos requiera mapeos en las asignaciones de los objetos, con la definición formal de las interfaces. Los diagramas son utilizados para mostrar cómo los objetos se comunican unos con otros. El diseño orientado a objetos toma el modelo conceptual extrayéndolo de la fase de análisis, tomando los detalles de las reglas del negocio y dentro de todos los detalles encontrados asigna la cooperación entre los objetos a través de la definición formal de las interfaces (métodos). Lo que se realiza por medio de diagramas de burbujas, los que probablemente se han utilizado para un análisis y diseño orientado a objetos, aún sin conocerlo.

Para el propósito de diseñar sistemas distribuidos cliente/servidor, los beneficios de la orientación a objetos puede obtenerse sin programar el sistema completamente orientado a objetos. Para pensar en términos de objetos y entonces implementar en términos de procesos, el resultado puede ser un sistema completamente orientado a objetos, a un alto nivel, con reutilización, interoperabilidad, flexibilidad y la mejor operación distribuida, pero sin programarse orientado a objetos.

Si la meta es la reutilización, entonces el desarrollo orientado a objetos en procesos es significativo al final. Pero si el objetivo es una orientación a objetos definida puramente, entonces la programación orientada a objetos es el único camino a seguir. Entonces, el resultado no es necesariamente un mejor sistema, y en términos de costo se tiene el riesgo de fallar, además de incrementarse el tiempo de entrenamiento, entre otras cosas.

Los procesos son objetos desde una perspectiva de implementación. Y los objetos son procesos desde una perspectiva de diseño. Al unirlos surge una solución completa. Se utilizan los objetos para conceptualizar los procesos de negocios y sus interacciones. Entonces, se usan los procesos (técnicamente) para implementar estos objetos cuando llega el tiempo de construir el sistema. Por supuesto, el conocimiento interno del sistema en que se trabaja, a un bajo nivel, puede no ser orientado a objetos porque los procesos son muy extensos como para utilizar pequeños objetos. A un nivel más detallado, los objetos son difíciles de visualizar. Pero, en un lenguaje de programación orientada a objetos llegan a ser más accesibles.

Son dos las perspectivas fundamentales que se manejan para la construcción de un nuevo marco de trabajo. Al nivel más alto se realiza la introducción de dos mecanismo fundamentales de abstracción que permiten dividir el gran sistema en partes mas pequeñas. Y la asignación fundamental de modelos que se utiliza para manejar el diseño subsecuente. El orden para ver esta división es comparando la nueva metodología con la anterior. La figura III.11 hace esta comparación.

	Pasado	Futuro
Arquitectura		Modelo de interacción de objetos
Aplicación		Objetos; Prototipos
Procesos; Comportamiento	Modelo funcional	Objetos; Modelo funcional
Base de datos	Modelo de datos	Modelo de datos

Figura III.11 El camino viejo y nuevo para el diseño dividido en cuatro fases.

La figura III.12 muestra una pequeña aplicación que puede ser descompuesta utilizando los mecanismos de abstracción. En primera, el sistema completo tiene que ser dividido arquitectónicamente en tres capas: aplicación, reglas del negocio y base de datos. Pensando antes de llevarlo a cabo, que esta arquitectura de la aplicación ayuda a pensar en el manejo en términos del flujo de trabajo, flujo de procesos y a la comprensión del modelo de datos del negocio. Además, separar la interfaz, el comportamiento y los datos; ya que esto ayuda a asegurar los componentes interoperables antes de que el diseño sea llevado a cabo. Después, los componentes ejecutados en cada capa son conceptualizados y diseñados en términos de la separación de los componentes que interactúan. El diagrama muestra los tipos de figuras que pueden ser dibujados para mostrar cómo interactúan los componentes unos con otros.

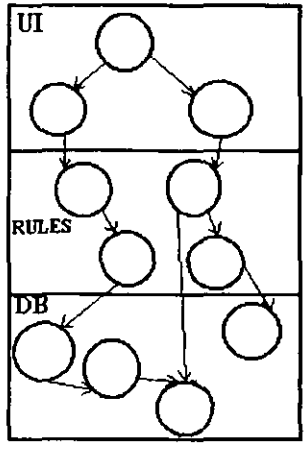


Figura III.12 Descomposición de una pequeña aplicación.

Visualizando la tabla con más detalle, hay básicamente tres procesos de diseño fundamentales que se usan:

Modelos dinámicos (interacción de objetos): En toda la arquitectura, el flujo de trabajo de los usuarios y los modelos de procesos de alto nivel (reglas) son mejor descritas en términos de objetos por el modo en que interactúan unos con otros. Las figuras de burbuja mostradas son más evocativas que descriptivas, pero la idea básica es correcta: capturar los componentes fundamentales y las interacciones de unos con otros. Los modelos dinámicos son también bastante apropiados para describir las transacciones y *queries* que son mapeados a la base de datos y sus interacciones con otras transacciones y componentes.

Modelo estructural estático: El modelo de datos en particular se describe mejor mediante un modelo, con cajas y flechas que lo muestran adecuadamente, ya que capturan la estructura estática de los componentes individuales (entidades y sus atributos, objetos y sus intefaces) y las relaciones entre estos componentes. Los procesos de las reglas del negocio, componentes activos que pueden ser descritos estáticamente, pero el modelo estático da por sí mismo la base de datos. Históricamente, los modelos de datos han sido basados en **diagramas entidad-relación**. Más recientemente, los aficionados a los objetos con frecuencia hablan acerca del modelado de objetos. Existe muy poca diferencia entre los dos, viéndolo desde el punto de vista estático. Pero es verdad, que un modelo de objetos también puede mostrar los métodos (operaciones) que pueden ser aplicados para un objeto particular o una tabla. Por lo cual, el diagrama entidad-relación puede facilitar extender el contenido de esta información. Por lo tanto si hablamos de modelo de objetos estático o de un diagrama entidad-relación, se habla de lo mismo.

Prototipos: Hasta ahora, todos los aspectos del procesos de diseño que se han discutido están basados en el modelo manejado. Por lo que el modelo de diseño, es un principio central detrás de todo lo que se hace. Pero de lo que no se ha tratado es sobre la interfaz de usuario. Para esto, hay que tomar en cuanta que las actividades antes descritas requieren un alto nivel de análisis y abstracción, lo cual no es requerido para el diseño de la interfaz de usuario. En contraste, el diseño de la interfaz de usuario requiere espontaneidad, sin estructuración y mucha creatividad. Por lo que no es diseñada con base en el modelo, por lo tanto es un proceso creativo, con base en prototipos y un tanto artístico. Pero independientemente de este aspecto, las dos técnicas de abstracción son el centro de la metodología que proporcionará la solución.

III.1.4 Principios de Diseño y Desarrollo - Paul E. Renaud (Principles of Design and Development)

Este autor nos muestra una metodología de diseño y desarrollo para sistemas cliente/servidor, pero sólo en el aspecto de la arquitectura del sistema. A continuación, se puntualizarán las veinticinco reglas de dedo que da Renaud como punto de partida antes de exponer la metodología:

1. Poner todo el procesamiento que sea posible dentro del cliente.

2. Hacer todas las actividades intensivas de cómputo en el cliente. Dichas actividades incluyen las siguientes:
    - Todo dibujo de líneas y gráficos bitmap
    - Clasificación y búsqueda dentro de las tablas
    - Todas las operaciones aritméticas de punto flotante y punto fijo
    - Compresión y descompresión de datos
    - Inferencia y otras reglas manejadas
    - Encriptación/desencriptación de datos
    - Captura de sonido o *playback*
    - Captura de video o *playback*
    - Compilación del código fuente
    - Optimización de *queries*
    - Análisis estadístico
    - Procesamiento de texto
3. Separar los contextos de procesamiento que se dan al actuar en el ambiente de un solo usuario (validación al introducir en pantalla, edición) o en el ambiente multiusuario (el modo del flujo de trabajo es esencialmente multiusuario, por lo que el estatus del trabajo de cada usuario puede ser conocido para rutear algún otro asunto).
4. Administrar todos los recursos compartidos con servidores de procesamiento.
5. Mantener una visión virtual de los servidores durante el diseño. Un exitoso diseño cliente/servidor requiere una perspectiva virtual. Durante la etapa de diseño se debe pensar de forma lógica y no física, por lo que los servidores ofrecerán servicios virtuales.
6. Administrar todos los datos con servidores de procesamiento.
7. Evitar la centralización de servicios. Administrar los recursos compartidos usando servidores de procesamiento no implica usar necesariamente un solo servidor. Los sistemas cliente/servidor son inherentemente distribuidos.
8. Estar seguro que los datos locales son verdaderamente locales y administrados de la misma forma.
9. Utilizar procesamiento por niveles para mejorar la escalabilidad.
10. Asegurar que el servidor administre las grandes transferencias de datos. Y no dejar que los clientes decidan unilateralmente cuándo se inician dichas transferencias. Esto es particularmente importante en grandes sistemas. Si algún sondeo es realizado, asegurar que haya sido hecho por el servidor, de otra forma el diseño no puede ser escalable.
11. Minimizar la transferencia de datos entre cliente y servidor. A mayor transferencia de datos, se incrementan las oportunidades de tener dificultades y la necesidad de la recuperación de errores.
12. *Cache* para modificaciones o datos estáticos. Ésta evita tener transferencia de datos más de lo necesario.
13. Compresión de largas transferencias de datos si es posible.
14. Puntos de chequeo en grandes transferencias de datos.
15. Utilizar clientes portadores pata implementar el flujo de datos multiservidor. Usar el paradigma cliente/servidor para simular un procesamiento punto-a-punto donde sea necesario. Lo que reduce la complejidad del diseño y permite el rehuso de las misma herramientas de desarrollo que son utilizadas en la programación cliente/servidor para la implementación de flujos de datos multiservidor.

16. Usar un esquema de llamadas para implementar un filtro de procesamiento donde sea necesario. Los filtros de procesamiento usualmente involucran un camino de servidor-a-servidor. Un ejemplo de esto es un monitor de seguridad que filtra ciertas salida de datos antes que éstas sean regresadas al cliente.
17. Diseño para el caso de entrar en un estado de falla. Y de esta forma si se presentan fallas no tener una crisis operacional, así el sistema pueda seguir funcionando adecuadamente.
18. Administración centralizada dentro de un orden distribuido.
19. Diseño de la administración remota y el monitoreo.
20. Construir perímetros de seguridad en los sistemas y aplicaciones.
21. Analizar la confiabilidad de la arquitectura. El uso de técnicas de modelado de la confiabilidad para analizar la vulnerabilidad dentro de la arquitectura. Se puede explotar la redundancia natural que ocurre dentro de la arquitectura por niveles, ya que la topología de la red puede facilitar cambios para proporcionar rutas alternativas, los cuales pueden ser utilizados cuando una falla ocurre.
22. El uso de **ordenamientos encadenados** para los puntos conflictivos, en los que se puede presentar un embotellamiento. En el análisis de embotellamiento se toma el tiempo en que los componentes son introducidos a la cadena, y el orden en que irán saliendo lo más rápidamente posible para controlar los embotellamientos de la mejor forma posible.
23. El **ordenamiento encadenado** no dice cuántos están tratando de utilizar un componente, el uso de los **modelos de colas** evitan los retrasos debido a la sobre utilización de los componentes.
24. Diseño de compatibilidad con la generación de clientes y servidores.
25. Medida independiente de la demanda de capacidad ofrecida. Por lo regular existen varias combinaciones posibles cuando se está decidiendo dónde será colocada la aplicación dentro de la red. La clave para obtener la solución adecuada es teniendo una visión clara de que los vectores de carga de la aplicación son independientes de la capacidad potencial de las diferentes máquinas. Entonces se pueden mapear estos vectores de carga en diferentes alternativas de configuración hasta que se encuentre un balance costo-efectividad entre la demanda y la capacidad ofrecida.

Ahora, se expondrán los pasos requeridos en la metodología. El autor enfatiza que la clave de esta metodología está en enfocarse primero en la estructura lógica del sistema antes de involucrarse en los detalles físicos. Dicha metodología se muestra en la figura III.3.13.

1. El primer paso es asignar los requerimientos funcionales entre los clientes y los servidores.

- Un buen punto de comienzo es identificar el balance de procesamiento. El procesamiento cliente/servidor ocurre en algún punto de balanceo a lo largo del segmento de procesamiento, aun en los extremos como lo podemos ver en la figuras III.3.14.a y b. Si el cliente es solo un *front-end GUI* para una aplicación anfitrión, dicho sistema está en el principio de ser cliente/servidor. En el otro extremo, el servidor puede simplemente ser un motor de base de datos SQL para red, recobrando y almacenando registros en la base de datos. Aún así, ésta es una aplicación cliente/servidor, ya que el procesamiento SQL ocurre en la capa de aplicación.

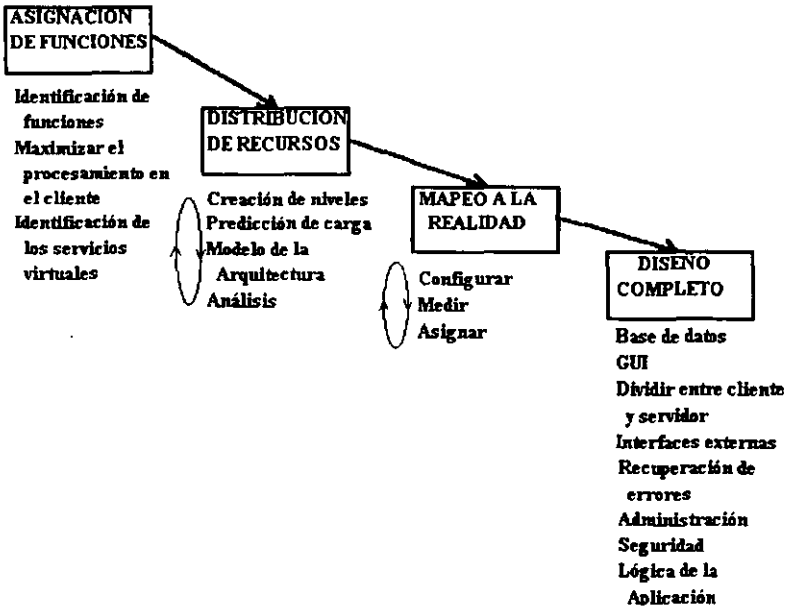


figura III.3.13 Metodología de desarrollo.

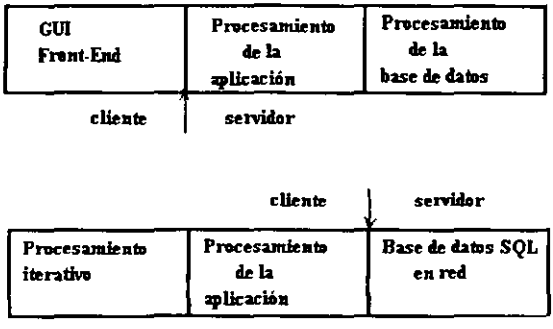


Figura III.3.14.a Extremos del procesamiento cliente/servidor.

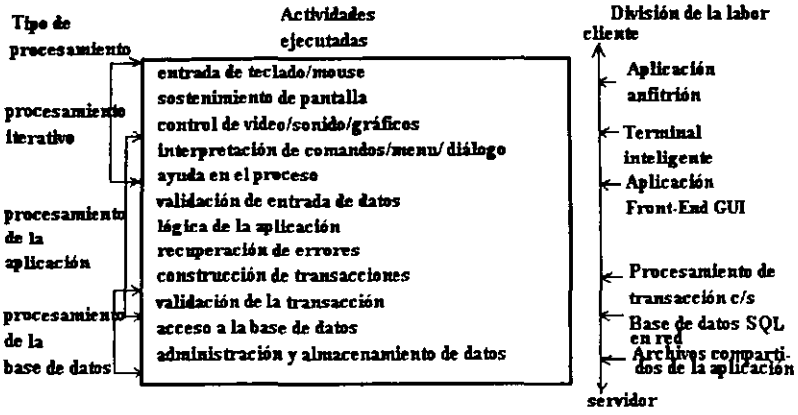


Figura III.3.14.b Continuidad de procesamiento.

- Aplicar las reglas de dedo que gobiernan los contextos monousuario y multiusuario, con la meta de tratar de poner tanto procesamiento como sea posible en el cliente.
- Identificar los servidores potenciales para tratar de que todas las funciones o recursos compartidos se consideren como servidores virtuales.

2. El segundo paso es distribuir los recursos entre los servidores virtuales. Esto es un ciclo iterativo de los siguientes pasos:

- Distribuir los datos y otros recursos entre los servidores virtuales, creando niveles de procesamiento donde sea necesario.
- Predecir la carga resultante de la aplicación por la distribución de los recursos. Esto puede requerir la simulación de la aplicación que será utilizada para medir ciertas partes del vector de carga.
- Las iteraciones del modelo, debido a las implicaciones de fiabilidad y eficiencia en la distribución de los recursos.
- Analizar los resultados de los modelos, y repetir el ciclo hasta que la arquitectura sea la más adecuada.

3. El siguiente paso es mapear los servidores virtuales a configuraciones físicas. Esto también involucra un ciclo de lo siguiente:

- Seleccionar la tecnología y los productos para la configuración.
- Examinar la integración de estos productos para asegurar su interoperatividad y eficiencia.
- Mapear la capacidad de demanda en la actual configuración, y repetir el ciclo hasta que se obtiene un equilibrio óptimo.

4. El último paso es para completar el diseño de la arquitectura y en el diseño de alto nivel establecido en los pasos anteriores. Ocasionalmente, se podrá descubrir algo durante el

diseño detallado que no pudo haber sido contemplado en los pasos anteriores. Si esto sucede, se pueden revisar las suposiciones hechas anteriormente y ver cómo puede ser cambiada la arquitectura. Antes de que algún desarrollo se realice (o prototipo), se debe estar seguro de lo siguiente:

- El diseño de base de datos completa.
- El diseño de la interfaz gráfica de usuario (GUI) completa.
- Dividir la interfaz entre el cliente y el servidor.
- Detallar alguna interfaz con otro sistema.
- Diseñar cómo trabajará la recuperación de errores.
- Diseñar cómo el sistema podrá ser administrado remotamente.
- Diseñar los perímetros de seguridad.
- Diseño de la lógica de la aplicación completa.

III.1.5 Construyendo un Nuevo Sistema Automatizado - Paul Kavanagh (Building a New Automated System)

El enfoque de Paul Kavanagh es dirigido al Desarrollo Rápido de Aplicaciones, lo cual es impulsado por la necesidad de obtener resultados rápidos en la dinámica de los sistemas distribuidos actuales. Para lo cual, menciona que hay tres requisitos fundamentales para un desarrollo en poco tiempo:

1. Inversión: uso de herramientas integradas CASE y una metodología organizada para desarrollar grandes sistemas con importantes componentes de generación de código.
2. Alcance limitado: uso de herramientas de desarrollo cliente/servidor para desarrollar pequeños sistemas con un alcance controlado.
3. Reuso: utilización de trozos de código previamente escritos y examinados.

Existen tres opciones para el Desarrollo Rápido. La primera opción es la metodología RAD de la escuela de la ingeniería de la información, popularizada por James Martin. Hasta hace poco ésta fue el mejor modelo de desarrollos exitosos. La segunda opción es la metodología de desarrollo rápido de aplicaciones, la cual es basada en la arquitectura cliente/servidor. Y que se fundamenta en las herramientas que tienen base en la tecnología de bases de datos relacionales. Ésta es la metodología que expone Kavanagh, para el desarrollo de aplicaciones de información intensiva. La tercera opción es un modelo de componentes, pero es una metodología emergente que probablemente llegue a ser un modelo dominante en el futuro.

El principio fundamental de RAD es el uso de actividades paralelas para comprimir el ciclo del tiempo de desarrollo. JRP (Joint Requeriments Planning) y JAD (Joint Aplication Design) reemplazan las series de entrevistas y revisiones en sesiones de grupos (paralelos). El desarrollo se divide en múltiples actividades paralelas de diseño/construcción que van ocurriendo en paralelo. Para monitorear esto, son necesarios un modelo de datos complejo y una buena administración del proyecto. El proyecto debe ser planeado y revisado de una manera iterativa en tiempo real.

METODOLOGÍA

También es importante hacer uso de un buen *staff* y una adecuada metodología, la cual es específicamente diseñada para el desarrollo rápido de sistemas cliente/servidor, la cual debe asegurar que:

- Los objetos del proyecto son bien definidos y comprendidos por todo el *staff* del proyecto
- Las tareas son completamente definidas y asignadas al *staff* de acuerdo a sus habilidades
- Realizar revisiones de calidad en cada una de las fases del proyecto

En el comienzo de un proyecto, los estándares de desarrollo del sistema son establecidos. En todo el proyecto, el *staff* debe tener acceso al plan del proyecto que les permitirá revisar los objetivos del proyecto, sus asignaciones y el criterio de revisiones establecido para las tareas en las cuales son asignados.

La metodología del desarrollo rápido de aplicaciones utiliza prototipos que van evolucionando (figura III.15). La meta es producir una versión inicial que contenga del 60% al 80% de la solución en un 20% del tiempo.

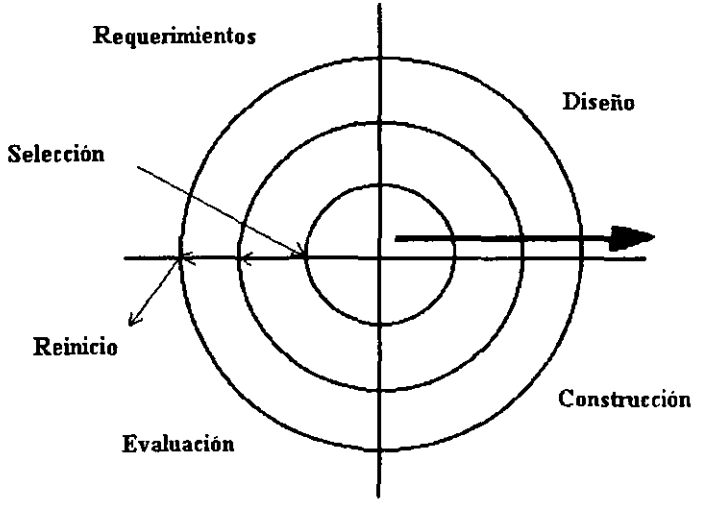


Figura III.15 EL Desarrollo Evolutivo en Espiral.

La metodología tiene seis fases:

- selección y planeación
- requerimientos
- diseño
- construcción
- decisión

- reinicio

Cada fase de la metodología va dando pie a la siguiente fase, pero esto no es un proceso rígido. Las siguientes son unas reglas importantes a seguir como estándar en la administración del proyecto:

1. Saber quién es el cliente
2. Documentación formal bien identificada
3. Permanecer familiarizado con el plan del proyecto en todo momento
4. Alcance de los cambios que pueden ser reconocidos y pactados durante los procedimientos de control

A continuación se expondrán cada una de las etapas de la metodología.

ESTABLECER EL PLAN DEL PROYECTO

El propósito de esta etapa es para organizar el proyecto tal que pueda ser exitosamente terminado. Además de establecer los principales aspectos en un plan de proyecto detallado. Negocios del proyecto.- este documento se desarrolla en esta fase. Inicialmente se revisan los documentos internos disponibles, tales como reportes, documentación del sistema actual y algún plan relevante en el área. Y trabajando con los usuarios finales de alto nivel, se puede confirmar lo siguiente:

- Alcance (qué problemas del negocio tienen que ser resueltos)
- Objetivos (qué funciones debe tener el sistema)
- Nivel de compromiso de la alta administración con el proyecto
- Los componentes de la aplicación
- Las funciones que deben de ser cubiertas de forma inicial

Esto puede hacerse en una serie de entrevistas o sesiones, dependiendo de la disponibilidad.

El siguiente paso es establecer los aspectos de la administración de los procesos. Se crea una Consultoría (a nivel de negocios) y una Dirección del Comité Técnico, a los cuales será reportado el proyecto. Y finalmente este trabajo es sumarizado en un proyecto.

AMBIENTE DE DESARROLLO

En esta etapa se establece por completo la plataforma tecnológica de desarrollo (figura III.16). La plataforma incluirá:

- hardware/sistema operativo
- lenguajes de desarrollo/herramientas
- software de administración de la base de datos
- software de redes, software del sistema y software de administración de la red
- necesidades de conectividad

- utilidades tales como editores y herramientas de análisis
- facilidades
- estándares de la documentación
- administración de la configuración

⇒ Prototipo evolutivo

⇒ Caja de tiempo (timebox) para el control de avance

⇒ Prototipo de Concepto -> Prototipo de Contenido -> Prototipo de Proceso

⇒ Actividades paralelizadas

⇒ Uso del modelo de datos como vehículo de integración

⇒ Obtener el 80% de la solución en 20% del tiempo

⇒ Inconvenientes que afectan a todo el sistema

Figura III.16 Desarrollo Rápido Cliente/Servidor.

ESTABLECER LOS REQUERIMIENTOS DEL NEGOCIO

Los requerimientos del negocio son establecidos en una serie de sesiones de Unión para la Planeación de Requerimientos (Joint Requeriments Planning - JRP). Las sesiones JRP son similares a talleres como los bien conocidos JAD, pero involucran diferentes participantes y produce diferentes resultados. El JRP tiene el poder de introducir un nuevo sistema. Hay muchas ventajas para llevar a cabo estos talleres (figura III.17).

• Formar equipos por consenso, además de compartir una visión del desarrollo ⇒ cortar a través de las barreras organizacionales ⇒ resolución de conflictos
• Recolectar detalles acerca de las operaciones ⇒ recopilar el conocimiento de los usuarios del negocio ⇒ resultados visuales
• Reducir el tiempo de solución ⇒ reduce las entrevistas y el tiempo de diseño ⇒ reduce la propagación de esfuerzos al realizar el *front-end*

Figura III.17 Ventajas que facilitan los talleres.

Entonces debe quedar claro que el propósito de una sesión JRP es para el desarrollo de los requisitos del negocio, llevando a cabo un convenio entre todas las partes afectadas. Con

objetivos bien claros: 1) examinar cuidadosamente los requisitos del negocio, particularmente el alcance y los objetivos; 2) enlazar el plan del proyecto para el sistema con las metas y los factores críticos para el éxito de la empresa; y 3) comprender y formar un consenso en la administración sobre los objetivos del proyecto.
Dentro de los participantes se incluyen los niveles gerenciales de cada área afectada en la organización. Entonces, aquí es donde los altos ejecutivos entran en la planeación. Los pasos a través de los cuales se lleva a cabo una sesión son los siguientes:

1. Explicar los conceptos de la sesión y las reglas fundamentales para la sesión (tal como no criticar las ideas expuestas).
2. Introducir y revisar el documento realizado especialmente para la sesión (el cual contiene los detalles del alcance y objetivos del proyecto).
3. Determinar las funciones del sistema.
4. Guiar a los participantes a través de los procesos del negocio de un modo ordenado para así producir un resultado definido.
5. Acordar el estado del proyecto y los pasos a seguir.

Después de la sesión se habrán obtenido varios productos de este trabajo, tales productos son los siguientes:

1. Lista de las operaciones del negocio que serán afectadas (departamentos, sitios, áreas externas).
2. Problemas del negocio direccionados hacia el sistema.
3. Objetivos del sistema.
4. Funciones del sistema, con beneficios y prioridades.
5. Descomposición y flujo de procesos.
6. Modelo de datos esencial.
7. Aspectos sin resolver con sus planes de acción para la solución.
8. Se determinan las fechas a cumplir en esta fase del proyecto.

DISEÑO DE LA APLICACIÓN

En la fase de diseño de la aplicación, se hará un prototipo de la interfaz externa y del contenido de la aplicación, tales como el desarrollo de los modelos de datos, de eventos y de procesos.

Inicialmente se establecerá el alcance y la arquitectura. El propósito de esta etapa es para definir los componentes de la aplicación y establecer cuáles funciones serán desarrolladas en la primera implementación de la aplicación, que irá evolucionando. Los siguientes datos deben de estar disponibles al comienzo del proyecto:

- eventos significativos del negocio
- principales procesos del negocio
- necesidades críticas de información

Se debe priorizar la funcionalidad a ser desarrollada. Primero, priorizar los procesos del negocio. Entonces, identificar las entidades de datos requeridas, el trabajo de soporte y seleccionar los primeros procesos a ser soportados. Como parte de esta priorización identificar los componentes de la arquitectura que serán desarrollados, como el desarrollo de la infraestructura (acceso a datos remotos o la seguridad).

PROTOTIPOS USANDO JAD

Frecuentemente, los prototipos se pueden producir tan rápidamente como los documentos de diseño. Y casi siempre son más completos y acertados.

Un buen prototipo puede ser construido fácil y rápidamente. Un prototipo es necesario cuando todos los detalles del sistema no han sido definidos, el usuario final tiene un interés en la operación del sistema, y el sistema es los suficientemente complejo para tener la necesidad de modelarlo. Una clara excepción de esto es cuando el sistema reemplazará uno existente, y en tal caso el viejo sistema servirá para esta función.

El sistema completo no es modelado en un prototipo. Por lo que se toman los principales componentes junto con las principales interacciones con el usuario final, para realizar un prototipo (entradas, edición de datos y flujos de diálogo). En este método se construye iterativamente, manteniendo el trabajo en el sistema en producción, sin desperdiciar esfuerzos.

Existen pros y contras en cuanto a la utilización de los prototipos. Los aspectos positivos son los siguientes:

- Los usuarios reciben una entrega anticipada
- Ayuda a los usuarios a visualizar los beneficios
- Ayuda a vender el sistema internamente
- Los usuarios sienten como propio el sistema, ya que ellos trabajan en el diseño
- Los usuarios expertos pueden ayudar, ya que es más efectivo porque pueden visualizar mejor el sistema que mediante un documento
- Puede ser rápido y menos caro que el diseño en papel
- Un diseño más pobre se obtiene en un ciclo de diseño tradicional
- Un buen camino para dar expectativas a los programadores
- Entrenamiento en las herramientas necesarias

Y algunos aspectos negativos:

- El uso de prototipos puede dejar muchos problemas en cuanto a aspectos que quedan incompletos
- Los usuarios pueden ser confundidos en cuanto al costo y el tiempo para tener el sistema real
- Los detalles del sistema pueden ser olvidados
- Puede dejar un enfoque prematuro en cuanto a los detalles
- Algunos usuarios no son involucrados
- Algunos sistemas son atrapados en un ciclo de cambios continuos

Ahora, en cuanto a la idea de una caja de tiempo (timebox) es un intento por derrumbar algunos de los problemas de los prototipos iterativos. El problema más grande es que no se sabe cuando se terminará. La caja de tiempo es simplemente una designación de fechas, las cuales no pueden ser quebrantadas. Si el sistema se descuida y funcionalmente se deja caer, la fecha no será cumplida. Un periodo de tiempo típico para un timebox es de tres meses.

Las etapas de los prototipos usados en esta metodología, son los siguientes:

1. Contenido: sólo la interfaz de usuarios, generalmente acompañada por el modelo de datos y otras actividades de diseño.
2.. Procesos: la interfaz de usuario con un limitado acceso a los datos, generalmente limitado en función de que no optimiza una producción eficiente.
3. Producción: aplicación funcional, con aproximadamente un 80% de las funciones necesarias, con una aceptable eficiencia en cuanto al volumen completo de la producción

Con frecuencia se anticipa un prototipo eficiente, para confirmar la factibilidad de la arquitectura.

Utilizando sesiones JAD, se desarrollan las pantallas, menús y el diálogo de la interfaz del sistema. Es un proceso iterativo, que repetidamente ejecuta actividades en la misma etapa hasta que es obtenida una aprobación administrativa.

JAD establece un consenso para el sistema entre los usuarios y obtiene su conocimiento para mejorar la calidad del mismo. Produce documentación visual (prototipos), lo que pueden comprender más fácilmente los usuarios; que no sucede con la documentación técnica, ya que por lo general no la comprenden. JAD introduce técnicas tales como los prototipos y el modelado de datos en las cuales está involucrado el usuario final. Este puede traer beneficios a la organización y solución a conflictos.

La técnica de desarrollo JAD es un diseño en lapsos de tiempo muy pequeños como en el análisis de sistemas tradicionales basado en entrevistas, pero las entrevistas son paralelizadas. Con esto también se pueden utilizar menos recursos cuando es bien realizado. Esto puede desgastar los recursos, si las sesiones no son bien preparadas y estructuradas. Por lo que para realizar una sesión JAD que es una herramienta pero costosa, se debe planear de antemano, para utilizarla efectivamente y con destreza.

Usualmente se da una serie de sesiones JAD, en las cuales su duración común es alrededor de una semana. Uno o dos días, después de las sesiones, son usados para revisar y confirmar los resultados. Si un sistema es muy grande, puede ser dividido en subsistemas que son desarrollados a través de varias semanas de sesiones JAD.

Entonces el propósito de una sesión JAD es el desarrollo de los documentos de diseño del sistema, llevando un consenso entre todas las partes afectadas. Con el objetivo de documentar el flujo de trabajo, el modelo esencial de datos, las pantallas y los reportes de los aspectos

funcionales del nuevo sistema. La gente que se reúne en estas sesiones son personas de cada una de las áreas afectadas, incluyendo gerentes y empleados. Pudiendo incluir viejos y nuevos empleados, ya que los nuevos empleados pueden tener una perspectiva diferente.

Los pasos a seguir en las sesiones son las siguientes:

1. Explicar el concepto del diseño de la aplicación y las principales reglas de la sesión (no criticar las ideas que surjan).
2. Guiar a los participantes a través de los requerimientos del sistema en una forma ordenada, siguiendo el flujo de trabajo, para obtener una lista ordenada.

Después de la sesión, se deben obtener una serie de productos provenientes del trabajo realizado:

1. Propósito, alcance y límites del sistema.
2. Direccionar los problemas del negocio en el sistema.
3. Interfaces del sistema.
4. Descomposición funcional.
5. Modelo de procesos, incluyendo los eventos esenciales.
6. Modelo de datos, incluyendo los atributos esenciales.
7. Pantallas y reportes trazados.
8. Requerimientos operativos y procedimientos.

MODELADO DE DATOS

Si se construye un sistema dividido en pequeñas unidades y en pequeños periodos de tiempo, esto permite delimitar el sistema que será construido. Por lo cual, grandes sistemas pueden ser construidos a través del desarrollo de varios y pequeños esfuerzos en paralelo y en coordinación con un modelo consolidado.

En la ingeniería de la información, los modelos de datos y procesos son preparados. Por el tipo de sistema empleado, el modelo de datos es un trabajo esencial. Mientras el modelo de datos es probablemente la guía más utilizada para la realización de un sistema, esto es especialmente verdadero para sistemas DSS y de Control Operacional.

Un modelo de información completo debe contener los siguiente elementos:

- modelado de eventos
- modelado de procesos
- modelado de datos
- análisis del ciclo de vida de entidades

Otro aspecto en el que se hace énfasis en el modelo de datos es el que debe ser estable. El cambio en los procesos es mucho más frecuente que en los datos. Un modelo de datos esencial es estable en la organización a través del tiempo, a través de compañías del mismo ramo, y

entre áreas funcionales que son similares en los diferentes ramos. Lo que un modelo de procesos no.

En la fase de requerimientos, el modelo de datos representa el tipo de datos mantenidos por el negocio y las relaciones e interdependencias entre los datos. Esto lo producen los usuarios para definir y priorizar sus metas y objetivos en la fase de planeación de los requerimientos. Por lo que este modelo es usado en la identificación de metas y objetivos. Lo cual define el alcance del proyecto en términos de las necesidades de información para soportar el área de negocios en estudio, además de las interacciones con agentes externos. En la fase de diseño de la aplicación, los modelos de datos esenciales, de procesos y de eventos son construidos correctamente con un prototipo de Contenido. Esto permite tener un claro entendimiento del área de negocios, todos los problemas asociados a dicha área y de las fuentes de datos apropiadas.

En la fase de desarrollo, los modelos son expandidos y refinados para incorporar los requerimientos específicos de la implementación y para incluir detalles adicionales.

Existen muchas herramientas CASE disponibles para construir modelos de datos. El ambiente más natural para cliente/servidor son los CASE basados en windows, que sino incluyen características de generación de código son accesibles. Estos CASE son más que simples herramientas de dibujo, ya que pueden generar diagramas del esquema de base de datos y reportes. Ejemplos de estos son los siguientes: System Architect, Erwin y EasyCASE Plus.

Como conclusión el autor enfatiza que los factores críticos para el desarrollo rápido de aplicaciones son:

1. Compromiso y participación de los administradores y de los usuarios finales.
2. Una buena administración del proyecto.
3. Motivación, contando con un *staff* hábil y una metodología apropiada.
4. Un modelo de datos compartido.

III.2 Selección de la metodología.

En esta sección se visualizará lo que va a tomarse como guía metodológica para el análisis y diseño del problema a resolver en el siguiente capítulo. Comenzando por la identificación de los aspectos más relevantes de cada una de las metodologías que se han analizado. Para así tomar los elementos más importantes que aportan, y tratar de ubicar lo más adecuado para la solución del problema planteado en este trabajo.

En el caso de la metodología propuesta por Inmon, un punto de partida muy relevante que explica el autor, es la identificación y separación fundamental de lo que es un Sistema Operacional y un Sistema de Soporte de Decisiones. Analizando y dando bases firmes para la separación de dichos tipos de sistemas. Y basándose en esto va presentando la ruta metodológica a seguir en el desarrollo de un sistema cliente/servidor. Además de enfatizar la

naturaleza iterativa de un desarrollo de este tipo, lo cual es un aspecto que no se permite olvidar al llevar a cabo esta metodología.

Para efectuar esta metodología se hace necesaria la utilización de dos herramientas técnicas fundamentales: los Diagramas de Flujo de Datos (DFD) y los Diagramas Entidad-Relación (ER). Los cuales son básicos en el desarrollo de dicha metodología.

La siguiente metodología analizada fue la de Vaughn. Esta metodología en particular llamó mucho mi atención, porque conjunta elementos que otras metodologías no toman en cuenta o consideran como aspectos algo aislados. Ya que no se les da la importancia central para la evolución del sistema. Estos aspectos son: la Reingeniería de Procesos y la Arquitectura en sus aspectos lógicos y físicos.

Esta metodología es algo peculiar por la introducción del concepto de Reingeniería de Procesos, que es un término generalmente manejado en el área de administración de empresas. Pero Vaughn la introduce como un elemento fundamental en la iteración de los procesos que se llevan a cabo en la metodología. Tal es la importancia de la Reingeniería de Procesos, que lo pone como una de las tres etapas para el desarrollo de un sistema. Y de alguna forma repercute en las dos etapas posteriores.

El otro aspecto importante es la Arquitectura. Lo relevante es que le da la misma importancia que a la Reingeniería de Procesos y al Desarrollo del Sistema. Ya que constituye la tercera etapa de la metodología.

Algo también muy importante de mencionar es la clara esquematización de los procesos iterativos y de la ejecución paralelizada de las actividades. Y además del énfasis que pone el autor en la necesidad de contar con una metodología flexible que permita adaptarle herramientas técnicas viejas y nuevas, que ayuden al buen desempeño de la metodología.

La tercera metodología que se analizó fue la de Vaskevitch, la metodología de este autor se inclina a visualizar las etapas del proyecto (Interfaz, Reglas del Negocio, Base de Datos y Arquitectura) desde el punto de vista del modelado de datos. Ya que en cada una de éstas se distingue lo que es el modelado Conceptual, Lógico y Físico.

A mi ver el elemento más relevante de esta metodología es la introducción del concepto de OBJETO. El autor destaca la importancia de considerar el Análisis y Diseño orientados a objetos dentro del desarrollo de un sistema cliente/servidor. Lo cual puede traer ventajas relevantes como la reutilización, flexibilidad e interoperabilidad. Además, aclara que la programación totalmente orientada a objetos puede implicar un grado de complejidad muy elevado, sin incluir el costo.

Pero aunque no se haga una programación totalmente orientada a objetos, se podría utilizar dicho enfoque y así aprovechar sus ventajas. De hecho en las herramientas de desarrollo actuales (Power Builder, Delphi, etc.) ya existe el concepto de objeto, aunque sigan siendo

herramientas de desarrollo para bases de datos Relacionales. Con lo que se aprovechan algunas ventajas que ofrece la orientación a objetos.

Respecto a la cuarta metodología de Renaud, la cual se enfoca solamente a lo que es la Arquitectura de un sistema cliente/servidor; que es un punto que muy pocas metodologías prestan la atención que requiere. De algún modo, esta metodología podría ser un complemento a la metodología presentada por Vaughn, en cuanto a la etapa de la arquitectura.

La metodología propuesta por este autor abarca puntos muy importantes de lo que es el modelo cliente/servidor, que ya se han analizado en el capítulo II. Y que además permite identificar claramente dónde encajan todos estos conceptos en el desarrollo real de un sistema.

La quinta y última metodología analizada es propuesta por Kavanagh. Lo más relevante de esta metodología es que está basada en el Desarrollo Rápido de Aplicaciones, que se lleva a cabo mediante prototipos. Que además se apoya en herramientas tales como JRP y JAD, las cuales son fundamentales para el desarrollo de esta metodología. Lo más peculiar de este tipo de desarrollo es que se puede introducir dentro de otras metodologías, como lo menciona Inmon, Vaughn y Vaskevitch; ya que ellos contemplan la posibilidad de introducir el desarrollo mediante prototipos dentro de sus metodologías. Aunque hablando estrictamente de sistemas que puedan ser considerados para RAD, las metodologías analizadas podrían no caer dentro de esta categoría, dependiendo de las características particulares del problema a resolver y de los recursos con que se cuenten. El desarrollo mediante prototipos es un enfoque muy tentador ya que debido a la dinámica de las organizaciones, se requieren soluciones rápidas y eficientes.

Después de haber destacado los aspectos más relevantes de cada metodología que se analiza en el presente capítulo, es necesario definir la metodología a seguir para solucionar la problemática que se presenta en el "Sistema de Control de Gestión" que me ocupará en el siguiente capítulo. La metodología de Vaughn me parece la más completa, ya que abarca los tres aspectos fundamentales: Reingeniería de Procesos, Desarrollo del Sistema y Arquitectura. Pero sólo para desarrollar el sistema operacional, como indica Inmon. Además, se complementará en algunos aspectos en la etapa de la Arquitectura con la metodología de Renaud. Considerando la posible utilización de las herramientas JRP y JAD, para el análisis y diseño de la aplicación.

A continuación se presenta un cuadro que esquematiza los aspectos más relevantes de cada metodología.

Algo a destacar es que en el cuadro se puede visualizar que las metodologías varían unas con otras en algunos aspectos, pero algo que es común a todas ellas es la importancia fundamental de un buen análisis y diseño, sin importar las herramientas técnicas que se utilicen. Pues esto será determinante para el éxito del proyecto. Y algo que ya se ha marcado varias veces, es la naturaleza iterativa de todas estas metodologías, ya que es algo inherente a las metodologías de desarrollo de sistemas cliente/servidor.

METODOLOGÍA

	Herramientas Técnicas	Análisis y Diseño	Arquitectura	Metodología para Sistemas Operacionales	Metodología para DSS
Inmon	• Diagramas de Flujo de datos DFD • Diagrama Entidad Relación D E-R	SI	NO SE PROFUNDIZA	SI	SI
Vaughn	* Cualquier técnica o herramienta, vieja o innovadora que ayude al objetivo.	SI	SI	SI	NO
Vaskevitch	• Modelado conceptual, lógico y físico en los diferentes aspectos del sistema • Análisi y diseño orientado a objetos • Tal vez la programación orientada a objetos	SI	NO SE PROFUNDIZA	SI	NO
Renaud	• Diagramas de distribución de recursos, escalabilidad y seguridad • Técnicas de fiabilidad, eficiencia y capacidad.	SI	SI	ES INDEPENDIENTE DEL TIPO DE SISTEMA	ES INDEPENDIENTE DEL TIPO DE SISTEMA
Kavanagh	• Sesiones para la Planeación de Requerimientos (JRP) • Sesiones de Diseño de la Aplicación (JAD)	SI	NO SE PROFUNDIZA	SI	NO

* DIAGRAMAS DE FLUJO DE DATOS, DIAGRAMA DE FLUJO DE TRABAJO, DIAGRAMAS CAUSA-EFECTO, BRAINSTORMING, PROTOTIPOS RAPIDOS, MODELADO DE DATOS, ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS, ETC.

CAPÍTULO IV

Análisis y diseño del "Sistema de Control de Gestión"

IV ANÁLISIS Y DISEÑO DEL "SISTEMA DE CONTROL DE GESTION"

El crecimiento de una organización implica que la comunicación entre las unidades que la componen se va volviendo cada vez más compleja, por lo cual se debe de buscar la forma de que el flujo y compartición de la información no se vea afectado. El deterioro en el control de la información puede tener graves consecuencias en perjuicio del crecimiento sano de la organización.

Nuestro caso de estudio se centra en una organización en la que es necesario llevar un control estricto de los trámites que se dan a asuntos que son gestionados hacia áreas tanto internas como externas a dicha organización. Todas las unidades que forman parte de esta organización, comparten cierto tipo de información, y tienen la necesidad de obtener a su vez algún tipo de información que involucra a todas ellas, las cuales se rigen por una misma normatividad de gestión. Tomando en cuenta lo anterior, todas las unidades que forman parte de la organización tienen la imperante necesidad de compartir recursos, con ello se tiene como un camino a seguir la creación o integración de la información corporativa, necesaria para lograr un funcionamiento uniforme e integrado.

REINGENIERÍA DE PROCESOS

DEFINICIÓN DE LA SOLUCIÓN

CASO DE ANÁLISIS

La problemática a resolver se encuentra en la Secretaría de Desarrollo Social (SEDESOL). El área que visualiza la necesidad de contar con un sistema que comprenda toda la organización es la Secretaría Técnica del C. Secretario del Ramo. Dicha área en particular tiene como objetivos generales:

1.- Recibir las peticiones que por escrito presenten los ciudadanos y las organizaciones sociales, de acuerdo a lo establecido en la ley.

2.- Dar curso y atención a las peticiones de los ciudadanos.

3.- Coordinar con las áreas de la Secretaría de Desarrollo Social la atención y respuesta a las peticiones de los ciudadanos y organizaciones sociales, formuladas a la Secretaría.

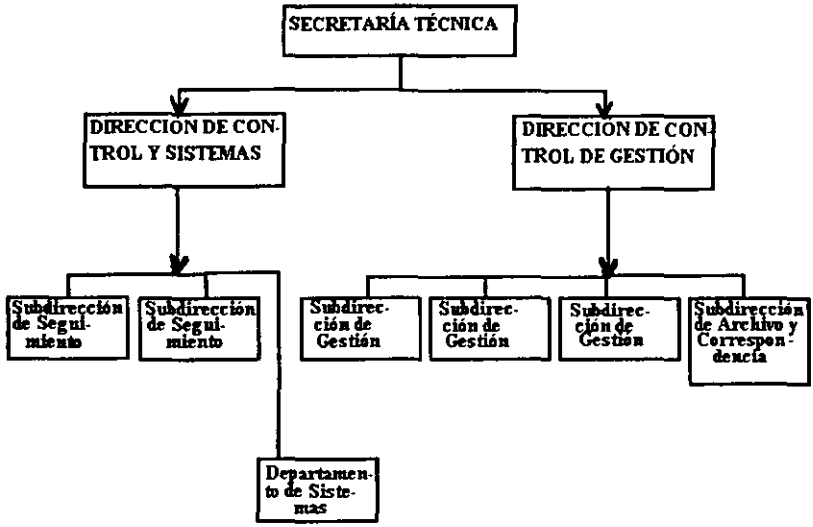
4.- Dar respuesta a las peticiones de los ciudadanos en un plazo no mayor a 80 días hábiles, dependerá del asunto mismo, por lo que el tiempo puede ser menor ó mayor en ocasiones.

5.- Resolver las solicitudes de audiencia con el C. Secretario.

6.- Informar oportunamente de los asuntos más relevantes al C. Secretario.

7.- Custodiar y archivar las solicitudes de los ciudadanos durante el tiempo que establece la ley.

La estructura orgánica de la Secretaría Técnica es la siguiente:



A continuación se enlistan las actividades de cada una de las áreas que forman la estructura orgánica:

Dirección de Control de Gestión.

- ⇨ Analizar las solicitudes de los ciudadanos y turnar sobre las atribuciones de las diferentes áreas Internas (Subsecretaría de Desarrollo Regional, Subsecretaría de Desarrollo Urbano y Vivienda, Oficialía Mayor, Coordinación General de Delegaciones y Delegaciones Estatales, FONAES, Coordinaciones e Institutos, Áreas de Apoyo, Organismos Descentralizados) y externas (otras Dependencias Federales).

- ⇨ Elaborar memorándum y enviar a captura.

- ⇨ Revisar captura.

- ⇨ Someter a firma del titular del área.

Subdirección de Archivo y Correspondencia.

- ⇨ Recibir la correspondencia, documentos y solicitudes, vía correo o personal, dirigidas al C. Secretario del Ramo, Secretario Particular, Secretario Auxiliar y Secretaría Técnica.

- ⇨ Separar y ordenar de acuerdo al tipo de documento (peticiones, información personal, información periodística, revistas, invitaciones, felicitaciones, etc.).

- ⇨ Recibir los documentos que envía la Coordinación de Atención Ciudadana de la Presidencia de la República, que competan a la Secretaría de Desarrollo Social.

- ⇨ Fotocopiar los documentos que se gestionan a las áreas internas y externas de la Secretaría, y pasarlos a Control de Gestión, para tramitar la atención que requieran.

- ⇨ Registrar sistemáticamente la documentación correspondiente, mediante el folio de entrada.

- ⇨ Enviar los documentos vía correo o con mensajero.

- ⇨ Archivar acuses de envío de documentos, copias u originales de las solicitudes enviadas al C. Secretario, Secretario Particular, Secretario Auxiliar, Secretaría Técnica, Diario Oficial de la Federación y diversos documentos de la competencia de la Secretaría.

- ⇨ Clasificar con base al tema, abrir expediente, registrar en catálogo alfabético y numérico, y archivar.

- ⇨ Enviar a captura el número de memorándum y expediente.

Dirección de Control y Sistemas.

- ⇨ Capturar la información y generar los documentos de: síntesis, turnos u oficios.

- ⇨ Analizar, registrar y enviar a captura en el sistema de cómputo los oficios que indiquen respuesta o gestión a las solicitudes planteadas a esta Secretaría y en su caso, dar respuesta a solicitudes específicas.

- ⇨ Mandar al Archivo

Subdirección de Seguimiento

Seguimiento Normal

- ⇨ Revisar y analizar los documentos que son copias marcadas para el C. Secretario, Secretaría Particular y Secretaría Técnica.

- ⇨ Separar los que correspondan a las respuestas de las solicitudes y lo que es archivo.

- ⇨ Revisar las respuestas y proponer status (***en espera de respuesta***, ***en análisis*** y ***concluido***) de la solicitud.

⇒ Enviar a captura.

Seguimiento trimestral

⇒ Solicitar a cómputo la emisión del listado trimestral por área de los asuntos que no estén concluidos.

⇒ Enviar a las áreas el listado con oficio rubricado por el titular.

⇒ Revisar las respuestas de lo solicitado a las áreas y proponer status de la petición.

⇒ Enviar a captura.

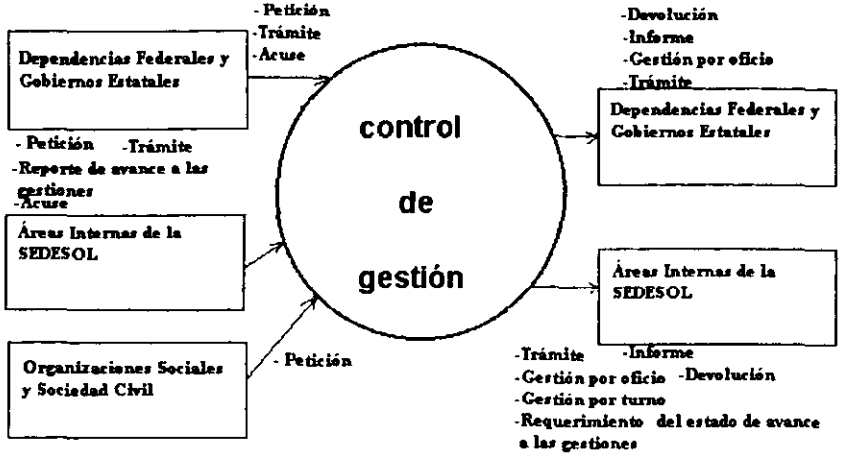
Nota: Los dos tipos de seguimiento se hacen simultáneamente.

IDENTIFICACIÓN DE LA PROBLEMÁTICA QUE EXISTE (DFD ACTUAL). DETERMINACIÓN DE LAS CAUSAS Y SOLUCIÓN PROPUESTA.

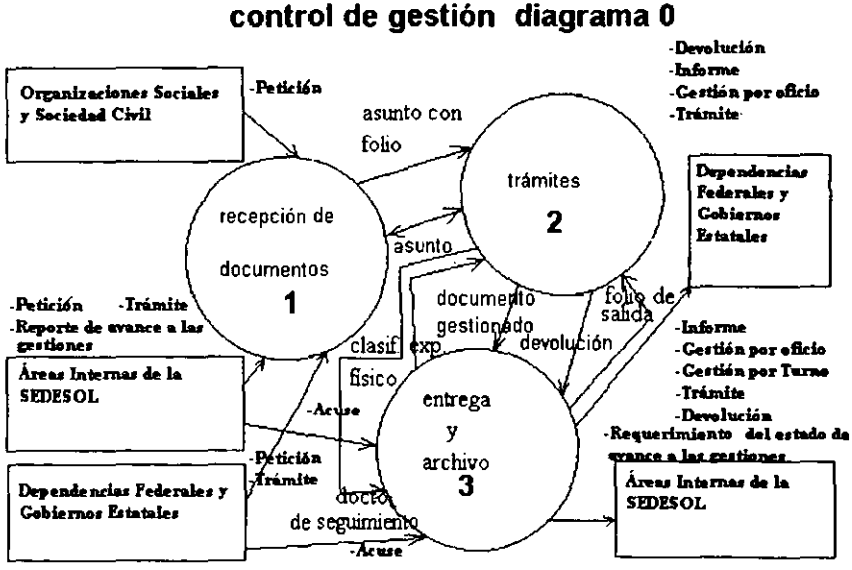
Inicialmente, debemos ver el contexto en que se encuentra el área en estudio. Como podemos ver en el Diagrama de Contexto, se consideran todas las entidades que interactúan, ya sea involucradas como entradas y/o salidas. Quedando claro que existe la interacción con Dependencias Federales, Gobiernos Estatales, Áreas Internas a la SEDESOL, Organizaciones Sociales y Sociedad Civil; como origen de la información.

DIAGRAMA DE CONTEXTO

SISTEMA DE CONTROL DE GESTIÓN



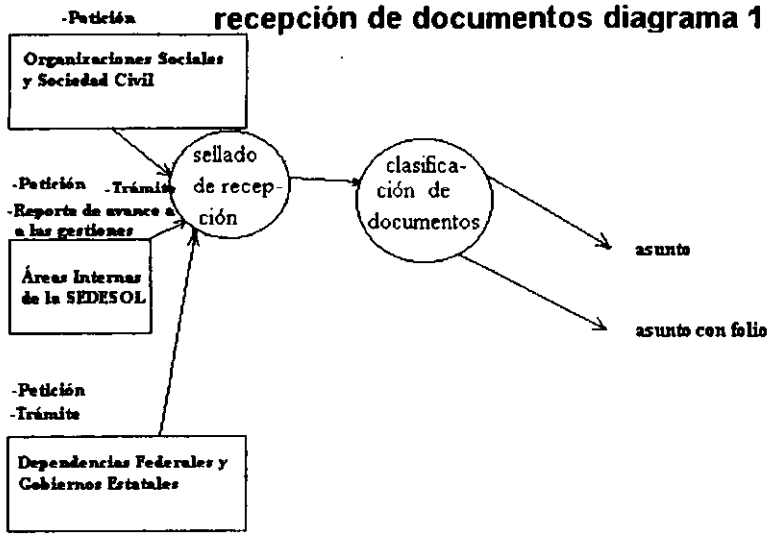
Adentrandonos más en lo que es el Control de Gestión, como lo podemos ver en el diagrama 0, se muestran los principales procesos que se involucran en el sistema.



En el proceso de Recepción de Documentos existen dos actividades básicas: el sello que se da a los documentos al momento de recibirse y la clasificación de los documentos dependiendo si son: peticiones, información personal, invitaciones, felicitaciones, revistas o información periodística. Las invitaciones se consideran como peticiones, y todos los demás tipos de documentos como trámites (Diagrama 1).

La problemática que se presenta en este proceso es que todo es mediante un control manual, y como consecuencia cuando se requiere saber si en realidad entró algún documento no se puede saber y por lo tanto no se puede obtener un conteo de lo que se va recibiendo. Porque cuando los asuntos son reintegrados del proceso de Trámites sólo se saca fotocopia a los documentos que serán gestionados y sólo éstos llevarán folio de entrada. Entonces, para todos los demás documentos no habrá forma de constatar su entrada. Ya que debido al flujo que se da después de clasificarse, pasando de mano en mano sin un control, es más probable que se puedan llegar a extraviar.

Hay documentos que pueden no tener mucha importancia (como las revistas), pero por ejemplo los SEGUIMIENTOS son documentos de fundamental importancia, ya que es la respuesta a las gestiones realizadas. Por lo que esto ha llegado a ser un punto de falla que es necesario corregir. Entonces, sería necesario introducirlo como parte del Sistema de Información para llevar un registro automatizado de todo lo que es recibido. Llevando a cabo este proceso en el momento mismo de la recepción, así como el tipo de documento en que se ha clasificado, antes de pasar al proceso de trámite.



Como ya dije, las invitaciones serán consideradas como peticiones, pero a su vez las peticiones serán clasificadas dependiendo del asunto, especificamente como: solicitud de peticiones diversas, convenio o designación. Además, tomando en cuenta la confidencialidad o importancia del asunto se puede tratar como una SÍNTESIS (Diagrama 2).

Tenemos dos procesos principales que son la Gestión de Peticiones y el Seguimiento a Peticiones, las cuales están constituidas por subprocesos.

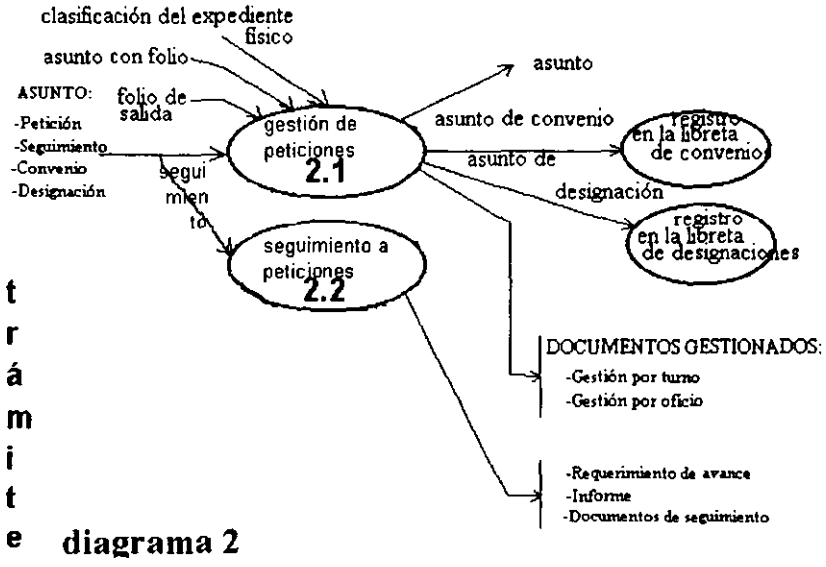


diagrama 2

En el diagrama 2.1, el cual muestra los subprocesos de la Gestión de Peticiones, aquí las peticiones son analizadas a fondo para definir las características particulares del asunto, y así determinar si necesita ser gestionado. En caso afirmativo se determina el conducto por el que se tramitará; dependiendo de su procedencia, urgencia, confidencialidad y tiempo crítico de respuesta, cuando lo tenga.

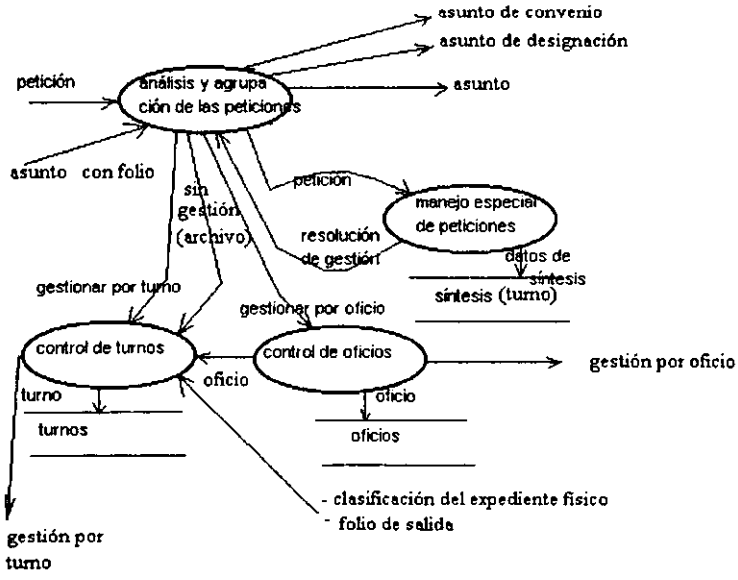
Si después de hacer este análisis, se define que varios asuntos serán enviados a una misma área de gestión y son asuntos afines, el conjunto será manejado como un solo asunto. Ya clasificados se regresan al proceso de Clasificación de Documentos en la Recepción para que sean foliados con un número de entrada para después regresarlos a Análisis y Agrupación de Peticiones para enviarlos a la gestión determinada.

Cuando un documento es considerado como Designación o Convenio se anota en una libreta respectivamente, ya que después es muy importante identificar estos documentos por la importancia y relevancia que pueden tener tiempo después. Cuando los documentos se consideran importantes, se le da un trato especial. Se le presentan al Secretario del Ramo para que determine la gestión a darles. Estos asuntos son llamados SÍNTESIS. Cuando se tienen las instrucciones a seguir de estos asuntos se regresan a Análisis y Gestión de Peticiones para tramitarlos.

Después de ser analizados no todos los documentos tendrán un proceso de trámite, algunos solamente son registrados como Archivo. Los asuntos gestionados como TURNO y los que no tienen gestión son registrados en el Sistema de Información. El cual genera el documento que se envía. Los oficios se hacen en un editor de texto fuera del sistema, para darles un formato

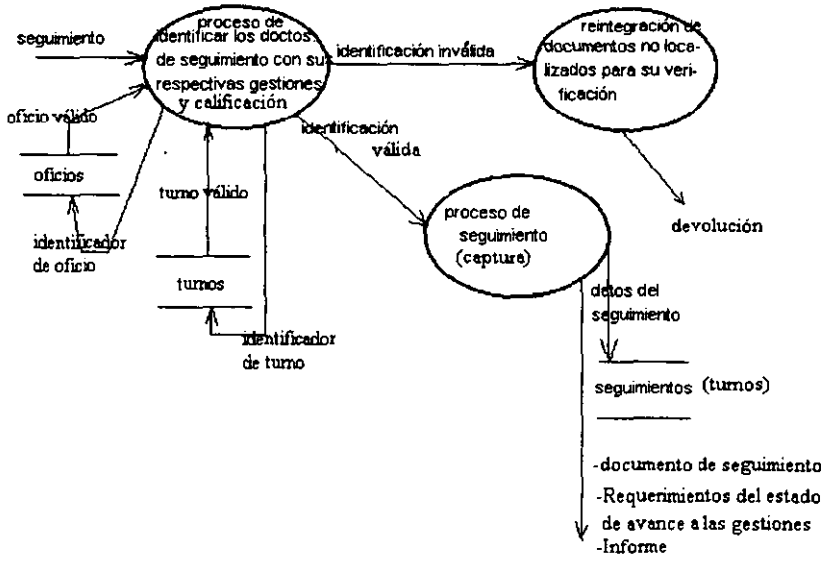
más elaborado. Pero después los datos del oficio son vaciados en el sistema, ya que el asunto ha recibido un folio de entrada que deberá ser debidamente registrado, para poder localizarlo posteriormente.

gestión de peticiones diagrama 2.1



Los documentos que son seguimientos a asuntos gestionados son localizados donde corresponden. Cuando traen como referencia el folio de salida se pueden localizar sin problema; en caso contrario se tienen que buscar entre todos los registros. Cuando un documento no es localizado se reintegra mediante un oficio para que se especifique el folio de salida que le corresponde (diagrama 2.2).

seguimiento a peticiones diagrama 2.2

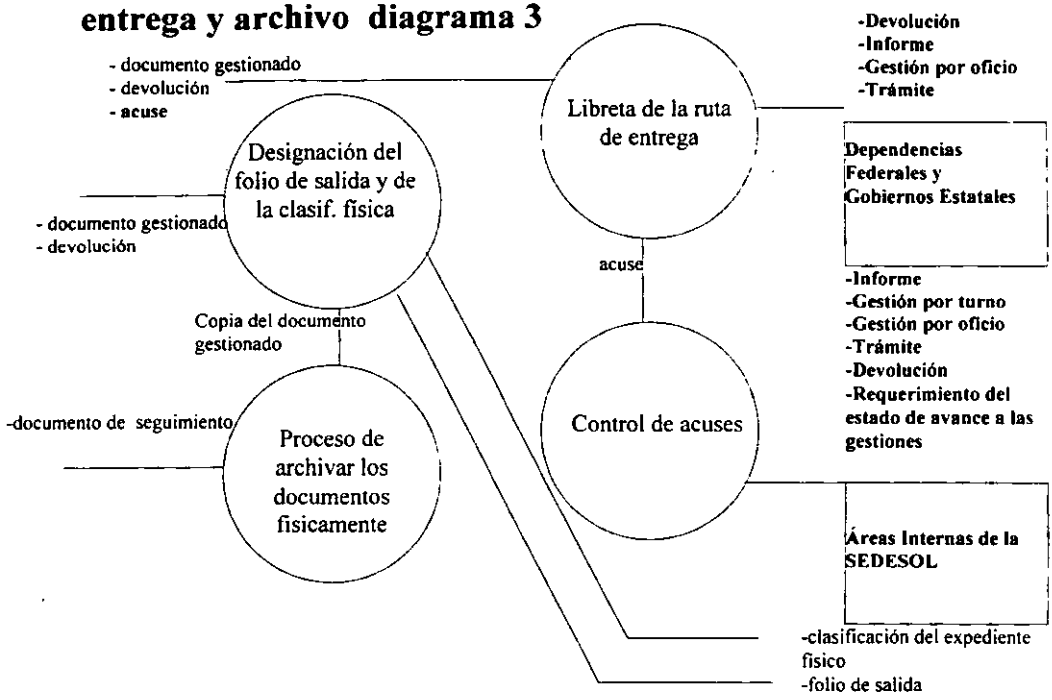


Al ser ubicado se califica el estado de avance, y son transferidos a capturar en el sistema. Cuando se registran se anota el expediente físico donde se encuentra el asunto para que le sea anexado.

A los documentos gestionados y a las devoluciones se les asigna el folio de salida y el expediente donde serán guardados, y se regresan al proceso de Control de Turnos (diagrama 2.1) para que se capturen. Los documentos de seguimiento, gestionados y devoluciones son archivados físicamente. Los documentos a ser entregados a otras áreas son registrados en la libreta de entrega indicando quién entrega, qué ruta llevará, cuántos documentos entregará en cada lugar, el kilometraje y algunas anotaciones especiales. Al entregar los documentos, el mensajero debe requerir un acuse, que avale la entrega del mismo. El cual será integrado al control de acuses que se lleva de forma manual (diagrama 3).

SISTEMA DE CONTROL DE GESTIÓN

entrega y archivo diagrama 3



Después de haber analizado el flujo entre los procesos actuales de Trámite y Entrega y archivo, es necesario identificar los puntos críticos en que se presentan los problemas. En el diagrama 2, a primera vista se observa que el control de convenios y designaciones es manual, lo cual implica imprecisión y lentitud en el trato con estos documentos que son relevantes, pues no existe un enlace con la gestión realizada ni el avance o modificación que pudieron haber tenido.

Al ir más a fondo en la Gestión de Peticiones en el diagrama 2.1, lo primero que hay que notar es que todo se guarda en un solo archivo. Los documentos con gestión, sin gestión, el seguimiento, el folio de salida y el expediente físico, son almacenados en un solo archivo. En primera, el archivo crece mucho por toda la información innecesaria que se repite, por los campos nulos, la información operativa se mezcla con la información histórica. Además, en cuanto a los oficios se duplica el trabajo al vaciar los oficios al sistema, ya que su elaboración es externa al sistema.

En cuanto a las síntesis, aunque este proceso pertenece al mismo sistema, no permite ningún tipo de vinculación , por lo que en el control de turnos no podemos saber cuáles recibieron un trato de síntesis.

Lo anterior más que ser un problema en los procesos, es más bien una falla en el modelo de datos del sistema actual, el cual afecta significativamente el buen desempeño del área. Ejemplo

de ello son la falta de integración entre Turnos y asuntos de Síntesis y el modo externo en que se manejan los oficios, provocando el doble registro de los oficios.

Se detecto que se realiza el proceso de Agrupación de Peticiones. Que se presenta cuando analizan y agrupan las peticiones porque ese conjunto agrupado se manejará como un solo documento, y cuando elaboran el documento de gestión ya sea turno u oficio no hacen un desglose de todas las peticiones involucradas. Por lo que al llegar los documentos de seguimiento no se pueden ubicar a qué gestión corresponden en el caso de no tener referencia del folio de salida, y en caso de que hagan referencia no se puede verificar que en realidad corresponden y hay que referirse al expediente físico. Algunas veces se hacen listados de estas peticiones de forma manual, sin ningún control y orden. Pero de nuevo llegamos al punto anterior, concluyendo que se debe a que el sistema no permite este tipo de relación (uno a muchos) por el problema en su modelado.

Como ya se ha visto en capítulos anteriores, un análisis y diseño para el modelado de datos es fundamental para el buen funcionamiento de la organización, ya que cuando es utilizado el sistema, de algún modo afecta en forma positiva o negativa en los objetivos de la empresa. En el proceso de Seguimiento a Peticiones del diagrama 2.2, un punto a resaltar es que la captura de seguimientos se hace como un archivo de texto, por lo que las búsquedas implican dificultad y tiempo. Y mucha veces se encuentra más rápido en el archivo físico que en el sistema de búsqueda.

Un punto importante a resolver en el proceso de registro de seguimientos es encontrar cómo poder enlazar y obtener eficientemente el seguimiento a las peticiones y los oficios con que llegan relacionados. Pues los seguimientos pueden remitirse de las siguientes formas: 1) un oficio de seguimiento que responde a una sola gestión, 2) un oficio de seguimiento que responde a varias gestiones, 3) un oficio de seguimiento que anexa un conjunto de oficios, y cada oficio corresponde a una sola gestión o 4) un oficio de seguimiento que anexa un conjunto de oficios, y cada oficio corresponde a más de una gestión. También hay que considerar que cada gestión puede tener más de un seguimiento.

Este procedimiento permitiría un eficiente control en el seguimiento a las gestiones, que es uno de los objetivos principales de la organización, por lo que es preciso realizar un buen modelado que cubra las necesidades.

En el proceso de Entrega y Archivo será necesaria la automatización de la Ruta de Entrega y el Control de Acuses. En cuanto al proceso de Designación de Folio de Salida y de la Asignación de la Clasificación Física, creo que sería pertinente realizar esta actividad en este mismo proceso y no tener que pasarlo a Control de Turnos. Porque aquí se tiene la certeza de los datos asignados, existiendo la posibilidad de verificación.

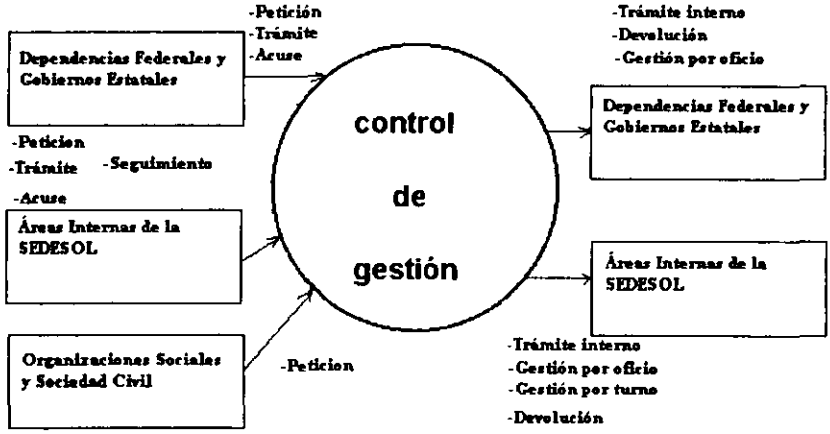
DESARROLLO DE LA SOLUCIÓN

DETERMINAR LA SOLUCIÓN ÓPTIMA (DFD PROPUESTO)

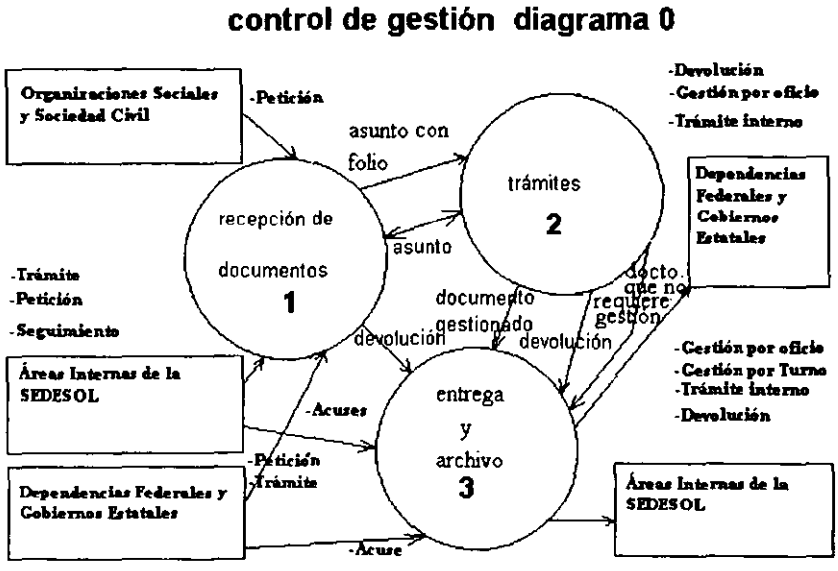
Antes de comenzar, hay que aclarar que se hará una separación de lo que es el sistema operacional, de lo que hasta cierto punto involucraría lo que es un DSS. Como por ejemplo, los informes y reportes que se obtienen a partir de la información. Y como el objetivo de este trabajo es el análisis y diseño de un sistema operacional es necesario hacer una división clara.

DIAGRAMA DE CONTEXTO

SISTEMA DE CONTROL DE GESTIÓN

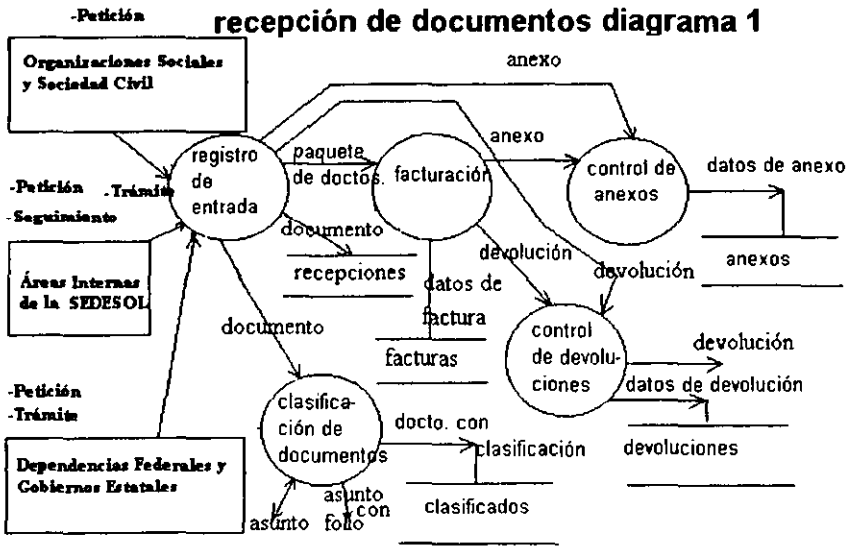


Las primeras modificaciones se pueden observar en el diagrama 0, donde dos de los flujos: clasificación del expediente físico y folio de salida; que van hacia el proceso 2 (trámite) desaparecen. Esto se debe a que los procesos que involucran estos dos flujos se realizan en el proceso 3 (entrega y archivo), como se verá más adelante en los siguiente diagramas. Además de incluir el flujo de devolución.



Se van a analizar cada uno de los procesos por niveles, iniciando por Recepción de Documentos diagrama 1. Como ya se mencionó este proceso se lleva a cabo de forma manual. Por lo que el diagrama 1 sería la propuesta de solución. Principalmente, hay que distinguir entre dos tipos de recepción: los documentos que llegan de forma individual y los que llegan en forma de FACTURA, la cual enlista una serie de documentos. Dicha FACTURA contiene datos específicos que la identifican como única. Cualquiera de estos tipos de entrega pueden contener anexos; como por ejemplo cajas u otro artículo que debe ser debidamente registrado. Y de igual manera, si algún documento a sido recibido equivocadamente, debe ser devuelto por no corresponder al ámbito de atención del área, se deberá registrar su devolución.

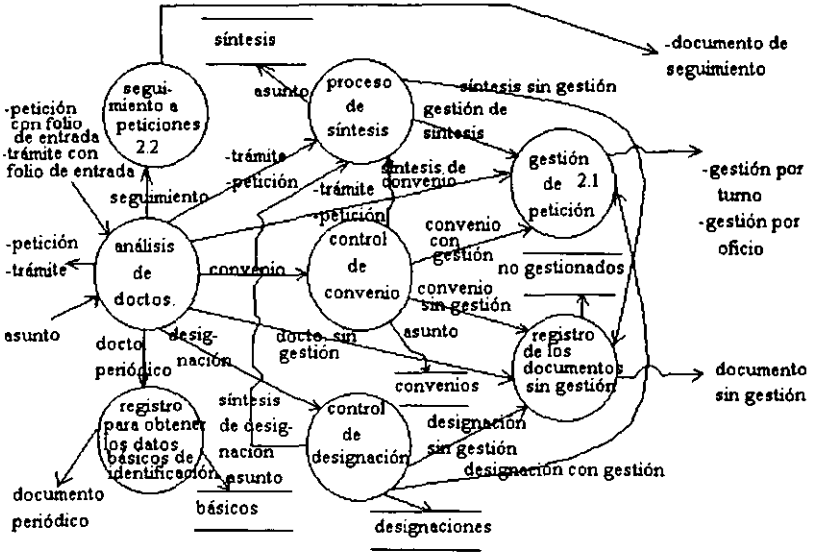
Entonces, después de ser registrado adecuadamente todos los documentos se pasan al proceso de Clasificación de Documentos para su selección. Y así, ser transferidos al proceso 2 (trámite). Pero después son reintegrados al proceso de Clasificación de Documentos, ya identificados por asuntos son clasificados en: peticiones, seguimientos, trámites, documentos sin gestión y documentos periódicos. Sólo las peticiones y algunos trámites son foliados con un número de entrada, para ser reintegrados al proceso 2. Pero antes de esto se llevará a cabo el proceso de registrar la clasificación del asunto y en su caso el número de entrada. Para que se lleve a cabo el objetivo de este proceso será necesario que el proceso de Registro de Entrada se realice de esta forma, ya que sólo en este momento se puede tener la certeza de lo que se está recibiendo. Porque si se deja para después, en el ir y venir de los documentos se puede perder algo importante.



A continuación veremos el proceso de Trámite más a fondo en el diagrama 2. Como ya se dijo hay diferentes tipos de asuntos, y cada uno de ellos requiere un proceso especial. El proceso que requieren las peticiones y los trámites involucran la Gestión de Peticiones. Y los seguimientos involucran el proceso de Seguimiento a Peticiones, estas son más extensas como se analizará en los siguientes diagramas. Los documentos sin gestión requieren un proceso de registro algo detallado, ya que aunque no tengan una gestión es importante tenerlos bien identificados por posibles referencias a su contenido. Los documentos periódicos o no relevantes tienen un proceso de registro muy básico, debido a que sólo sirve para verificar si llegan en la periodicidad adecuada, en caso de requerirse.

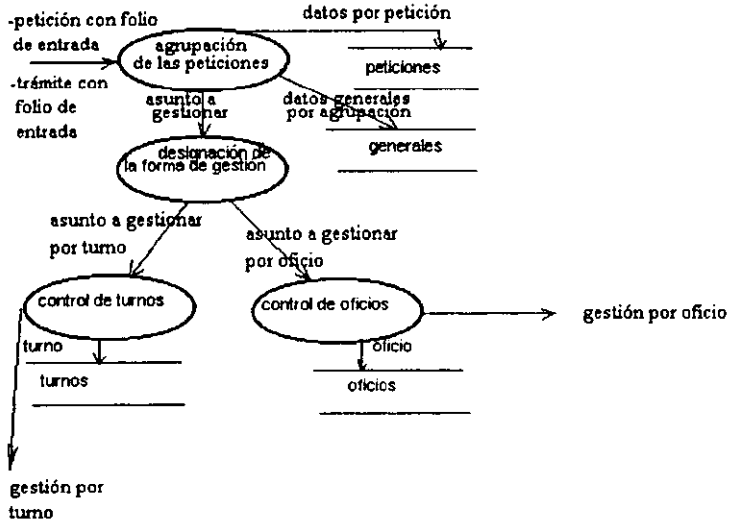
En el proceso de Análisis de Documentos se determinará el tipo de documento y la gestión a seguir. Los asuntos pueden ser tratados como Síntesis, lo cual está determinado por la importancia del asunto. En el Proceso de Síntesis se determina si se le hará una Gestión o no. Los documentos también pueden pasar directamente a ser gestionados o al registro sin gestión. Cuando un documento se identifica como Convenio o Designación, también puede ser tratado como Síntesis o pasarlo a gestión directamente, o también decidir que será un documento sin gestión.

trámite diagrama 2



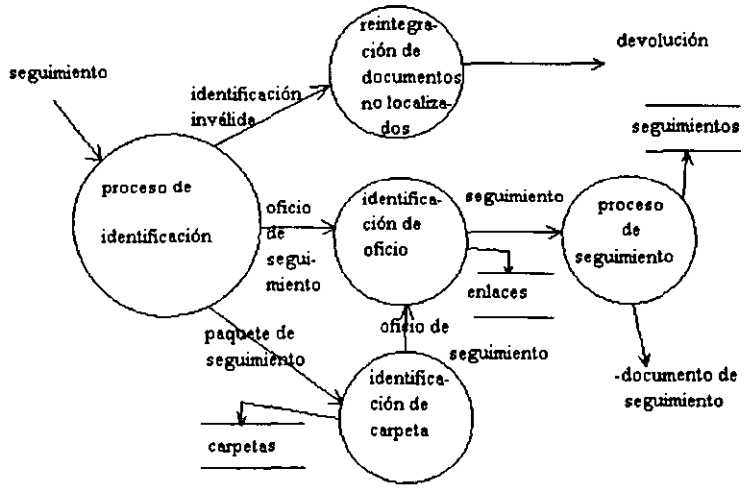
En cuanto al proceso de Gestión de Peticiones diagrama 2.1, uno de los cambios más notorios es que el Proceso de Síntesis ya no se ve dentro del contexto de Gestión de Peticiones, ya que no todos los documentos que entran al Proceso de Síntesis serán gestionados, además de que también se involucran en Síntesis los Convenios y las Designaciones. El proceso de Agrupación de las Peticiones consiste en agrupar las peticiones o trámites que son afines en el tema y el área a que serán gestionados, para así enviar un solo oficio o turno que involucre al conjunto de asuntos. Por lo cual, se registran los datos generales del conjunto de asuntos, además de registrar los datos necesarios de cada una de las peticiones o trámites involucrados en la agrupación. Después se determina la forma de gestión, pues se pueden gestionar por turno, por oficio, más de uno de los casos o ambos. La diferencia entre estos tipos de gestión radica en que los turnos se envían a las áreas internas, y los oficios por lo regular son enviados a áreas externas a la secretaría o también para asuntos importantes que ameriten este tipo de trámite. Un aspecto a hacer notar, es que los oficios también serán obtenidos del sistema, y no se manejarán de forma externa como se realizan actualmente. Pues así se evitará la duplicación de actividades al vaciar los datos del oficio ya generado hacia el sistema.

gestión de peticiones diagrama 2.1



El Seguimiento a Peticiones como se puede ver en el diagrama 2.2. son todos los procesos que se involucrarán en el seguimiento. El Proceso de Identificación consiste en identificar el formato físico de los documentos y verificar la relación de los documentos de seguimiento con alguna gestión. Sino se identifica la gestión a que corresponde se pasa una Identificación Inválida al Proceso de Reintegración de Documentos, para que sean devueltos y verificados. Cuando se identifica un paquete de seguimientos lo consideraremos como una CARPETA si el conjunto de oficios de seguimiento van acompañados por un oficio que los identifica a todos. Por lo que será necesario registrar los datos de este oficio en el proceso de identificación de Carpeta, el cual estará relacionado con todos los seguimientos que integran el paquete. El Proceso de Identificación de Oficio consiste en registrar los datos individuales de cada oficio de seguimiento, independientemente de que corresponda a una o varias gestiones. El Proceso de Seguimiento es donde se registra la contestación de una o más gestiones que corresponden a un oficio y a su vez a una carpeta.

seguimiento a peticiones diagrama 2.2



Si los documentos no llegan en carpeta, sino en oficios individuales pasarán directamente al proceso de Identificación de Oficios y después al Proceso de Seguimiento. En el diagrama 2.2 se cubren los cuatro casos posibles de seguimiento, que son:

1. Un oficio de seguimiento que corresponde a una sola gestión. Este caso es cuando pasa directamente al Proceso de Identificación de Oficio.

2. Un oficio de seguimiento responde a varias gestiones. Este caso es cuando pasa directamente al Proceso de Identificación de Oficios, y en el Proceso de Seguimiento podemos relacionar tantas gestiones como sean necesarias con el oficio registrado en la Identificación de Oficio.

3. Un oficio que anexa un conjunto de oficios, y cada oficio corresponde a una gestión. Este caso es cuando pasa por el Proceso de Identificación de Carpeta y después al Proceso de Identificación de Oficio para así relacionarse con una sola gestión en el Proceso de Seguimiento.

4. Un oficio que anexa un conjunto de oficios, y cada oficio corresponde a más de una gestión. En este caso también pasa por el Proceso de Identificación de Carpeta para después ir al Proceso de Identificación de Oficio y al llegar al Proceso de Seguimiento se puedan asociar tantas gestiones como sean necesarias al oficio de seguimiento que a su vez se asocia con la carpeta.

En el diagrama 2.2 tenemos los procesos que se llevan a cabo en el Seguimiento a Peticiones, y es precisamente la parte central de todo. Esto se debe a que este proceso es el objetivo

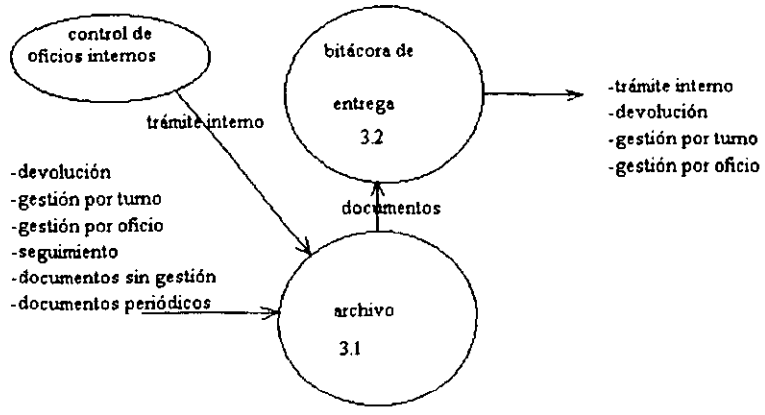
mismo de la organización, el cual es "Tener un seguimiento a todas las gestiones realizadas a una petición o trámite, hasta llegar a su conclusión", y sólo cumpliendo con esto se justifica la existencia de esta parte de la organización.

Ahora, hay que notar que el seguimiento no sólo involucra a la Secretaría Técnica, sino que está involucrando a toda la organización. La Secretaría Técnica cumple con la función de gestionar los asuntos a las áreas encargadas de atenderlos, pero el compromiso de dar solución a las Peticiones o Trámites es de toda la organización. Por lo que el Seguimiento no se da necesariamente en el área que da origen a la gestión.

Aunque todas las áreas tengan el acceso a los asuntos que les son gestionados y puedan dar el seguimiento a los mismos, el flujo de documentos físicos es inevitable ya que sólo así se tiene un sustento legal.

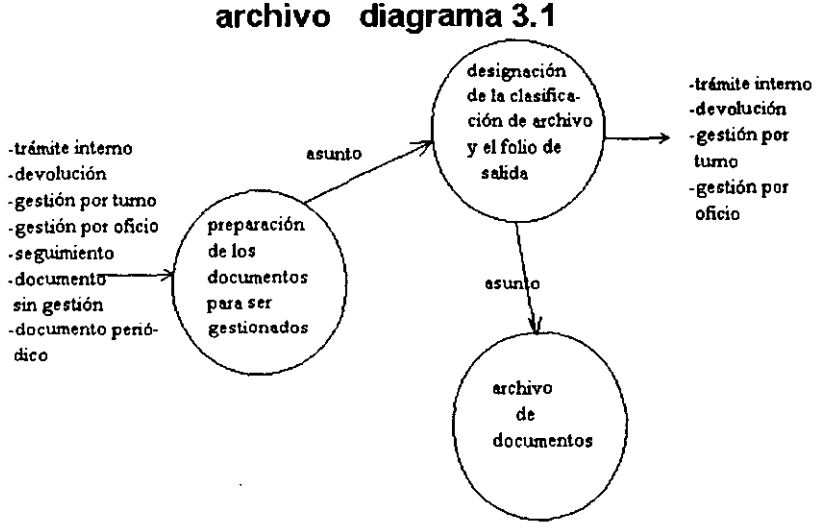
El diagrama 3 muestra los subprocesos que se involucran en el Proceso de Entrega y Archivo. Estos subprocesos son el Proceso de Archivo y la Bitácora de Entrega, las cuales veremos con más profundidad en los siguientes diagramas. Y el Proceso de Control de Oficios Internos, que consiste en asuntos que surgen dentro de la Secretaría Técnica. Por lo tanto éstos no tienen un número de entrada, y siempre son tramitados mediante oficio.

entrega y archivo diagrama 3



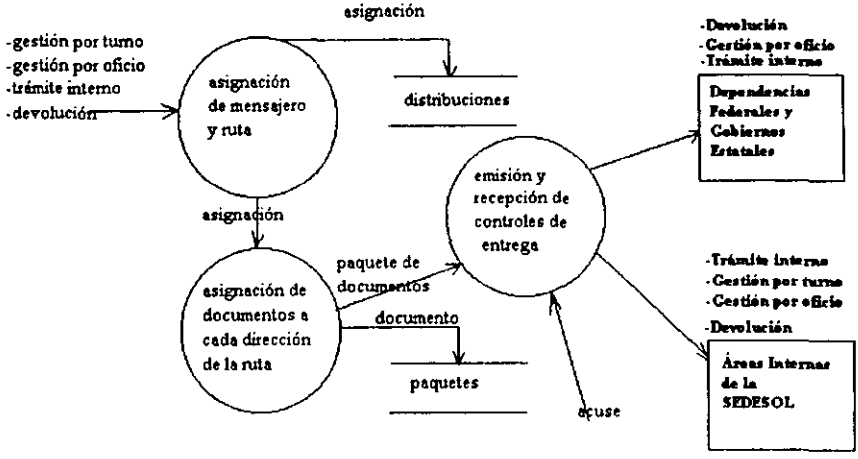
El Proceso de Archivo es la etapa previa al envío de los documentos, diagrama 3.1. El primer subproceso es la Preparación de los Documentos para ser Archivados, aquí se sacan las fotocopias necesarias de documentos, y se ordenan los que serán enviados por lo que requieren folio de salida y clasificación de expediente. Los documentos que sólo van al archivo se les da la clasificación del expediente. Todos son transferidos al Proceso de Designación de la

Clasificación de Archivo y el Folio de Salida, en donde les son asignados los datos correspondientes. Los originales o fotocopias de todos los documentos pasan al Proceso de Archivo de Documentos para ser archivados físicamente. Y los asuntos a ser enviados pasan a la Bitácora de Entrega.



La Bitácora de Entrega diagrama 3.2, es el control de la entrega de los documentos a las áreas internas y externas. El primer subproceso es la Asignación de Mensajero y Ruta, en donde de acuerdo a las direcciones a ser entregados los documentos y a la prioridad de entrega se arma la ruta y se asigna al mensajero. Después se pasa al proceso de Asignación de Documentos a cada Dirección de la Ruta, en donde se arman los paquetes a entregar en cada dirección física, pero haciendo las separaciones de los documentos de cada una de las áreas que pueden existir en una misma dirección física. Después de esta clasificación pasan los documentos al proceso de Emisión y Recepción de Controles de Entrega, aquí se emitirá la orden de entrega de la ruta asignada, y se recibirán los ACUSES de los documentos que hayan sido entregados o no, para llevar el control de que en realidad son entregados los documentos.

bitácora de entrega diagrama 3.2



En los anteriores diagramas se ha presentado la propuesta de cambios a los procesos de "Control de la Gestión" de la organización que se está analizando. Cambios tales como la automatización de ciertos procesos y el traslado de funciones entre los procesos.

DESARROLLO DE SISTEMAS

Como se pudo observar en la metodología seleccionada, para realizar las tres etapas principales: Reingeniería de Procesos, Desarrollo de Sistemas y Arquitectura, en el equipo de desarrollo se debe dar una actividad paralelizada, por lo que cada una de las etapas se van ejecutando al mismo tiempo. La etapa que inicia primero es la de Reingeniería de Procesos, pero al llegar el punto en que se tiene la solución propuesta mediante los diagramas de flujo de datos, es donde inicia la actividad de las etapas de Desarrollo de Sistemas y Arquitectura. Entonces, es aquí cuando se inicia una actividad paralelizada.

Para el mejor entendimiento del flujo que se da dentro de la metodología de desarrollo, pasaré de una etapa a otra como vaya siendo necesario, procurando no perder la claridad de la metodología. Entonces, a partir de la solución obtenida en la Reingeniería de Procesos se definirá la solución óptima en cuanto al Modelo de Datos.

DEFINICIÓN DE LA SOLUCIÓN

REVISIÓN DEL SISTEMA ACTUAL

Existe un sistema actual en operación, el cual fue hecho en clipper. De este sistema se puede obtener información útil para el diseño del modelo de datos. Ya que sirve como referencia al tomarlo como un Prototipo de Concepto (como define Renaud), para así poder determinar visualmente qué es lo que se requiere para el sistema a desarrollar. Por la forma en que fue diseñado el sistema actual se pueden identificar dos entidades, que son: TURNO y SÍNTESIS.

Las cuales se reflejan en dos archivos (.DBF) principales, y se llaman: CDG y TARJACDO. A continuación se describirá la información contenida en estos archivos. Los cuales son relacionados solamente por medio del memorándum. Además de que contienen la información, en el caso de TURNO, de una forma poco óptima debido a que se da un almacenamiento redundante.

ATRIBUTOS DE LA ENTIDAD TURNO
fecha de entrada
folio del número de entrada
entidad
tipo de documento: turno archivo oficio
tema del asunto (catálogo de temas por área)
remitente
cargo del remitente
procedencia (dependencia, organismo, sociedad o ciudadano)
origen (remitente original)
organización social específica
número de peticiones
asunto breve
asunto
fecha de evento
oficio de referencia
fecha de salida
memorándum de salida
área de turno (catálogo)
tipo de instrucción (conocimiento, opinión, manuscritas, procedente, asistir)
instrucción
fecha de contestación
instrucción del C. Secretario
tipo de documento (normal, confidencial)
expediente del archivo
seguimiento
estado de avance

SÍNTESIS
fecha de entrada
prioridad (normal, urgente)
número de entrada
indicador de acuerdo (para acordar con el Secretario Particular o con el C. Secretario)
fecha del evento o compromiso
memorándum de salida
remitente
asunto
datos de la gestión (turno, oficio)
observaciones
expediente del archivo

Como la información anterior es considerada como base, debe ir debidamente ajustada al modelo de datos que se propondrá.

MODELO DE DATOS INICIAL

Después de la rápida revisión del sistema actual, habiendo analizado todos los campos que constituyen dicho sistema. A continuación se presenta el modelo de datos inicial, mediante un DIAGRAMA ENTIDAD-RELACIÓN elaborado en Erwin. El cual mostrará a NIVEL DE CAMPOS LLAVE las relaciones que existen entre las entidades y su cardinalidad.

El significado de cada uno de los campos llave de las entidades se podrán ver más adelante en la especificación completa del modelo. Ahora, para un entendimiento más claro del modelado, se ha dividido en cinco áreas principales:

- **Relación entre catálogos**
- **Recepción de documentos**
- **Trámite**
- **Seguimiento**
- **Entrega**

Relación entre catálogos

Como se puede ver en el diagrama correspondiente, la entidad ENTIDAD que corresponde a todos los estados de la república tiene una relación de UNO A MUCHOS con la entidad MUNICIPIO, ya que cada estado está constituido por al menos un municipio. La entidad PROCEDENCIA contiene todas las áreas (internas y externas) y está relacionado UNO A MUCHOS con la entidad ÁREA, pues incluye las subáreas en que está estructurada cada área. A su vez la entidad ÁREA tiene una relación UNO A MUCHOS con la entidad CARGO, la cual está formada por los diferentes cargos con la llave del nombre de cada una de las personas que conforman dicha área. También se tiene que la entidad CARGO tiene una relación de UNO A UNO con la entidad NOMBRE, pues cada cargo puede tener sólo un nombre. Además la entidad CARGO se relaciona como MUCHOS A UNO con la entidad DIRECCIÓN, considerando que en cada dirección física pueden encontrarse varios cargos. Estos catálogos se utilizarán en varias secciones del modelo, por lo que era importante aclarar la forma en que se

relacionan, además existen otros catálogos pero que no tienen relación con otros catálogos por lo que son más simples de distinguir.

Recepción de documentos

En el diagrama de RECEPCIÓN DE DOCUMENTOS se puede observar que el modelo inicia con la entidad ENTREGA, la cual puede estar constituida por un conjunto de documentos que se desglosan en la entidad DOCUMENTO, por lo tanto éstas guardan una relación de UNO A MUCHOS. Cada uno de los elementos de la entidad ENTREGA puede contener uno o más ANEXOS o DEVOLUCIONES, por lo que guardan una relación de UNO A MUCHOS. Pero además el conjunto de varios elementos de ENTREGA pueden estar contenidos para efectos de recepción de los documentos en una FACTURA, por lo que la relación entre FACTURA y ENTREGA es UNO A MUCHOS.

La entidad DOCUMENTO es una generalización del tipo de documento, los cuales se clasifican en: TRÁMITE, CONTESTACIÓN Y DOCTO-IDENTIFI. La relación entre las entidades es UNO A UNO, pero cada DOCUMENTO puede clasificarse en más de una de las opciones. Las diferentes entidades llamadas ARCHIVO*X* contienen la clave de la clasificación del archivo físico donde será anexado el documento y la entidad TIPO-CLASIFI es el catálogo donde se encuentran las descripciones de estas clasificaciones. La relación de cualquiera de las entidades con su respectiva entidad de ARCHIVO*X*, mantienen una relación UNO A MUCHOS, ya que cada tipo de documento clasificado puede guardarse en diferentes archivos físicos.

Trámite

El diagrama de TRÁMITE se referirá a los documentos que son clasificados como TRÁMITE. Un documento TRÁMITE puede tomar varios caminos, se puede considerar como CONVENIO, DESIGNACIÓN o SÍNTESIS, o también pasar directamente a la entidad GESTIÓN. Lo cual dependerá de las características del asunto, entonces tenemos una relación de UNO A UNO con cualquiera de las opciones. Cuando tenemos una DESIGNACIÓN varias personas pueden estar involucradas con diferentes funciones, por lo que tenemos una relación de UNO A MUCHOS entre DESIGNACIÓN y PERSONAS, además de una relación de UNO A MUCHOS entre el catálogo TIPO-DESIGNACIÓN y la entidad PERSONA.

Cada uno de los elementos de GESTIÓN puede estar constituido por varias peticiones particulares dentro del mismo documento, por lo que existirá una relación de UNO A MUCHOS con la entidad PETICIÓN. También, la entidad GESTIÓN guarda una relación de UNO A MUCHOS con la entidad ORIGEN, ya que cada gestión puede contener varias procedencias, lo cual se da cuando varias instancias van pasando el asunto entre ellas hasta que llega al destino de su tramitación. En este punto se define si el documento clasificado como TRÁMITE pasará a la generalización en TURNO u OFICIO para ser gestionado, o se determina que pasará a ser un documento perteneciente a la entidad DOCTO-S-GESTIÓN los cuales no tienen gestión, por lo que tenemos que existe una relación de UNO A UNO entre las entidades de GESTIÓN Y DOCTO-S-GESTIÓN. Ahora, GESTIÓN tiene una generalización

por tipo de trámite, que son: TURNO y OFICIO. Y cada GESTIÓN puede clasificarse como TURNO, OFICIO, MÁS DE UN TURNO, MÁS DE UN OFICIO y TURNOS además de OFICIOS. Por lo que hay una relación de UNO A MUCHOS entre GESTIÓN - TURNO Y GESTIÓN-OFICIO. Cada OFICIO está relacionado UNO A MUCHOS con la entidad COPIA, ya que cada oficio puede tener varias copias marcadas.

Tenemos otra entidad llamada INTERNO, el cual es un documento que surge de manera interna que tiene que ser gestionado. Para estos casos se hacen oficios, y es importante mantenerlos dentro del sistema porque posteriormente tendrán un seguimiento. Guarda una relación de UNO A MUCHOS con la entidad COPIA2, pues se puede tener más de una copia marcada en cada oficio. Y la entidad FIRMA que contiene a las personas que pueden firmar los oficios INTERNOS y los trámites de OFICIO.

Seguimiento

En el SEGUIMIENTO a las peticiones tenemos a la entidad CONTESTACIÓN en la cual se registran todos los documentos que son la contestación de los trámites realizados a cada asunto. Si un seguimiento está constituido por un conjunto de oficios, entonces entre las entidades CARPETA y CONTESTACIÓN tendremos una relación UNO A MUCHOS. Y se tiene una relación UNO A MUCHOS entre las entidades CONTESTACIÓN E INVOLUCRADO ya que cada oficio de CONTESTACIÓN puede tener varias copias marcadas para conocimiento. Ahora, cada contestación puede responder a más de un asunto por lo que existe una relación UNO A MUCHOS entre las entidades CONTESTACIÓN y SEGUIMIENTO. Cada seguimiento tiene un estado de avance. Entonces se da una relación UNO A MUCHOS entre las entidades TIPO-ESTADO y SEGUIMIENTO.

La entidad SEGUIMIENTO guarda una relación de MUCHOS A MUCHOS con las entidades INTERNO, OFICIO, TURNO y PETICIÓN. Porque un conjunto de seguimientos pueden corresponder a alguno de los asuntos tramitados en alguna de estas opciones o también en el caso de la entidad PETICIÓN si se quiere hacer el seguimiento hasta ese nivel. Para implementar esta relación se genera una entidad llamada SEG-GESTIÓN.

Entrega

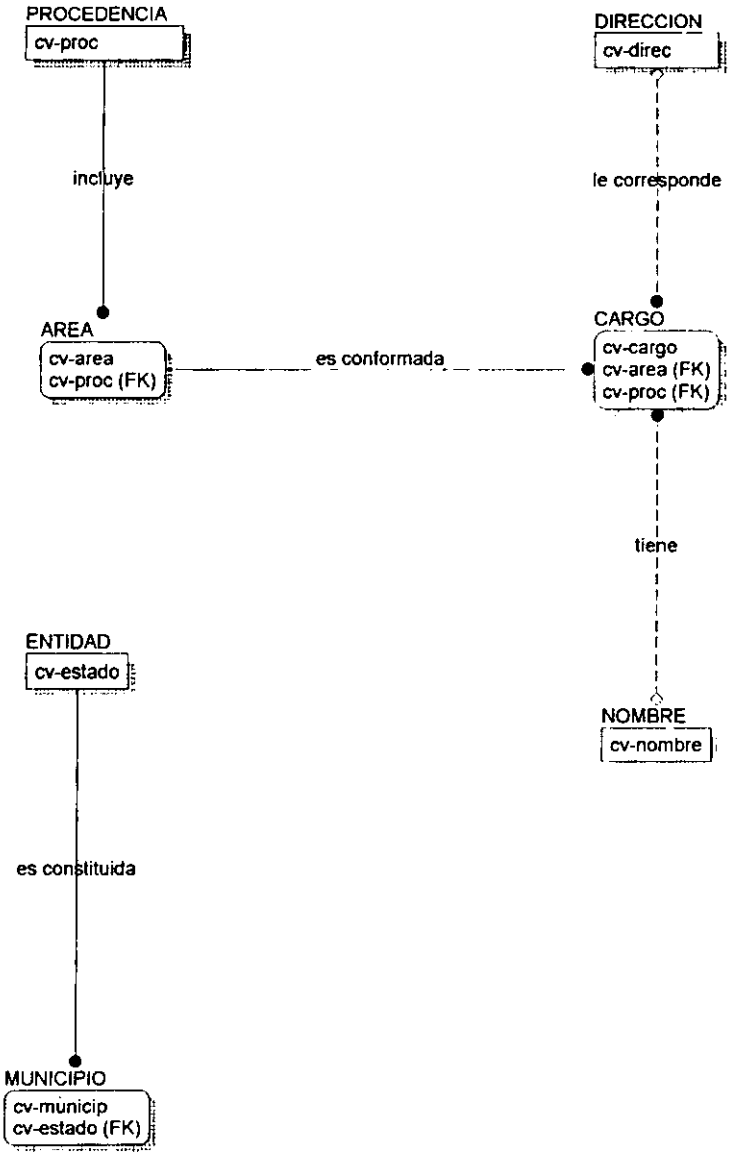
Para cumplir con la normatividad establecida, se tiene que hacer la entrega física de los documentos tramitados. Para esto tenemos la entidad DISTRIBUCIÓN que contará con los datos generales del momento en que se emitirá la entrega y quién lo hará. La entidad MENSAJERO es un catálogo que contiene los datos de las personas encargadas de la distribución física de los documentos y guardan una relación de UNO A MUCHOS entre MENSAJERO y DISTRIBUCIÓN.

Cada DISTRIBUCIÓN estará constituida por uno o más PAQUETES, por lo que entre las entidades DISTRIBUCIÓN y PAQUETE existe una relación UNO A MUCHOS. Cada PAQUETE tendrá una dirección física, la cual se obtiene de la entidad DIRECCIÓN, entonces entre el catálogo DIRECCIÓN y la entidad PAQUETE existe una relación UNO A MUCHOS.

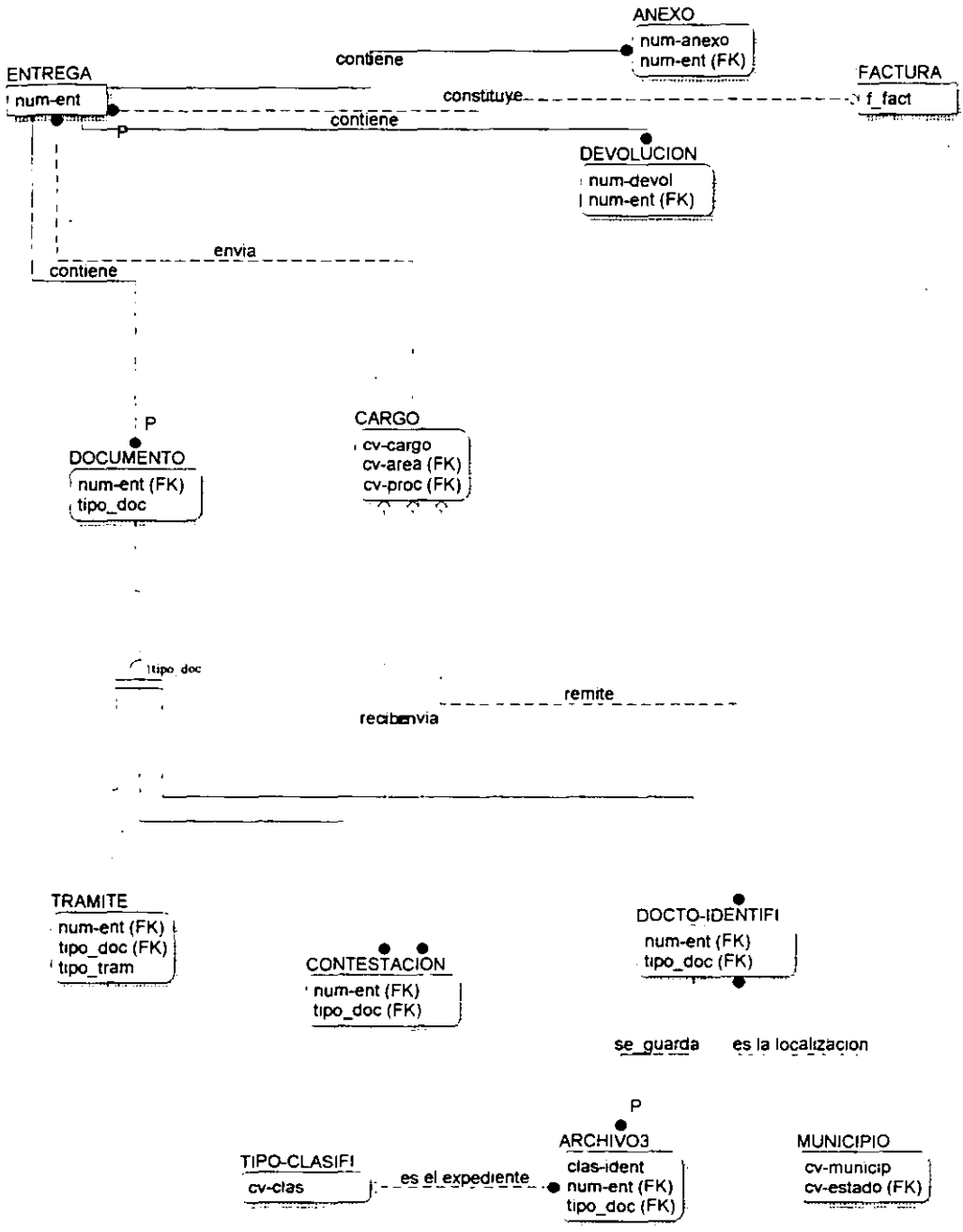
A su vez cada paquete puede estar constituido por TURNOS, OFICIOS y oficios INTERNOS que van a una área específica de la dirección física. Como esto nos está reflejando una relación MUCHOS A MUCHOS, tenemos que implementar una entidad intermedia llamada PAQ-GEST. Entendiendo que una dirección física puede estar constituida por más de una área específica. Por lo que un paquete contiene las entregas para cada una de las áreas que constituyen una dirección física.

Para concluir este punto se puede ver el modelado de todo el sistema en el último diagrama llamado MODELO DE DATOS SIN CATÁLOGOS, en el cual podemos observar todas las entidades y cómo se relacionan, para así tener un panorama completo. Para lo cual se eliminarán los catálogos para quitar la complejidad que se agrega con tantas relaciones que involucran.

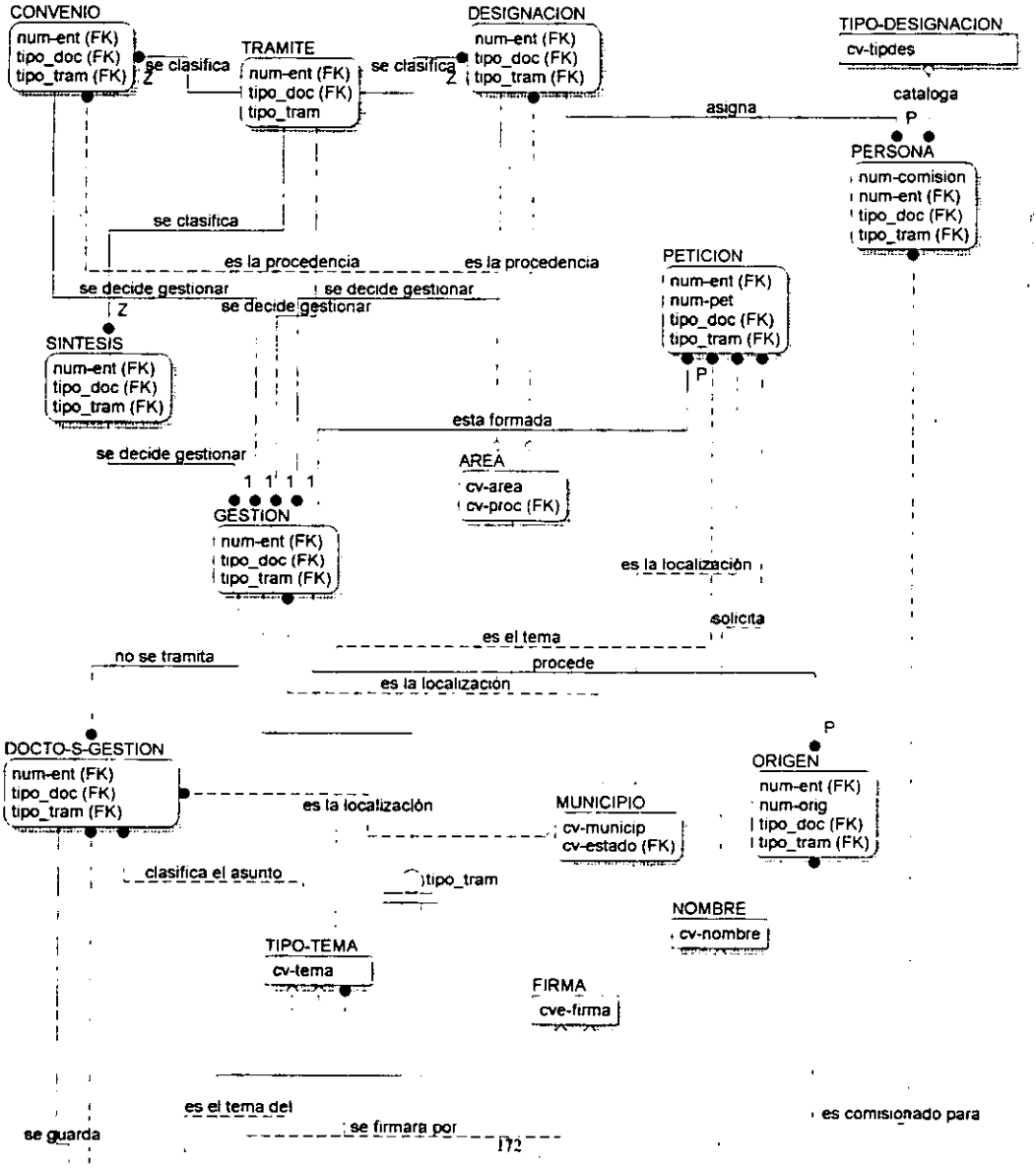
RELACION ENTRE CATALOGOS

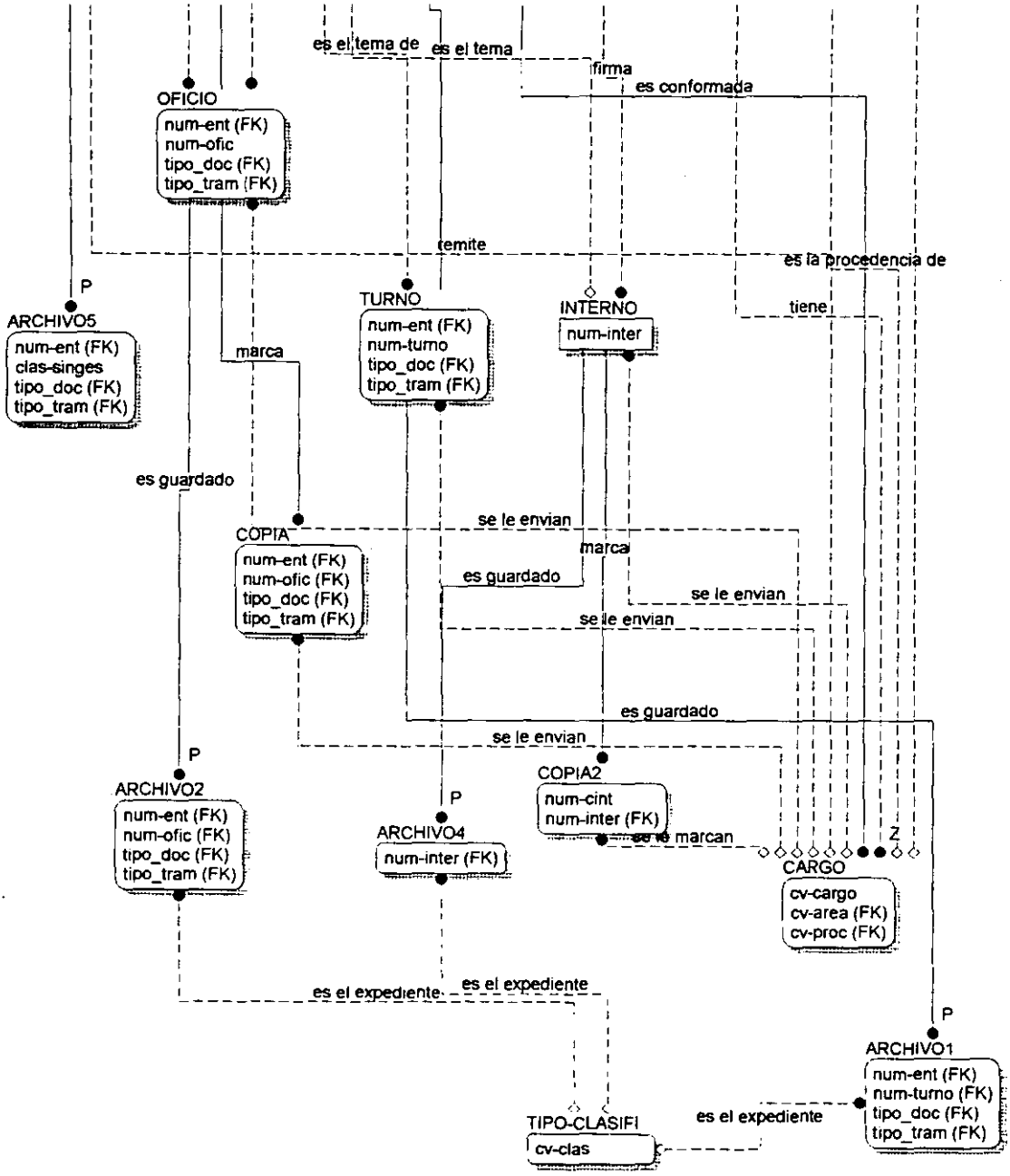


RECEPCION DE DOCUMENTOS

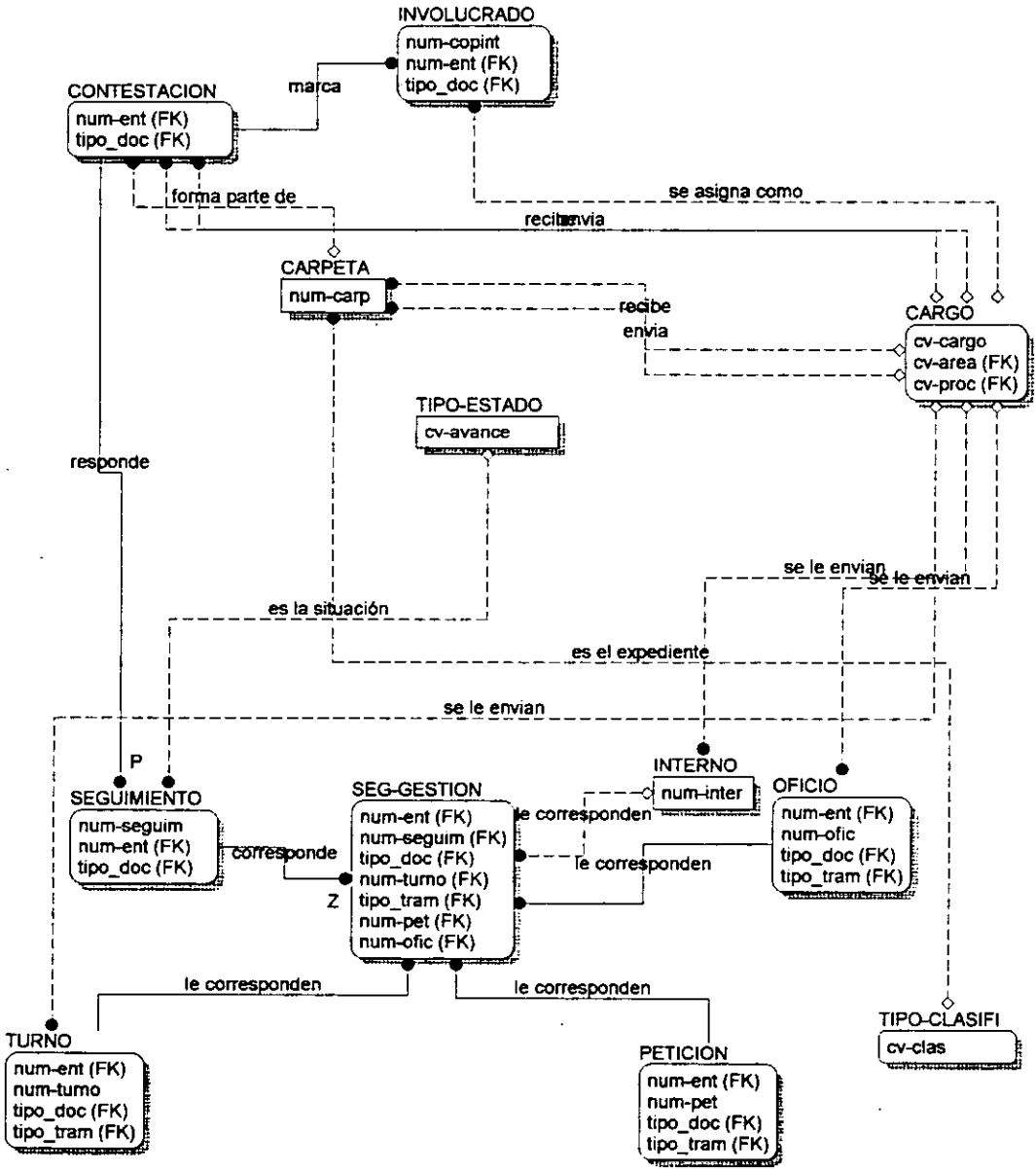


TRAMITE

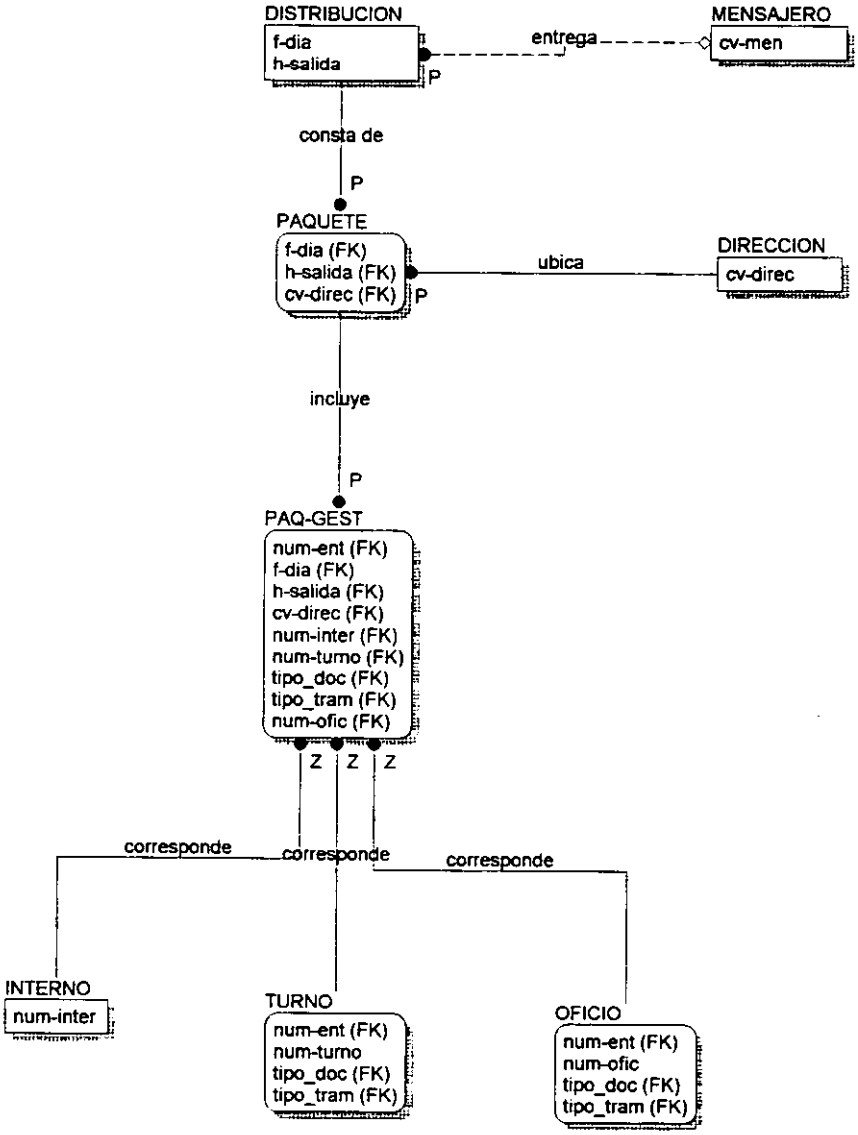

CONVENIO
num-ent (FK)
tipo_doc (FK)
tipo_tram (FK)
se clasifica
Z
TRAMITE
num-ent (FK)
tipo_doc (FK)
tipo_tram
se clasifica
Z
DESIGNACION
num-ent (FK)
tipo_doc (FK)
tipo_tram (FK)
TIPO-DESIGNACION
cv-tipdes
cataloga
asigna
P
PERSONA
num-comision
num-ent (FK)
tipo_doc (FK)
tipo_tram (FK)
se clasifica
es la procedencia
es la procedencia
PETICION
num-ent (FK)
num-pet
tipo_doc (FK)
tipo_tram (FK)
P
se decide gestionar
se decide gestionar
se decide gestionar
Z
SINTESIS
num-ent (FK)
tipo_doc (FK)
tipo_tram (FK)
esta formada
se decide gestionar
1 1 1 1
AREA
cv-area
cv-proc (FK)
GESTION
num-ent (FK)
tipo_doc (FK)
tipo_tram (FK)
es la localización
solicita
es el tema
no se tramita
procede
es la localización
P
DOCTO-S-GESTION
num-ent (FK)
tipo_doc (FK)
tipo_tram (FK)
es la localización
MUNICIPIO
cv-municip
cv-estado (FK)
ORIGEN
num-ent (FK)
num-orig
tipo_doc (FK)
tipo_tram (FK)
clasifica el asunto
tipo_tram
NOMBRE
cv-nombre
TIPO-TEMA
cv-tema
FIRMA
cve-firma
es el tema del
se guarda
se firmara por
172
es comisionado para


es el tema de
es el tema
firma
es conformada
OFICIO
num-ent (FK)
num-ofic
tipo_doc (FK)
tipo_tram (FK)
remite
es la procedencia de
P
ARCHIVO5
num-ent (FK)
clas-singes
tipo_doc (FK)
tipo_tram (FK)
TURNO
num-ent (FK)
num-turno
tipo_doc (FK)
tipo_tram (FK)
INTERNO
num-inter
tiene
marca
es guardado
COPIA
num-ent (FK)
num-ofic (FK)
tipo_doc (FK)
tipo_tram (FK)
se le envian
marca
es guardado
se le envian
se le envian
es guardado
se le envian
P
ARCHIVO2
num-ent (FK)
num-ofic (FK)
tipo_doc (FK)
tipo_tram (FK)
P
ARCHIVO4
num-inter (FK)
COPIA2
num-cint
num-inter (FK)
se le marcan
Z
CARGO
cv-cargo
cv-area (FK)
cv-proc (FK)
es el expediente
es el expediente
P
ARCHIVO1
num-ent (FK)
num-turno (FK)
tipo_doc (FK)
tipo_tram (FK)
TIPO-CLASIFI
cv-clas
es el expediente

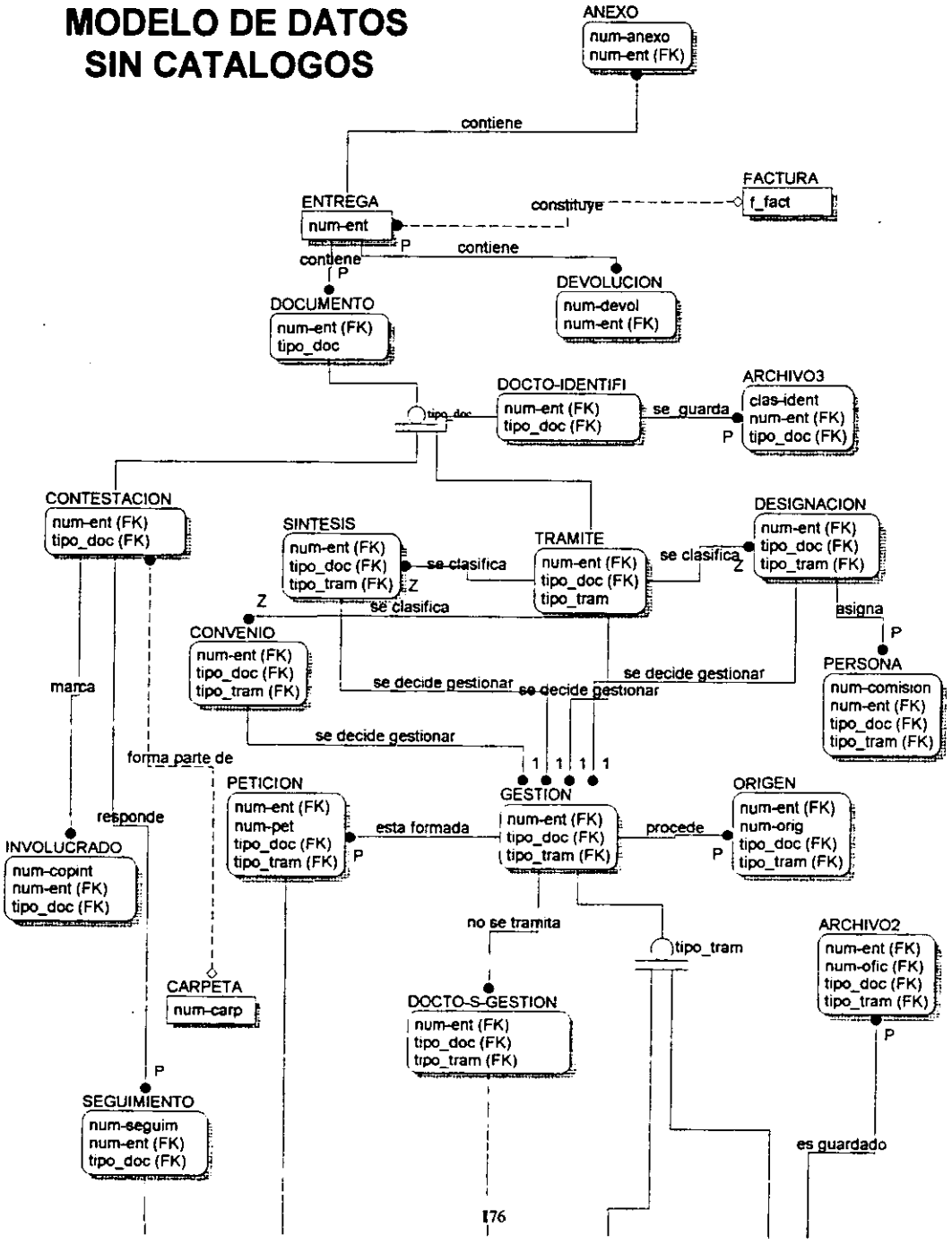
SEGUIMIENTO

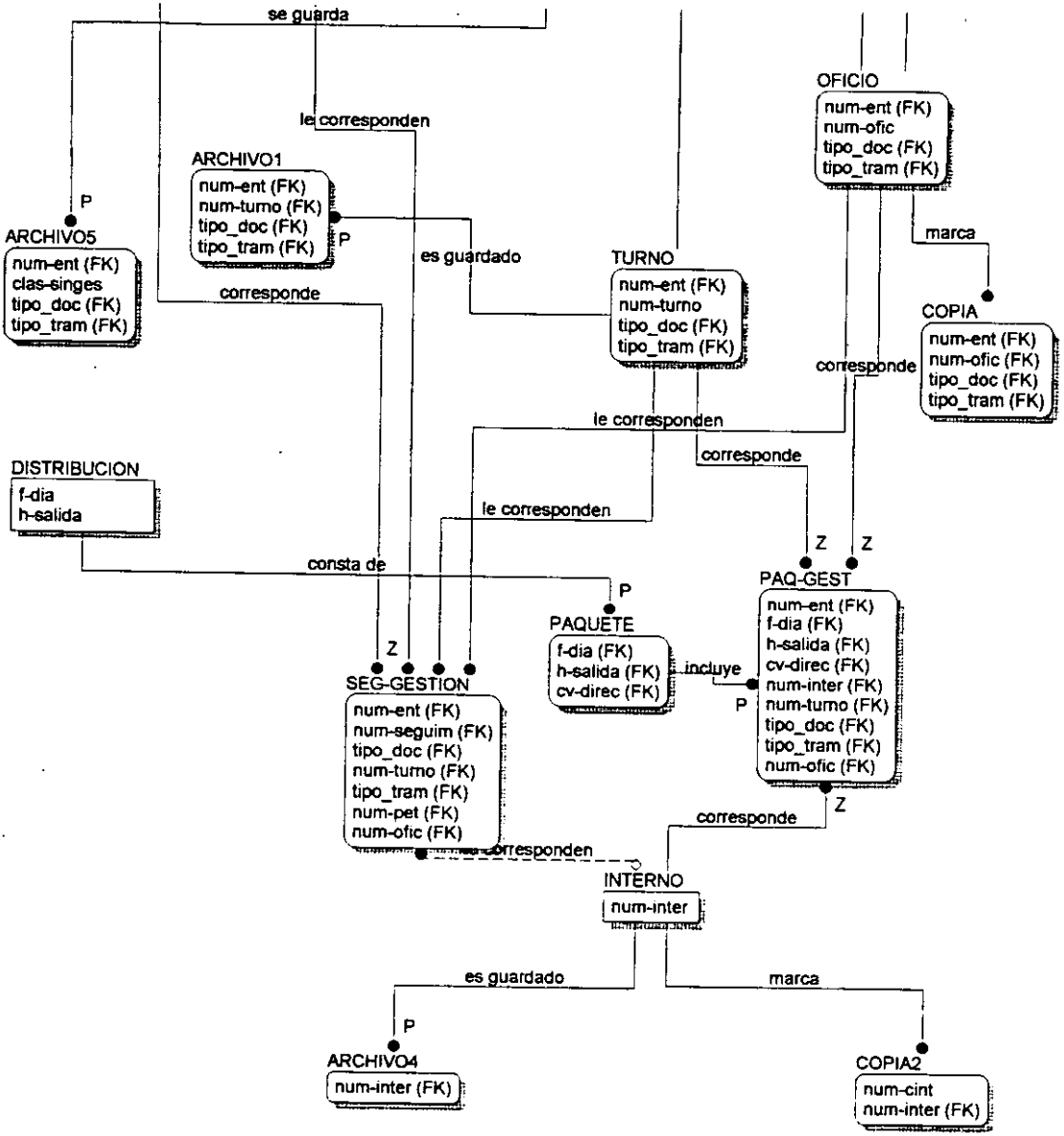


ENTREGA


DISTRIBUCION
f-dia
h-salida
P
entrega
MENSAJERO
cv-men
consta de
P
PAQUETE
f-dia (FK)
h-salida (FK)
cv-direc (FK)
P
ubica
DIRECCION
cv-direc
incluye
P
PAQ-GEST
num-ent (FK)
f-dia (FK)
h-salida (FK)
cv-direc (FK)
num-inter (FK)
num-turno (FK)
tipo_doc (FK)
tipo_tram (FK)
num-ofic (FK)
Z
Z
Z
corresponde
corresponde
corresponde
INTERNO
num-inter
TURNO
num-ent (FK)
num-turno
tipo_doc (FK)
tipo_tram (FK)
OFICIO
num-ent (FK)
num-ofic
tipo_doc (FK)
tipo_tram (FK)

MODELO DE DATOS SIN CATALOGOS




se guarda
le corresponden
OFICIO
num-ent (FK)
num-ofic
tipo_doc (FK)
tipo_tram (FK)
ARCHIVO1
num-ent (FK)
num-turno (FK)
tipo_doc (FK)
tipo_tram (FK)
P
P
ARCHIVO5
num-ent (FK)
clas-singes
tipo_doc (FK)
tipo_tram (FK)
es guardado
TURNO
num-ent (FK)
num-turno
tipo_doc (FK)
tipo_tram (FK)
marca
COPIA
num-ent (FK)
num-ofic (FK)
tipo_doc (FK)
tipo_tram (FK)
corresponde
corresponde
le corresponden
corresponde
DISTRIBUCION
f-dia
h-salida
le corresponden
Z
Z
consta de
P
PAQ-GEST
num-ent (FK)
f-dia (FK)
h-salida (FK)
cv-direc (FK)
num-inter (FK)
num-turno (FK)
tipo_doc (FK)
tipo_tram (FK)
num-ofic (FK)
PAQUETE
f-dia (FK)
h-salida (FK)
cv-direc (FK)
incluye
P
Z
SEG-GESTION
num-ent (FK)
num-seguim (FK)
tipo_doc (FK)
num-turno (FK)
tipo_tram (FK)
num-pet (FK)
num-ofic (FK)
Z
corresponde
corresponden
INTERNO
num-inter
es guardado
marca
P
ARCHIVO4
num-inter (FK)
COPIA2
num-cint
num-inter (FK)

MODELO DE DATOS COMPLETAMENTE ESPECIFICADO Y LOS REQUERIMIENTOS PARA LA FUNCIONALIDAD A SER PROPORCIONADA

Para la especificación completa del modelo, a continuación se definirán a detalle las ENTIDADES, ATRIBUTOS y RELACIONES entre las entidades, mediante los siguientes listados:

- **Reporte de entidades**
- **Reporte de atributos**
- **Reporte de relaciones**

En el reporte de entidades tenemos por orden alfabético el nombre de las entidades, la definición y el tipo. Una entidad puede ser Dependiente o Independiente. Dependiente significa que no tendría significado su existencia sino estuviera relacionada con otra entidad, entonces la entidad con que se relaciona es la entidad PADRE y la entidad dependiente es la entidad HIJO. Por lo que la existencia de una entidad Independiente no depende de las relaciones con otras entidades.

En el reporte de atributos tenemos los atributos de cada entidad con su respectiva definición, el tipo de atributo y el tipo de dato. El tipo de dato indica si el atributo es una llave o no, para lo cual se presentan los siguientes casos:

- **Owned Key**: Forma parte de la llave principal de la tabla.
- **Owned Non-Key**: Atributo que no forma parte de la llave principal.
- **Foreign Key**: Atributo que es llave principal de una primera tabla, pero por la relación que existe con una segunda tabla y de acuerdo a la teoría relacional pasa a formar parte de la segunda tabla.
- **Foreign Non-Key**: Es un atributo que forma parte de la llave principal de una primera tabla, pero por el tipo de relación el atributo pasa a ser parte de una segunda tabla pero no como parte de la llave principal.

El tipo de dato se refiere a un atributo que es carácter, numérico, de tiempo, de fecha o tipo carácter pero como un texto largo. Los cuales son los siguientes:

- **Char**
- **Integer**
- **Time**
- **Date**
- **Long Varchar**

El Reporte de Relaciones contiene la entidad PADRE con su entidad HIJO y su respectiva frase con que se relaciona, además del tipo de relación y cardinalidad. En el tipo de relación tenemos que puede ser una relación Identificada (Identifying), No-Identificada (No - Identifying) o Subtipo de una generalización (Subtype). Una relación Identificada significa

que la entidad PADRE hereda su llave principal a la entidad HIJO formando parte de su llave principal, por lo que se puede saber qué registro o registros de la entidad HIJO le corresponden a cada uno de los registros de la entidad PADRE.

Una relación No-Identificada es cuando la entidad PADRE hereda su llave principal a la entidad HIJO, pero no para formar parte de la llave principal, sino como un atributo más. Por lo que no podemos saber mediante las llaves principales qué registros de la entidad HIJO corresponden a cada registro de la entidad PADRE.

En la Cardinalidad se presentaron los siguientes casos:

- uno-para-cero-uno-o-más (one-to-zero-one-or-more)
- cero-o-uno-para-cero uno-o-más (zero-or-one-to-zero-one-or-more)
- uno-para-uno-o-más (one-to-one-or-more (P))
- uno-para-exactamente-1 (one-to-exactly-1)
- cero-o-uno-para-uno-o-más (zero-or-one-to-one-or-more (P))
- uno-para-uno, entidad que forma parte de una generalización (is a)
- uno-para-cero-o-uno (one-to-zero-or-one(Z))
- cero-o-uno-para-cero-o-uno (zero-or-one-to-zero-or-one (Z))

Requerimientos funcionales

Teniendo como base el Modelo de Datos antes definido, se debe construir una aplicación que tenga las funciones adecuadas para cubrir los requerimientos. Esta aplicación tendrá dos módulos o subsistemas fundamentales:

- **Subsistema Central de Gestión**
- **Subsistema de Seguimiento a la Gestión**

El Subsistema Central de Gestión tendrá dos funciones principales: Recepción de Documentos y Gestión de los asuntos para su atención. Por lo que éste será el punto de distribución de las tareas hacia las áreas internas de la institución.

La parte correspondiente al Subsistema de Seguimiento a la Gestión se ubicará en las áreas internas de la organización, que están encargadas de darle una atención a los asuntos que les son remitidos. A los cuales deben de darles una resolución positiva o negativa. Por lo que el papel principal de este subsistema es el seguimiento a las gestiones. Considerando que si cada área controla el seguimiento de sus asuntos, se evitará que halla duplicación del trabajo, al llevarse el seguimiento en cada área y de forma central. Entonces, con el sistema tendremos una fuente de información corporativa. Pero también, en el área central (Secretaría Técnica) será posible dar seguimiento a cualquier asunto gestionado, considerando que no todas las áreas a las que se gestionan los asuntos estarán incorporadas a la RED AMPLIA de la institución.

REPORTE DE ENTIDADES

NOMBRE DE LA ENTIDAD	DEFINICIÓN DE LA ENTIDAD	TIPO DE ENTIDAD
ANEXO	Cualquier tipo de objeto que llega junto con los documentos de la ENTREGA, el cual debe ser identificado.	Dependent
ARCHIVO1	Clasificación del lugar en que será almacenado físicamente el documento gestionado por TURNO.	Dependent
ARCHIVO2	Clasificación del lugar en que será almacenado físicamente el documento gestionado por OFICIO.	Dependent
ARCHIVO3	Clasificación del lugar en que será almacenado físicamente el documento tramitado como DOCUMENTO IDENTIFICADO.	Dependent
ARCHIVO4	Clasificación del lugar en que será almacenado físicamente el documento gestionado por OFICIO INTERNO.	Dependent
ARCHIVO5	Clasificación del lugar en que será almacenado físicamente el documento identificado como DOCUMENTO SIN GESTIÓN.	Dependent
AREA	Áreas en que se puede dividir cada procedencia.	Dependent
CARGO	Catálogo de cargos o puestos que existen en cada una de las áreas de cada procedencia.	Dependent
CARPETA	Datos generales del documento que respalda el listado de contestaciones a las gestiones.	Independent
CONTESTACION	Documento(s) de seguimiento(s) a las peticiones gestionadas a las diversas áreas de atención.	Dependent

REPORTE DE ENTIDADES

NOMBRE DE LA ENTIDAD	DEFINICIÓN DE LA ENTIDAD	TIPO DE ENTIDAD
CONVENIO	Documento referente a CONVENIOS efectuados entre organizaciones o dependencias gubernamentales.	Dependent
COPIA	Copia marcada que se le hace al OFICIO.	Dependent
COPIA2	Copia marcada que se le hace al OFICIO INTERNO.	Dependent
DESIGNACION	Designación de comisiones, cargos o representaciones, en eventos o ante otras dependencias u organizaciones.	Dependent
DEVOLUCION	Documentos que por no corresponder al ámbito de competencia de esta secretaría son reintegrados.	Dependent
DIRECCION	Catálogo de direcciones físicas del catálogo de cargos.	Independent
DISTRIBUCION	Asignación de los documentos para entregarse al área correspondiente.	Independent
DOCTO-IDENTIFI	Documento que por ser no relevante, o que llegue de forma periódica, solo se registra con los datos mas básicos para llevar un control.	Dependent
DOCTO-S-GESTION	Documento que requiere ser registrado, pero que no es necesario darle algún trámite.	Dependent
DOCUMENTO	Documento (documentos) clasificado(s) para su tramitación. Los tipos de tramitación son: TRAMITE CONTESTACION Y DOCUMENTO - IDENTIFICADO	Dependent

REPORTE DE ENTIDADES

NOMBRE DE LA ENTIDAD	DEFINICIÓN DE LA ENTIDAD	TIPO DE ENTIDAD
ENTIDAD	Catálogo que contiene los estados de la república.	Independent
ENTREGA	Una entrega es un conjunto de documentos o un solo documento que son recibidos.	Independent
FACTURA	Relación que incluye documentos recibidos por correo y mensajería, los cuales fueron recibidos en otra área de la secretaría; además de los procedentes de palacio nacional.	Independent
FIRMA	Los datos correspondientes a las personas que tienen la autoridad de firmar un oficio.	Independent
GESTION	Documento(s) que para su atención debe(n) ser remitido(s) a la área de atención correspondiente. Pudiendo clasificarse de acuerdo a: CONVENIO DESIGNACIÓN SÍNTESIS NINGUNO CONVENIO - DESIGNACIÓN CONVENIO - SÍNTESIS DESIGNACIÓN - SÍNTESIS	Dependent
INTERNO	Oficio que se hace con un formato determinado por ser un asunto que no corresponde con alguna entrega, por lo que son oficios que se elaboran de manera interna.	Independent
INVOLUCRADO	Copia marcada al interesado en los oficios de contestación.	Dependent
MENSAJERO	Datos de las personas que distribuyen los documentos.	Independent
MUNICIPIO	Catálogo que contiene los	Dependent

REPORTE DE ENTIDADES

NOMBRE DE LA ENTIDAD	DEFINICIÓN DE LA ENTIDAD	TIPO DE ENTIDAD
	municipios de cada uno de los estados de la república.	
NOMBRE	Catálogo que contiene nombres de persona.	Independent
OFICIO	Oficio que se hace con un formato determinado por ser un asunto especial o por corresponder a alguna área externa a la secretaría.	Dependent
ORIGEN	Procedencia de la petición.	Dependent
PAQ-GEST	Enlace entre el paquete y cada uno de los documentos gestionados que se llevaran al área correspondiente.	Dependent
PAQUETE	Agrupación de los documentos a ser entregados de acuerdo a las direcciones físicas del recorrido.	Dependent
PERSONA	Datos de la persona designada.	Dependent
PETICION	Petición específica que puede ir incluida en un documento.	Dependent
PROCEDENCIA	Catálogo de la clasificación de dependencias, organismos y tipos de organizaciones.	Independent
SEG-GESTION	Identificación del enlace del seguimiento con su correspondiente turno u oficio.	Dependent
SEGUIMIENTO	Respuesta a la petición del documento.	Dependent
SINTESIS	Documento que por su importancia se le da un trato especial para ser revisados por el secretario.	Dependent
TIPO-CLASIFI	Catálogo de las clasificaciones físicas de los expedientes en donde se guardan documentos.	Dependent

REPORTE DE ENTIDADES

NOMBRE DE LA ENTIDAD	DEFINICIÓN DE LA ENTIDAD	TIPO DE ENTIDAD
TIPO-DESIGNACION	Tipos de designación que se le asigna a una persona para el cumplimiento de su comisión.	Independent
TIPO-ESTADO	Catálogo de SITUACIONES DE AVANCE que se tiene en cuanto a la gestión de un asunto.	Independent
TIPO-TEMA	Catálogo que contiene clasificaciones temáticas en las cuales se incluyen los diversos asuntos.	Independent
TRAMITE	Documento(s) que para su atención debe(n) ser analizado(s) para definir si son: TURNO, OFICIO o SIN GESTIÓN.	Dependent
TURNO	Oficio con formato especifico que es firmado por la Secretaria Técnica.	Dependent

REPORTE DE ATRIBUTOS

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
ANEXO	desc-anexo	Descripción física de cada objeto que se considera un ANEXO.	Owned Non-key	CHAR(100)
ANEXO	num-anexo	Número consecutivo que se le da a cada ANEXO para su identificación.	Owned Key	INTEGER
ANEXO	num-ent		Foreign Key	INTEGER
ARCHIVO1	cv-clas		Foreign Non-key	CHAR(35)
ARCHIVO1	num-ent		Foreign Key	INTEGER
ARCHIVO1	num-turno		Foreign Key	INTEGER
ARCHIVO1	tipo_doc		Foreign Key	CHAR(2)
ARCHIVO1	tipo_tram		Foreign Key	CHAR(2)
ARCHIVO2	cv-clas		Foreign Non-key	CHAR(35)
ARCHIVO2	num-ent		Foreign Key	INTEGER
ARCHIVO2	num-ofic		Foreign Key	INTEGER
ARCHIVO2	tipo_doc		Foreign Key	CHAR(2)
ARCHIVO2	tipo_tram		Foreign Key	CHAR(2)
ARCHIVO3	clas-ident	Número consecutivo que se le asigna a cada clasificación física del documento tramitado como DOCUMENTO IDENTIFICADO.	Owned Key	INTEGER
ARCHIVO3	cv-clas		Foreign Non-key	CHAR(35)
ARCHIVO3	num-ent		Foreign Key	INTEGER
ARCHIVO3	tipo_doc		Foreign Key	CHAR(2)
ARCHIVO4	cv-clas		Foreign Non-key	CHAR(35)
ARCHIVO4	num-inter		Foreign Key	CHAR(2)
ARCHIVO5	clas-singes	Número consecutivo asignado a cada clasificación física que se le da a un documento tramitado	Owned Key	INTEGER

REPORTE DE ATRIBUTOS

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
		como SIN GESTIÓN.		
ARCHIVO5	cv-clas	Clasificación del archivo físico.	Owned Non-key	CHAR(35)
ARCHIVO5	num-ent		Foreign Key	INTEGER
ARCHIVO5	tipo_doc		Foreign Key	CHAR(2)
ARCHIVO5	tipo_tram		Foreign Key	CHAR(2)
AREA	cv-area	Clave del área para su identificación.	Owned Key	INTEGER
AREA	cv-proc		Foreign Key	INTEGER
AREA	desc-area	Nombre de cada área de la procedencia.	Owned Non-key	CHAR(50)
CARGO	cv-area		Foreign Key	INTEGER
CARGO	cv-cargo	Clave del cargo para su identificación.	Owned Key	INTEGER
CARGO	cv-direc		Foreign Non-key	INTEGER
CARGO	cv-nombre		Foreign Non-key	INTEGER
CARGO	cv-proc		Foreign Key	INTEGER
CARGO	desc-cargo	Nombre del cargo.	Owned Non-key	CHAR(100)
CARPETA	cont-seg	Número de seguimientos que tiene el listado de respuestas.	Owned Non-key	INTEGER
CARPETA	cv-area		Foreign Non-key	INTEGER
CARPETA	cv-cargo		Foreign Non-key	INTEGER
CARPETA	cv-cargo		Foreign Non-key	INTEGER
CARPETA	cv-clas		Foreign Non-key	CHAR(35)
CARPETA	cv-proc		Foreign Non-key	INTEGER
CARPETA	f-ofcont	Fecha del oficio de respuestas.	Owned Non-key	DATE
CARPETA	num-carp	Número consecutivo que se asigna a la CARPETA para su identificación.	Owned Key	INTEGER
CARPETA	ofic-cont	Número de oficio con que llega	Owned Non-key	INTEGER

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
		foliado el listado de respuestas.		
CARPETA	text-cont	Texto del oficio de contestación del listado de respuestas.	Owned Non-key	LONG VARCHAR
CONTESTACION	cv-area		Foreign Non-key	INTEGER
CONTESTACION	cv-cargo		Foreign Non-key	INTEGER
CONTESTACION	cv-cargo		Foreign Non-key	INTEGER
CONTESTACION	cv-proc		Foreign Non-key	INTEGER
CONTESTACION	fecha-ofic	Fecha que tiene el documento de contestación a la gestión.	Owned Non-key	DATE
CONTESTACION	num-carp		Foreign Non-key	INTEGER
CONTESTACION	num-ent		Foreign Key	INTEGER
CONTESTACION	num-ofcont	Número de oficio con que viene foliado el documento de contestación a la gestión.	Owned Non-key	CHAR(35)
CONTESTACION	tipo_doc		Foreign Key	CHAR(2)
CONVENIO	asunto-conv	Asunto especifico que se tratara en dicho convenio.	Owned Non-key	LONG VARCHAR
CONVENIO	cv-area		Foreign Non-key	INTEGER
CONVENIO	cv-clas	Clasificación del archivo físico.	Owned Non-key	CHAR(4)
CONVENIO	cv-proc		Foreign Non-key	INTEGER
CONVENIO	desc-conv	Descripción del convenio a que se hace referencia.	Owned Non-key	CHAR(100)
CONVENIO	num-ent		Foreign Key	INTEGER
CONVENIO	refer-ext	Número de oficio con que esta foliado el convenio, si tiene.	Owned Non-key	CHAR(35)
CONVENIO	tipo_doc		Foreign Key	CHAR(2)
CONVENIO	tipo_tram		Foreign Key	CHAR(2)
COPIA	cv-area		Foreign Non-key	INTEGER
COPIA	cv-cargo		Foreign Non-key	INTEGER

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
COPIA	cv-proc		Foreign Non-key	INTEGER
COPIA	num-ent		Foreign Key	INTEGER
COPIA	num-ofic		Foreign Key	INTEGER
COPIA	tipo_doc		Foreign Key	CHAR(2)
COPIA	tipo_tram		Foreign Key	CHAR(2)
COPIA2	cv-area		Foreign Non-key	INTEGER
COPIA2	cv-cargo		Foreign Non-key	INTEGER
COPIA2	cv-proc		Foreign Non-key	INTEGER
COPIA2	num-cint	Número que se asigna a la copia marcada de las gestiones por OFICIO INTERNO para su identificación.	Owned Key	INTEGER
COPIA2	num-inter		Foreign Key	CHAR(2)
DESIGNACION	asunto-des	Descripción del asunto para la que se requiere la designación.	Owned Non-key	LONG VARCHAR
DESIGNACION	cv-area		Foreign Non-key	INTEGER
DESIGNACION	cv-clas	Clasificación del archivo físico.	Owned Non-key	CHAR(4)
DESIGNACION	cv-proc		Foreign Non-key	INTEGER
DESIGNACION	num-ent		Foreign Key	INTEGER
DESIGNACION	refer-exter	Número de oficio con que esta foliada la designación.	Owned Non-key	CHAR(35)
DESIGNACION	tipo_doc		Foreign Key	CHAR(2)
DESIGNACION	tipo_tram		Foreign Key	CHAR(2)
DEVOLUCION	desc-devol	Descripción del documento que será reintegrado.	Owned Non-key	CHAR(100)
DEVOLUCION	num-devol	Número consecutivo que identifica las devoluciones.	Owned Key	INTEGER
DEVOLUCION	num-ent		Foreign Key	INTEGER
DIRECCION	calle	Nombre de la calle.	Owned Non-key	CHAR(50)

REPORTE DE ATRIBUTOS

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
DIRECCION	codpostal	Código postal.	Owned Non-key	CHAR(6)
DIRECCION	colonia	Colonia en que se localiza.	Owned Non-key	CHAR(30)
DIRECCION	cv-direc	Clave de la dirección para su identificación.	Owned Key	INTEGER
DIRECCION	deleg-est	Delegación política o estado de la república.	Owned Non-key	CHAR(30)
DIRECCION	num-i-e-p	Número externo, interno y piso del lugar.	Owned Non-key	CHAR(30)
DISTRIBUCION	cv-men		Foreign Non-key	INTEGER
DISTRIBUCION	f-dia	Fecha en que se registra la distribución.	Owned Key	DATE
DISTRIBUCION	h-llegada	Hora en que llega el mensajero.	Owned Non-key	TIME
DISTRIBUCION	h-salida	Hora en que sale el mensajero.	Owned Key	TIME
DISTRIBUCION	kilometraje	Kilómetros recorridos en la distribución.	Owned Non-key	INTEGER
DOCTO-IDENTIFI	cv-area		Foreign Non-key	INTEGER
DOCTO-IDENTIFI	cv-cargo		Foreign Non-key	INTEGER
DOCTO-IDENTIFI	cv-estado		Foreign Non-key	INTEGER
DOCTO-IDENTIFI	cv-municip		Foreign Non-key	INTEGER
DOCTO-IDENTIFI	cv-proc		Foreign Non-key	INTEGER
DOCTO-IDENTIFI	num-ent		Foreign Key	INTEGER
DOCTO-IDENTIFI	obser-ident	Alguna observación específica del DOCUMENTO IDENTIFICADO.	Owned Non-key	CHAR(100)
DOCTO-IDENTIFI	refer-ident	Número de oficio con que está foliado el DOCUMENTO IDENTIFICADO.	Owned Non-key	CHAR(35)
DOCTO-IDENTIFI	tipo_doc		Foreign Key	CHAR(2)
DOCTO-S-	asun-singes	Descripción del asunto del	Owned Non-key	LONG

REPORTE DE ATRIBUTOS

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
GESTION		DOCUMENTO SIN GESTIÓN.		VARCHAR
DOCTO-S-GESTION	cv-area		Foreign Non-key	INTEGER
DOCTO-S-GESTION	cv-cargo		Foreign Non-key	INTEGER
DOCTO-S-GESTION	cv-estado		Foreign Non-key	INTEGER
DOCTO-S-GESTION	cv-municip		Foreign Non-key	INTEGER
DOCTO-S-GESTION	cv-proc		Foreign Non-key	INTEGER
DOCTO-S-GESTION	cv-tema		Foreign Non-key	CHAR(4)
DOCTO-S-GESTION	num-ent		Foreign Key	INTEGER
DOCTO-S-GESTION	refer-singes	Número de oficio con que está foliado el DOCUMENTO SIN GESTIÓN.	Owned Non-key	CHAR(35)
DOCTO-S-GESTION	tipo_doc		Foreign Key	CHAR(2)
DOCTO-S-GESTION	tipo_tram		Foreign Key	CHAR(2)
DOCUMENTO	copia	Indica si se le a sacado fotocopia parcial o completa al documento a tramitar.	Owned Non-key	CHAR(2)
DOCUMENTO	desc-brev	Descripción breve del asunto del documento a tramitar.	Owned Non-key	CHAR(100)
DOCUMENTO	num-ent		Foreign Key	INTEGER
DOCUMENTO	num-refer	Número con que llega foliado el	Owned Non-key	CHAR(35)

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
		documento a ser tramitado.		
DOCUMENTO	tipo_doc	El documento se puede clasificar en: TRÁMITE CONTESTACIÓN DOCUMENTO IDENTIFICADO y COMBINACIONES ENTRE ELLOS.	Owned Key	CHAR(2)
ENTIDAD	cv-estado	Clave del estado de la república para su identificación.	Owned Key	INTEGER
ENTIDAD	desc-edo	Nombre de estado de la república.	Owned Non-key	CHAR(100)
ENTREGA	ab-cerr	Dato que indica si la recepción llega en SOBRE CERRADO.	Owned Non-key	CHAR(2)
ENTREGA	conf	Dato que indica si la recepción es CONFIDENCIAL.	Owned Non-key	CHAR(2)
ENTREGA	cv-area		Foreign Non-key	INTEGER
ENTREGA	cv-cargo		Foreign Non-key	INTEGER
ENTREGA	cv-proc		Foreign Non-key	INTEGER
ENTREGA	f_fact		Foreign Non-key	DATE
ENTREGA	num-ent	Número consecutivo asignado a cada recepción.	Owned Key	INTEGER
ENTREGA	piezas	Número de documentos recibidos en la entrega.	Owned Non-key	INTEGER
ENTREGA	urg-nor	Dato que indica si la recepción es URGENTE O NORMAL.	Owned Non-key	CHAR(2)
FACTURA	f_fact	Fecha en que es recibida la FACTURA.	Owned Key	DATE
FACTURA	fact-corr	Número con que viene foliada la factura de los documentos que llegan por correo.	Owned Non-key	CHAR(35)

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
FACTURA	fact-men	Número con que viene foliada la factura de los documentos que llegan por mensajería.	Owned Non-key	CHAR(35)
FACTURA	tot-corr	Total de documentos que integran la factura de correo.	Owned Non-key	INTEGER
FACTURA	tot-men	Total de documentos que integran la factura de mensajería.	Owned Non-key	INTEGER
FACTURA	tot-pal	Total de documentos que son enviados de palacio nacional.	Owned Non-key	INTEGER
FIRMA	cargo-firma	Cargo que desempeña.	Owned Non-key	CHAR(100)
FIRMA	cve-firma	Clave que identifica a cada persona autorizada.	Owned Key	INTEGER
FIRMA	encabezado	Encabezado que tendrá el oficio.	Owned Non-key	CHAR(100)
FIRMA	nom-firma	Nombre de la persona autorizada.	Owned Non-key	CHAR(100)
GESTION	cv-estado		Foreign Non-key	INTEGER
GESTION	cv-municip		Foreign Non-key	INTEGER
GESTION	num-ent		Foreign Key	INTEGER
GESTION	tipo_doc		Foreign Key	CHAR(2)
GESTION	tipo_tram		Foreign Key	CHAR(2)
INTERNO	cv-area		Foreign Non-key	INTEGER
INTERNO	cv-cargo		Foreign Non-key	INTEGER
INTERNO	cv-proc		Foreign Non-key	INTEGER
INTERNO	cve-firma		Foreign Non-key	INTEGER
INTERNO	f-salint	Fecha en que se emite el OFICIO INTERNO.	Owned Non-key	DATE
INTERNO	num-inter	Número consecutivo que se le asigna cada oficio que se genera como OFICIO INTERNO.	Owned Key	CHAR(2)
INTERNO	num-salint	Número del foliador para la salida	Owned Non-key	INTEGER

REPORTE DE ATRIBUTOS

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
		que se le asigna al OFICIO INTERNO.		
INTERNO	texto-int	Texto del OFICIO INTERNO.	Owned Non-key	LONG VARCHAR
INTERNO	tipo-salint	Tipo del foliador que se utiliza de acuerdo a la persona que firmo el OFICIO INTERNO.	Owned Non-key	CHAR(35)
INVOLUCRADO	cv-area		Foreign Non-key	INTEGER
INVOLUCRADO	cv-cargo		Foreign Non-key	INTEGER
INVOLUCRADO	cv-proc		Foreign Non-key	INTEGER
INVOLUCRADO	intere-interm	Indicar en función de que se envía la copia, puede ser como INTERESADO DIRECTO o como INTERMEDIARIO.	Owned Non-key	CHAR(20)
INVOLUCRADO	num-copint	Número consecutivo de copia marcada a los interesados, que se les envía copia de la contestación a su asunto.	Owned Key	INTEGER
INVOLUCRADO	num-ent		Foreign Key	INTEGER
INVOLUCRADO	tipo_doc		Foreign Key	CHAR(2)
MENSAJERO	cv-men	Clave del mensajero para su identificación.	Owned Key	INTEGER
MENSAJERO	materno-men	Apehido materno del mensajero.	Owned Non-key	CHAR(25)
MENSAJERO	nombre-men	Nombre del mensajero.	Owned Non-key	CHAR(25)
MENSAJERO	paterno-men	Apehido paterno del mensajero.	Owned Non-key	CHAR(25)
MUNICIPIO	cv-estado		Foreign Key	INTEGER
MUNICIPIO	cv-municip	Clave del municipio para su identificación.	Owned Key	INTEGER

REPORTE DE ATRIBUTOS

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
MUNICIPIO	desc-municip	Nombre del municipio.	Owned Non-key	CHAR(100)
NOMBRE	cv-nombre	Clave del nombre para su identificación.	Owned Key	INTEGER
NOMBRE	materno	Apehido materno.	Owned Non-key	CHAR(25)
NOMBRE	nombre	Nombre(s).	Owned Non-key	CHAR(25)
NOMBRE	paterno	Apehido paterno.	Owned Non-key	CHAR(25)
NOMBRE	titulo	Título con que cuenta la persona.	Owned Non-key	CHAR(15)
OFICIO	cv-area		Foreign Non-key	INTEGER
OFICIO	cv-cargo		Foreign Non-key	INTEGER
OFICIO	cv-proc		Foreign Non-key	INTEGER
OFICIO	cv-tema		Foreign Non-key	CHAR(4)
OFICIO	cve-firma		Foreign Non-key	INTEGER
OFICIO	f-salofic	Fecha en que se emite el OFICIO.	Owned Non-key	DATE
OFICIO	num-ent		Foreign Key	INTEGER
OFICIO	num-ofic	Número consecutivo que se asigna al documento gestionado por OFICIO para su identificación.	Owned Key	INTEGER
OFICIO	num-salofic	Número de foliador que se le asignara al OFICIO de acuerdo a la persona que lo firme.	Owned Non-key	INTEGER
OFICIO	texto-ofic	Texto del asunto de la gestión por OFICIO.	Owned Non-key	LONG VARCHAR
OFICIO	tipo-salida	Tipo de foliador que se utilizara para la salida del OFICIO.	Owned Non-key	CHAR(35)
OFICIO	tipo_doc		Foreign Key	CHAR(2)
OFICIO	tipo_tram		Foreign Key	CHAR(2)
ORIGEN	cv-area		Foreign Non-key	INTEGER
ORIGEN	cv-cargo		Foreign Non-key	INTEGER
ORIGEN	cv-proc		Foreign Non-key	INTEGER

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
ORIGEN	num-ent		Foreign Key	INTEGER
ORIGEN	num-orig	Número consecutivo que se asignará a cada una de las procedencias del documento a gestionar.	Owned Key	INTEGER
ORIGEN	tipo_doc		Foreign Key	CHAR(2)
ORIGEN	tipo_tram		Foreign Key	CHAR(2)
PAQ-GEST	cv-direc		Foreign Key	INTEGER
PAQ-GEST	f-dia		Foreign Key	DATE
PAQ-GEST	h-salida		Foreign Key	TIME
PAQ-GEST	num-ent		Foreign Key	INTEGER
PAQ-GEST	num-inter		Foreign Key	CHAR(2)
PAQ-GEST	num-ofic		Foreign Key	INTEGER
PAQ-GEST	num-turno		Foreign Key	INTEGER
PAQ-GEST	tipo_doc		Foreign Key	CHAR(2)
PAQ-GEST	tipo_tram		Foreign Key	CHAR(2)
PAQUETE	cv-direc		Foreign Key	INTEGER
PAQUETE	f-dia		Foreign Key	DATE
PAQUETE	f-entregapaq	Fecha en que es entregado el paquete.	Owned Non-key	DATE
PAQUETE	h-entregapaq	Hora en que se entrega el paquete.	Owned Non-key	TIME
PAQUETE	h-salida		Foreign Key	TIME
PAQUETE	obser-paq	Observaciones particulares sobre la entrega del paquete.	Owned Non-key	CHAR(100)
PERSONA	cv-area		Foreign Non-key	INTEGER
PERSONA	cv-cargo		Foreign Non-key	INTEGER
PERSONA	cv-proc		Foreign Non-key	INTEGER
PERSONA	cv-tipdes		Foreign Non-key	INTEGER
PERSONA	f-desig	Fecha de la designación.	Owned Non-key	DATE

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
PERSONA	nivel-desig	Nivel de la designación, lo cual implica una designación como PROPIETARIO o SUPLENTE.	Owned Non-key	CHAR(35)
PERSONA	num-comision	Número consecutivo que se asigna a las designaciones para su identificación.	Owned Key	INTEGER
PERSONA	num-ent		Foreign Key	INTEGER
PERSONA	tipo_doc		Foreign Key	CHAR(2)
PERSONA	tipo_tram		Foreign Key	CHAR(2)
PETICION	cv-estado		Foreign Non-key	INTEGER
PETICION	cv-municip		Foreign Non-key	INTEGER
PETICION	cv-nombre		Foreign Non-key	INTEGER
PETICION	cv-tema		Foreign Non-key	CHAR(4)
PETICION	desc-pet	Descripción de la PETICIÓN específica que se hace.	Owned Non-key	CHAR(100)
PETICION	num-ent		Foreign Key	INTEGER
PETICION	num-pet	Número consecutivo que se asigna a las peticiones como identificador único.	Owned Key	INTEGER
PETICION	ofic-refer	Número con que llega foliada la PETICION específica.	Owned Non-key	CHAR(35)
PETICION	tipo_doc		Foreign Key	CHAR(2)
PETICION	tipo_tram		Foreign Key	CHAR(2)
PROCEDENCIA	abrev-proc	Abreviatura del nombre de la dependencia, organismo u organización.	Owned Non-key	CHAR(20)
PROCEDENCIA	cv-proc	Clave de la procedencia.	Owned Key	INTEGER
PROCEDENCIA	desc-proc	Nombre de la dependencia, organismo u organización.	Owned Non-key	CHAR(50)

REPORTE DE ATRIBUTOS

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
SEG-GESTION	num-ent		Foreign Key	INTEGER
SEG-GESTION	num-inter		Foreign Non-key	CHAR(2)
SEG-GESTION	num-ofic		Foreign Key	INTEGER
SEG-GESTION	num-pet		Foreign Key	INTEGER
SEG-GESTION	num-seguim		Foreign Key	INTEGER
SEG-GESTION	num-turno		Foreign Key	INTEGER
SEG-GESTION	tipo_doc		Foreign Key	CHAR(2)
SEG-GESTION	tipo_tram		Foreign Key	CHAR(2)
SEGUIMIENTO	cv-avance		Foreign Non-key	CHAR(25)
SEGUIMIENTO	f-seguim	Fecha de captura del seguimiento.	Owned Non-key	DATE
SEGUIMIENTO	num-ent		Foreign Key	INTEGER
SEGUIMIENTO	num-seguim	Número de seguimiento que corresponde a cada petición que se identifica por número de salida.	Owned Key	INTEGER
SEGUIMIENTO	text-seguim	Texto de respuesta a la petición.	Owned Non-key	LONG VARCHAR
SEGUIMIENTO	tipo_doc		Foreign Key	CHAR(2)
SINTESIS	f-inicio	Si se trata de algún evento, se registra la fecha en que se llevará a cabo.	Owned Non-key	DATE
SINTESIS	f-rsint	Fecha en que se registra el documento como síntesis.	Owned Non-key	DATE
SINTESIS	f-termino	Fecha en que termina el evento.	Owned Non-key	DATE
SINTESIS	loc-sint	Indicar si el evento es LOCAL, DEL INTERIOR O EXTRANGERO.	Owned Non-key	CHAR(35)
SINTESIS	num-ent		Foreign Key	INTEGER
SINTESIS	texto-sint	Texto del asunto a tratar como síntesis.	Owned Non-key	LONG VARCHAR

REPORTE DE ATRIBUTOS

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
SINTESIS	tipo_doc		Foreign Key	CHAR(2)
SINTESIS	tipo_tram		Foreign Key	CHAR(2)
TIPO-CLASIFI	cv-clas	Clave de la clasificación para su identificación.	Owned Key	CHAR(35)
TIPO-CLASIFI	desc-clas	Descripción de la clasificación.	Owned Non-key	CHAR(100)
TIPO-CLASIFI	estancia	Periodo que se debe mantener el expediente.	Owned Non-key	INTEGER
TIPO-CLASIFI	mantiene	Año del mas antiguo expediente que se mantiene.	Owned Non-key	INTEGER
TIPO-CLASIFI	periodo	Periodo de tiempo relacionado con la permanencia física de los expedientes.	Owned Non-key	CHAR(20)
TIPO-CLASIFI	primero	Año en que se abrió el expediente.	Owned Non-key	DATE
TIPO-CLASIFI	tot-conc	Número de veces que se ha concentrado el expediente.	Owned Non-key	INTEGER
TIPO-CLASIFI	ultimo	Año del último expediente.	Owned Non-key	DATE
TIPO-DESIGNACION	cv-tipdes	Clave del tipo de designación para su identificación.	Owned Key	INTEGER
TIPO-DESIGNACION	desc-tipdes	Descripción del tipo de designación.	Owned Non-key	CHAR(100)
TIPO-ESTADO	cv-avance	Clave del nivel de avance para su identificación.	Owned Key	CHAR(25)
TIPO-ESTADO	desc-avance	Descripción del nivel de avance.	Owned Non-key	CHAR(100)
TIPO-TEMA	cv-area	Clave del área que identifica que temas corresponden a cada área de atención.	Owned Non-key	INTEGER
TIPO-TEMA	cv-proc	Clave de procedencia que identifica junto con la clave de	Owned Non-key	INTEGER

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
		área. los temas de clasificación de documentos que corresponden a cada área de atención.		
TIPO-TEMA	cv-tema	Clave de tema para su identificación.	Owned Key	CHAR(4)
TIPO-TEMA	desc-tema	Descripción del tema.	Owned Non-key	CHAR(100)
TIPO-TEMA	num-inter		Foreign Non-key	CHAR(2)
TRAMITE	num-ent		Foreign Key	INTEGER
TRAMITE	tipo_doc		Foreign Key	CHAR(2)
TRAMITE	tipo_tram	Cuando un documento se clasifica en TIPO DE DOCUMENTO como TRÁMITE. a su vez puede ser clasificado como TIPO DE TRÁMITE: CONVENIO, DESIGNACIÓN, SÍNTESIS. NINGUNO O ALGUNA COMBINACIÓN PERMITIDA.	Owned Key	CHAR(2)
TURNO	cv-area		Foreign Non-key	INTEGER
TURNO	cv-cargo		Foreign Non-key	INTEGER
TURNO	cv-proc		Foreign Non-key	INTEGER
TURNO	cv-tema		Foreign Non-key	CHAR(4)
TURNO	f-saltur	Fecha en que se emite el TURNO.	Owned Non-key	DATE
TURNO	num-ent		Foreign Key	INTEGER
TURNO	num-saltur	Número del foliador que se asigna al documento gestionado como TURNO para su salida.	Owned Non-key	INTEGER
TURNO	num-turno	Número consecutivo que se asigna a los TURNOS para su identificación.	Owned Key	INTEGER
TURNO	tex-int	Indicación específica de la	Owned Non-key	LONG

ENTIDAD A QUE PERTENECE	NOMBRE	DEFINICIÓN	TIPO DE ATRIBUTO	TIPO DE DATO
		instrucción de acuerdo al asunto que es tratado en el TURNO.		VARCHAR
TURNO	tex-turno	Texto del asunto que se gestionó como TURNO.	Owned Non-key	LONG VARCHAR
TURNO	tipo-int	Tipo de la instrucción del TURNO, la cual puede ser: ATENCIÓN, OPINIÓN O CONOCIMIENTO.	Owned Non-key	CHAR(30)
TURNO	tipo_doc		Foreign Key	CHAR(2)
TURNO	tipo_tram		Foreign Key	CHAR(2)

REPORTE DE RELACIONES

ENTIDAD PADRE	FRASE	ENTIDAD HIJO	TIPO DE RELACIÓN	CARDINALIDAD
AREA	es conformada	CARGO	Identifying	One-to-Zero-One-or-More
AREA	es la procedencia	CONVENIO	Non-identifying	Zero-or-One-to-Zero-One-or-More
AREA	es la procedencia	DESIGNACION	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	envía	CARPETA	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	envía	CONTESTACION	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	envía	ENTREGA	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	es comisionado para	PERSONA	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	es la procedencia de	ORIGEN	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	recibe	CARPETA	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	recibe	CONTESTACION	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	remite	DOCTO-IDENTIFI	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	remite	DOCTO-S-GESTION	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	se asigna como	INVOLUCRADO	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	se le envían	COPIA	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	se le envían	INTERNO	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	se le envían	OFICIO	Non-identifying	Zero-or-One-to-Zero-One-or-More

REPORTE DE RELACIONES

ENTIDAD PADRE	FRASE	ENTIDAD HIJO	TIPO DE RELACIÓN	CARDINALIDAD
CARGO	se le envian	TURNO	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARGO	se le marcan	COPIA2	Non-identifying	Zero-or-One-to-Zero-One-or-More
CARPETA	forma parte de	CONTESTACION	Non-identifying	Zero-or-One-to-Zero-One-or-More
CONTESTACION	marca	INVOLUCRADO	Identifying	One-to-Zero-One-or-More
CONTESTACION	responde	SEGUIMIENTO	Identifying	One-to-One-or-More (P)
CONVENIO	se decide gestionar	GESTION	Identifying	One-to-Exactly-1
DESIGNACION	asigna	PERSONA	Identifying	One-to-One-or-More (P)
DESIGNACION	se decide gestionar	GESTION	Identifying	One-to-Exactly-1
DIRECCION	le corresponde	CARGO	Non-identifying	Zero-or-One-to-One-or-More (P)
DIRECCION	ubica	PAQUETE	Identifying	One-to-One-or-More (P)
DISTRIBUCION	consta de	PAQUETE	Identifying	One-to-One-or-More (P)
DOCTO-IDENTIFI	se guarda	ARCHIVO3	Identifying	One-to-One-or-More (P)
DOCTO-S-GESTION	se guarda	ARCHIVO5	Identifying	One-to-One-or-More (P)
DOCUMENTO	is a	CONTESTACION	Subtype	Is a
DOCUMENTO	is a	DOCTO-IDENTIFI	Subtype	Is a
DOCUMENTO	is a	TRAMITE	subtype	Is a
ENTIDAD	es constituida	MUNICIPIO	Identifying	One-to-One-or-More (P)
ENTREGA	contiene	ANEXO	Identifying	One-to-Zero-One-or-More
ENTREGA	contiene	DEVOLUCION	Identifying	One-to-Zero-One-or-More
ENTREGA	contiene	DOCUMENTO	Identifying	One-to-One-or-More (P)
FACTURA	constituye	ENTREGA	Non-identifying	Zero-or-One-to-One-or-More (P)
FIRMA	firma	INTERNO	Non-identifying	Zero-or-One-to-Zero-One-or-More
FIRMA	se firmara por	OFICIO	Non-identifying	Zero-or-One-to-Zero-One-or-More
GESTION	esta formada	PETICION	Identifying	One-to-One-or-More (P)
GESTION	is a	OFICIO	Subtype	Is a
GESTION	is a	TURNO	Subtype	Is a

ENTIDAD PADRE	FRASE	ENTIDAD HIJO	TIPO DE RELACIÓN	CARDINALIDAD
GESTION	no se tramita	DOCTO-S-GESTION	Identifying	One-to-Zero-One-or-More
GESTION	procede	ORIGEN	Identifying	One-to-One-or-More (P)
INTERNO	corresponde	PAQ-GEST	Identifying	One-to-Zero-or-One (Z)
INTERNO	es el tema	TIPO-TEMA	Non-identifying	Zero-or-One-to-Zero-One-or-More
INTERNO	es guardado	ARCHIVO4	Identifying	One-to-One-or-More (P)
INTERNO	le corresponden	SEG-GESTION	Non-identifying	Zero-or-One-to-Zero-One-or-More
INTERNO	marca	COPIA2	Identifying	One-to-Zero-One-or-More
MENSAJERO	entrega	DISTRIBUCION	Non-identifying	Zero-or-One-to-One-or-More (P)
MUNICIPIO	es la localización	DOCTO-S-GESTION	Non-identifying	Zero-or-One-to-Zero-One-or-More
MUNICIPIO	es la localización	GESTION	Non-identifying	Zero-or-One-to-Zero-One-or-More
MUNICIPIO	es la localización	PETICION	Non-identifying	Zero-or-One-to-Zero-One-or-More
MUNICIPIO	es la localización	DOCTO-IDENTIFI	Non-identifying	Zero-or-One-to-Zero-One-or-More
NOMBRE	solicita	PETICION	Non-identifying	Zero-or-One-to-Zero-One-or-More
NOMBRE	tiene	CARGO	Non-identifying	Zero-or-One-to-Zero-or-One (Z)
OFICIO	corresponde	PAQ-GEST	Identifying	One-to-Zero-or-One (Z)
OFICIO	es guardado	ARCHIVO2	Identifying	One-to-One-or-More (P)
OFICIO	le corresponden	SEG-GESTION	Identifying	One-to-Zero-One-or-More
OFICIO	marca	COPIA	Identifying	One-to-Zero-One-or-More
PAQUETE	incluye	PAQ-GEST	Identifying	One-to-One-or-More (P)
PETICION	le corresponden	SEG-GESTION	Identifying	One-to-Zero-One-or-More
PROCEDENCIA	incluye	AREA	Identifying	One-to-Zero-One-or-More
SEGUIMIENTO	corresponde	SEG-GESTION	Identifying	One-to-Zero-or-One (Z)
SINTESIS	se decide gestionar	GESTION	Identifying	One-to-Exactly-1
TIPO-CLASIFI	es el expediente	ARCHIVO1	Non-identifying	Zero-or-One-to-Zero-One-or-

ENTIDAD PADRE	FRASE	ENTIDAD HIJO	TIPO DE RELACIÓN	CARDINALIDAD
TIPO-CLASIFI	es el expediente	ARCHIVO2	Non-identifying	Zero-or-One-to-Zero-One-or-More
TIPO-CLASIFI	es el expediente	ARCHIVO3	Non-identifying	Zero-or-One-to-Zero-One-or-More
TIPO-CLASIFI	es el expediente	ARCHIVO4	Non-identifying	Zero-or-One-to-Zero-One-or-More
TIPO-CLASIFI	es el expediente	CARPETA	Non-identifying	Zero-or-One-to-Zero-One-or-More
TIPO-DESIGNACION	cataloga	PERSONA	Non-identifying	Zero-or-One-to-Zero-One-or-More
TIPO-ESTADO	es la situación	SEGUIMIENTO	Non-identifying	Zero-or-One-to-Zero-One-More
TIPO-TEMA	clasifica el asunto	DOCTO-S-GESTION	Non-identifying	Zero-or-One-to-Zero-One-or-More
TIPO-TEMA	es el tema de	PETICION	Non-identifying	Zero-or-One-to-Zero-One-or-More
TIPO-TEMA	es el tema de	TURNO	Non-identifying	Zero-or-One-to-Zero-One-or-More
TIPO-TEMA	es el tema de	OFICIO	Non-identifying	Zero-or-One-to-Zero-One-or-More
TRAMITE	se clasifica	CONVENIO	Identifying	One-to-Zero-or-One (Z)
TRAMITE	se clasifica	DESIGNACION	Identifying	One-to-Zero-or-One (Z)
TRAMITE	se clasifica	SINTESIS	Identifying	One-to-Zero-or-One (Z)
TRAMITE	se decide gestionar	GESTION	Identifying	One-to-Exactly-1
TURNO	corresponde	PAQ-GEST	Identifying	One-to-Zero-or-One (Z)
TURNO	es guardado	ARCHIVO1	Identifying	One-to-One-or-More (P)
TURNO	le corresponden	SEG-GESTION	Identifying	One-to-Zero-One-or-More

Como el objetivo de este trabajo no es la implantación del sistema, el segundo y tercer paso de esta etapa en el Desarrollo de Sistemas no serán abarcados completamente. Los cuales consistirían en la construcción física de la estructura de la base de datos, la construcción de la aplicación, pruebas de la aplicación y la construcción y prueba de las funciones de conversión. Así como los pasos necesarios para la instalación de la aplicación, inicio de la conversión, desactivación de la aplicación anterior y el inicio de la operación del nuevo sistema. Pero también está implicado el mejoramiento continuo con el monitoreo rutinario y la optimización de la eficiencia de la aplicación, y la rápida adecuación cuando se presenten problemas, el soporte de la reingeniería de procesos para una mejora continua, lo cual requerirá regresar a pasos anteriores y se reinicia parte del ciclo, puesto que las tres etapas están íntimamente relacionadas.

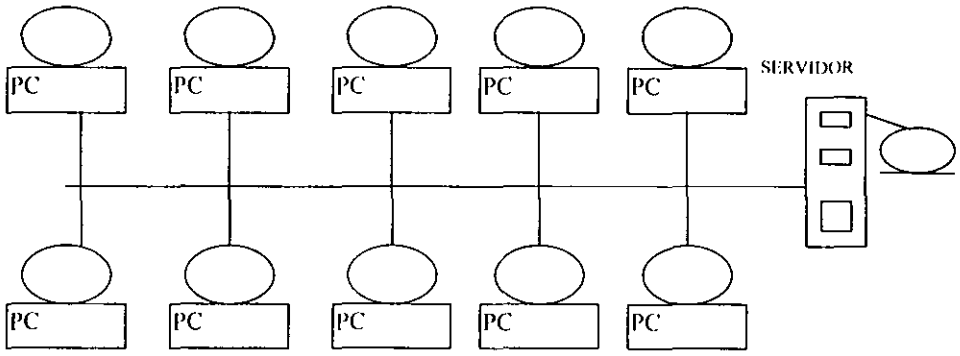
ARQUITECTURA

DEFINICIÓN DE LA SOLUCIÓN

Esta etapa inicia en paralelo con la etapa de Desarrollo de Sistemas, un poco después de haber iniciado la etapa de Reingeniería de Procesos. Todas las etapas se retroalimentan de forma continua.

REVISIÓN DE LA TECNOLOGÍA EN USO Y DETERMINACIÓN DE LA TECNOLOGÍA APROPIADA

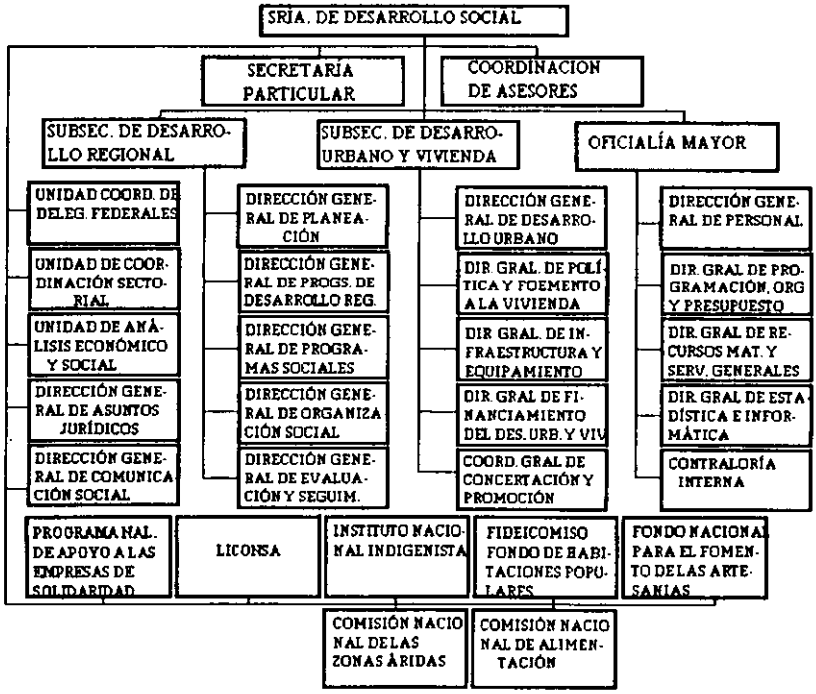
Actualmente se tiene una red local Novell Netware, que se ubica en la Secretaría Técnica (S.T.). Dicha LAN cuenta con un servidor con procesador 486 a 60 MHZ, capacidad de 16 MB en RAM y 500 MB en disco duro, y la posibilidad de diez conexiones al servidor; así como su conexión a la red amplia de SEDESOL. Toda la información de los asuntos que son gestionados a las áreas internas son registrados en este sistema local. Y todos los seguimientos que envían las áreas internas, ya sea en papel o en algún medio magnético, son recapturados en el sistema actual. Lo anterior implica una duplicación de esfuerzos. Además de que puede haber pérdida u omisión de información.



Plataforma Actual. Red Novell Netware con 10 conexiones.

La aplicación actual esta programada en Clipper 5.1 para red. Dicha aplicación cuenta con dos módulos principales: Turnos y Síntesis; que se trataron anteriormente.

Como se vio en la sección II.3 referente a Sistemas Cliente/Servidor, en el que se plantean ciertos aspectos que son recomendados por los expertos para definir si es necesario un sistema c/s. Una de las interrogantes es ¿CUANTOS USUARIOS INTERVIENEN?, la respuesta en este caso en particular es que toda la organización está involucrada en este proceso. El organigrama más detallado de la secretaría se muestra en el siguiente diagrama. La mayoría de los usuarios además de consultar la información tendrán la facultad de agregar y modificar la información de la base de datos en lo que se refiere al seguimiento.



Estructura Orgánica de la Secretaría de Desarrollo Social.

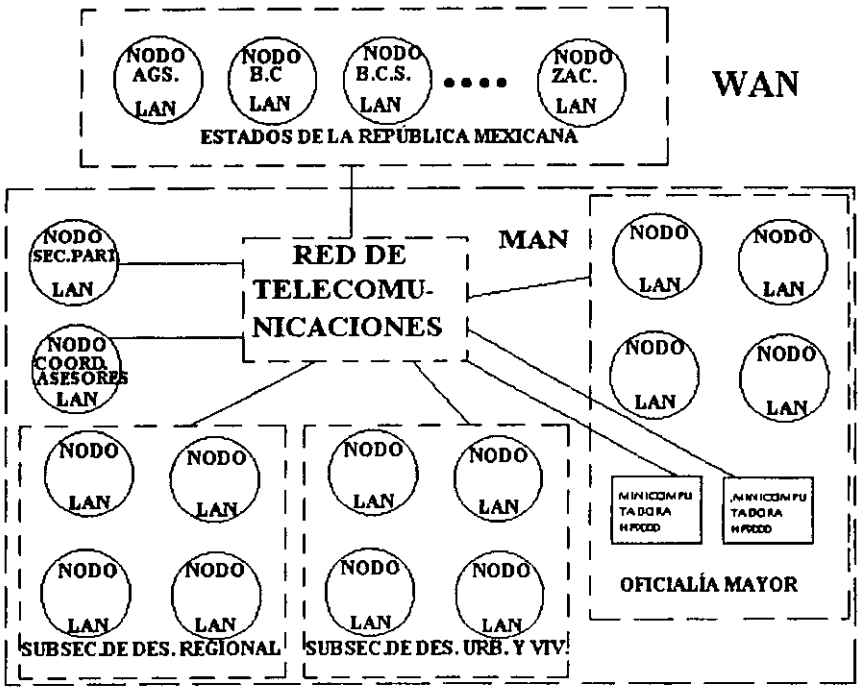
Un punto de mucho peso que impulsó a que se diera el cambio es que la institución cuenta recientemente con la infraestructura de una Red de Área Amplia (WAN) a nivel nacional. La cual enlaza las Delegaciones Estatales con las Oficinas Centrales en el Distrito Federal. Se le ha dado uso a la WAN a través del Coordinador y el acceso a la Internet, pero se está iniciando el esfuerzo para impulsar un verdadero aprovechamiento de las telecomunicaciones existentes a nivel nacional para crear sistemas que sean verdaderamente institucionales.

Observando el diagrama de la estructura orgánica, se puede dar una idea general de cómo están entrelazados los nodos en la red amplia. A continuación se enlista la estructura de comunicaciones con que cuenta SEDESOL:

- ***41 Redes Locales Metropolitanas Novell***
- ***34 Redes locales Estatales Novell***
- ***1 Nodo Central de Telecomunicaciones manejando cuatro enlaces de fibra óptica, un canal UHF, cinco enlaces de microondas, un canal E1 para la comunicación de voz y datos a las Delegaciones Estatales. Protocolos de comunicación PPP, Frame Relay.***
- ***2 Equipos Centrales HP9000 UNIX serie 827, con 256Mb RAM y 10 Gb en disco duro, para cada uno.***

La inclusión de todas las áreas faltantes se hará de forma gradual hasta que halla una cobertura completa. La responsabilidad de control y administración de toda la red de telecomunicaciones recae en OFICIALÍA MAYOR, específicamente en la Dirección General de Estadística e Informática.

Volviendo al diagrama de la estructura orgánica y al cuadro de distribución de las LAN, podemos observar que las entidades coordinadas sectorialmente no están incluidas en la WAN, ya que por ser órganos que aunque dependen de cierta forma de SEDESOL tienen una organización muy independiente de su órgano rector. Por lo que es difícil su inclusión en este plan corporativo, al menos en cuanto a la utilización de un sistema. El siguiente diagrama esquematiza lo expuesto anteriormente.



Red de Área Amplia de la Secretaría de Desarrollo Social (SEDESOL)

Todas las redes LAN están bajo una misma plataforma, ya que son redes Novell Netware con características básicas como las descritas para la red local de la Secretaría Técnica o superiores (que está incorporada a la Secretaría Particular de acuerdo al diagrama organizacional). En cuanto a las terminales de cada LAN, no todas cuentan con la capacidad mínima para poder trabajar con un sistema como el que se está diseñando, pero ya existe un plan global de reemplazo para las PC's obsoletas, por lo que en un principio no todas las terminales de los nodos podrán tener acceso al sistema implantado, pero se tiene contemplado que sea de forma gradual.

En cuanto a las Minicomputadoras HP 9000 es necesaria su existencia considerando que la base de datos de este trabajo no es la única que utiliza los recursos de la WAN, sino que existen otras bases de datos que también son prioritarias. Por lo que es requerida una gran capacidad de procesamiento y almacenamiento.

Retomando la figura II.3.b "Guía para Base de Datos Cliente/Servidor" del capítulo II, hay que tratar de ubicar la categoría en que caería el sistema que se trata en este trabajo. Siguiendo el diagrama de flujo:

⇒ Se necesitan usuarios múltiples

⇒ Se necesitan más de 20 usuarios simultáneos
⇒ Se necesitan transacciones múltiples con modificaciones de datos

Por lo tanto, es necesario un sistema CLIENTE/SERVIDOR que nos proporcione gran integridad en los datos, aunque traerá consigo un incremento en la complejidad. Que además:

⇒ No existe un DBMS multiusuario basado en PC
⇒ Existe una minicomputadora
⇒ No se realizará el downsize
⇒ Los distribuidores del DBMS tienen opciones c/s
⇒ Si trabaja con protocolos comunes para PC

Por lo anterior, obtendremos un sistema cliente/servidor con compuertas o gateway, entonces la minicomputadora podrá funcionar como servidor.

SELECCIÓN DE LAS HERRAMIENTAS DE DESARROLLO

Como ya se vio en el capítulo II, fueron listadas algunas de las herramientas disponibles para las diferentes plataformas. La plataforma en que va a funcionar este sistema está basado en microcomputadoras, pero por la cantidad de información que se maneja se verá apoyada con una minicomputadora.

Este sistema no es el único que se está desarrollando en la institución bajo ambiente cliente/servidor, ya existen herramientas de Desarrollo de Aplicaciones y el DBMS que están siendo utilizadas para tal objetivo. Por lo que se usarán estas mismas herramientas, lo cual permitirá estandarizar el desarrollo de aplicaciones en la institución. La capacidad informática con que cuenta la secretaría esta constituida de los siguientes elementos:

BASES DE DATOS RELACIONALES

- ***Manejador de Base de Datos Relacional Sybase SQL System 11 para equipos bajo el Sistema Operativo UNIX.***
- ***Manejador de Base de Datos Relacional Sybase SQL Anywhere para equipos bajo el Sistema Operativo Novell.***

HERRAMIENTAS PARA LA EXPLOTACIÓN DE LA BASE DE DATOS

- ***InfoMaker***

HERRAMIENTAS PARA EL DESARROLLO DE SISTEMAS

- ***Power Builder Enterprice*** — ***Desarrollo de Aplicaciones***
- ***Erwin*** — ***Modelado de Bases de Datos***

Tecnología de bases de datos

Para el manejo de grandes volúmenes de información que puedan combinar y relacionar diversos aspectos como datos nominales, estadísticos y geográficos, se cuenta con el Manejador de Bases de Datos Relacionales *Sybase SQL Server System 11* para plataformas UNIX, y *Sybase SQL Anywhere* para plataformas Novell y/o Windows NT.

Las principales características de estos productos se muestran en la siguiente tabla:

Relacional	Capacidad de establecer diferentes tipos de relaciones entre las tablas de una base de datos a través de sus atributos, con lo que se garantiza la integridad de la información.
Esquema Cliente/Servidor	Capacidad para que los procesos de aplicación o manejo de datos se realicen en el equipo en que está instalado el manejador de la base de datos, con lo que se liberan de estas tareas los equipos de los usuarios y permite un mejor desempeño de los sistemas.
Seguridad	Capacidad de establecer diferentes esquemas de seguridad a nivel de la base de datos y la información, a través de accesos y permisos a los usuarios.
Replicación de Datos	Capacidad de transferir y actualizar diferentes bases de datos distribuidas en distintos sitios geográficos, a través de una red, con mecanismos rigurosos de seguridad mediante comunicación encriptada de datos.
Multiusuarios	Capacidad para que varios usuarios dentro de una red, trabajen concurrentemente con la misma base de datos.
Procedimientos Almacenados	Capacidad de crear rutinas dentro de la base de datos para realizar diversos procesos como: validación de las normas operativas del Programa (validación de las reglas del negocio) y vistas parciales de la información, entre otras aplicaciones.
Portabilidad de la información	Capacidad para exportar e importar datos en diferentes formatos ampliamente utilizados y disponibles en el mercado.

Modelado de bases de datos

Se cuenta con ERWIN como herramienta para construir modelos de bases de datos a partir de un modelo de procesos. Esta herramienta permite diseñar el modelo lógico de una base de datos, lo que comprende la identificación de los atributos, de sus características y de las relaciones que guarda con los demás elementos de la base de datos. Una vez terminado el modelo lógico, se establece una conexión física a la base de datos (en este caso SYBASE) para que se genere la estructura física (Modelo Físico) y definitiva de la misma.

Programación de las aplicaciones

Se cuenta con Power Builder como herramienta de programación de sistemas, dicha herramienta está diseñada para generar sistemas y aplicaciones en ambientes gráficos, tales como Windows. Dentro de sus características principales destacan:

Desarrollo Rápido de Aplicaciones (RAD)	Capacidad de construir programas con gran rapidez y versatilidad, lo que garantiza total control sobre los productos creados.
Orientado a Objetos	Capacidad para construir diferentes tipos de objetos con características y funciones propias. Dentro del manejo de objetos, Power Builder permite la utilización de las siguientes propiedades: ↦ Clases ↦ Herencias ↦ Polimorfismo ↦ Encapsulado A su vez, a través de métodos y/o funciones, se manejan y controlan cada uno de los objetos.

DESARROLLO DE LA SOLUCIÓN

Esta etapa de la ARQUITECTURA consiste en el diseño y especificación física de la *aplicación* y la *base de datos;* el diseño físico, construcción y prueba de todas las interfaces con los sistemas externos; la instalación, integración y examen de alguna nueva estación de trabajo, servidor o tecnología de red que esté siendo utilizada; y el desarrollo y documentación de un plan completo de conversión. Pero como el alcance de este trabajo se limita al análisis y diseño solo se desarrollarán los siguientes puntos: DISEÑO DE LA DISTRIBUCIÓN DE LA BASE DE DATOS y DISEÑO DE LA DISTRIBUCIÓN DE LOS COMPONENTES DE LA APLICACIÓN.

Por lo tanto tampoco se abarcará la IMPLEMENTACIÓN DE LA SOLUCIÓN para la arquitectura, la cual consiste en el monitoreo eficiente bajo las actuales condiciones de operación de la red y un monitoreo eficiente para la recuperación de datos.

DISTRIBUCIÓN DE LA BASE DE DATOS

Con base en el diagrama de la Red de Área Amplia, la forma en que están distribuidos los nodos y las funciones que tiene cada una de las áreas. Puntos que ya se han cubierto en secciones anteriores, puede tomarse como base para la distribución de los datos, teniendo siempre presente que la base de datos se distribuye única y exclusivamente cuando es necesario. De forma general podría considerarse que la BASE DE DATOS será UBICADA, PARTICIONADA y/o REPLICADA de la siguiente forma:

MODULO	TABLA	REPLICADO		PARTICIONADO		UBICACIÓN	
		NODO DE LA S.T.	NODOS DE LAS ÁREAS INTERNAS	NODOS DE LAS ÁREAS INTERNAS	MINI	NODO	MINI
Recepción de documentos	ENTREGA	■					■
	FACTURA	■					■
	ANEXO	■					■
	DEVOLUCIÓN	■					■
	DOCUMENTO	■					■
	DOCTO-IDENTIFI	■					■
	ARCHIVO3	■					■
Trámite	TRÁMITE	■					■
	CONVENIO	■					■
	DESIGNACIÓN	■					■
	TIPO-DESIGNACIÓN	■					■
	PERSONA	■					■
	SÍNTESIS	■					■
	ARCHIVO5	■					■
	DOCTO-S-GESTIÓN	■					■
	GESTIÓN	■		■			■
	PETICIÓN	■		■			■
	ORIGEN	■		■			■
	OFICIO	■		■			■
	TURNO	■		■			■
	COPIA	■		■			■
	ARCHIVO1	■		■			■
	ARCHIVO2	■		■			■
	INTERNO	■		■			■
	COPIA2	■		■			■
	ARCHIVO4	■		■			■
Seguimiento	CONTESTACIÓN			■			■
	INVOLUCRADO			■			■
	CARPETA			■			■
	SEGUIMIENTO			■			■
	SEG-GESTIÓN			■			■
Entrega	DISTRIBUCIÓN	■					■
	PAQUETE	■					■
	MENSAJERO	■					■
	PAQ-GEST	■					■
Catálogos	PROCEDENCIA	■	■				■
	ÁREA	■	■				■
	CARGO	■	■				■
	DIRECCIÓN	■	■				■
	NOMBRE	■	■				■
	ENTIDAD	■	■				■
	MUNICIPIO	■	■				■
	TIPO-CLASIFI	■	■				■
	TIPO-TEMA	■	■				■
	FIRMA	■	■				■
	TIPO-ESTADO	■	■				■

Analizando la tabla de Distribución de Datos tenemos tres columnas que nos indican DÓNDE los datos serán UBICADOS, REPLICADOS o PARTICIONADOS, pudiéndose dar alguna combinación entre ellas, siempre y cuando no sean excluyentes. Además, otra

columna que indica a qué tabla nos estamos refiriendo y en la parte más a la izquierda tenemos a qué módulo pertenece cada tabla.

Como punto de partida, podemos observar en la tabla de distribución que la base de datos se ubicará en la MINICOMPUTADORA, considerando factores como la capacidad de procesamiento para administrar la base de datos. Pero como el objetivo no es crear un sistema centralizado, en el cual se adolece del embotellamiento debido a que un gran número de solicitudes de acceso al servidor pueden requerirse de forma simultánea, presentandose una sobrecarga de procesamiento. Por lo tanto, la carga de procesamiento se dividirá entre los clientes y los servidores, y se habla de servidores porque la base de datos será PARTICIONADA y REPLICADA. Esto implica, que parte de la base de datos residirá también en otros nodos diferentes al servidor principal identificado en la MINI.

La base de datos será REPLICADA parcialmente en el nodo de la Secretaría Técnica (S.T.), con el objetivo de evitar un tráfico excesivo hacia la MINI. Tomando en cuanta que el sistema tiene un alto nivel de utilización en esta área, por la centralización de ciertas tareas de control, además de realizar la tarea operativa de las áreas internas que no tienen acceso al sistema por no contar con la infraestructura adecuada. El modulo de seguimiento no será replicado en la S.T. debido a que no se tendría la capacidad de almacenamiento , además hay que considerar que el sistema sea multiejercicio.

La REPLICACIÓN de todos los catálogos de la Aplicación en los nodos de las áreas internas, se visualiza como una decisión acertada considerando el alto nivel de acceso a los catálogos, que es una característica inherente a este tipo de tablas.

La PARTICIÓN de la base de datos se hará en los nodos de las áreas internas, colocando en los nodos sólo la información que involucra a cada área. Por lo tanto, se implementará una PARTICIÓN HORIZONTAL de las tablas que corresponden a los documentos gestionados a cada área y al seguimiento de tales asuntos. Todas las tablas que corresponden al subsistema "Seguimiento" serán PARTICIONADAS en los NODOS de las áreas internas. Este es el subsistema más importante, ya que es precisamente el POR QUÉ de la necesidad de un sistema c/s. Al igual que parte del subsistema de "Entrega", que es precisamente la información que corresponde a la gestión de los documento a cada área en particular.

COMPONENTES DE LA APLICACIÓN

Para comenzar con este aspecto tan importante del diseño del sistema, es necesario retomar lo visto en la sección *II.2.3 Estructura del procesamiento cooperativo*, del capítulo II. En donde se analizaron los componentes fundamentales de una aplicación, que son ilustrados en la figura *II.6 Componentes típicos de una aplicación*, del mismo capítulo. Entonces, como se vieron los componentes son los siguientes:

⇒ PROCESAMIENTO LÓGICO DE LA PRESENTACIÓN

⇒ PROCESAMIENTO LÓGICO DE NEGOCIOS

⇒ FUNCIONES DE ADMINISTRACIÓN DE DATOS: PROCESAMIENTO LÓGICO DE DATOS Y PROCESAMIENTO DE LA BASE DE DATOS

La división entre cada uno de estos elementos puede llegar a no ser clara, pero es necesario delimitar las fronteras cuando ya se tiene un problema particular que resolver. Haciendo referencia a los diferentes estilos de procesamiento cooperativo considerados como estilos elementales, es necesario ubicar en cuál de ellos ajustaremos el "Sistema de Control de Gestión". A continuación, podemos ver la tabla de componentes de la aplicación para cada estilo elemental del procesamiento cooperativo ilustrado en el capítulo II.

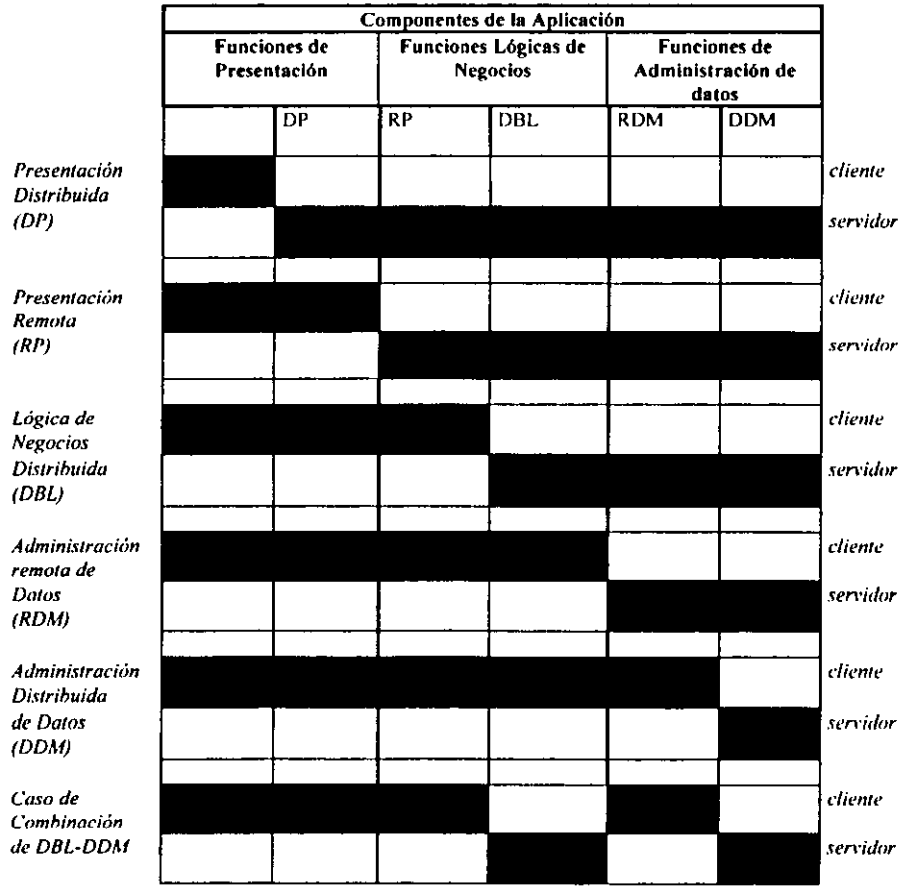


Figura II.9 Componentes de la Aplicación.

- Presentación Distribuida (Distributed Presentation - DP)

- Presentación remota (Remote Presentation - RP)
- Negocios Lógicos Distribuidos (distributed Business Logic - DBL)
- Administración de Datos Distribuido (Distributed Data Management -DDM)
- Administración de Datos Remota (Remote Data Management - RDM)

De acuerdo al *MODELO DE DATOS* diseñado, las *NECESIDADES DE FUNCIONALIDAD* a ser proporcionadas, la *PLATAFORMA* especificada, y la *DISTRIBUCIÓN DE DATOS* propuesta. Podemos clasificar nuestra aplicación con base a la figura II.9, como un caso de **COMBINACIÓN DE NEGOCIOS LÓGICOS DISTRIBUIDOS (DBL)-ADMINISTRACIÓN DE DATOS DISTRIBUIDOS (DDM).**

Algo muy determinante para ubicar la aplicación en este estilo de procesamiento, radica en que la base de datos será distribuida en múltiples nodos. Lo que da como consecuencia que las funciones de administración de datos también serán distribuidas, por lo que cada nodo que contenga datos, también contendrá la porción del PROCESAMIENTO LÓGICO DE DATOS y PROCESAMIENTO DE LA BASE DE DATOS. Y además, cada nodo contendrá la PRESENTACIÓN, por lo tanto no se presenta el caso de PRESENTACIÓN REMOTA, ya que parte de las funciones de administración de datos y parte del procesamiento de negocios estarán también en el nodo cliente. Considerando que el PROCESAMIENTO DE NEGOCIOS también será distribuido implica que será necesario un SISTEMA DE PROCESAMIENTO DE TRANSACCIONES EN LÍNEA para controlar todos los procesos distribuidos y asegurar la integridad de la base de datos, realizando el control sistemático de todas las transacciones. Lo cual implica que cualquier tipo de operación sobre la base de datos será totalmente realizada o abortada.

Con la ubicación del "SISTEMA DE CONTROL DE GESTIÓN" dentro de un estilo básico de procesamiento. Podemos analizar que se están implicando los datos, el control, las funciones y el procesamiento. El procesamiento es cooperativo lo cual es lo que hace posible que sea un sistema cliente/servidor, y además que al distribuir los datos se está distribuyendo el control ya que se permite tener cierta independencia en el nodo que contiene parte de la base de datos. Y las funciones del sistema que son el por qué de la existencia de una aplicación, tiene un papel fundamental, que en nuestro caso estarán ubicadas en el DBMS y como parte de la PRESENTACIÓN.

REINGENIERÍA DE PROCESOS

Lo último que se realizó de la reingeniería de procesos fue la *determinación de la solución óptima* en la etapa de *DESARROLLO DE LA SOLUCIÓN*, pero este aspecto de la metodología también implica: la preparación de la documentación, programas de entrenamiento para la implementación y determinación de las métricas que serán usadas para juzgar la efectividad de la solución óptima después de su implantación.

En lo que se refiere al entrenamiento es necesario capacitar a las personas que participarán en las diferentes etapas del sistema. Lo que implicaría un calendario de cursos de capacitación en las herramientas de diseño y desarrollo, sabiendo de antemano cual es el

presupuesto disponible para ello y las personas que se requieren. En cuanto a las métricas para determinar la efectividad están dadas principalmente por la disponibilidad de la información, considerando que el "Sistema de Control de Gestión" es un sistema integral que no tiene redundancia de información y permite obtener los mismos datos en cualquier nodo del sistema, en comparación con el sistema anterior se debe evaluar el tiempo de respuesta entre otros aspectos.

En la etapa de IMPLEMENTACIÓN DE LA SOLUCIÓN se realiza el entrenamiento que sea necesario, la validación de los datos en la conversión y la implementación de procesos y cambios en el flujo de trabajo.

Como ya se ha mencionado anteriormente, mientras se realizan estas actividades las otras etapas de la metodología ya han iniciado y se van realizando todas las actividades necesarias de forma paralela. Además hay que considerar que las tres etapas se van retroalimentando con los resultados obtenidos, y de está forma se van encontrando errores e inconsistencias que se van corrigiendo de forma iterativa dentro del mismo ciclo en que se van efectuando las diferentes etapas de la metodología. Hasta que es encontrada la solución óptima, la cual es desarrollada e implantada. Pero aún después de que es implantado el sistema, a través del ciclo de vida que tenga debe de haber un mejoramiento continuo, ya que es el periodo más importante dentro del ciclo de vida del sistema. Esto implica que después de que el sistema está en producción y en uso continuo, debe ser sucesivamente refinado y extendido para una mejor funcionalidad. Lo que está implicando regresar a ciertos puntos de la metodología y efectuar el ciclo.

CAPÍTULO V

Conclusiones

V CONCLUSIONES

En el desarrollo de este trabajo se han tratado diversos aspectos que de forma gradual han ido dando forma al objetivo. Se estructuraron los elementos necesarios para poder determinar la conveniencia de migrar o implantar un sistema cliente/servidor, lo cual se ha dado fundamentalmente respondiendo a las siguientes interrogantes:

⇒*¿Qué es un sistema Cliente/Servidor?*

⇒*¿Cuáles son los elementos que conforman un sistema Cliente/Servidor?*

⇒*¿Cuándo es necesario un sistema Cliente/Servidor?*

⇒*¿ Cuál es la tecnología disponible para este tipo de sistemas?*

⇒*¿Existe algún tipo de metodología para el desarrollo de sistemas Cliente/Servidor?*

Los anteriores son algunos de los puntos más relevantes que se trataron para lograr obtener el diseño del "Sistema de Control de Gestión". Comenzando con la definición de lo que es un sistema Cliente/Servidor quedo entendido que al tomar este concepto nos referimos a aquel sistema en que el procesamiento se divide entre dos elementos, para esto cada elemento se especializa en las funciones a realizar. De aquí se desprende el por qué es llamado cliente/servidor. El cliente es el que inicia la sesión haciendo requerimientos al servidor. Y es en este momento cuando el servidor inicia sus funciones de responder a los requerimientos. Por lo tanto, un sistema c/s es aquel que divide el procesamiento entre el cliente y el servidor, y por lo consiguiente se está bajo un ambiente distribuido.

El modelo cliente/servidor es consecuencia de la evolución del procesamiento utilizado en los sistemas de cómputo, así como del desarrollo y el avance de la tecnología. Lo cual ha permitido dar forma al modelo cliente/servidor.

En relación a la tecnología es necesario tener claro algunos conceptos básicos que dan la pauta para el entendimiento de este tipo de sistemas en lo relacionado al diseño de la arquitectura en que van a ser implantados. Que de forma general abarcan los conceptos de redes en relación a su tamaño (LAN, MAN, WAN), topología (bus, anillo, estrella, malla), medios físicos de transmisión, métodos de codificación, transmisión y control de acceso, y la arquitectura básica de los protocolos de comunicación.

De forma general, los sistemas cliente/servidor son divididos en dos grandes clasificaciones: Sistema Operacional y Sistema de Soporte de Decisiones. La importancia de esta diferenciación radica en que dependiendo del tipo de sistema variará la metodología de diseño. Considerando que la metodología de desarrollo de sistemas es un punto de fundamental importancia para la obtención de los resultados deseados. Recordando lo que dicen algunos expertos en la materia "no importa qué tan buenas o malas son las herramientas de desarrollo, cualquier tipo de dificultad al respecto es minimizada con un buen diseño". Por lo tanto,

aunque se tuvieran las mejores herramientas de desarrollo y no se realiza de la mejor forma posible el diseño o simplemente no existe ningún tipo de diseño previo al desarrollo, lo más seguro es que los resultados que se obtendrán no van a satisfacer los requerimientos del problema.

Como se especificó en su momento el sistema diseñado está definido como un SISTEMA OPERACIONAL, pues cumple con ser un sistema que irá generando los datos en el momento de usarlo, se mantendrá en uso constante y se manipularán los datos registro por registro para modificar o consultar.

Conceptualmente existen dos elementos que dan forma al modelo cliente/servidor, dichos elementos son: LAS BASES DE DATOS DISTRIBUIDAS y EL PROCESAMIENTO COOPERATIVO. La relación entre estos dos elementos es lo que da fuerza y ventaja sobre otros tipos de procesamiento.

Cuando una organización es inherentemente distribuida los elementos que permiten sumergir a la organización dentro de este tipo de sistemas, van surgiendo internamente debido a que es necesario cumplir los requerimientos para lograr los objetivos. Por lo tanto, si la compartición de la información a lo largo de toda la organización está condicionando el diseñar una base de datos distribuida, se tiene como consecuencia la probabilidad de tener un procesamiento cooperativo. Lo que lleva a plantear en primera instancia la posibilidad de implantar un sistema cliente/servidor. En el caso de estudio de este trabajo se tiene que es necesario distribuir la información concerniente a cada área, lo cual es la materia prima para que la organización cumpla ciertas funciones. Y que después de que se analiza y se gestiona, es necesaria una retroalimentación entre las diferentes áreas de la organización. Lo que implica un procesamiento cooperativo, que permite tener la misma información en cualquier sitio, y por lo tanto se elimina la posibilidad de tener redundancia innecesaria y se garantiza la integridad de los datos en cualquier punto, además de la accesibilidad inmediata de cualquier asunto independientemente del área a que pertenezca. Ya que por definición, cuando se habla de una base de datos distribuida tenemos que los elementos de datos están bajo un control múltiple y por lo tanto obtendremos los beneficios de la distribución de la información al obtener transparencia, sincronización, distribución basada en replicación o partición, integridad en los datos y cierto grado de independencia local en los nodos donde haya una porción de la base de datos.

De forma general se pueden visualizar las ventajas y desventajas que se adquieren al implantar una base de datos distribuida. Las ventajas más relevantes son las siguientes:

- Autonomía local
- Mejoramiento de la eficiencia (en el acceso a los datos)
- Fiabilidad/disponibilidad

CONCLUSIONES

- La economía relacionada con que gran parte del procesamiento se realiza de forma local y que en general se utilizan computadoras pequeñas y de bajo costo
- Expandibilidad
- Compartición

Y las desventajas adquiridas son:

- Por ser una tecnología relativamente nueva no existe gran experiencia al respecto
- Se agrega complejidad
- Software y hardware adicional en lo relacionado al sistema de comunicación
- Incremento del personal para dar mantenimiento al equipo en cada nodo
- La distribución del control también puede verse como una desventaja ya que es más difícil tener un punto central donde se puedan controlar ciertos aspectos
- La seguridad del sistema se vuelve extremadamente complicado debido a que se deberá tener un control de acceso estricto
- Cuando la organización tiene un sistema centralizado y quiere migrar a un sistema distribuido es bastante complejo este cambio ya que no existe una metodología para realizar esta transición

En el diseño de una base de datos distribuida se entrelazan diferentes aspectos, tales como:

- La administración del directorio
- El procesamiento de *queries*
- La confiabilidad
- El control de concurrencia
- La administración de puntos muertos

La solución de cada uno de estos aspectos de alguna forma determina o condiciona la solución de otro aspecto. Ya que con este conjunto de elementos se obtiene el diseño más óptimo de la base de datos.

Cuando se diseñó la forma en que estarían distribuidos los datos del sistema de control de gestión a lo largo de la red de la secretaría, se determinó la partición y replicación que era

necesaria. Con base en esta distribución se alimentará el DICCIONARIO/DIRECTORIO DE DATOS con las rutas de acceso de la localización de todos los datos que están distribuidos a través de la red. Con lo cual las funciones de administración de datos cuentan con las rutas alternativas de acceso a los datos para poder llevar a cabo el procesamiento de los *queries*, además de que permite administrar el acceso concurrente de los usuarios. Al tener nodos alternativos donde esté replicada cierta información nos permite tener un mayor grado de confiabilidad debido a que si llega a haber algún punto muerto en la red, esto no impedirá el acceso a los datos.

Por el grado de complejidad en las comunicaciones se hace necesario un Sistema Administrador de Transacciones el cual tiene la función de garantizar la INTEGRIDAD REFERENCIAL, sincronizando que los nodos participantes en una transacción lleven a su fin las tareas que les corresponden, por lo que si la transacción no puede ser totalmente realizada debe ser abortada.

Como ya se mencionó, existe una base de datos distribuida se puede tener como consecuencia un procesamiento cooperativo lo que implica que dos o más computadoras comparten el procesamiento de un programa.

El procesamiento cooperativo trae consigo grandes ventajas como:

- Una Interfaz Gráfica de Usuario amigable
- Se reduce la cantidad de procesamiento realizado en el servidor
- El procesamiento se divide entre el cliente y el servidor
- Además de la consideración de que la unidad de trabajo realizada en el cliente es más económica

Un aspecto muy importante que va íntimamente ligado al modelo cliente/servidor es el fenómeno llamado DOWNSIZING, el cual consiste en que las organizaciones en las cuales la plataforma de trabajo sea la minicomputadora o mainframe con sistemas centralizados que se van degradando con el tiempo y se pueden ir volviendo deficientes con respecto al cumplimiento de los nuevos requerimientos de la organización. Y llega el momento en que pueden ser reemplazables por microcomputadoras enlazadas a través de una red de comunicaciones.

Pero por otro lado, hay empresas u organizaciones que cuentan por lo regular con modernas minicomputadoras, debido a la enorme cantidad de información que manejan y procesan. Por lo tanto sería sumamente difícil manejar esta cantidad de información en un sistema basado en potentes microcomputadoras. Para estos casos lo más óptimo es enlazar la minicomputadora con la red de comunicaciones basada en PC's, y utilizarla como servidor de base de datos. Para aprovechar su gran capacidad de almacenamiento y procesamiento, al sumergirlo en un ambiente cliente/servidor.

CONCLUSIONES

En el caso del Sistema de Control de Gestión es necesario mantener la minicomputadora (HP-9000) debido a la gran cantidad de información que se va generando, y no tan sólo por este sistema sino también por otros que cumplen parte de los objetivos de la secretaría. Considerando que tales equipos todavía tiene un cierto periodo de vida al cual se le puede sacar provecho. Entonces, basándonos en lo anterior nuestro sistema será de tres niveles: el primer nivel consiste en la minicomputadora que fungirá como servidor de base de datos, el segundo nivel está constituido por potentes microcomputadoras que pueden contener alguna partición o réplica de la base de datos y el tercer nivel lo están formando todas las terminales que forman parte de la red local de cada nodo (clientes).

Por todo lo anterior el modelo cliente/servidor toma dos significados; primeramente el procesamiento cooperativo entre los componentes de software entre el cliente y el servidor. Y en segunda es la relación entre el hardware del sistema cliente y el sistema servidor. Esto se debe a que se da una especialización de las funciones tanto a nivel de software como de hardware.

Como se vio anteriormente, cualquier sistema o aplicación está constituido de varios componentes fundamentales, los cuales deben ser debidamente identificados para poder diseñarlos de la manera más adecuada. Dichos elementos son:

- La Presentación
- El Procesamiento Lógico de Negocios
- Las Funciones de Administración de Datos, que están constituidas por: el Procesamiento Lógico de Datos (Lenguaje de Manipulación de Datos) y el Procesamiento de la Base de Datos (DBMS)

El límite entre cada uno de los elementos anteriores es bastante difícil de definir, pero es necesario delimitarlos para realizar una conceptualización más clara en el análisis y diseño del sistema. En el caso de estudio de este trabajo la PRESENTACIÓN será ubicada en el cliente junto con las FUNCIONES DE ADMINISTRACIÓN DE DATOS que correspondan a cada nodo, en el caso de que el nodo contenga alguna partición. Y por lo tanto, cada nodo que contenga parte de la base de datos necesita el software necesario para administrar la información. En cuanto a la LÓGICA DE NEGOCIOS, también se encuentra distribuida debido a que los procesos que son necesarios se realizan a través de validaciones desde la PRESENTACIÓN, ubicada en el nodo cliente; o desde la base de datos a través de los STORE-PROCEDURE o los TRIGGERS.

De forma muy general se pudieron identificar cinco categorías de sistemas cliente/servidor, pero para llegar a tomar la decisión de ubicarse en una categoría es necesario tomar ciertos principios importantes. Por lo tanto, la primera interrogante es:

¿SE NECESITAN USUARIOS MÚLTIPLES? Si la respuesta es NO, entonces no se necesita un sistema cliente/servidor. Pero si la respuesta es afirmativa, se debe considerar la siguiente interrogante.

¿SON MÁS DE 20 USUARIOS SIMULTÁNEOS? Si la respuesta es NO, entonces tampoco es necesario un sistema cliente/servidor por lo que en estos casos la solución más adecuada en un sistema multiusuario. Pero en caso de ser afirmativa esta respuesta, se debe de considerar la siguiente interrogante.

¿EXISTIRÁN TRANSACCIONES MÚLTIPLES CON MODIFICACIONES DE DATOS? Si la respuesta es un NO, tampoco es necesario un sistema cliente/servidor para lo cual es más recomendable un sistema multiusuario. Pero si la respuesta es afirmativa, entonces lo más adecuado es la implantación de un sistema cliente /servidor. Pero para esto hay que tomar en cuenta otro tipo de aspectos que nos guiarán para poder ubicar en qué categoría de sistema c/s caería nuestro sistema.

¿EXISTE UN DBMS MULTIUSUARIOS BASADO EN PC? Si la respuesta es afirmativa, y además se tiene alguna aplicación en lenguaje DBASE y no se quieren reescribir las aplicaciones, con estas condiciones se puede obtener un sistema cliente/servidor Clase 4 con una PC como servidor, para lo cual se tendrá facilidad para modificar los sistemas existentes pero el costo será tener una baja integridad en los datos. Pero en el caso de que se requieran reescribir las aplicaciones o no existan aplicaciones hay que tomar en cuenta si existe algún equipo RISC o equipos con sistema operativo UNIX, que en tal caso se tendría un sistema cliente/servidor Clase 2 con un servidor RISC que proporcionará un alto poder de procesamiento pero con el inconveniente de que este sistema operativo además de ser muy grande agrega complejidad en las comunicaciones. Pero si no existe este tipo de equipo se puede presentar un tipo de sistema cliente/servidor Clase 2 con una PC como servidor. Ahora, en caso de que no exista un DBMS multiusuario basado en PC hay que considerar la siguiente pregunta.

¿EXISTE UNA MINICOMPUTADORA O MAINFRAME? En caso de que no exista se cae nuevamente en el caso anterior en donde hay que seleccionar si existe un sistema RISC. Pero en caso de que si exista una mini o una mainframe es necesario determinar si se realizará el DOWNSIZE, porque de ser así tendríamos un sistema cliente/servidor Clase 2 con una PC como servidor. Y en el caso de que no se cambie de plataforma se considera la siguiente pregunta.

¿LOS DISTRIBUIDORES DEL DBMS TIENEN OPCIONES C/S? Si se tiene una respuesta negativa se caería en un sistema cliente/servidor Clase 3 con compuertas, con el cual se incrementa la complejidad por el enlace de los sistemas. Y en el caso de que se tengan opciones es necesario determinar lo siguiente.

¿TRABAJAN CON PROTOCOLOS COMUNES PARA PC O WORKSTATION (NetBIOS, TCP/IP, IPX)? Si la respuesta es negativa tendremos un sistema cliente/servidor con hardware para puentes (bridge) y compuertas (gateway). A través del cual, se obtendrá la comunicación

deseada pero será necesario hardware adicional que agrega complejidad y costo al sistema. Pero en caso de que se pueda trabajar con protocolos comunes se tendrá un sistema cliente/servidor con software adicional para compuertas (gateway), para lo cual dicho software se ejecuta en el equipo anfitrión pero requerirá poder adicional del CPU. Este caso es precisamente el que se presentó en el "Sistema de Control de Gestión", ya que se cuenta con una red WAN basada en microcomputadoras que tiene integrada la minicomputadora HP9000.

En cuanto a la tecnología disponible para este tipo de sistemas se presentó, en el capítulo II en la sección 3, una panorámica de los DBMS's disponibles en cada una de las plataforma. Para el caso particular que se trata en el "Sistema de Control de Gestión" se presenta un sistema cliente/servidor basado en UNIX como servidor de base de datos. Para lo cual, se consideraron los siguientes aspectos como base para la decisión en la elección de la minicomputadora como el servidor de archivos:

- **Hardware**: La minicomputadora con que se cuenta es un equipo al cual se le puede obtener todavía mucha ventaja. Además, de lo necesaria que es debido a la capacidad de almacenamiento con que cuenta, considerando las necesidades crecientes de la secretaría.

- **Compatibilidad con las redes existentes**: El hardware de la minicomputadora tiene soporte para la topología de red de las LAN en que se ejecutará.

- **Soporte y entrenamiento**: Este aspecto es ampliamente cubierto por el proveedor (Hewlet Packard).

- **Monitoreo de la eficiencia**: Los sistemas basados en UNIX son bastante eficientes en este aspecto debido al tiempo que tienen de existir.

- **Software de front-end**: Como en este caso se requiere un sistema ***gateway*** adicional para las comunicaciones, con lo que se está agregando costo extra. Se seleccionó el software adecuado de acuerdo a todas las características antes mencionadas. Y se eligió la herramienta de desarrollo de aplicaciones **POWER BUILDER ENTERPRISE** por haberse considerado la más adecuada.

Y como consecuencia de los puntos tratados se decidió utilizar como Sistema Administrador de la Base de Datos (DBMS) el **SYBASE SQL SERVER**, por adecuarse a los requerimientos del sistema.

En lo relacionado a las metodologías de desarrollo sería bueno retomar el cuadro de las diferentes metodologías analizadas tratadas en el capítulo III sección 2, que se muestra a continuación.

CONCLUSIONES

	Herramientas Técnicas	Análisis y Diseño	Arquitectura	Metodología para Sistemas Operacionales	Metodología para DSS
Inmon	• Diagramas de Flujo de datos DFD • Diagrama Entidad Relación D E-R	SI	NO SE PROFUNDIZA	SI	SI
Vaughn	* Cualquier técnica o herramienta, vieja o innovadora que ayude al objetivo.	SI	SI	SI	NO
Vaskevitch	• Modelado conceptual, lógico y físico en los diferentes aspectos del sistema • Análisis y diseño orientado a objetos • Tal vez la programación orientada a objetos	SI	NO SE PROFUNDIZA	SI	NO
Renaud	• Diagramas de distribución de recursos, escalabilidad y seguridad • Técnicas de fiabilidad, eficiencia y capacidad.	SI	SI	ES INDEPENDIENTE DEL TIPO DE SISTEMA	ES INDEPENDIENTE DEL TIPO DE SISTEMA
Kavanagh	• Sesiones para la Planeación de Requerimientos (JRP) • Sesiones de Diseño de la Aplicación (JAD) • RAD	SI	NO SE PROFUNDIZA	SI	NO

* DIAGRAMAS DE FLUJO DE DATOS, DIAGRAMA DE FLUJO DE TRABAJO, DIAGRAMAS CAUSA-EFECTO, BRAINSTORMING, PROTOTIPOS RÁPIDOS, MODELADO DE DATOS, ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS, ETC.

La metodología que se tomó como base para el diseño del "Sistema de Control de Gestión" fue la de Vaughn, debido a la flexibilidad que presenta en la utilización de herramientas que pueden ir fortaleciendo el análisis y diseño de un sistema sin que se desvirtúe el principio que guía la metodología. Además de que trata de manera especial lo relacionado con la ARQUITECTURA del sistema y la forma iterativa en que se van sucediendo las tres etapas de

que consta dicha metodología. Con lo que permite incorporar ciertos elementos de otras metodologías para ir complementando en los aspectos que se crea más conveniente.

Al entrar al capítulo V e iniciar el análisis y diseño del "Sistema de Control de Gestión" se fueron explicando las diferentes etapas para obtener el diseño más óptimo. Y como se mencionó varias veces el alcance de este trabajo se limita hasta el diseño, por lo que no se llevaron a cabo todos los pasos que implica la metodología de Vaughn. Pero se trató de dar un esquema básico de los elementos necesarios a considerar en el diseño de un sistema cliente/servidor.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

PC Week, Sep. 12, 1994. V11 N.36.
"Perceived and actual risks of client/server"
Comaford Christine

"Redes de área local"
Thomas W. Madron
Editorial Limusa, México, D.F., 1993.

Byte México, Junio 1995, Año 9, N. 89.
"Arquitectura cliente/servidor"
Gerardo Quiroz Vieyra

"Developing client/server aplication"
W. H. Inmon
QBE Publishing Group, 1993. USA.

"Local area networks (arquitecture and implementations)"
James Martin with Kathleen K. Chapman.
The Arben Group, inc. Prentice Hall Englewood Cliffs
New Jersey 1989, USA.

"Lan applications - client/server databases"
Jim Krochmal and Larry Morris
New Riders Publishing. USA 1993.

"Open systems networking (TCP/IP and OSI)"
David M. Piscitello and Lyman Chapin.
Addison wesley professional computing series, 1993
Second printing, march 1994, USA.

"Redes de computadoras (Protocolos, normas e interfases)"
Uyless Black
Ra-Ma, Madrid España, 1989. Macrobit, México D.F., 1990.

"Distributed Database, Cooperative Processing, & Networking"
Shaku Atre
Database Experts' Series, Published by McGraw-Hill
Primera edición, U.S.A. 1992

BIBLIOGRAFÍA

"Client/Server Achitecture"
Alex Berson
Jay Ranade, Series Advisor, McGraw-Hill
Edición 1994, Singapore

"Guide to Client/Server Database"
Joe Salemi
U.S.A. 1995

"Concepción y Diseño de Bases de Datos"
(del Modelo E/R al Modelo Relacional)
Adoración de Miguel Castaño
Mario Gerardo Piattini Velthuis
Addison -Wesley Iberoamericana. Serie paradigma,
USA 1993.

"Principles of Distributed Database Systems"
M. Tamer Ozsu
Patrick Valduriez
Prentice Hall, Englewood Cliffs, New Jersey, 1991.

"Fundamentos de Bases de Datos"
Henry F. Korth
Abraham Silberschatz
Segunda edición, McGraw Hill, España 1993.

"Client/Server System Design & Implementation"
Larry T. Vaughn
Shaku Atre, Series Advisor, McGraw-Hill
U.S.A., 1994.

"Client/Server Strategies"
David Vaskevitch
IDG Books, U.S.A., 1993.

BIBLIOGRAFÍA

"Introduction to Client/Server Systems - A practical guide for systems professionals"
Paul E. Renaud
John Wiley & Sons, Inc., U.S.A., 1993.

"Downsizing for Client/Server Applications"
Paul Kavanagh
AP Professional, U.S.A., 1995.