8
29

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

"LA TECNOLOGIA OLTP"

P R E S E N T A :

SARA EUGENIA ESPARZA REGALADO

SEMINARIO DE INVESTIGACION INFORMATICA

PARA OBTENER EL TITULO DE:

LICENCIADO EN INFORMATICA

DIRECTOR DEL SEMINARIO:
C.P.M.B.A. JOSE ANTONIO ECHENIQUE GARCIA

DIRECTOR DE LA FACULTAD DE CONTADURIA Y ADMINISTRACION

1 9 9 3

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DEDICATORIAS

Gracias Dios Padre por tu grandeza y por seguir alimentandome cada día de tu espíritu.

PADRES: Les dedico este estudio en agradecimiento por su amor, entendimiento y confianza. A ti mamá Sarita especialmente por la fuerza que me has infundido con mi amor y admiración.

✞ Sra. Socorro Hernandez en mi corazón te llevo siempre y en tí me inspiro para amarte.

Sandy Esparza que tu nobleza, amor y todas tus cualidades hermosas te acompañen en toda la vida.

Marianela Durán Vasconcelos: Gracias por tu amistad e inteligencia, con estimación Sara.

C.P. M.B.A. JOSE ANTONIO ECHENIQUE GARCIA: Gracias por su colaboración durante toda la Licenciatura.

Al M. en C. Ing. Anibal Bustillo Leal. Le agradezco la orientación para la selección de mi tema de tesis, así como su apoyo y colaboración durante la elaboración de ésta.

Dra. Concepción Astudillo y Al Ing. Emilio Dorado les agradezco su colaboración.

Y en general todos mis amigos y familiares.

La sabiduría infunde vida a sus hijos, y acoge a los que la buscan, y va delante de ellos en el camino de la justicia.
Evan.Sabiduría.

INDICE

PAGINA

I.- INTRODUCCION

El estudio de esta tesis tiene la finalidad de proveer un conocimiento más claro y conciso acerca de la tecnología de Procesamiento de Transacciones en Línea (On Line Transaction Processing ***"OLTP"***), así como los principales tópicos y las características técnicas que coadyuven a una adecuada selección de este tipo de herramientas informáticas, con base en los productos comerciales que actualmente se encuentran disponibles en el mercado y de sus avances futuros más aptos a la estructura informática de las Empresas.

Un OLTP es de gran importancia y utilidad en aquellas Empresas que desarrollan aplicaciones tales como:

- Bancos.
- Financieras.
- Casas de Bolsa.
- Manufacturación.
- Manejo y Control de Inventarios.
- Reservaciones aéreas, hoteles, etc.
- Y otras Empresas que requieran de aplicaciones orientadas a transacciones.
- Empresas de Telefonía (Red Digital de Servicos Integrados ***"RDSI"*** (Integrated Services Digital Network ***"ISDN"***).

La tecnología OLTP es un sistema orientado a transacciones con un alto "performance", diseñado para asegurar y optimizar la integridad, confibilidad, actualización, recuperación de fallas, estandarización y control de la información; con tiempos de respuesta rápida para un amplio número de usuarios para las actividades de las Empresas que requieren de altos volumenes de peticiones a la ***"B.D."*** (Base de Datos) tanto descentralizadas como distribuidas, en esta última, que es la tendencia a seguir, es mayor su utilización y su adecuada colaboración en la problemática de áreas (sucursales, puntos de venta, etc) foráneas o geográficamente distribuidas. En este estudio también se contemplan los cuatro diferetes puntos de vista que definen un ***"SPT"*** (Sistema de Procesamiento de Transacciones es la implantación del la tecnología OLTP)con base a un mismo ejemplo, en este caso, un Sistema de Correo el cual tipifica exáctamente estas funciones.

Asimismo se contemplan una serie de servicios que proporcióna la tecnología OLTP, con objeto de facilitar las comunicaciónes de las transacciones, el "encolado" de peticiones, el cual funcióna semejante a un ***"spooler"***, pero más sofisticado, ya que éste es inteligente, sobre todo en el manejo de fallas (energía, etc), el manejo de bitácora, el cual es de gran utilidad para la administración y control de las transacciones y otros servicios como el de seguro y el de recuperación entre otros.

Otra parte de interés en este estudio, son los componentes o características que conforman un OLTP, destacando las del monitor de transacciones, también conocido como servicio, el de repositorio y los de la B.D. entre otras. Se dedica un capítulo a éste tema debido a su importancia, ya que estas características son la parte que sustentan el análisis, motivo principal de esta tesis.

II.- OBJETIVOS

Analizar la tecnología OLTP existente en el mercado tecnológico, a fín de conocer sus características, técnicas principales y para conformar un portafolio de opciones que permitan a los empresas la selección de la tecnología adecuada.

Evaluar los componentes principales del modelo transaccional y propuestas de estándares a fin de constituír una guía fundamental para el desarrollo de aplicaciones OLTP.

Comparar los modelos de monitores de transacciones más comerciales y la mayoría de éstos que se han basado en el modelo X/OPEN a fín de conocer las características funciónales de cada uno.

Proporcionar un enfoque sistemático que permita a los responsables de las áreas de informática contar con los criterios de decisión suficientes para la evaluación y selección adecuada de la tecnología en procesamiento de transacciones.

Seleccionar el OLTP adecuado en base a sus características funcionales para que pueda aplicarse en las necesidades específicas de un cliente.

Fomentar el conocimiento de esta herramienta de avanzada tecnología, para que coadyuve al incremento de la productividad dentro de las Empresas u Organismos.

Conocer las tendencias de la tecnología OLTP a fín de enriquecer los conocimientos y/o adoptar otras tecnologías computaciónales que sean amigables a ésta.

III.- SISTEMA DE PROCESAMIENTO DE TRANSACCIONES "SPT".

Hace seis mil años, los Sumerios inventaron la escritura para procesamiento de transacciones. La escritura más antigua es encontrada en tablas de arcilla, registrando el inventario real de impuestos, tierras, granos, ganado, esclavos y oro; las escrituras mantuvieron evidentemente, registros de cada transacción. Este antiguo sistema tenía los aspectos principales de un Sistema de Procesamiento de Transacciones *"SPT"*.

Con la escritura de los Sumerios favoreció el que se proporcionará un registro histórico sobre el actual y pasado estado de las transacciones. La tecnología del SPT basado en arcilla, evolucionó sobre miles de años a través de los papiros, pergamino y papel. En miles de años, el papel, la tinta y libros mayores, fueron la tecnología para procesamiento de transacciones. La más reciente inovación comenzo a fines de 1800, cuando Herman Hollerith construyó sistemas de computadoras de tarjetas perforadas para registrar y reportar los censos de Estados Unidos en 1890. Durante el primer medio del siglo veinte, la necesidad para procesamiento de transacciones completo la evolución y crecimiento de equipo de tarjetas perforadas. Estos primeros computadores fueron usados primariamente para control de inventario y contabilidad. En efecto, remplazaron las tablas de arcilla con tablas de papel (tarjetas); su virtud fue que el sistema pudiera buscar y actualizar alrededor de una "tabla" (tarjeta) por segundo. Con la inovación de las computadoras y el incremento de la cantidad de información en aplicaciones relacionadas con los negocios aumentan las transacciones y las necesidades de almacenar información (B.D.). Estos dos conceptos son interpretados a continuación en base al orígen y registro de las escrituras antíguas de los Sumerios

B.D.- Es un estado de sistema abstracto, representado como marcas o tablas de arcilla que fueron escritas. En la actualidad, llamamos esto la B.D.
Transacciones.- Las escrituras registran cambios de estado con nuevos registros (tablas de arcilla) en la B.D. En la actualidad, llamamos a estos cambios de estado, transacciones.

La segunda mitad del siglo veinte vio dos principales desarrollos en el Procesamiento de Transacciones: el Procesamiento de Transacciones Batch basados en almacenamientos magnéticos (cintas y discos), seguido por el **Procesamiento de Transacciones en Línea** (OLTP), basado en almacenamientos electrónicos y redes de computadoras. Estos dos desarrollos fueron grandemente responsables del crecimiento en la industria de la computación y las aplicaciones de procesamiento de transacciones. Hoy el uso primario de las computadoras de proposito general es el estilo de procesamiento de transacciones. Algunas aplicaciones típicas y ejemplos son incluídos a continuación:

- Finanzas.- Bancos, cuentas, ventas etc.
- Comunicaciónes.- telefonía, correo electrónico etc.

- Viajes.- Reservaciones y pagos para aerolíneas, hoteles, carros y trenes.
- Manufactureras.- Ordenes de entrada, trabajos y planeación de inventarios, programación y contabilidad,
- Control de procesos.- Control de fábricas, control de líneas de producción, almacén, acero, papel y plantas químicas, etc.

III.1.- DEFINICION DE UN SPT.

Un SPT tiene diferentes perspectivas: la de un usuario final, administrador/operador, diseñador de aplicaciones y del manejador de recursos. Cada uno de estos puntos de vista del sistema son diferentes, es difícil dar una sola definición de lo que es un SPT y qué hace.

Un Sistema de Procesamiento de **Transacciones** proveé herramientas para facilitar o automatizar la programación de aplicaciones, ejecución y administración. Las aplicaciones de procesamiento de transacciones típicamente soportan una red de dispositivos para consultas y actualización a la aplicación. La aplicación mantiene una B.D. que representa algunos estados del mundo real. Las respuestas de la aplicación y las salidas, manejan actuadores del mundo real y transductores (convertidor de energía) que alteran o controlan el estado. Las aplicaciones, B.D., y red, tienden a evolucionar sobre varias décadas. Aún más, los sistemas están geográficamente distribuidos, heterogéneos (que involucran equipo y software de distintos vendedores), disponibilidad continua y tienen respuestas en cadena en requerimientos de tiempo. A esta tecnología se le conoce como Procesamiento de Transacciones en Línea (On Line Transaction Processing ***"OLTP"***). Un sistema OLTP colecciona información acerca de las transacciones y destina los cambios de la información para las Organizaciones dictadas por una B.D. compartida o por un archivo. El Procesamiento de Transacciones es un estilo de computación caracterizado por el procesamiento concurrente de peticiones para la ejecución de transacciones que accesan archivos compartidos, B.D., y otros recursos (por ejemplos colas de peticiones) en un ambiente.

Un SPT incluye generadores de aplicación, herramientas de operaciones, uno o más sistemas de B.D., utilerías, todo lo relativo a redes y software del sistema operativo. Historicamente, los SPT significaron Sistemas de Tele-procesamiento y denotaron un programa que soporta una gran variedad de tipos de terminales y protocolos de red. Algunos sistemas de procesamiento actuales evolucionaron de los sistemas de teleprocesamiento. Dentro del SPT, existe una colección central de servicios, uno de estos servicios principalmente es llamado Monitor de Procesamiento de **Transacciones** ***"Monitor de PT"***, que maneja y coordina el flujo de las transacciones a través del sistema. También es definido como: Una herramienta que coordina el flujo de las peticiones de una transacción entre los procesos clientes, y como consecuencia de estas peticiones son los procesos. El término ***transacción*** puede tener algunos de los siguientes significados:

1.-	Los requerimientos o mensajes de entrada que arrancaron la operación.	Transacción petición/respuesta
2.-	Todos los efectos de la ejecución de la operación.	Transacción

3.-	El(los) programa(s) que ejecuta(n) la operación.	Transacción de programa
4.-	La unidad básica de procesamiento consistente y confiable ante un acceso concurrente de múltiples usuarios.	Transacción

La transacción está compuesta por dos estados principales, que son:

- Estado consistente.
- Estado confiable.

El estado consistente de una transacción sustenta la estabilidad de la información ante usuarios que efectúan acceso de lectura y actualización en cualquier B.D.

El estado confiable de una transacción soporta diversos tipos de fallas, es decir, alteraciones en los datos y aseguran la capcidad para recuperarse de estas fallas en las Bases de Datos.

Las transacciones se ejecutan en forma concurrente, por tal motivo pueden interferir entre sí, debido a la asignación y retiro del procesador conforme al *"quantum"* (Quantum es un subrango en cuantificación, es decir la subdivisión de un rango de valores de una variable dentro de un número finito de no sobrenlapados y no necesariamente subrangos iguales o intervalos, cada uno de los cuales está representado por un valor asignado dentro del subrango. Por ejemplo la edad de una persona es cuantificada por la mayoría de los propósitos con un quantum de un año) de tiempo estipulado; por lo tanto los problemas que causa la concurrencia son:

a).- Pérdida de actualización: dos transacciones leen un valor original y subsecuentemente lo actualizan.

b).- Salida inconsistente: una transacción de consulta lee un dato antes de que otra de actualización lo modifique.

El SPT fue el pionero de muchos conceptos en computación distribuida y computación tolerante a fallas. Estos introdujeron datos distribuidos para confiabilidad, disponibilidad y performance, ellos desarrollaron almacenamiento tolerante a fallas y procesos tolerantes a fallas para disponibilidad; y desarrollan el modelo cliente/servidor (ver punto III.1.1.) y llamados a procedimientos remotos (RPC ver punto III.1.3.) para computación distribuida. Lo más importante es que ellos introdujeron las propiedades de las transacciones conocidas como *"ACAD"* Atomicidad, Consistencia, Isolación (aislamiento) y Durabilidad que han surgido como los conceptos de unificación para la computación distribuida. La naturaleza de una transacción impone 4 propiedades básicas, en los sistemas OLTP.

ATOMICIDAD
La Atomicidad reconoce que una transacción compromete dos o más piezas de información. El trabajo hecho y todas las piezas de información en una transacción, deben ser *comitted* (todo o nada) o ninguna de estas.

CONSISTENCIA
Requiere que la transacción ya sea que cree un nuevo y válido estado de los recursos afectados, o si en caso de que la transacción *abort* (aborta)regresa el recurso a un estado previo antes de que haya comenzado la transacción. Esto quiere decir, que cada una de las transacciones creen un nuevo y válido estado de organización en los datos compartidos, o si la transacción abort, regresa el dato al estado previo.

AISLAMIENTO (ISOLACION)
Cuando una transacción está en progreso, el trabajo debe mantenerse separado y aislado de otras transacciones que están siendo procesadas.

DURABILIDAD
La durabilidad envolverá la habilidad del sistema para asegurar que el efecto de una transacción no sea afectada por una falla. Esto quiere decir, que un sistema OLTP salva una transacción *committed* (comprometiendo) de los cambios que se hagan a los datos compartidos, a pesar de que se presenten fallas subsecuentes. Muchos ambientes, sin embargo, tienen aplicaciones populares con un conocido y predecible grupo de atributos con usuarios que accesan estas aplicaciones repetitivamente.

Para ejemplificar las cuatro propiedades de la transacción se expone en una aplicación bancaria. Una transacción bancaria de débito, es atómica si ambos distribuyen dinero y actualizan su cuenta. Es consistente, si el dinero distribuido es el mismo débito para las cuentas. Es aislada, si el programa transacción es ajeno a otros programas leídos y escritos para su cuenta actual, (por ejemplo, su esposo hace un deposito concurrente). Y es durable si una vez que la transacción es completa, la cuenta del balance es seguro para reflejar retiro.

COMPONENTES DE UNA TRANSACCION

La transacción es una unidad lógica de ejecución que se compone de un conjunto de acciones (u operaciones) de lectura (Read) y escritura (Write), con delimitaciones de inicio (*start*) y terminación normal (*commit*) o anormal (*abort*). Los componentes principales de la transacción son:

Start.- Indica al Manejador de Sistemas de B.D. (Data Base Management System *"DBMS"*), que se inicia la ejecución de una nueva transacción.

Commit (Compromiso).- Indica al DBMS la terminación normal de la transacción y que todos sus efectos sean permanentes.

Abort.- Indica al DBMS la terminación anormal de la transacción y que todas sus acciónes no causarán efecto.

A continuación se presenta la parte de un programa desarrollado en lenguaje C, que efectúa la ejecución de los componentes de la transacción con campos de una operación bancaria.

```
                    Procedure TRANSFERENCIA
                    Begin
                      start;
                      Input (cuenta_fuente, cuenta_destino, cantidad);
                      temp:= Read(cuentas cuenta_fuente );
                      IF temp < cantidad THEN
                              Begin
                                 output( "fondos insuficientes" );
                                 Abort;
                              End
                      ELSE
                              Begin
                              Write(cuentas [cuenta_fuente] , temp - cantidad);
                              temp:= Read(cuentas [cuenta_destino] );
                              write(cuentas [cuenta_destino] , temp + cantidad);
                              Commit;
                              Output( transferencia completada );
                              End;
                      End
                       A2
T2 = S2 -->   r2 cf -->
                        w2 [cf] --> w2 [cd] --> C2
```

OPERACION DE LA TRANSACCION

La forma en que opera una transacción es de la siguiente manera:

El compromiso "commit" de una transacción marca un punto sin regreso que separa la transacción en dos fases:

1).- Antes del commit, el efecto de la transacción puede ser anulado y restaurado el estado inicial.

2).- Después del commit las modificaciones son irreversibles y la transacción debe ser llevada a su fín.

La transacción se lleva a cabo completa o no tiene efecto alguno. Es decir, la transacción es de naturaleza atómica. En el esquema 1, se ejemplifica cómo se recupera una transacción.

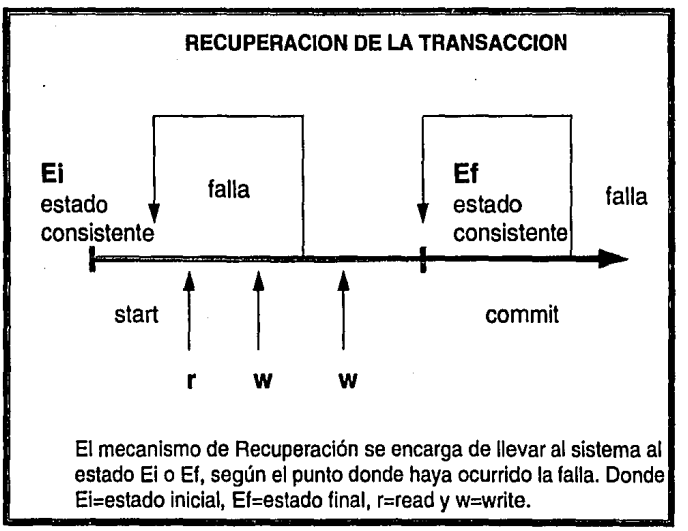

FIGURA III.1.- RECUPERACION DE LA TRANSACCION. El mecanismo de Recuperación se encarga de llevar al sistema el estado Ei o Ef, según el punto donde haya ocurrido la falla. Donde Ei= estado inicial, Ef=estado final, r=read y w=write.

III.1.1.- VISION DEL USUARIO FINAL DE UN SPT

La visión del usuario final en el ejemplo del sistema de correo, es que son buzónes asociados con gente y con mensajes (ver figura III.2). El usuario primero se identifica él mismo al sistema. Cuando ésta transacción se completa, el usuario es presentado con una lista de mensajes de entrada. El usuario puede leer un mensaje, replicarlo, copiarlo, borrarlo, enviar un nuevo mensaje o cancelar un mensaje que él envio. Cada una de estas operaciones son una transacción ACAD. La operación de leer puede ver el mensaje completo mientras la operación de borrado remueve el mensaje completo. Las interacciones entre enviar y cancelar, es el aspecto más ilustrativo de este ejemplo, si el sistema falla mientras el usuario está componiendo un mensaje y antes que el usuario emita un envío, entonces el mensaje no es enviado y la entrada del usuario se pierde. Después el usuario emitio satisfactoriamente el envío, el mensaje es entregado. Así, el envio es una transacción ACAD. Dos transacciones más son involucradas; el mensaje es entregado al buzón del receptor y el mensaje es leído por el receptor. Estas son también unidades ACAD. El enviador puede cancelar (borrar) un mensaje no leído, pero la cancelación no tendrá efecto si el mensaje ha estado listo para ser leído por el receptor. La cancelación es otra transacción ACAD; esto es llamado una **Transacción Compensada**, desde que trata de compensar los efectos de una transacción previamente committed.

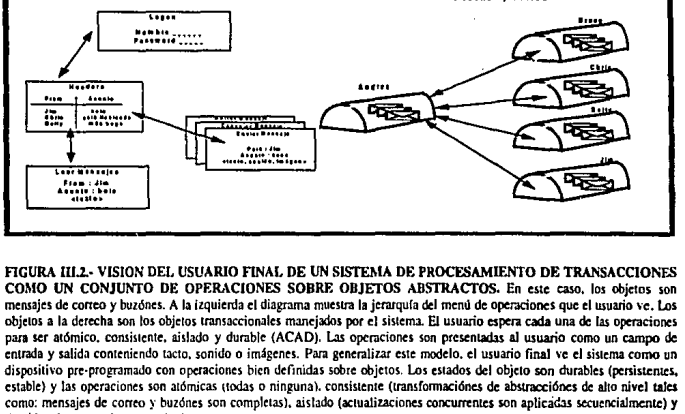

FIGURA III.2.- VISION DEL USUARIO FINAL DE UN SISTEMA DE PROCESAMIENTO DE TRANSACCIONES COMO UN CONJUNTO DE OPERACIONES SOBRE OBJETOS ABSTRACTOS. En este caso, los objetos son mensajes de correo y buzónes. A la izquierda el diagrama muestra la jerarquía del menú de operaciones que el usuario ve. Los objetos a la derecha son los objetos transaccionales manejados por el sistema. El usuario espera cada una de las operaciones para ser atómico, consistente, aislado y durable (ACAD). Las operaciones son presentadas al usuario como un campo de entrada y salida conteniendo tacto, sonido o imágenes. Para generalizar este modelo, el usuario final ve el sistema como un dispositivo pre-programado con operaciones bien definidas sobre objetos. Los estados del objeto son durables (persistentes, estable) y las operaciones son atómicas (todas o ninguna), consistente (transformaciónes de abstracciónes de alto nivel tales como: mensajes de correo y buzónes son completas), aislado (actualizaciones concurrentes son aplicadas secuencialmente) y durables (los mensajes no se pierden).

La operación de componer un mensaje de correo podría involucrar actualizar muchos registros de la B.D., representando el mensaje, y podría esto requerir preguntar el nombre del servidor remoto para verificar la correcta dirección del correo; todo este trabajo es empaquetado dentro de una unidad ACAD. Si cualquier cosa estuvo mal durante la composición del mensaje, todas las actualizaciones de la operación son regresadas y el mensaje es descartado; si todo va bien, todas las actualizaciones de la operación son hechas permanentes y el mensaje es aceptado para ejecución. Similarmente la ejecución del mensaje al destino podría encontrar muchos problemas. En tales casos, la transacción es abortada y una nueva ejecución de la transacción es arrancada. Eventualmente, el mensaje es ejecutado al buzón destino (insertado). Después éste, puede ser leído por el recipiente y replicado o borrado. Por supuesto, para hacer el mensaje permanente, esto es probablemente replicado (copiado en disco, cinta, etc) en dos o más lugares, que cuando se presente una simple falla no pueda dañarlo; así nuevamente, es transparente (no lo ve) al usuario final, quien piensa en términos de composición, envío, lectura y cancelación de mensajes. Para cada una de estas operaciones, la aplicación de correo electrónico proveé al usuario final con operaciones ACAD sobre objetos permanentes.

En este ejemplo, las operaciones son mostradas a humanos como una interface orientada a formas con texto, voz y entradas de boton. Las formas de salida contienen textos, sonidos e imágenes. Y en muchas otras aplicaciones, el usuario no es una persona, pero preferentemente un dispositivo físico tal como un switch de teléfono, como una ala de aeroplano o un robot. En estos casos, las entradas son las lectoras de posición y sensores táctiles mientras las salidas son un comando a actuadores. Los sensores presentan al SPT con mensajes de entrada que invocan una transacción y los transductores actúan sobre los mensajes de salida de la transacción.

MODELO CLIENTE SERVIDOR

Los Sistemas de Procesamiento de Transacciones se basan en la arquitectura conocida como el Modelo Cliente Servidor. El Cliente/Servidor es la plataforma de cómputo básica para proporcionar conectividad, interoperabilidad, cooperabilidad y colaborativídad. Un Sistema Distribuido (al igual que uno centralizado) puede ser visto como un entorno compuesto por un conjunto de objetos que interactúan entre sí.

Los objetos son recursos físicos (procesador, periféricos y memoria central) o lógicos (archivos, proceso, buzón y catálogo). Si tenemos un esquema cliente/servidor, entonces su cliente es un usuario de objetos (les aplica operaciones), mientras que su servidor administra uno o varios de ellos. Un servicio es cualquier función computaciónal (impresión, B.D., supercómputo, procesamiento de imágenes, etc.) que puede ofrecer un sistema distribuido. Un servidor es la entidad (hardware y/o software) capaz de proporcionar el servicio deseado.

III.1.2.- VISION DEL ADMINISTRADOR/OPERADOR DE UN SPT.

El administrador de este sistema de correo electrónico tiene usuarios en Asia, Australia, América, Europa y Africa (ver figura III.3). Existen staffs (grupo de apoyo para proyectos) administrativos en cada uno de estos continentes. Así mismo el sistema tiene comunicación a otros sistemas de correo y los administradores cooperan con los manejadores de aquellos sistemas.

La función primaria del administrador es agregar y borrar usuarios, proveerlos con documentación y entrenamiento y manejar la seguridad del sistema. La seguridad y la interface de administración proveen formas y programas de transacciones que permiten al administrador agregar usuarios, cambiar claves de acceso y examinar la seguridad de entrada para detectar violaciones, entre otras cosas.

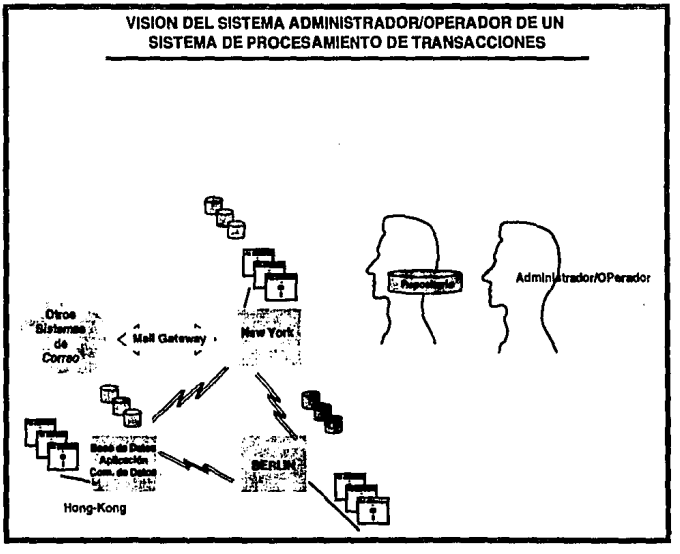

FIGURA III.3.- VISION DEL SISTEMA ADMINISTRADOR/OPERADOR DE UN SISTEMA DE PROCESAMIENTO DE TRANSACCIONES. Esta visión consiste de nodos físicos conteniendo hardware e interactuando software de modulos. La aplicación incluye interfaces para facilitar la administración y operación del sistema. El repositorio es una B.D. que registra la configuración del sistema. El administrador ve el sistema a través del repositorio.

Estas transacciones administrativas son operaciones ACAD en la B.D. del usuario y la B.D. de seguridad. Este es otro aspecto de la aplicación y está programado con la misma herramienta de Procesamiento de Transacciones de alto nivel usado por otros diseñadores de aplicaciones.

Además de éstas tareas, el administrador del sistema monitorea la confibilidad del sistema y planea el crecimiento del equipo y como los mensajes del correo consumen más almacenamiento conforme se incrementa el volumen del correo y el consumo de espacio en disco, procesadores y el ancho de banda de la red. Una vez que el sistema está instalado, el administrador obtiene reportes periodicos sobre la utilización del sistema, indexado por componente y por usuario. Otra vez, estas son aplicaciones estándar que usa la salida de un monitor de confibilidad para poblar una confiable B.D. ellos usan un escritor de reportes para generar pantallas formateadas de los datos.

La interface administrativa es una parte integral de cualquier sistema de procesamiento de transacciones y aplicaciones. Si desea cualquier otra aplicación de procesamiento de transacciones, esto es programado para proveer una interface transaccional orientado a formas a la configuración del sistema. La configuración del sistema es representado por la B.D., llamado el repositorio o diccionario del sistema.
El administrador puede alterar la configuración usando transacciones estándar, preguntar por la configuración usando reportes estándar o leer la configuración usando un lenguaje no procedural, tal como un ***"SQL"*** (Es un lenguaje de B.D. relacional definido por la International Standards Organization ***"ISO"*** en 1986.), para generar reportes especializados. El concepto repositorio está desarrollado con más detalle en el subcapítulo IV.2 (Características del repositorio). El punto clave es que la configuración del sistema debería estar representado como una B.D., que puede ser manipulada usando las herramientas de software de alto nivel del Sistema de Procesamiento de Transacciones.

El administrador del sistema, operador del sistema y el diseñador de aplicaciones también presentan problemas de confiabilidad. Los sistemas a veces corren ambas "transacciones interactivas", las cuales son procesadas mientras el cliente espera una respuesta y las "transacciones batch" que son enviadas a ejecución podrían ser procesadas más tarde. Las transacciones interactivas son necesariamente pequeñas, ya que deben ser procesadas en unos pocos segundos que el cliente está dispuesto a esperar. Si el sistema está sobrecargado o pobremente configurado, los tiempos de respuesta pueden incrementarse más allá de un nivel aceptable. En este caso, el operador y el administrador son llamados para poner a punto o afirmar el sistema para mejorar la confiabilidad (performance). Como un último recurso, el administrador puede instalar más equipo, incrementando así el proceso de afinación que está siendo automatizado.

La mayoría de los sistemas proveen herramientas que miden la confiabilidad, recomendando o ejecutando acciónes de afinación y recomendando el tipo y calidad de equipo necesario para procesar una carga proyectada.

Estimar la confiabilidad de un SPT es difícil porque los sistemas son complejos y porque su performance sobre diferentes problemas puede ser muy diferente. Un sistema es bueno para trabajos batch, otro excelente sobre transacciones interactivas y todavía un tercero es ajustado hacia aplicaciones de recuperación de información. Gradualmente, varios patrones usados están siendo reconocidos y los benchmarks estándar (pruebas de rendimiento estimado), están siendo definidos para representar la carga de trabajo de problemas que se presenten.

Un consorcio industrial, el Comité de Certificación del Rendimiento (performance) de Procesamiento de Transacciones ***"TPC"*** (Transaction Processing Performance Council), tiene definido tres benchmarks (pruebas de rendimiento). Con cada uno de estos benchmarks llamados A, B, y C, contiene una B.D. escalable y carga de trabajo, permitiendo a los benchmarks estar corriendo sobre los computadores, como también sobre redes y bases de datos de cualquier tamaño, desde las más pequeñas a las más grandes. Usando estos benchmarks, cada uno de los sistema obtiene tres *"throughput ratings"* censos de rendimiento llamados transacciones por segundo ***"tps"*** (tps A, tps B y tps C).

Como los tps levantan el porcentaje (rating), el tamaño de la red y el tamaño de la B.D. y se escalan de acuerdo al raiting. Este benchmark también mide la relación precio/confiabilidad del Sistema por el precio a cinco años del hardware, software y mantenimiento del vendedor. confiabilidad del vendedor. Este precio de 5 años es entonces dividido por el rating del tps para obtener un rating precio/performance ($/tps).

III.1.3.- VISION DE LOS DISEÑADORES DE APLICACIONES DE UN SPT

Basándonos en el ejemplo del sistema de correo la perspectiva del diseñador o implementador de las aplicaciones. En el pasado los roles del diseñador e implementador de aplicaciones fueron separados. Con la llegada de los generadores de aplicaciones, prototipos y lenguajes de programación de cuarta generación ***"4GL's"*** (Fourth Generation Language), muchas tareas de implementación han sido automatizadas, esto es, que el diseño es la implementación.

Las aplicaciones están ahora estructuradas rutinariamente como procesos cliente, acercan al usuario, hacen requerimientos para procesos del servidor corriendo en otras computadoras.

En el ejemplo del sistema de correo, el cliente es el programa de correo corriendo sobre una workstation Andreas y el servidor está corriendo sobre el host Berlín, ver figura III.3. Por supuesto, el cliente podría ser una bomba de gas, un robot o un lector de código de barras. En todos los casos, los conceptos deberán ser los mismos.

El programa cliente se ejecuta como un proceso en la workstation del usuario, proviendo una gráfica y una interface que sirve como respuesta al usuario.

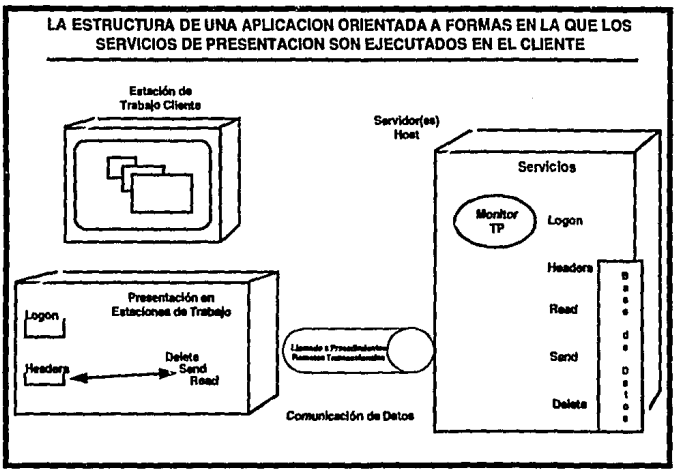

FIGURA III.4.- LA ESTRUCTURA DE UNA APLICACION ORIENTADA A FORMAS EN LA QUE LOS SERVICIOS DE PRESENTACION SON EJECUTADOS EN EL CLIENTE. El programa cliente fue generado por un generador de aplicación. El cliente genera accioes de llamados a procedimientos remotos para el Monitor de PT corriendo en el computador Host compartido que almacena los buzónes y mensajes. El Monitor de PT lanza uno u otros servicios para ejecutar las operaciones requeridas nuevamente en la B.D. Por esto, el host podría estar geográficamente distribuido (como en la figura III.3.), y el cliente podría almacenar parte de la B.D. y ejecutar algunas funciones de servicios local.

Este aspecto de la aplicación llamada servicios de presentación, recolecta datos y navega a través de las formas. Si los accesos a otros servicios es requerido, el cliente pasa el requerimiento a un servidor corriendo sobre un host remoto o local, como se muestra en la figura III.4.

El requerimiento al servidor (servicio) llega o arriba al host como un mensaje dirigido al Monitor de PT para este host. El Monitor de PT tiene una lista de servicios registrados con él.

En el ejemplo del sistema de correo estos servicios son funciones de aplicaciones específicas, tales como una petición de conexión al sistema de correo, leer un mensaje, borrar un mensaje, enviar un mensaje, etc. Cuando un servicio de requerimiento llega o arriva, el Monitor de PT busca el servicio en el repositorio y verifica que el cliente esté autorizado para usar este servicio. El Monitor de PT crea entonces un proceso para ejecutar este servicio pasando el texto del requerimiento inicial al proceso servidor. El proceso servidor ejecuta entonces esta función, accesando de la B.D. y coversando con el proceso cliente. Si el servicio es muy simple, lo accesa a la B.D. contesta y termina. Si el servicio es más complejo lo contesta y espera para requerimientos más fuertes del cliente. Un servicio podría llamar a otros servicios y podría implicar servicios de acceso remoto por acceso de datos almacenados en los centros de cómputo remotos.

El código cliente fue probablemente construído con un generador de aplicaciones que producen una subrutina para cada una de las formas entrada/salida mostrada en la figura III.2. El diseñador dibuja cada forma, definiendo campos de entrada, campos de salida y botones que navegan a otras formas. Por ejemplo la forma logon (solicitud o petición de acceso al sistema) recoge el nombre del cliente y el password, entonces lo pasa al host, invocando el servidor de correo, el servidor de logon vía un llamado de procedimiento remoto *"RPC"* (Remote Procedure Call). El servicio prueba el password y regresa el resultado de la prueba al cliente. Si el logon es satisfactorio, el servicio tendrá distribuido un contexto para el cliente en el servidor, para que las subsecuentes llamadas puedan rápidamente referirse a los apartados postales de los clientes y el estado generado en el host. La forma logon invoca entonces la forma del encabezado, la cual está en turno del servicio encabezado en el host del servidor de correo para enumerar la reciente entrada y salida de mensajes; como el código cliente está representado como la caja logon en la workstation de la figura III.2.

Esto es escrito en un lenguaje de programación con sintaxis estándar y control de flujo, pero también tiene las funciones de manejo de pantalla con presentación de servicio (los verbos ACCEPT y DISPLAY), las funciones de llamado a procedimientos remotos son los que permiten invocar a cualquier nodo en la red (el verbo ALL) y los verbos transacción que no permiten declarar términos de transacciones (verbo Begin y Commit).

El lenguaje de programación cliente es un mundo persistente. Esto significa que si el proceso cliente falla o si aborta la transacción, el proceso y el estado persistente reiniciarán o están reiniciando desde el comienzo de la transacción, por ejemplo, si la falla es de potencia y el procesador se reestablece, entonces el proceso cliente se recreará (volver a crear) como si la transacción simplemente abort. Los lenguajes de programación persistentes amarrán el estado de la aplicación junto con el estado de la B.D. y el estado de la red, haciendo cambio a todos los aspectos del estado atómico todo o nada.

El código servidor podría también haber sido producido por un generador de aplicaciones. Por ejemplo el servidor logon tiene para verificar que el password del usuario es correcto y entonces abre el apartado postal del usuario produciendo un token (carácter de identificación), que identificará este buzón del cliente en subsecuentes llamados. Esta aplicación lógica es también simple y puede ser un patrón presente en el sistema. Si no, el diseñador de aplicaciones tendrá que escribir unas cuantas líneas de código SQL y C para ejecutar estas funciones.

LLAMADO A PROCEDIMIENTOS REMOTOS TRANSACCIONALES

Un mecanismo de llamado a procedimientos es la forma para que un programa llame a otro.

Este mecanismo es parte de un modelo de Comunicaciónes. Un Modelo de Comunicaciones describe un conjunto abstracto de operaciones que permite la comunicación entre procesos y proporcióna una metodología para desarrollar programas distribuidos.

Los Modelos de Comunicaciónes permiten analizar, explicar y diseñar nuevas formas de comunicación entre procesos ***"IPC"*** (Interprocess Communications).

Un programa distribuido se compone de un conjunto de procesos distribuidos que interactúan de acuerdo con un modelo de comunicación.

En forma generalizada existen actualmente dos tipos de modelos de comunicación:

1).- Las primitivas de comunicación por mensajes.
2).- Los llamados a procedimientos remotos "RPC" (Remote Procedure Call).

Ambos modelos de comunicación conforman la base de las estructuras de control distribuido, tanto de Lenguajes de programación distribuida como de las Interfaces de programación.

En general, todos los procedimientos son parte de un espacio de dirección y procesos. Los llamados a procedimientos remotos (RPC), son una forma para que un proceso invoque un programa en otro proceso remoto, aunque la subrutina en el proceso remoto haya sido local. Lo manifiestan; el llamador y el llamado, estos son ambos en el mismo proceso; los RPCs buscan un llamado a procedimientos locales ***"LPC"*** (Local Procedure Call). Los RPC son una generalización del llamado de procedimiento ordinario, con la diferencia de que el llamante y el llamado se ejecutan en sitios distintos y geográficamente dispersos.

Los RPC no necesariamente tienen que ser remotos. En un esquema, cliente/servidor pueden estar en el mismo computador y todavía usar la misma interface para comunicarse. Los llamados a procedimientos locales o remotos (LPC o RPC) tienen la misma interface, así que el programa cliente se puede mover a cualquier lugar en la red y operar correctamente. El cliente y el servidor no necesitan conocer cada una de las otras localidades. Localizar el cliente en un site (centro de cómputo) en particular, es una decisión de confiabilidad y autorización. Si el cliente interactúa pesadamente con dispositivos de entrada/salida, entonces el cliente debe estar cerca de la fuente de datos; si el cliente interactúa más pesadamente con un simple servidor, entonces el cliente probablemente debe estar cerca del servidor.

Dadas estas definiciones aproximadas de llamados a procedimientos, ahora se definirá el concepto de llamado a procedimiento remoto transaccional ***"TRPC"*** (Transactional Remote Procedure Call). Esta es una manera para combinar el trabajo de varios clientes y servidores en una simple unidad de ejecución ACAD. La aplicación declara el límite de la transacción por el llamado begin_work ( ). Esto arranca una nueva transacción y crea un identificador de transacción única, mejor conocido como (trid). Una vez que una transacción es arrancada, todas las operaciones realizadas por el cliente son etiquetadas por el identificador de transacciones. El identificador de transacciones es enviado con todos los requerimientos de servicios, tal que la operación de estos servicios está también dentro de la vista de la transacción. Cuando la transacción hace commit, todas las transacciones que están participando del servicio también hacen commit sus operaciones. El commit lógico es implementado por los monitores de PT de cada uno de los nodos participantes. Todos ellos van a través de un protocolo de ceremonia de compromiso (Protocolo de compromiso en dos fases: ***"2PC"*** Two Phase Commit) coordinado por un proceso confiado.

El protocolo de compromiso en dos fases: 2PC: Es un protocolo que asegura el compromiso atómico de transacciones distribuidas. Cuando las acciónes de una transacción (generadas en un sitio coordinador) involucran varios sitios (participantes), es necesario sincronizar los procesos interactuantes, ya que cada sitio podría tener sus razones para no comprometer la transacción.

Volviendo al protocolo 2PC, pregunta: "Hace commit"?, respuesta: "Si hago". Pregunta: "El está committed", respuesta: "grandioso". Si cualquier cosa va mal, todas las cosas hacen rollback o abort (significa que la transacción abort) al arranque de la transacción.

En los sistemas transaccionales, los servidores son generalmente llamados manejadores de recursos. Un manejador de recursos son los datos, código y procesos que proveen acceso a algún dato compartido.

Los Sistemas de PT lo hacen fácil para construir e invocar manejadores de recursos. RPC es el estándar vía un programa de aplicación para invocar un manejador de recursos.

El llamado a procedimientos remotos transaccional "TRPC" es semejante a un llamado a procedimientos remotos "RPC", excepto que lo propaga el identificador de transacciones del cliente, con los otros parámetros, al servidor. El mecanismo TRPC para el servidor registra el servidor como parte de las transacciones del cliente. Figura III.5. muestra el flujo de mensajes a lo largo de los clientes y servidores en una transacción típica. Cada uno de los llamados a procedimientos locales y remotos en la figura III.5. están etiquetados con el identificador de transacciones del cliente.

FALLAS

Existen otros casos de fallas como por ejemplo, si un usuario oprime la tecla de cancel, o si el servidor decide que la entrada está mal o si algo también falla. Si la falla ocurre después de que se completa la operación entera, entonces el cliente y el servidor deberán recordar los cambios de estado y ambos deberán regresar al estado final si la falla ocurre antes de completar la operación, entonces algunos manejadores de recursos podrían no ser capaces de commit la transacción. Para estar seguros el cliente y el servidor deberán regresar a sus estados como al inicio de la transacción.

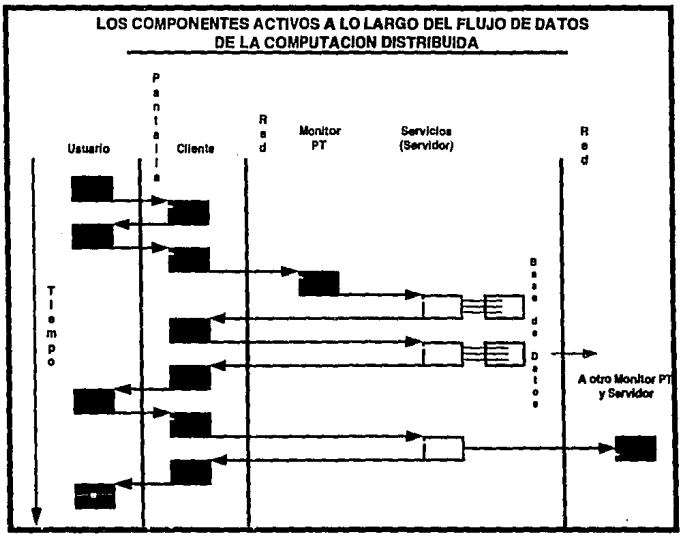

FIGURA III.5.- LOS COMPONENTES ACTIVOS A LO LARGO DEL FLUJO DE DATOS DE LA COMPUTACION DISTRIBUIDA. Mostrados en la figura III.4. El usuario interactúa con la interface de pantalla, el cliente lee, escribe y se comunica con los servidores vía llamado a procedimientos remotos RPC (como se muestra en las líneas con negrillas). La invocación inicial de los servicios es ejecutada por el Monitor de PT que crea el proceso servidor. El servidor entonces interactúa con la B.D. y el cliente. El servidor podría también hacer llamadas más poderosas a otros servidores y transparentemente accesar datos remotos vía el sistema de B.D. Esta figura muestra un cliente llamando a un simple servidor pero un cliente podría invocar varios servidores dentro de una transacción. Las interacciones cliente/servidor pueden ser estructuradas como una simple transacción, como una secuencia de transacciones.

Todas las actualizaciones subsecuentes deberán ser destruidas, la ventana en la estación de trabajo deberá regresar a sus estados originales y la B.D. deberá regresar al estado original, esta es la propiedad de Atomicidad.

El Monitor de PT automáticamente ejecuta este reset (reinicio) o reconstrucción lógica, en caso de falla. El programa de aplicación (cliente) declara el inicio de una computación que es para ser tratada como una unidad ACAD, por consecuencia de un verbo Begin_Work ( ). Cuando la computación es completada (cuando es una transformación correcta), el cliente emite un verbo commit_work ( ). Todas las acciónes de la transacción entre el begin_work ( ) y el verbo commit_work ( ) se tratará como un simple grupo ACAD. Si la computación falla antes del commit_work ( ), entonces el estado del sistema será regresado a la situación como la de Begin_work ( ). En particular, el cliente, los servidores, las pantallas y la B.D. serán todas reseteadas (reíniciadas) a ese estado. Esto simplifica el manejo de excepciones para el diseñador de aplicaciones. Todo este borrado de trabajo es automático.

III.1.4.- VISION DEL MANEJADOR DE RECURSOS DE UN SPT

Nosotros hemos visto que de esta manera un Sistema de Procesamiento de Transacciones incluye una colección de subsistemas, llamado manejador de recursos que juntos proveen las transacciones ACAD. Como Generadores de Aplicaciones, interfaces de usuario, lenguajes de programación y sistemas de B.D. proliferan y evolucionan, los manejadores de recursos están continuamente siendo agregados a los SPT.

El SPT debe ser extensible en el sentido que lo debe hacer fácil para agregar tales manejadores de recursos. Existen Sistemas de Bases de Datos (por ejemplo: DB2[1], Rdb[2], Oracle, Ingres, Informix, Sybase), muchos Lenguajes de Programación (por ejemplo: COBOL[3], FORTRAN, PL/I, Ada, C, C++), muchas redes (por ejemplo: SNA[4], OSI, TCP/IP[5], DECnet, LAN Manager), muchos manejadores de presentaciones (por ejemplo: X-Windows, News, PM, Windows) y muchos generadores de aplicaciones (por ejemplo: CPS, Cadre, Telos, Pathmaker). Cada cliente busca seleccionar un subgrupo random de este menú y construir una aplicación de esta base. Un SPT facilita esta selección random. El SPT provee una manera para interconectar aplicaciones y manejadores de recursos, mientras provee las propiedades ACAD para la computación entera.

La llave para esta interoperabilidad de recursos es una manera estándar de invocar servicios de aplicaciones y manejadores de recursos. Tales mecanismos deben permitir invocar ambos servicios tanto locales y remotos. El mecanismo estándar para hacer esto, es un llamado a procedimientos remoto (RPC). El TRPC es una extension del llamado a procesamiento simple. Esto permite el trabajo de muchos llamados y servidores ser coordinados como una unidad ACAD.

El resultado de la estructura de computación busca una semejanza gráfica, o árbol, de procesos que se comunican con otro vía Llamados a Procedimientos Remotos Transaccionales. El SPT provee los mecanismos de liga necesarios para unir el cliente a los servidores. La interface podría ser local o remota.

LOCAL

Si el cliente y el servidor están en el mismo espacio de dirección o proceso, en este caso el código del servidor es destinado al código del cliente y la invocación es un simple llamado a procedimientos.

[1]DB2.- Es un sistema SQL disponible en MVS de IBM. Actúa como Manejador de Recursos para IMS y CICS.
[2]Rdb, Oracle, Ingres, Informix y Sybase.- Son un sistema SQL portable, disponible en muchas plataformas. Actúan como Manejadores de Recursos para el SPT.
[3]COBOL, FORTRAN, PL/I, Ada, C, C++.- Lenguajes de Programación.
[4]SNA, OSI, TCP/IP, DECnet, LAN Manager.- Redes y Protocolos de Comunicaciones.
[5]CPS, Cadre, Telos, Pathmaker.- Generadores de Aplicaciones.

REMOTO

Si el cliente y el servidor están en un proceso diferente, en este caso la invocación consiste un mensaje enviado por el cliente al servidor y el servidor enviando una respuesta al cliente. Este es un llamado a procedimientos remoto (RPC).

El llamador no puede distinguir un llamado a procedimientos locales de un llamado a procedimientos remotos; ellos tienen la misma sintaxis. Los llamados locales generalmente tienen mejor performance, pero los llamados remotos permiten computación distribuida y a veces proveen mejor protección del servidor desde el cliente. La figura III.6. ilustra la estructura típica del llamado de una aplicación invocando varios servicios de aplicación y varios manejadores de recursos. Cada caja es opcionalmente un proceso, y la conexion a través de las cajas es vía un llamado a procedimientos locales o remotos "RPC o LPC".

Por otro lado manejando, la creación e intercomunicación de procesos ejecutando la transacción, cada Monitor de PT tienen un grupo de servicios principales que los proveen al manejador de recursos. Estos servicios ayudan al manejador de recursos para implementar operaciones ACAD y proveen ejecuciónes sobre todo de un control de programa de aplicaciones que invocan los manejadores de recursos individuales.

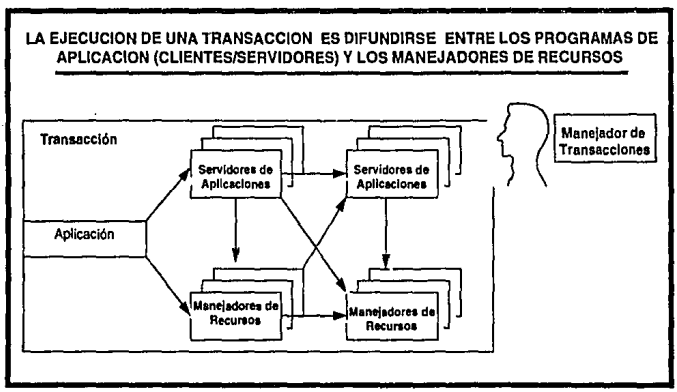

FIGURA III.6.- LA EJECUCION DE UNA TRANSACCION ES DIFUNDIRSE ENTRE LOS PROGRAMAS DE APLICACION (CLIENTES/SERVIDORES) Y MANEJADORES DE RECURSOS. Los servicios y manejadores de recursos pueden ser invocados por llamados a procedimientos locales o remotos (LPC o RPC). Los servicios locales y servidores están ligados al espacio de direcciónes del llamador, mientras algunos RPC o LPC corren en procesos, quizas sobre sistemas de computadores remotos. La computación entera ocurre dentro de la visión de una simple transacción. El manejador de transacciones monitorea el progreso de transacciones, conexión clientes a servidores, y coordina el commit y rollback de transacciones.

Este básico control de flujo de una aplicación es diagramado en la figura III.7. El begin_work ( ) arranca la transacción,.registrándolo con el manejador de transacciones y creando un único identificador de transacciones. Una vez que la aplicación ha arrancado una transacción, esto puede comenzar a invocar al manejador de recursos, leyendo y escribiendo en la terminal; así como en la B.D. y enviando requerimientos a los servicios locales y remotos.

Cuando un manejador de recursos obtiene el primer requerimiento asociado con la transacción, lo asocia la transacción, diciéndole al manejador de transacciones local, que éstos quieren participar en las operaciones de commitment y rollback de la transacción. Esto es típico por varios manejadores de recursos para asociar la transacción. Como estos manejadores de recursos ejecutan trabajo en beneficio de la transacción, los mantienen listos de los cambios que han hecho a los objetos. Como una regla, ellos registran tanto los viejos y nuevos valores del objeto. El SPT provee un servicio de petición de acceso (logging) para registrar estos cambios. El manejador de bitácora (log manager) eficientemente implementa un archivo secuencial de todas las actualizaciones de transacciones a objetos. Por supuesto, los otros manejadores de recursos tienen que decirle al manejador de bitácora que estos son actualizados.

Para proveer aislamiento, los manejadores de recursos aseguran (lock significa un candado de seguridad) los objetos accesados por la transacción; éstos previenen otras transacciones desde la visión de las actualizaciones uncommitted de esta transacción y las previenen a éstas de alterar los datos de lectura o escritura por esta transacción uncommitted. El SPT provee un manejador de lock que otros manejadores de recursos pueden usar.

Cuando la transacción emite commit_work ( ), el manejador de transacciones ejecuta el protocolo 2PC. Primero, todos los queries de los manejadores de recursos están asociados a la transacción, preguntando si piensan que la transacción es una consistente y completa transformación. Cualquier manejador de recursos puede votar no, en este caso el commit falla. Pero si todos los manejadores de recursos votan sí, entonces la transacción es una transformación correcta y el manejador de transacciones registra este acto en el log, informando a cada uno de los manejadores de recursos que la transacción está completa. Para este punto los manejadores de recursos pueden liberar los locks y ejecutar cualquier otra operación necesaria para completar la transacción.

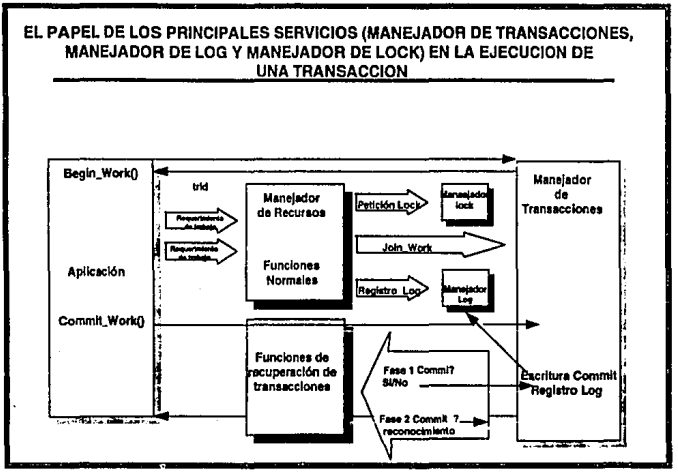

FIGURA III.7.- EL PAPEL DE LOS PRINCIPALES SERVICIOS (MANEJADOR DE TRANSACCIONES, MANEJADOR DE LOG, Y MANEJADOR DE LOCK) EN LA EJECUCION DE UNA TRANSACCION. El planeador de la transacción (no se muestra) coloca los procesos que ejecuta la transacción.

Si la transacción debe fallar durante la ejecución, o si un manejador de recursos no vota no durante la fase 1 del 2PC, entonces el manejador de transacciones instrumenta la transacción de rollback. En este caso, el manejador de transacciones lee el log de las transacciones y para cada registro log invoca, el manejador de recursos que escribío el registro, diciéndole al manejador de recursos que deshaga la operación. Una vez que el *undo scan* (deshacer la búsqueda) es completa, el manejador de transacciones invoca cada manejador de recursos que asoció la transacción y dice que la transacción fue aborted (abortado).

El manejador de transacciones también instrumenta la recuperación de la transacciones si un nodo o site falla. Esto proveé servicios genéricos para la falla de un simple objeto, la falla de un manejador de recursos y la falla de un site completo. Los siguientes párrafos dan un bosquejo de cómo el manejador de transacción ayuda en la recuperación de los sistemas.

Si un site falla, el SPT reinicia todos los manejadores de recurso. Varias transacciones podrían haber estado en progreso en el tiempo de la falla. El manejador de recursos contacta al manejador de transacciones como parte de su lógica de reinicio.

Para este tiempo el manejador de transacciones informa a ellos del resultado de cada transacción que estuvo activa en el tiempo de la falla. Algunos podrían haber committed, algunos podrían haber abortd y algunos podrían todavía estar en los proceso de committing. El manejador de recursos puede recuperar estos estados committed independientemente o puede participar en el *undo y redo scan* (deshacer y rehacer la búsqueda) del log del manejador de la transacción.

Si un manejador de recursos falla pero el resto del SPT continúa operando, el manejador de transacciones aborts todas las transacciones uncommitted (sin compromiso)involucradas con este manejador de recursos. Cuando el manejador de recursos regresa a servicio, el manejador de transacciones informa al manejador de recursos acerca de los resultados de esas transacciones. El manejador de recursos puede usar esta información y el log de transacción para reconstruir este estado.

Si un objeto en particular es perdido pero el manejador de recursos y es por otro lado operaciónal, entonces el manejador de recursos puede continuar ofreciendo servicio sobre otros objetos, mientras el objeto fallado es reconstruído de una copia de archivo y en el log quedan registrados los cambios committed a esta copia. El manejador de transacción y el manejador de log ayudan a recuperar de una copia archivo del objeto.

Cada site usualmente tiene un manejador de transacciones separado. Esto permite a cada site operar independientemente de los otros, proviendo autonomía local. Cuando la ejecución de las transacciones es distribuida a través de varios sites, esto es distribuido a través de varios manejadores de transacciones. En este caso, el protocolo 2PC para múltiples procesos generaliza facilmente para múltiples manejadores de transacciones.

El modelo definido por el consorcio X/OPEN está dibujado en la figura III.7. Asume éste, que cada manejador de recursos tiene un log privado y un manejador de lock privado, y X/OPEN asume que los manejadores de recursos ejecutan sus propios rollback, basados en un simple llamado de rollback del manejador de transacciones en caso de que la transacción aborts. El más simple resultado del dibujo se muestra en la figura III.8.

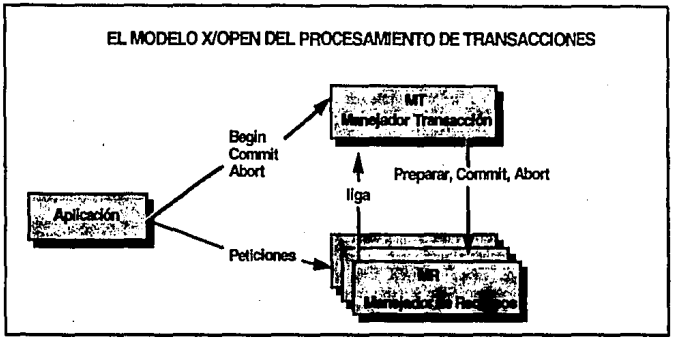

FIGURA III.8.- EL MODELO X/OPEN DEL PROCESAMIENTO DE TRANSACCIONES. Como en la figura III.7, la aplicación arranca un manejador de transacciones por el gestor de transacciones local. Como la aplicación invoca los manejadores de recursos, éstos acoplan la transacción. Cuando la transacción commit o abort, el manejador de transacciones envía un mensaje de salida al manejador de recursos. En este modelo, el manejador de recursos tiene cerrada privada (lock) y el manejador de transacciones no provee una búsqueda de la bitácora de transacción.

Las figuras III.7. y III.8. describen las transacciones centralizadas involucrando en un manejador de transacción simple. Cuando el manejador de transacciones es involucrado, la figura viene más complicada ya que el manejador de la transacción debe cooperar en el commit de la transacción. La figura III.9. diagrama este diseño.

III.1.5.- SERVICIOS PRINCIPALES DEL SPT.

Los principales servicios del Monitor de PT, son los siguientes:

TRANSACCIONAL REMOTE PROCEDURE CALL (TRPC)
Programa, autoriza e invoca la ejecución de servicios (server).

MANEJADOR DE TRANSACCION.- Instrumenta el commit y rollback de las transacciones así como la recuperación de objetos, manejadores de recursos o site, despues de que éstos fallan.

MANEJADOR DE BITACORA (LOG)
Registra un log de cambios hechos por transacciones, así como una versión consistente de todos los objetos puede ser reconstruida en caso de falla.

MANEJADOR DE SEGUROS (LOCKS)
Provee un mecanismo genérico para regular accesos concurrentes a objetos. Estos manejadores de recursos ayudan a proveer aislamiento de la transacción.

Los manejadores de recursos extienden el SPT para el uso de estos servicios principales. Generalmente, esto es un separado SPT para cada site o cluster de una red de computadoras. Estos monitores de PT cooperan para proveer un ambiente de ejecución distribuida al usuario.

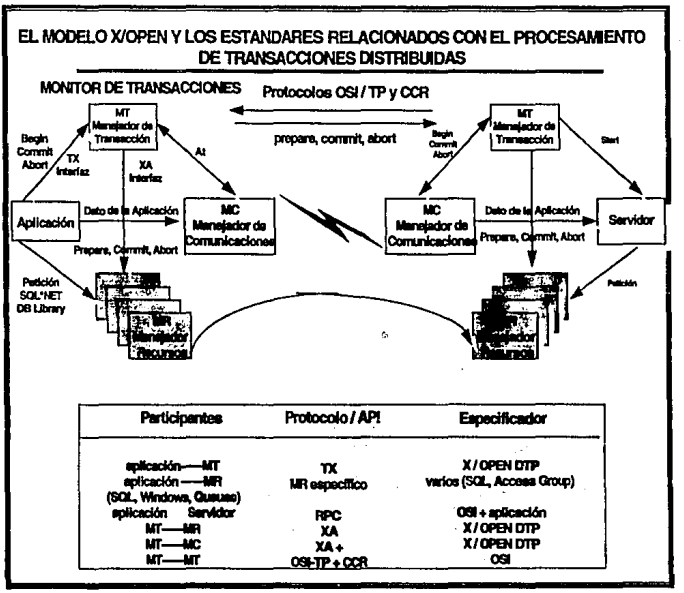

FIGURA III.9.- EL MODELO X/OPEN Y LOS ESTANDARES RELACIONADOS CON EL PROCESAMIENTO DE TRANSACCIONES DISTRIBUIDAS. Como en la figura III.7 y III.8, la aplicación arranca un manejador de transacciones por el manejador de transacciones local. Cuando la aplicación o algunos manejadores de recursos actúan en beneficio de la aplicación, hacen un requerimiento remoto; el manejador de Comunicaciones informa para cada uno de los nodos sus manejadores de transacciones local de la entrada o salida de transacciones. El manejador de transacciones para cada uno de los nodos maneja el trabajo de la transacción por ese nodo. Cuando la transacción commit o abort, el manejador de transacciones coopera para proveer el commit, atómico y durable.

IV.- LISTA DE CARACTERISTICAS DE UN SPT.

Más que una visión del sistema desde la perspectiva de un usuario particular, se pueden observar las características típicas de un SPT. La más obvia de estas características son el diseño de aplicaciones y las herramientas de generación que, a su vez, son construidas alrededor del **Repositorio**. Los Generadores de Aplicaciones y el Repositorio son un Sistema de B.D. usados para almacenar datos persistentes (a veces una B.D. SQL), y un Sistema de Comunicación de Datos usado para proveer una interface abstracta a terminales, workstations, y otros dispositivos de entrada/salida y solicitantes remotos. Detras de este arreglo B.D. - Comunicación de datos (B.D.-BC) esta el **Monitor de PT** esto mismo, con los servicios principales. En la periferia de estas características principales son herramientas auxiliares usadas para administrar, afinar y operar el sistema. Por supuesto, todas corren sobre un substracto consistente de un sistema operativo, una red y un hardware de computadora. La lista de características se enfoca sobre una Generación de Aplicaciones, el Repositorio, la B.D., la Comunicación de Datos, Manejador de Transacciones y Operaciones.

IV.1.- CARACTERISTICAS DEL DESARROLLO DE APLICACIONES

El objetivo y compromiso de los Generadores de Aplicaciones es automatizar la programación por el translado de diseños directamente dentro de los Sistemas Ejecutables. Estos grandes sucesos han sido en el área de Procesamiento de Transacciones. Los proyectos típicos reportan que el 90% de los clientes y servidores son automáticamente generados vía una interface de programación gráfica punto y clik. Los Generadores de Aplicaciones permiten rápidamente hacer prototipos, reduciendo riesgo en los proyectos y mejorando el uso de los sistemas. Lo que realmente esto significa, es que los Generadores de Aplicaciones son capaces de generar la mayoría de las funciones genéricas de una aplicación desde una caja de herramientas estándar, por esto libera el resultado complejo de dominio específico (tales como descuentos, códigos de impuestos, y programas de trabajo), para la programación convencional. Este recidual o código principal es usualmente escrito en un lenguaje de programación estandar (Cobol, C, Fortran, SQL).

En suma, éstas son herramientas que analizan una descripción esquemática de un diseño de B.D. y últimamente producen un optimizado grupo de estados de definición de datos de SQL para representar la B.D. La ayuda de diseño de pantallas permite al diseñador pintar las pantallas asociadas con cada transacción. Las gráficas resultantes son transladadas a programas que leen y escriben tales formas. Una vez que la aplicación ha sido generada de esta forma, las herramientas son provistas para popularizar la B.D. con datos sintetizados y generar una carga sintetizada sobre el sistema. Las herramientas de depuración son usadas para probar y certificar las correcciones de los programas.

La mayoría de estos sistemas proveen un sistema arrancador, una aplicación simple que no obstante incluye totalmente compresion administrativa y operaciones de funciones, un sistema de seguridad, un sistema de contabilidad y procedimientos de operaciones. Estas herramientas preprogramadas son entonces aplicables a nuevas aplicaciones construidas con el generador de aplicaciones, éstas también direccionan la importancia del entrenamiento resultado del operador y administrador.

IV.2.- CARACTERISTICAS DEL REPOSITORIO

El repositorio mantiene descripciones de los objetos en el sistema y registra la interdependencia a través de estos objetos. Los repositorios son también llamados diccionarios o catálogos. Los catálogos de el sistema de B.D. SQL proveen un ejemplo concreto de un repostorio. Estos rastrean las tablas de la B.D., índices, vistas, y programas. El catálogo del sistema SQL también rastrea las dependencias a través de estos objetos, por ejemplo este programa usa estas vistas, que en turno usa esos índices y tablas. Cuando alguien cambia una tabla, el sistema SQL automáticamente recompila todos los programas afectados. Haciendo antes los cambios, el administrador puede correr un reporte estándar para desplegar el programa que será afectado por el cambio. Todo esto es inherente en catálogos SQL. La deficiencia de este ejemplo es que el sistema SQL es solamente enterado de la parte de B.D. de la aplicación. Si el cambio debía propagarse al cliente, el sistema SQL no tendrá conocimiento acerca de esto.

Lo necesario para un repositorio está implícito en la figura III.3. El sistema típico tiene miles de terminales, y usuarios y cientos de aplicaciones, pantallas, tablas, reportes, procedimientos, y copias de archivo de estos objetos. Cuando las versiones viejas de aplicaciones, procedimientos y otros objetos son considerados, el número de objetos crece más halla de diez mil. Recordar ésta información es difícil y cambiarlos es un error, porque cualquier cambio involucra cambiar muchos componentes del sistema. Por ejemplo agregar un campo a una pantalla, altera la pantalla, probablemente altera el software del cliente y la interface del servidor y probablemente agregar una columna a una tabla. Todos estos cambios deben ser hechos juntos (como una unidad ACAD). El cambio deberá ser expresado una vez y deberá automáticamente ser propragado a cada consumidor del objeto. Si los cambios no trabajan fuera, entonces todos ellos deberán ser regresados, tal que el sistema regrese al estado previo.

Diseño.- captura todas las decisiónes de diseño de la aplicación en una B.D.
Dependencias.- captura todos los diseños de la aplicación y modulo de dependencia en una B.D.
Cambio.- expresa los procedimientos de cambio como transacciones sobre esta B.D. Un buen repositorio necesita estar completo.

Lo necesario para describir todos los aspectos del sistema no así la B.D., los programas o la red. El repositorio almacena el sistema de materiales. Para estar completo el repositorio debe ser extensible, esto debe ser posible para agregar nuevos objetos, relaciónes, y procedimientos. El diseñador de aplicaciones debe ser capaz de definir un nuevo tipo de objetos sin reimplementar el repositorio. La prueba real de extensibilidad es si el repositorio tiene un núcleo que implementa los objetos del repositorio, con extenciones que implementan los otros objetos (tales como pantallas, tablas, clientes y servidores), por ejemplo, los catálogos SQL deben ser una extensión del repositorio. Si el vendedor ha extendido el repositorio en esta forma, usando el mecanismo de extención interconstruído del repositorio, entonces del diseñador de aplicaciones probablemente será capaz de hacer extenciones comparables usando el mismo mecanismo.

Un repositorio deberá ser activo; esta es la descripción del objeto que debe ser consistente con el estado actual del objeto. Por ejemplo si un dominio es agregado a un archivo, o si un campo es agregado a una pantalla, entonces la descripción del objeto deberá reflejar este cambio; los repositorios sin esta propiedad son llamados pasivos.

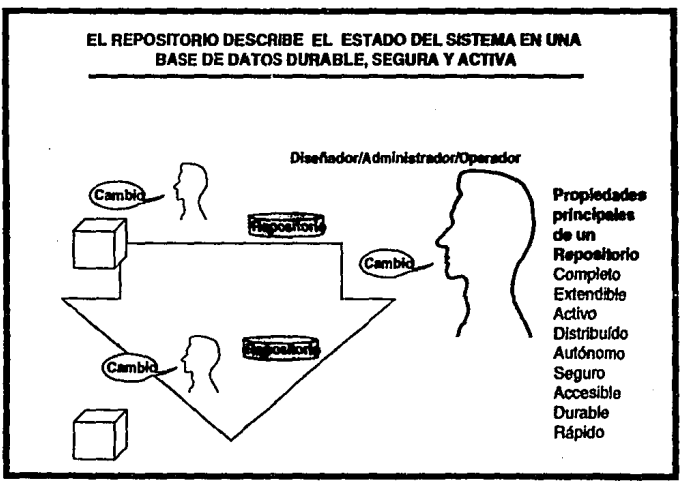

FIGURA IV.1.- EL REPOSITORIO DESCRIBE EL ESTADO DEL SISTEMA EN UNA B.D. DURABLE SEGURA Y ACTIVA. Esta B.D. es cambiada por las transacciones que altera el repositorio, por medio del estado del sistema. La propiedades deseables de un repositorio están listadas a la derecha.

IV.3.- CARACTERISTICAS DEL MONITOR DE PROCESAMIENTO DE TRANSACCIONES

El Monitor de PT provee un ambiente de ejecución para manejadores de recursos y aplicaciones. Cuando los requerimientos llegan de un cliente local o remoto, el Monitor de PT lanza un servidor a ejecutar el requerimiento. Si el requerimiento llega con un identificador de transacción, entonces el servidor se convierte en parte de esta transacción pre-existe. Si el requerimiento no es está listo para transacciones protegida, entonces el Monitor de PT o servidor arranca una nueva transacción para cubrir la ejecución del requerimiento. Así, el Monitor de PT provee un mecanismo RPC transaccional.

Antes de crear un servidor para un requerimiento, el Monitor de PT autoriza el cliente al servicio. El cliente usualmente ha sido autentificado por el Monitor de PT como una persona particular o grupo de personas (todos los empleados del almacén). El Monitor de PT verifica que el cliente está autorizado para invocar este servicio. Esta autorización verifica este estado como una función del grupo de clientes, localidad del cliente y tiempo. Por ejemplo los empleados del almacén local podrían correr durante la transacción invocada en horas de oficina. Opcionalmente el Monitor de PT registra la verificación de seguridad o violación de seguridad en una pista de auditoría.

Si el cliente está autorizado para invocar el servicio, entonces arrancando el servicio se vuelve un programa de decisión. Si el sistema está congestionado, el arranque podría ser retardado. El Monitor de PT podría tener un prelocalizado *pool* de procesos servidores y podría asignar los requerimientos sobre demanda. Este pool de procesos es llamado un servidor de clase. Si todos los miembros del servidor de clase están ocupados, entonces el requerimiento debe esperar, o un nuevo proceso servidor debe ser asignado. El Procesamiento de Transacciones este asignado el requerimiento debe esperar, o un nuevo proceso de servidor debe ser asignado.

El Monitor de PT puede crear un servidor, poner el requerimiento en un queue (cola) del servidor de clase, o igual poner el requerimiento en una cola para que el servicio sea atendido más tarde (a media noche). El balanceo de cargas y la planeación del servidor de clases es la principal responsabilidad del Monitor de PT.

Una vez que el proceso es programado y está ejecutando el requerimiento, el servicio puede invocar otros manejadores de recursos y otros servicios en turno, que asocian la transacción. El manejador de transacciones sigue la pista a todos los servidores y manejadores de recursos asociados a la transacción y los invoca para hacer un commit y rollback de la transacción. Si la transacción es distribuida a lo largo de muchos sites de una red de computadoras, el manejador de transacciones media la transacción *commitment* con los manejadores de transacciones operando en otros sites.

La visión de la aplicación de transacciones es que una transacción consiste de una secuencia de acciones en medio de paréntesis por un par de begin_commit o begin_rollback. El begin asigna un identificador de transacción (trid). Cada una de las subsecuentes acciones es una operación ejecutada por algún manejador de recursos como ambos un llamado a Procedimientos Remotos o Locales (RPC - LPC). Este llamado de procedimientos implícitamente lleva el trid. Todas las operaciones sobre datos recuperables generan información (datos del log), que permiten la transacción hacer rollback (desecha) o atómicamente y committed durablemente. En adición, para aislar la transacción de actualizaciones concurrentes por otras, cada manejador de recursos ordinariamente adquiere locks sobre los objetos accesados por la transacción. El trid es la etiqueta usada para registro de lock y registro de log. Esto es el identificador de objetos ACAD.

Cualquier proceso ejecutando el nombre de la transacción tendrá el trid de la transacción. Si el proceso está corriendo sobre un sistema operativo orientado a transacciones, el trid es parte del estado de proceso. Si las transacciones han sido insertadas dentro del sistema operativo sobre el mecanismo RPC del Monitor de PT, entonces el trid es una variable global del proceso.

Las transacciones pueden declarar puntos de salvación, puntos dentro de la ejecución de la transacción para que la aplicación pueda más tarde rollback. Este parcial rollback crea la habilidad para servidores o subrutinas para levantar y limpiar procesos por excepción. Un punto de salvación es establecido cuando la invocación arranca. Si cualquier cosa va mal, el servicio puede rollback a este punto de salvación y entonces regresar un diagnostico. El sevidor puede decirle al cliente que la llamada falló y fue operación nula.

LOS DIFERENTES TIPOS DE EJECUCION EN TRANSACCIONES

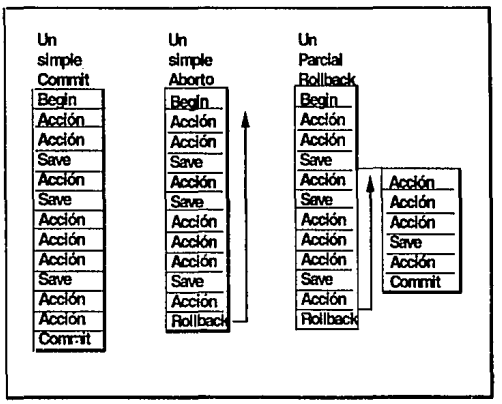

FIGURA IV.2.- LOS DIFERENTES TIPOS DE EJECUCION EN TRANSACCIONES. En los casos normales (97%) la transacción ejecuta y commit. En algunos casos (aproximadamente el 3%) la transacción abort, ya sea por llamados rollback o esto es rolledback por el sistema. En otros casos, la transacción hace un rollback parcial para un punto de salvación, intermedio y entonces continuan adelante con el proceso.

Esto provee simples errores de semántica del servidor sin abort la transacción entera. Aplicando este concepto más ampliamente, el trabajo después de un punto de salvación observa una transacción anidada dentro de otra. La substracción puede abort independientemente de la transacción origen, pero la transacción puede commit solamente si todas las transacciones anteriores commit.

IV.4.- CARACTERISTICAS DE COMUNICACIONES DE DATOS

Las Comunicaciones de Datos se consideran en el ambiente de mainframes manejando terminales tontas, (terminales que no aplican la lógica) y el mundo cliente servidor de clientes asiendo todo el manejo de presentación y manejando servidores vía TRPC's. En ambos diseños el clásico mainframe y el cliente servidor, las Comunicaciones de datos proveen una interface de programación de aplicaciones para dispositivos remotos.

El software de comunicación de datos corren en la parte alta del sistema operativo nativo y del software de la red del computador host. Estos son muchos tipos de redes de computadoras y protocolos de red. El subsistema de comunicación de datos del SPT provee una conveniente y uniforme interface de programación de aplicaciones para todos estos diferentes protocolos de red. En el modelo OSI, las Comunicaciones de datos están en el arreglo (capa 6 o 7) de aplicación o presentación.

FIGURA IV.3.- EL SUBSISTEMA DE COMUNICACION DE DATOS Y SERVICIOS DE PRESENTACION PROVEEN UNA INTEFACE UNIFORME A LOS PROTOCOLOS DE COMUNICACION DE DATOS USADOS POR LOS CLIENTES Y TERMINALES. Este protocolo deberá agregar muhas características a protocolos simples, tales como SNA/LU o TCP/IP[1], pero necesita a agregar menos funciónalidad a un stack de protocolos tal como SNA/LU6.2 o OSI/TP.

El sistema de comunicación de datos ideal provee una interface uniforme a los diferentes protocolos de Comunicaciones, así que como a las capas más altas que no necesitan estar enteradas de los diferentes estándares.

Las aplicaciones quieren una simple forma de llamado a procedimientos remotos y no quieren aprender acerca de todas las diferentes formas de nombres de red y sus transacciones (SNA/LU6.2[6], OSI/TP[8], IMS Tandem TMF, DECdtm[9], y Tuxedo todas tienen diferentes esquemas de nombrarlas). El software de Comunicación de datos provee estos arreglos, y en cómo hacerlos, proveee una interface uniforme a la mayoría de los protocolos de red. El diagrama resultante se ve algo semejante en la figura IV.3

[6]SNA.- System Network Architecture.
[7]TCP/IP.- Transmision Control Protocol/Internet Protocol
[8]OSI/TP.- Open System Interconection/Transaction Processing.
[9]DECdtm.- Un manejador de transacción integrado con el sistema operativo VMS por DEC.

IV.4.1.- COMUNICACION DE DATOS CLASICA

En un sistema de comunicación de datos clásica, la terminal es un dispositivo orientado a formas, o un lector de código de barras con poca o no lógica interna. Toda la lógica de servicios de presentación (formateo de la pantalla) es manejado por la aplicación corriendo en el host. Esto es extremadamente diverso a lo largo de los tipos de terminales y subtipos.

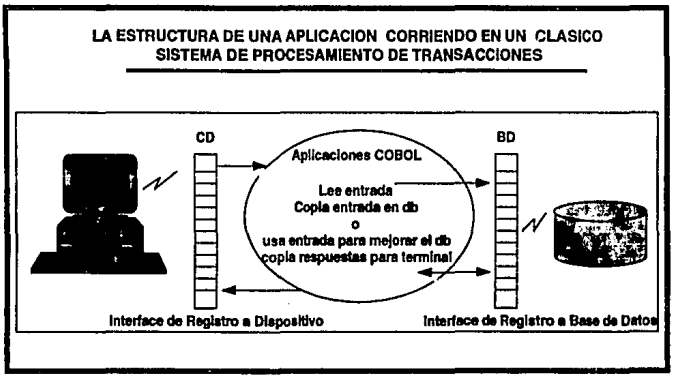

FIGURA IV.4.- LA ESTRUCTURA DE UNA APLICACION CORRIENDO EN UN CLASICO SISTEMA DE PROCESAMIENTO DE TRANSACIONES. El sistema B.D.-C.D. (Base de Datos - Comunicaciones de Datos) provee una interface de orientada-registro- tanto para la B.D. (disco) y las redes (terminales). La aplicación mueve registros de un lugar a otro. El detalle de los formatos terminal están a la izquierda para el sistema de Comunicaciónes de datos.

Los servicios de presentación proveen independencia de dispositivos por el uso de las técnicas diagramadas en el figura IV.6. Primero un pintor de pantallas interactivo produce una forma de descripción. El diseñador especifica como se verá en la pantalla por el pintado. El pintor de pantallas translada éstas pintadas en una descripción de forma abstracta, la cual puede ser entonces compilada para trabajar para diferentes tipos de dispositivos (por ejemplo, cada uno de los diferentes pantallas de modo caracter y cada uno de diversos sistemas de ventana). La descripción especifica como la forma ve en la pantalla en términos abstractos. En el pintado de la pantalla, el diseñador le dice las cosas que quiere "este es un encabezado", "este es una campo de entrada", "este es un campo de salida", " esta es una decoración". La integridad verifica que puede ser colocada en campos de entrada.

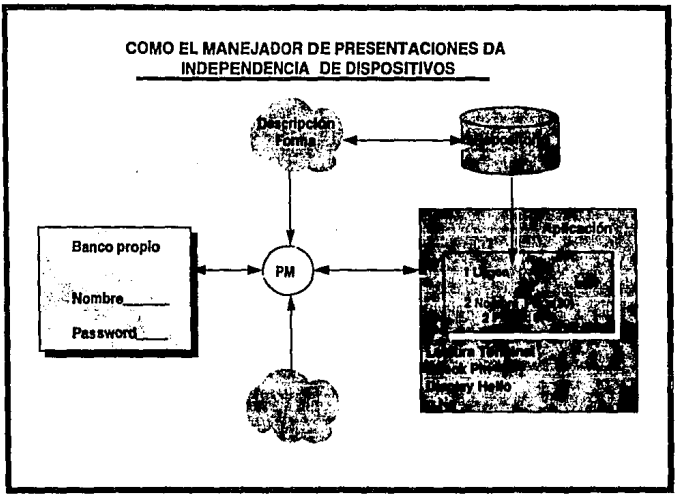

FIGURA IV.5.- COMO EL MANEJADOR DE PRESENTACION DA INDEPENDENCIA DE DISPOSITIVOS. El generdor de formas impresas en una descripción abstracta de la forma. Los campos de entrada y salida de la forma implican una entrada/salida de la estructura de registro para el programa. La forma abstracta, cuando es combina con una descripción particular del dispositivo puede ser usado para lectura y escritura del dispositivo. El repositorio almacena descripciones de las formas, kas descripciónes de los registros y las descripciones de los dispositivos.

Idealmente, todos estos están preespecificados, ya que muchas entradas son los dominios de la B.D. con tipos bien definidos y constantes de integridad. Por ejemplo, el repositorio define el dominio del número de parte y el dominio del nombre del cliente inmediatamente implicados en un completo grupo de atributos de pantalla, textos de ayuda, y verificación de la integridad.

Una vez que la forma está definida, esto implica dos registros usados para comunicar el programa de aplicación: (1) un registro de entrada consistente de los campo de entrada y sus tipos, y (2) un registro de salida que consistente de los campos de salida y sus tipos. La forma podría tener teclas de función, menús, botones u otros atributos que también aparecen como parte del registro de entrada.

Cuando la aplicación lee una forma, el manejador de presentaciones despliega la forma, recoge los datos de entrada, verifica además la integridad de las constantes de las formas, y entonces presenta los datos de entrada de la forma como un registro para la aplicación.

A la inversa, cuando la aplicación escribe un registro a la forma, el manejador de presentación toma la descripción de la forma, la descripción del dispositivo y los campos de datos de la aplicación y construye una imágen específica del dispositivo. De esta forma, el mismo programa puede leer y escribir un teletipo, una terminal en modo caracter de 24x80, una ventana de una terminal X-Windows o una ventana de una PC corriendo Windows de Microsoft. Esto es una independencia de dispositivos.

Con el servicio principal de independencia de dispositivos, el manejador de presentaciónes también provee el pintor de pantalla y una facilidad de debuggin de pantalla (depurador u optimizador). Esto maneja la forma, la pantalla, el dispositivo y registro de objetos en el repositorio. El manejador de presentaciónes es la más grande y visible parte del sistema de Comunicaciónes de datos.

IV.4.2.- COMUNICACIONES DE DATOS CLIENTE/SERVIDOR

El Sistema de procesamiento de transacciones moderno está moviendo el volumen del manejador de presentación al cliente. La interface del cliente al Monitor de PT por la negociación del transaccional RPC. En este mundo, los servicios de presentación son una parte integral del sistema cliente.

Cuando los servicios de presentación mueven al cliente, el servidor tiene control acerca de cómo los datos son acumulados o desplegados. El Monitor de PT para el servidor recibe un llamado a procedimientos remotos del cliente. Moviendo las funciones del servicio de presentación a las estaciones de trabajo, es una amplia simplificación porque la estación de trabajo tiene solamente un simple tipo de dispositivo, comunmente una ventana.

La administración del sistema todavía presenta el problema de muchos diferentes clientes, cada uno con su propio ambiente, (DOS, OS/2, UNIX, Macintosh), sus propios servicios de presentación (Presentation Manager, X Windows, Motif, OPEN LOCK, Next Step, Macintosh) y sus propias observaciones y sentimientos. El problema administrativo entonces, es todavía aún pesadilla, porque cada una de las terminales ahora han sido reemplazadas por una completa estación de trabajo compleja con una B.D. local, sistema operativo y sistema de red.

Si el sistema es administrado centralmente, los administradores pueden simplificar el número de casos por el dictado de un ambiente de grupos soportado.

En este modelo computacional cliente/servidor, los servidores exportan servicios transaccionales y las estaciones de trabajo clientes pueden romper cualquier formato de presentación que deseen. El principal trabajo del sistema de comunicación de datos Cliente servidor es proveer TRPC a esos clientes y servidores. Para una ilustración de esta situación referirse a la figura III.4.

IV.5.- CARACTERISTICAS DE LA BASE DE DATOS

Los Sistemas de B.D. almacenan y llevan cantidades masivas de datos estructurados. Estos proveen un mecanismo para definir estructuras de datos que representen objetos, y una interface no procedimental para manipular estos datos (leer y escribir). La interface no procedimental en el sentido en que un programa o las personas que accesan los datos necesitan no conocer la localidad de los datos (local, remoto, memoria principal, disco, archivo) o el detalle de la trayectoria de acceso usada para encontrar los datos, ya que la aplicación requiere operaciones sobre los registros abstractos u objetos, mientras la B.D., es responsable de mapear tales abstracciónes al hardware concretamente. La definición de los datos incluye afirmaciones acerca de los valores de los datos y las relaciónes. El sistema de B.D. exige estas afirmaciones cuando los datos están actualizados o cuando la transacción commit. Cada uno de los sistemas de B.D. actúa como un manejador de recursos y por lo tanto provee operaciones ACAD sobre los datos. Cada sistema de B.D. también proveen utilerías para manejar y controlar el uso de los datos.

Las B.D. debe separar las aplicaciones de la localización de los datos y de la representación exacta de los datos. Estas ideas *transparencia de localización* e *independencia de datos* son muy similares a la independencia de dispositivos discutido en la subsección IV.4.1.

Para facilitar la programación y las operaciones, el sistema de B.D., debería estar integrado con un repositorio y con uno o más generadores de aplicaciones. Estos generadores de aplicaciones dan una interface visual para definir B.D., manipular y controlarla, como también automatizan muchos de los procesos de diseño.

El sistema de B.D. debe permitir al diseñador colocar parte de la B.D. en diferentes lugares. Como una regla, los datos son colocados sobre computadoras cerca de la fuente de datos o de los consumidores de datos (o ambos). Esto crea datos particionados, algunas partes de las tablas están aquí y algunas partes están allá. Si los mismo datos son colocados en muchos lugares, esto quiere decir que son replicados. El repositorio dio un buen ejemplo de esto. Por el motivo de la autonomía local, las partes del repositorio son replicadas para cada sitio.

Algunas otras partes del repositorio son particionadas, ya que solamente el nodo local necesita conocer acerca de los objetos locales. Los sistemas de B.D. proveen acceso transparente a datos replicados y particionados, los cuales son llamados Sistemas de Bases de Datos Distribuidas ***"S.B.D.D"***.

IV.5.1- DEFINICION DE DATOS

El estándar SQL define un grupo de tipos atómicos (números, cadena de caracteres, usuarios y otros). Los tipos de usuarios definidos, llamados dominios pueden ser definidos en términos de tipos atómicos y otros dominios. Por ejemplo, número de parte o nombre del cliente puede ser definido como dominio.

Las tablas SQL son grupos de registro, cada uno de los registros es una secuencia de valores. La característica que distinguen a una B.D. relaciónal es que todos los registros de una tabla tienen el mismo formato (la misma secuencia de tipos o dominios). Esto uniformemente significa que cada registro de la tabla tiene exactamente la misma forma.

Como cada operación produce una tabla como salida, es posible definir tablas virtuales, llamadas vistas, que son computadas de una o más tablas subrayadas. Las vistas proveen independencia de datos: si una definición de tabla es alterada, el formato de la tabla IV.4., puede ser definida como una vista del nuevo formato de tabla y los programas viejos pueden operar sin cambiar por el uso de la vista para accesar la tabla.

Toda la información lógica y física es registrada por el sistema SQL en grupos de tabla, llamados los catálogos SQL, dentro del repositorio. Si una tabla particular es particionada o replicada en muchos lugares, entonces, por motivos de autonomía de nodo, la definición de la tabla es replicada en catálogos para cada uno de esos lugares.

IV.5.2- MANIPULACION DE LOS DATOS

Uno de los principios del modelo relacional es el uso de un lenguaje para definición de datos, manipulación de datos y recuperación de datos. En adición, el mismo lenguaje es usado para ambos queris interactivos y para accesos programáticos. Usando el mismo lenguaje para una interface conversacional y programática, también provee una manera de ayudas para provar queries: se puede interactivamente introducir programas y ver la respuesta. El principal concepto que hace este trabajo es que cada uno de los operadores toma relaciónes como parámetros y una operación de lectura regresa una relación con los resultados. Las relaciones tienen un formato de despliegue natural y pueden ser alimentados para operadores más robustos. En particular las vistas pueden ser definidas en términos de queries.

El lenguaje de definición de datos toma prestado el manejo de expresion y procedimientos de los lenguajes de datos, para definir coacciónes y triggers. El operador select SQL forma el principio de esta lógica.

IV.5.3.- CONTROL DE DATOS.

El control de datos es un "recibe-todo" (catch-all). Esto incluye verbos de seguridad (GRANT y REVOKE), control de verbos concurrentes (LOCK y SET TRANSACTION) y verbos de transacción (BEGIN WORK y COMMIT WORK).

El modelo de seguridad básico de SQL es que las tablas están apropiadas por usuarios. Los programas corren como agentes para los usuarios. Los sistemas SQL permiten al propietario otorgar y revocar accesos a sus tablas. Por el uso de vistas, el propietario puede otorgar a otros la autoridad para accesar un subgrupo de valores específicos de la B.D. El modelo de seguridad elaborado de los SQL's se desarrollo de un sistema de tiempo compartido en el cual los usuarios comparten archivos. Este modelo es inapropiado para un SPT, en el que los clientes invocan servidores que en turno accesan los datos.

En resumen, la seguridad y el control de transacciones son resultado de los sistemas. La B.D. debe interoperar con la seguridad y los mecanismos de transacción del sistema operativo del host y el SPT del host.

IV.5.4- DESPLIEGUE DE DATOS

Una vez que las tablas han sido definidas y pobladas con datos, lo esencial es tener herramientas para desplegar los datos y generar reportes acordes. Tales herramientas generalmente soportan un escritor de reportes orientado a batch, que dan un query SQL y un arreglo de reporte, produce un archivo o lista conteniendo las respuestas. El reporte podría consistir de tablas, cartas y gráficas o el archivo de salida podría ser alimentado a una herramienta Lotus o Excel para análisis de datos y presentación.

Virtualmente todos los sistemas constan de un desplegador (brouser significa visualizador) en la B.D.; que produce una forma por registro o un despliegue tabular; tal visualizador permite a los usuarios hacer querys y editar la B.D. muchos deberían editar archivos de textos y hacer queries. Estos despliegues pueden automáticamente ser generados de la información descriptiva acerca de la tabla en el repositorio.

IV.5.5.- UTILERIAS DE OPERACIONES DE LA B.D.

Las utilerías de las B.D. proveen las herramientas de operación necesarias para manejar la B.D. Cada vez más estas utilerías corren automáticamente, ejecutando en la B.D. el archivado, recobro, reorganización y rediseño. El performance es usualmente un mejor resultado para las utilerías. Por ejemplo, si la carga y archivado corre en un megabyte/segundo, entonces el cargado o descarga de una B.D., de un terabyte tomará un millon de segundo, o alrededor de 12 días. Estas no son respuestas fáciles para este problema. Historicamente, las utilerías requirieron exclusivamente accesar la B.D.; todas las otras actividades fueron suspendidas. Este aprovechamiento individual no es aceptable para sistemas que deben estar continuamente disponibles para transacciones en línea. Los requerimientos son claros: (1) todas las utilerías deben correr en un segundo plano sin interrumpir el servicio y (2) las utilerías no deben tomar más de un día para hacer el trabajo.

En el presente, estas son actividades considerables para hacer todas las utilerías de la B.D. en línea (operaciones sobre los datos mientras otras están leyendo y escribiendo), incremento (trabajos en unos pocos megabytes para un tiempo), recomenzar (continuar si se detuvo debido a una falla o sobrecarga) y paralelo (operar en múltiples procesadores y discos en paralelo para subir la velocidad de la operación). Tales utilerías ahora existen para carga, descarga, recuperación y reorganización de B.D. y para colectar basura y compactar discos. Se conocen algoritmos para construir índices mientras el sistema está operando y la tabla base está cambiando, pero no una tiene que ser incorporada a estos algorítmos dentro de un sistema en producción. Ciertas operaciones excluyen una columna de la tabla, invalidando la tabla. Tales operaciones podrían forzar la aplicación para ser recompiladas o tal vez reprogramadas. Estas no son soluciones propuestas para tales problemas.

Otras utilerías producen, ya sea, reportes interactivos o batch sobre el performance del sistema . En el presente, el estilo típico es proveer un reporte diciendo a cuales programas accesar y a cuales tablas. Asociado con cada una de las entradas está el costo de un procesador, el costo de un disco I/O y si el sistema es distribuido, el costo de un mensaje. Esta información puede ser usada para analizar el performance del sistema.

Las herramientas que están surgiendo analizan la información del performance y sugieren diseños físicos mejores. En el futuro las herramientas estarán integradas con las utilerías para hacer al sistema autoafinable.

IV.5.6.- CARACTERISTICAS DE LAS OPERACIONES

Como se explico en la subcapítulo III.1.2., el sistema de administración y operaciones son una mayor parte del costo del propietario de un sistema de procesamiento de transacciones. La rutina deberá ser automatizada, reduce los costos del staff y reduce los errores. Los humanos deben solamente ser cuestionados para tratar situaciones excepcionales. El archivado y reorganización de la B.D. son ejemplos de tales tareas de rutina.

Estas deben ser interfaces orientadas a formas para desplegar el estado que guardan los sistemas y performance para varios niveles de abstracción (por ejemplo: aplicación, procesos, archivos o dispositivos). Estos despliegues deben incluir mecanismos para detectar problemas, diagnosticarlos y sugerir remedios. El sistema deberá también rastrear los problemas (por ejemplo: caídas de hardware y software) y generar reportes sobre el estado que guardan durante los problemas. El repositorio es el mecanismo principal para almacenar toda esta información. Esto es también lo básico para escribir aplicaciones de operaciones usando herramientas de alto nivel.

Los sistemas proveen los mecanismos para seguridad, recobro y control de cambios, pero el staff (grupo de acesores) debe diseñar las políticas de operaciones y administración, especificando como estos mecanismos serán usados.

V.- ESTUDIO COMPARATIVO ENTRE LOS SPT

Gran parte de esta presentación es histórica, mostrando cómo los Sistemas han evolucionado sobre el tiempo. Para el estudio de esta tesis se han seleccionado algunos de los más importantes (IMS, CICS, TUXEDO, ENCINA, TOP-END), los cuales definirán el comportamiento del futuro de los Sistemas de Procesamiento de Transacciones. Un típico monitor de PT y su infraestructura, refleja el esfuerzo de cientos de gentes sobre varias décadas (miles de personas - años). Se escogen algunos de los más populares y se selecciona estándares (LU6.2 y X/Open DTP) que creo definirán el comportamiento de los futuros sistemas., Los manejadores de B.D. son vistos como manejadores de recursos *"MR"* en un SPT. En esta tesis se tomaron algunas características de cada OLTP y se ignoraron otras que son muy específicas de cada producto, que se describen en la tabla comparativa del subcapítulo V.3.6.

V.1.- IMS

Comenzó en los sesentas como un sistema de inventarios de los E.U.A. En ese tiempo, el modelo de hardware tenía un procesador de medio mip con 100 kb de memoria, unos pocos drives de disco y un grupo de teclados remotos, lectora de tarjetas e impresoras como dispositivos periféricos.

FIGURA V.1.- LA ESTRUCTURA COMPLETA DE IMS. El sistema operativo proporciona básicos servicios para la red, el manejador de transacción "MT" y las aplicaciones usan la B.D. de IMS. El cuadro representa los procesos. No todos los procesos están incluídos. Una pequeña suma de terminales de contexto son mantenidas por los *threads* (proceso) en los espacios de dirección de IMS/MT. Este contexto es usado para servicios de presentación de drive como el switcheo de mensajes de entrada/salida. Las aplicaciones IMS mantienen uno o más servidores de clases manejados vía colas de recuperación. Las aplicaciones hacen llamadas a los sistemas de B.D. IMS (DL/I) que este corre con un dominio de protección de los procesos de aplicación. Este da protección y paralelismo a los procesos de aplicación.

Antes del IMS, el modelo de procesamiento fue un prototipo viejo muy grande, que fue creciendo y aprovechó el nuevo prototipo que está basado en cintas. Las actualizaciones en línea para archivos indexados fue considerado un avanzado concepto. La información idéntica fue replicada en muchos archivos; los accesos a datos fueron a través de lectoras-impresoras de tarjertas locales o remotas y trabajos batch. IMS introdujo un sistema integrado de ***BD-CD.*** (Base de Datos - Comunicación de Datos). Desde el inicio IMS, dió una interface uniforme de programación tanto a la B.D. y como a las terminales. El modelo de datos fundamental consistía de una jerarquía basada en segmentos de datos (el modelo de datos jerárquicos).

IMS aplicó este modelo de datos a la interface de terminales como a la interface de B.D. Hoy el sistema típico IMS tiene miles de terminales distribuidas alrededor del mundo manejando un gran sistema multiprocesadores con una B.D. basada en discos en el rango de 100 Gb. Existen en el mundo alrededor de 10,000 de dichos sistemas, y ellos forman el núcleo de los sistemas de PT que están implícitos en todas las grandes corporaciones. Esto es un tributo al diseño original y para los desarrolladores del IMS que los sistemas fueron capaces de evolucionar. Con la evolución IMS fue remplazado por un Monitor de PT y una B.D. separada, y el Monitor de PT, red, y los sistemas operativos fueron adoptando soporte para un mecanismo TRPC (LU6.2, APCC)[10].

IMS tiene dos partes principales: (1) ***IMS/MT*** (Manejador de Transacciones),[11] el Monitor de PT que incluye planeación, autorización, servicios de presentación y funciones de operaciones; y (2) ***IMS/BD*** mejor conocido como ***DL/I***[12], un sistema de B.D. soportando un modelo de datos jerárquico. Las aplicaciones corriendo como servidores en IMS usualmente accesan B.D. DL/I, pero éstos pueden también accesar otros manejadores de recursos (DB2 y Oracle) y otros servicios de sistemas operativos. Como se muestra en la figura V.1, las aplicaciones pueden accesar directamente a la red, con lo cual permiten llamados a procedimientos remotos.

V.1.1.- AMBIENTE HARDWARE Y SISTEMAS OPERATIVOS

IMS está interesado en remplazar el sistema operativo. IMS usa procesos de sistemas operativos, planeación y protección, archivos del sistema operativo, administración de programas *binding* (poner a disposición) y autorización del sistema operativo. Esto está en contraste para CICS, el cual usa simples procesos del sistema operativo (espacio de dirección y tareas) y entonces implementa cada una de las funciones del sistema operativo, especialmente con espacios de dirección. CICS implementa sus propios procesos (tareas, threads) con espacio de dirección, tiene su propio cargador de programas y binding, así como su propia planeación y manejador de memoria.

[10] Esta interface es inherente al protocolo peer-to-peer (punto-por-punto) transaccional definido por IBM especificado como un formato y protocolo *"FAP"*.
[11] Este fue llamado IMS/CD hasta 1990
[12] DL/I.- Data Language/I.

Cada una de éstas funciones CICS, son simples y más rápido que la correspondiente función del sistema operativo de propósito general. El aprovechamiento de IMS usa más recursos de hardware (por ejemplo más instrucciones de procesador, y más bytes de memoria), pero ésto es más general y da mayor funcionalidad que un aprovechamiento de proceso simple. El aprovechamiento que hace CICS es más portable a través del sistema operativo.

Las aplicaciones IMS en la figura V.1., operan en sus propios procesos del sistema operativo. Así, ellos pueden llamar cualquier servicio del sistema operativo y está protegido de uno a otro. La falla de una aplicación no afectará las otras. Por contraste en la figura V.5 todas las aplicaciones corren dentro del simple proceso CICS (espacio de dirección). Cada aplicación corre como un *thread* (proceso) en este espacio de dirección, el cual es un simple dominio de protección; como resultado, cada aplicación puede romper las otras. Una aplicación CICS esta llamando al sistema operativo está haciendo una solicitud o requerimiento con la autorización de CICS; como por ejemplo un llamado DELLOCATE FILE, el proceso es autorizado por CICS. Las aplicaciones en CICS deberán por lo tanto no hacer llamados directos sobre el sistema operativo. Esta prohibición es administrativamente forzada; el administrador CICS lee cualquier programa que esté instalado en el sistema que él administra.

IMS corre solamente sobre los sistemas operativos *"MVS"* (Multiple Virtual System.- es el sistema operativo de las máquinas grandes de IBM) que, en turno corre sobre variantes de la familia de procesadores IBM 370 (4300, 9370, 3090, system/9000, etc). La figura V.2. muestra una colección de 16 sistemas MVS todos compartiendo un terabyte en disco. El sistema accesa un pool de discos, cintas y controladores de comunicaciones.

MVS provee una imágen del sistema simple para procesos corriendo en un sistema. Las tareas (el término MVS para procesos) son la unidad de envío y cada tarea corre en algún espacio de dirección. MVS y el hardware ofrece una interface del manejador de recursos eficiente llamado; *servicios a través de la memoria* o más comunmente ***XA***, para *cruce de espacio de dirección*. Este mecanismo permite a una aplicación invocar un manejador de recursos corriendo en un dominio de protección diferente (espacio de dirección). XA permite un proceso para eficientemente switchar desde un espacio de dirección a otro, protegiendo al manejador de recursos en el llamado de la aplicación. La B.D IMS (DL/1) y la B.D. SQL (DB2) actúan como manejadores de recursos dentro de un sólo sistema MVS. El subsistema y las interfaces XA son públicas, permitiendo a otros subsistemas (por ejemplo el sistema B.D. Oracle) actuar como un manejador de recursos.

MVS provee IMS con servicios básicos I/O para periféricos. Un sistema de archivo básico llamado *"VSAM"* (Virtual Storage Access Method), provee soporte para discos, archivos distribuidos y métodos simples de acceso (no estructurados, entrada secuencial, relativa y llave).

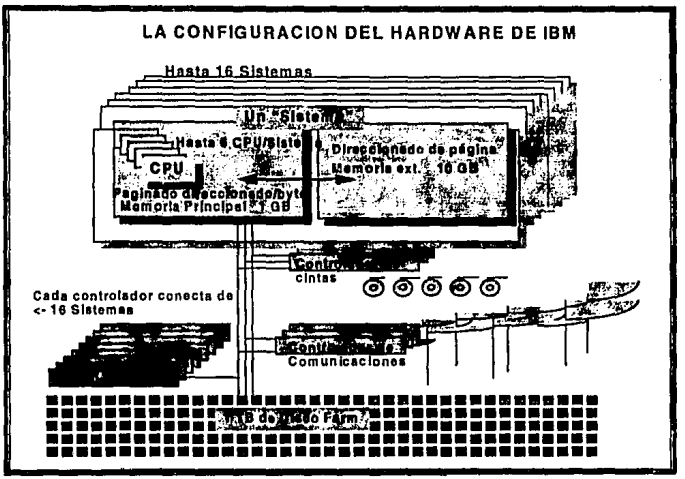

FIGURA V.2.- LA CONFIGURACION DEL HARDWARE DE IBM. Muchos sistemas pueden compartir un común pool de discos y cintas. Los 16 sistemas mostrados arriba, pueden ser unidos al pool del disco. Cada sistema soporta una sesión de la red.

V.1.2.- MODELO DE WORKFLOW (FLUJO DE TRABAJO)

IMS originalmente adoptó procesamiento de Transacciones encolado como paradigma básico computacional. En este modelo, un mensaje llega conteniendo un código de Transacción y un bloque de datos. El código de Transacción es un acrónimo (sigla-palabra formada con letras y sílabas iniciales de un nombre compuesto) corto para la transacción, semejante a una transacción de débito (TDB). IMS/MT mantiene un descriptor de contexto para cada terminal conocida o cliente. Basada sobre este descriptor de contextos y el código de Transacción, IMS/MT hace servicios de presentación sobre la entrada de datos, convirtiéndolos a registros de datos jerárquicos. Este registro de datos es insertado dentro de una cola para más tarde ser procesado por un servidor.

Desde estas colas son objetos recuperables, la inserción genera un committ. Una vez que el mensaje de entrada es formateado e insertado en la cola y la inserción es committed, IMS/MT reconoce la entrada. Los mensajes de entrada son entonces procesados asincrónamente. Cada cola de entrada representa un servicio, (todos los mensajes en la cola son similares). Un *servidor de clase* (es una colección de uno o más procesos servidor ofreciendo el mismo servicio) es asociado con cada cola.

Todos los miembros de un servidor de clases son programas de aplicaciones idénticas corriendo como procesos del sistema operativo estándar.

IMS/MT predistribuye los servidores, como parte del arranque o reconfiguración basados sobre comandos de operador. Un servidor de clase puede crecer o encoger. La creación de servidores puede ser periódica (por ejemplo una vez, un día) para obtener el efecto de procesamiento batch. Teniendo un pool predistribuído de procesos servidor reusables esperando para entrar desde la red, amortize los costos de la creación de procesos a través de muchas Transacciones y reducir dramáticamente el tiempo de respuesta de los requerimientos. Este estilo de *espera para entrada* de procesamiento fue agregado a IMS a mediados de los 70´s y está en uso hoy.

Un servidor de clases puede ofrecer muchos servicios; estos podrían ser capaces de aceptar entradas de distintas colas. Un IMS planea y asigna mensajes de entrada encolados a servidores basados sobre parámetros de cola especificados por el usuario, tales como, servidores-por-cola ("severs-per-queue"), la prioridad de mensajes y entradas de encolmiento múltiple(threshold queue depth).

Las aplicaciones pueden también hacer llamadas SQL a DB2, Oracle o Teradata; pueden hacer llamadas al sistema operativo para MVS; y pueden hacer iguales llamadas a procedimientos remotos a otros sistemas usando el soporte MVS para RPC proporcionado por la red (ver el subcapítulo V.2. Comunicación Programa a Programa). Todos estos llamados son implementados por llamados XA (cruce de memoria), el cual provee una interface uniforme al llamador y permite al llamado correr en un dominio de protección (espacio de dirección) separado de el llamador.

Cuando el mensaje de salida es committed (cuando el servidor llama SYNC que significa sincronización), IMS/MT formatea los mensajes de salida y los presenta a la terminal. Cuando la terminal reconoce el mensaje de salida, el mensaje es borrado desde el espacio de colas y el borrado es committed.

Como se describió, el procesamiento de una transacción IMS actualmente consiste de tres propiedades ACAD de la transacción. 1) Inserta el espacio de cola, (2) hace el trabajo, (3) libera la respuesta. Este modelo captura las necesidades de la mayoría de las aplicaciones, provee una integridad de mensajes excelente y tiene algunos beneficios reales para la planeación.

Pero el paradigma de transacción encoladas también hacen transacciones conversacionales y distribuidas completamente difíciles. El modelo encolado tiene también un costo de performance significativo, ya que cada una de las peticiones genera tres pasos de commit. Para salvar los dos commit extras, IMS ofrece una opción de "Manejo de Mensajes Expedito", el cual inserta el mensaje de entrada en espacio de cola sin reconocimiento a la terminal o cliente. La aplicación entonces corre, consume el mensaje de entrada y genera una réplica. Estas son todas las partes de una transacción (propiedades ACAD). IMS/MT entonces libera el mensaje de salida al cliente y obtiene el siguiente mensaje de entrada. Este es el flujo de transacción de Manejo de Mensajes Expedito (ver figura V.3).

El Manejo de Mensajes Expedito resuelve el problema del performance del procesamiento encolado (reemplazando tres commits con uno), mientras mantiene la integridad de los mensajes de salida. Los mensajes de salida son deliberados por lo menos una vez, porque las colas son durables. El uso de sesiones y los números de la secuencia de sesión eliminan duplicados, esto produce procesamiento exáctamente una vez de mensajes de entrada y libera exáctamente una vez de mensajes de salida.

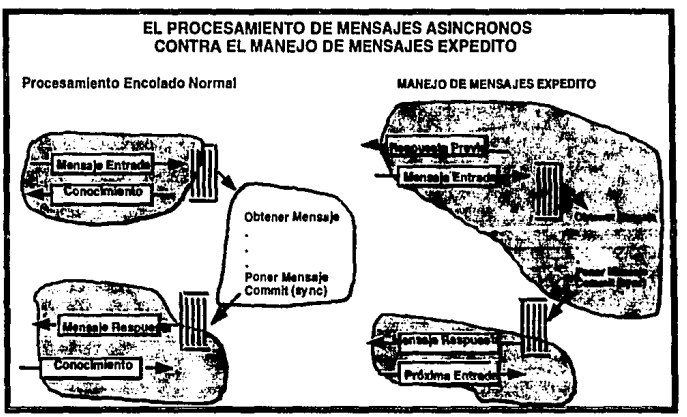

FIGURA V.3.- EL PROCESAMIENTO DE MENSAJES ASINCRONOS CONTRA EL MANEJO DE MENSAJES EXPEDITO. Las colas de mensajes de IMS pueden ser inmediatamente reconocidas, y entonces ser procesadas más tarde. Estas dán surgimiento a las tres transacciones de ACAD, mostradas en las áreas sombreadas de la parte izquierda de la figura. Si la transacción responde en poco tiempo y tiene un sólo mensaje de salida, entonces la respuesta del mensaje puede reconocer el mensaje de entrada previo, y el próximo mensaje de entrada puede reconocer la respuesta. Como se muestra en la figura del lado derecho, este es un manejo de mensajes expedito que reduce los números de transacciones ACAD por la entrada de un factor de tres.

Para mejorar las limitaciones del procesamiento de transacciones distribuido y conversacional de un modelo de procesamiento encolado, IMS soporta dos mecanismos de ruteo de transacción: 1) Un protocolo propietario IMS llamado Acoplamiento de Sistemas Múltiples *"ASM"* que rutea mensajes a lo largo de los sistemas IMS y 2) un protocolo "abierto" llamado Comunicación Inter-Sistema (CIS) que avanzará un mensaje de entrada a otro sistema IMS, para un sistema CICS, o para otros vendedores de sistemas (por ejemplo DEC, Tandem, Unisys).

La lógica avanzada es basada ya sea sobre el código de transacción de entrada o sobre una subrutina escrita por el usuario que analiza los mensajes y los avanza al otro site. Una vez lanzados al nodo remoto IMS/MT, la aplicación es liberada para accesar el espacio de colas local IMS/MT y llamar sobre otros manejadores de recursos locales. Esto no es soportado para transacciones conversacionales, pero es excelente para soportar transacciones pseudo-conversacionales y para transacción de contexto.

En suma, para programas que manejan mensajes, IMS también soporta accesos a manejadores de recursos desde sesiones interactivas (Los programas MVS de "OTC" Opción de Tiempo Compartido) MVS *"OTC"* y hasta programas batch. Muchas utilerías corren en este modo batch. Estos programas tienen accesos a servicos IMS, incluyendo las B.D. (DL/1, DB2, etc) y los espacios de cola.

V.1.3.- AISLAMIENTO DEL PROGRAMA

Una aplicación IMS DL/I declara al tiempo de compilación y qué tipos de registros accesará, ya sea que los accesos sean para lectura o para lectura y escritura.
La implementación inicial de IMS intentó hacer una planeación, significa que ellos no hicieran locking (aseguramiento) de registro-tipo. Si dos programas actualizaron el mismo tipo de registro, ellos no podrán concurrentemente programar. Esto es equivalente a una tabla granulada locking asegurado para sistema relacional. Por 1970 estó fue claramente inadecuado para sistemas en línea, y el locking de registros-granulado fueron agregados. La implementación señalada proprociona registros de seguro exclusivos y compartidos (estabilidad del cursor). Al mismo tiempo, el sistema de recuperación fue reescrito para incluir dos bitácoras: una bitácora REDO (rehacer) para reiniciar la recuperación y una bitácora UNDO (deshacer) llamada también la bitácora dinámica de la transacción (en la actualidad la bitácora REDO fue duplexada sobre una cinta, mientras la bitácora dinámica fue simplexada a disco). Dos manejadores de recursos escriben para la bitácora REDO: el sistema IMS B.D. DL/1, y el manejador de espacio de colas IMS/MT. Estos dos manejadores de recursos, cada uno son usados en diferente estrategia de recuperación. El manejador de B.D. usa el protocolo commit disponiendo la necesidad para cualquier REDO en caso de un reinicio. El manejador de colas usa una muy moderna estrategia UNDO-REDO, generando algunas bitácoras de registro necesarias para la fase 1 e inicializando la liberación de mensajes para la fase 2. Esto usa compensación de bitácora para limpiar en caso de que abort la transacción.

A este paquete, es llamado Aislamiento de Programa y fue liberado para IMS alrededor de 1974.

V.1.4.- LLAMADAS DE CAMPO Y PRINCIPAL ALMACENAMIENTO DE BASES DE DATOS

Los usuarios IMS encontraron simples registros locking inadecuados para sistemas de procesamiento de transacción de alto volumen. A mediados de los 70's, el grupo IMS obtuvo la meta de ejecutar 100 transacciones por segundo (tps) sobre un procesador dual 370/168 (alrededor de 5 mips). El resultado, llamado IMS de trayectoria rápida (Fast Path), introdujo commit de grupo, llamados de campo, B.D., de almacenamiento-principal y manejo de mensajes expedito.

En esencia, IMS FastPath implementó un nuevo sistema de B.D. con lenguajes de manipulación de datos propios y sistemas de recuperación. Esto hizo el tercer manejador de recursos familia IMS. Los llamados tradicionales DL/1 son soportados en estas B.D., pero la opción de alto performance fue para usar llamadas de campo (operaciones en custodia). Cuando el llamado de campo es entregado, el predicado es probado y un registro de bitácora es generado. Para la fase 1 de commit, un lock sobre el registro es adquirido, y el predicado es nuevamente aprobado. Si todos los predicados pasan, la transformación es aplicada y los locks liberados (todo como parte de la fase 1 de commit). Esto es una aplicación de la optimización del último manejador de recursos.

El FastPath también introdujo commit agrupado batch las bitácoras de escritura de varias transacciones en un simple par de bitácoras escritas. La B.D. de almacenamiento principal tuvo una interesante estrategia de recuperación: esto no fue una bitácora UNDO para llamadas de campo (solamente una bitácora REDO). Para reiniciar, la más reciente copia buena es leída dentro de memoria, y entonces una búsqueda de la bitácora REDO está corriendo. IMS FastPath también tiene una B.D. de insertación (Entrada de datos en la B.D la Dependiente Secuencial) que tiene actualizaciones diferidas con sólo una bitácora REDO. Consecuentemente, el FastPath por sí mismo es una pura actualización diferida, manejador de recursos solamente REDO. En el final, el grupo FastPath vino a cerrar esta meta, llevándose a cabo alrededor de 97 tps en sistemas de procesador dual 370/168 en 1976.

V.1.5.- COMPARTICION DE DATOS

Las demandas del performance IMS creció mucho más rápido que otros procesadores o softwares de aumento de producción. Los multiprocesadores de memoria compartida fueron una solución a este problema. A través del tiempo, IBM liberó 2 formas, después 4 formas y posteriormente 6 formas de multiprocesamiento de memoria compartida. Pero éstos no mantuvieron el paso con la demanda.

Como un resultado, IMS permitió dos diferentes sistemas IMS corriendo en diferentes sistemas MVS para accesar y actualizar una B.D. compartida residente en un disco entre los dos sistemas (ver figura V.2). Cada sistema puede actualizar una página en particular de la B.D. y los registros de la bitácora de una página en particular puede estar fundada en las bitácoras de dos diferentes sistemas. Solamente en los datos tradicionales DL/1 pueden ser compartidos de esta forma; espacio de cola, almacenamiento de B.D. principal, dependencia secuencial de entrada de datos a las B.D., y las B.D. DB2 no pueden ser compartidas en esta forma.

Esta idea de compartir datos entre múltiples sistemas de recuperación semi-independientes son genéricamente llamados; *"compartición de datos"*. Esto da surgimiento a muy interesantes seguros (loking) y problemas de recuperación. IMS usa seguros de granularidad de página para regular la consistencia de la página a través de los sistemas IMS. Estos seguros son proporcionados por el manejador de seguros (*Inter-system resource lock manager "IRLM"*), algunas veces llamado *el manejador global de seguro*. El aseguramiento es hecho sobre nombres, pero una parte del nombre es usada por razones de performance. IRLM soporta seguros exclusivos y compartidos. Esto es un tocken que permite al propietario de este proponer actualizaciones a la parte de la tabla compartida; la parte de la tabla es replicada para cada sistema. El tocken conjuntamente con los nuevos requerimientos de seguro, circulan en el sistema varias veces en un segundo. Cuando el manejador de seguro local necesita un seguro sobre un objeto compartido, esto parte del nombre de seguro.

Estos son tres casos:

1) Una parte está ya reservada por el sistema.
2) Una parte es reservada por otros; en este caso, agrega la petición de reservación al buffer de tocken de buffer.
3) Una parte está libre, en este caso, agrega el requerimiento de reservación al buffer tocken.

Cuando el tocken llega con la nueva parte de la tabla y peticiones, el manejador de seguro local trata de liberar cualquier requerimiento pero la parte entra en conflicto y entonces trata de conceder cualquier nuevo requerimiento. Esto significa que si un requerimiento es permitido, ambos sistemas saben acerca de él. El manejador de seguros pasa entonces el tocken, nuevas partes de la tabla y nuevos requerimientos al peer-system.

Este algoritmo ha sido mostrado para ejecutarse bien en muchos casos. Si cada sistema tiene buena localidad de referencia, entonces cada uno de ellos adquiere un subgrupo de la parte del cubo para "esta mitad" de la B.D. Sólo ocasionalmente una transacción tiene que preguntar al otro sistema por un seguro.

La lógica IRLM para manejar las fallas del sistema es complejo. Si un sistema falla, entonces todas las páginas cubiertas por los elementos de la parte de la tabla locked exclusivamente por este sistema debe permanecer locked hasta que las transacciones del sistema son recuperadas. Esto es porque IRLM toma tales aproximaciones conservativas para actualizar la parte de la tabla, insistiendo el peer system conozca acerca de cualquier parte de entradas nuevas anterior a los permitidos. Desde que DL/1 usa el commit forzado para los recursos compartibles, sólo la transacción UNDO es necesaria. Las bitácoras están residentes en discos compartidos, lo cual hace posible para el peer-system correr la recuperación sobre las transacciones fallidas. Una vez que estas transacciones son regresadas, las entradas de la parte de la tabla IRLM del sistema fallidos pueden ser limpiadas. Para los medios de recuperación, las jornadas de los dos sistemas son fusionadas en un equivalente en el orden en que fueron llegando al sistema (sellos de tiempo).

V.1.6.- DISPONIBILIDAD MEJORADA Y SISTEMAS DUPLICADOS

Los sistemas IMS son generalmente sistemas grandes. Como consecuencia, están constantemente cambiando. Las terminales y las líneas de comunicación están siendo agregadas o movidas, los discos son agregados, los archivos están creciendo, los campos de B.D. y tipos de registros están siendo constantemente agregados, los programas de aplicación son agregados y reparados, entre otros.

Al final de la década de los 80's los sistemas fueron creciendo y cambiando, mientras estaban operando. Esta capacidad es genéricamente llamada *"cambios en línea"*. En el presente, casi cualquier aspecto de un sistema IMS y del sistema operativo (MVS) y la red pueden ser cambiados mientras el sistema está operando.

Un segundo cambio, aún mayor ayudó a mejorar la disponibilidad de IMS sobre el manejo de errores. Originalmente, IMS fue de falla rápida (fail-fast). Esto tuvo pequeños errores de adherencia lógica. Como los sistemas crecieron para soportar miles de componentes, esto creó el problema que una falla donde fuera podría fallar el sistema entero. Por ejemplo, si un disco conteniendo un fragmento de archivo fue indisponible, entonces el archivo entero fue marcado indisponible, aunque los otros 500 discos conteniendo fragmentos del archivo estuvieran disponibles. Si una página de un archivo fue invalidado, entonces el archivo fue tratado como invalido. IMS adoptó varias tácticas para trabajar con esto. (1) Esto permitió algunos archivos de datos para ser replicados (hasta 7 veces) en orden para desvanecer las fallas de hardware. (2) Esto limitó la granularidad de la falla introduciendo la idea de indisponibilidad de datos. Ya que marcando un archivo entero como indisponible o inválido, solamente las partes fueron marcadas.

La regla fue esta: Si además un dato y la trayectoria de acceso están ahí, entonces el programa deberá ser capaz de accesar un dato; sí el dato además no está ahí y además pueda ser invalidado el programa deberá ser informado y deberá tener permiso para brincar este "hoyo" en el dato, de manera que este pueda procesar el dato remanente. (3) En una característica relaciónada, si una página escribe fallas, IMS recuerda la página alterada vía un mecanismo interno, hasta que el medio de recuperación es conveniente. Como una consecuencia de estas aproximaciones un programa de reporte batch que busca un terabyte de B.D., puede encontrar unas pocas páginas invalidadas, pero esto puede todavía procesar toda la información disponible. Hoy, la mayoría de grandes sistemas de B.D. adoptan una similar aproximación. Esta aproximación para datos desaparecidos es particularmente importante para sistemas distribuidos, donde algunos nodos o un archivo distribuído grande puede estar indisponible.

En *IMS/XRF*[13] estos son dos sistemas IMS que operan como sistemas pares (ver figura V.4). Estos son sólo copia de la B.D. que reside en el pool de discos compartidos entre el primario y el de respaldo. Ambos sistemas el primario y el respaldo están en sesión con todas las terminales activas. Los controladores de terminales ven esta sesión como una sesión primaria y una sesión de respaldo. El usuario a la terminal es ajeno de esta sesión par (la sesión lógica par está en el controlador de terminal local). Durante la operación normal, el sistema IMS primario ejecuta como usual y la terminal se comunica solamente con la sesión primaria. El sistema de respaldo es una reproducción del primario. Esto lo tienen todos los servidores de clases, manteniendo las estructuras de datos y generalmente simulan al sistema primario, sin hacer ningún trabajo actual.

El sistema de respaldo está constantemente ejecutando operaciones de "recuperación" REDO sobre los estados, aunque el respaldo estuvo ejecutando reinicios después de una falla del sistema primario. Todas las sesiones de respaldo están sin uso. El sistema de respaldo busca la bitácora de disco y ejecuta REDOES (rehacen) de cualquier registro de bitácora que deberá cambiar el respaldo del estado de memoria principal.

La recuperación de la B.D. en el disco están solamente ejecutadas para tomar el mando "control". Y como resultado, el estado del sistema de respaldo está totalmente al corriente con el primario. IMS/XRF es factible de ser llamado un "sistema en espera de que el otro falle" (*Hot standby system*). El respaldo cuidadosamente sigue la pista del estado del primario en órden para minimizar los tiempos fuera de la toma de mando. El seguimiento de esto describen el cargado dentro de los pools, la preasignación de los servidores de clases y de sus crecimientos. Estos seguimientos y su ejecución verifican la autorización y los archivos abiertos necesarios por la B.D.

[13]IMS/XRF.- Extended Recovery Feature.- es un mecanismo de sistemas pares con dos sistemas IMS adyacentes de discos compartidos. Si un sistema falla, el segundo sistema abort la transacción sin ser uncommitted, completando unos commitung y continúa ofreciendo servicios.

FIGURA V.4.- IMS/XRF PROPORCIONA UNA FORMA DE SISTEMAS PARES CON DISCOS COMPARTIDOS. El sistema primario IMS modifica la B.D. y genera registros de bitácoras. Este continúa enviando mensajes de "Estoy funcionando" para el sistema backup esto es manteniendo una copia del estado del sistema leyendo la bitácora de disco es generado. Tanto el primario y el backup tienen accesos a los discos y terminales. El primario está activo, mientras el backup está pasivamente leyendo los cambios de estado del primario.

Estos siguen las pistas de todas las terminales y sesiones del intersistemas, mensajes de encolado. Esto también sigue la pista de los locks, tal que los respaldos puedan terminar el procesamiento normal en paralelo para el rollback de la transacción para una toma de mando.

IMS/XRF usa tres mecanismos de vigilancia para detectar falla en el primario: (1) el primario envía mensajes "Estoy vivo" al respaldo vía una sesión de comunicaciones, (2) el respaldo vigila los registros de bitácora que están llegando sobre los discos compartidos y (3) el respaldo vigila para un especial "latido de corazón" sobre el reinicio de grupos de datos (indicando que el primario es punto de verificación). Basados sobre límites de tiempo especificados por el administrador, la vigilancia disparará automáticamente tiempo fuera. El diseñador del sistema puede poner el operador en el loop por la colocación de límites de tiempo muy altos.

Cuando el respaldo decide que éste es el nuevo primario, el respaldo instruye al controlador de disco rechazar el requerimiento del disco del primario hasta que el tiempo fuera esté completo. Esto detiene al primario de más fuertes accesos y actualización de la B.D. y asegura que el primario conoce que un tiempo fuera ha ocurrido. En este caso, el nuevo primario completa la búsqueda REDO y entonces comienza un UNDO de cualquier transacción uncommitted.

Una vez que el REDO está completo, el sistema de respaldo envía un mensaje de "Aquí estoy" sobre las sesiones de resplado que ahora están en el primario. Si la terminal estuvo esperando una respuesta desde el primario, esto ahora obtendrá esta respuesta desde la sesión de respaldo. El hecho de que los mensajes de entrada y salida son registrados en espacio de colas recuperables asegura que cada uno de los mensajes de entrada reconocidos deberán ser procesados por lo menos una vez y cada uno de los mensajes de salida committed serán entregados por lo menos una sola vez. El uso de los números de secuencia y la coordinación de estos números de secuencia entre el primario y las sesiones de respaldo aseguran que los mensajes serán entregados y procesados exactámente una vez.

V.1.7.- DB2

DB2 es una implementación de IBM SQL sobre los mainframes. Estos actúan como un manejador de recursos tanto para IMS, CICS. Desde el punto de vista del procesamiento de transacciones, DB2 implementa un manejador de buffer de uso privado, con bitácora de tipo de escritura-ahead. Esto tiene una bitácora privada (no compartida con los monitores de transacciones). El locking aseguramiento es hecho para granularidad de página y sub-página, y DB2 soporta la interface IRLM. Como en 1991, DB2 no soporta el compartimiento de datos a lo largo de los sistemas MVS. CICS provee una facilidad DB2 XRF (pero IMS no lo hace). DB2 tiene utilerías excelentes para acumulación de cambios y para recuperación.

V.1.8.- EVOLUCION RECIENTE DE IMS

Por 1991, IBM tuvo generalizados substancialmente IMS. Las comunicaciones de datos y los componentes de procesamientos de transacción de IMS han sido renombrados IMS/Manejador de Transacción "MR". Los componentes de B.D. fueron empaquetados como un manejador de recursos. El sistema operativo y el software de red ha sido extendido para soportar la invocación de programas de transacción (MVS/*APPC*) pero no permiten soportar ***SYNCPT***[11] (ACAD).

IMS soporta muchos generadores de aplicaciones, herramientas de ingeniería de software, y herramientas administrativas. Mucho de la evolución, ha sido en el área de estos capACADad de niveles grandes, y la administración de sistemas.

V.2.- CICS Y LU.6.2.

CICS es el más popular monitor de PT, mientras LU6.2 es un protocolo de sesión transaccional de estándar de facto IBM. Esta sección primero describe la estructura y funciones de CICS. Esto entonces da una breve revisión de LU6.2 y esto es usado por CICS.

[11]SYNCPT - Synchronization point

V.2.1.- REVISION DE CICS

La implementación original de CICS (1968), ha sido portada a muchos derivados de OS/360 y DOS/360 (por ejemplo MVS y DOS), y esto ha sido también implementado sobre OS/2, AS/400 y UNIX.

El más nuevo suceso de CICS, fue debido a la aproximación tomada en relación del sistema operativo. CICS y todos estas aplicaciones corren en un simple espacio de dirección común del sistema operativo (ver figura V.5). Un simple proceso del sistema operativo corre el sistema completo CICS, estas aplicaciones y estos servicios. CICS hace demandas mínimas sobre el sistema operativo y es por esto altamente portable de un sistema operativo a otro.

FIGURA V.5.- CICS CORRE TODAS LAS APLICACIONES Y SERVICIOS CONTENIDAS EN ESPACIOS DE DIRECCION. Los manejadores de recursos pueden optar correr una parte de espacio de dirección (en este ejemplo, es DOS DL/I) en espacios de direcciones separadas en las aplicaciones corren como threads dentro del espacio de dirección. Los servicios CICS y todos los manejadores de recurso son accesados vía la interface del manejedor de recurso.

Los servicios CICS son típicamente diez veces menos expansivos que los correspondientes servicios del sistema operativo. Por ejemplo, CICS puede crear un proceso (thread) en menos de mil instrucciones y puede despachar un thread en menos de 500 instrucciones. Pocos sistemas operativos de propósito general pueden igualar estos números. Por ejemplo, los procesos no tienen protección y el planeador es un simple uniprocesador.

Una segunda razón de la popularidad de CICS es que ha sido abierto a otros subsistemas (manejadores de recursos). Sobre lo más reciente, cada uno de los vendedores de los sistemas de B.D. descubrieron que podrían instalar su producto como una aplicación en el ambiente CICS, acomplando un sistema de B.D. con un monitor de PT. Como una concecuencia, CICS rápidamente tuvo muchos diferentes sistemas de B.D anexados a éstos (notablemente DL/1, System 2000, Mark 4, Adabase y Cullinet).

Para estos primeros anexos fueron claramente de acuerdo, a veces corriendo dentro del espacio de dirección CICS. Esto permitió diagnosticar fallas reales y contener problemas reales. El simple espacio de dirección contenido en CICS, el sistema de B.D. y las aplicaciones. Si esto fue un software clandestino, el espacio de dirección debería fallar, y esto fue difícil para diagnosticar la fuente del problema. En suma, los productos de B.D. aprenden más y más acerca de CICS internos y comenzaron a leer y a modificar los bloques de control de CICS (los cuales son todos visibles en espacios de dirección). Cuando la siguiente versión de CICS cambió la definición de estos bloques de control, los productos de B.D. fallaron extrañamente.

CICS desarrolló una interface de manejador de recursos, tal que los subsistemas pudieran tener una limpia y bien diseñada interface para estos servicios. La interface del manejador de recursos tiene dos aspectos; 1) Los llamados CICS- MR que son usados para arrancar, detener y recuperar el manejador de recursos y 2) Los llamados de la aplicación MR que son usados para invocar el manejador de recursos. Inicialmente, este mecanismo de aplicación MR fueron prometidos para manejadores de recursos que cohabitaran los espacios de dirección con CICS. Con el tiempo, la interface del manejador de recursos fue generalizado, permitiendo al manejador de recursos fragmentos en el sistema CICS para invocar procesos del manejador de recursos protegido, ya sea vía intercomunicación de interprocesos o vía servicios a través de la memoria (XA). Esta interface permite a manejadores de recursos correr en espacios de dirección separadas, mejorando las fallas contenidas y la seguridad.

El concepto de manejador de recursos evolucionó de CICS para ser la base de la estructuración DB2, y esto eventualmente trajo el concepto de estructuración para X/OPEN DTP. Esto es un concepto principal de la estructuración para sistemas operativos.

CICS fue también el primer sistema comercial en ofrecer computación distribuida de transacciones. El modelo de Comunicación Inter-sistemas de CICS evolucionó dentro del protocolo estándar de IBM para Comunicación Programa a Programa. Este protocolo es conocido como SNA LU6.2 o *"APPC"* (Application to Program Communication). LU6.2, en turno, fue la inspiración para el OSI-TP y el estándar ISO para el protocolo de transacciones commit.

La Interface de Programación de Aplicaciones *"API"* CICS para Comunicación Inter-sistemas no ha sido ampliamente adoptada. IBM tiene otros cuatro APIS para LU6.2 siendo un total de 5. Los otros cuatro son (1) ***CPI-C*** (Common Programming Interface- Communications), el estándar SAA que deriva del diseño VM/370 y también exite en OS2/EE, (2) APPC/PC, una inteface implementada sobre MS/DOS, (3) MVS/APPC, (4) una variante de APPC para el AS/400.

V.2.2.- SERVICIOS DE CICS

CICS es un ambiente de programación contenido por sí mismo. Este proporciona servicios del sistema operativo, servicios de B.D., servicios de presentación, autorización y un ambiente debugging. Muchos sistemas de generación de aplicaciones han sido construídos en la parte alta de CICS para automatizar aspectos del desarrollo y manteniento de procesos. Desde el punto de vista del programador, CICS ofrece las siguientes características:

SERVICIOS DE PRESENTACION

Estos mapean mensajes lógicos de entrada/salida para y desde formatos específicos de dispositivos. Por ejemplo, un mensaje lógico puede ser enviado a la mayoría por cualquier tipo de terminal y CICS producirá el mensaje físico apropiado y lo liberará al dispositivo.

SERVICIOS DE SESION
Estos proveen una interface open( ) read( ) write( ) close( ) para permitir a un programa de transacción conversar con dispositivos inteligentes o procesos.

ALMACENAMIENTO
CICS provee dos diferentes manejadores de almacenamiento: *almacenamiento temporal* y *datos transitorios*. Estos dos manejadores de almacenamientos son la B.D. de alto performance optimizados para almacenar terminales de contexto o colas de mensajes de entrada/salida.

ALMACENAMIENTO TEMPORAL
Los objetos de almacenamiento temporal son opciónalmente transacciones protegidas; esto significa que los cambios de objeto son enganchados en la bitácora a una jornada, y el objeto es reconstruído al reinicio del sistema. El almacenamiento temporal es usado para pasar datos a lo largo de los programas. CICS provee manejadores de recursos para terminales y contexto de transacciones, ruteo de transacciones y regularmente colas de transacción siendo ruteados a otros nodos. Estos manjedores de recursos implementan sus objetos en almacenamientos temporales.

DATOS TRANSITORIOS

Los datos transitorios están en un manejador de almacenamiento diferente con su propio mecanismo de recuperación. Estos clientes principales son varios manejadores de colas provistos por CICS. Los datos transitorios soportan ya sea no recuperación (objetos no protegidos) y recuperación de transacción (recuperación lógica), o rehacer todas las operaciones de la bitácora (recuperación física). Los datos transitorios proveen acceso a dispositovos de almacenamiento externo (cintas, spool de archivos y regularmente impresoras).

ARCHIVOS

Un simple sistema de B.D. secuencial, indexado y accesos de archivos son provistos por CICS como un servicio. Los archivos pueden ser transacciones protegidas o no.

MANEJADORES DE TRANSACCIONES

CICS registra manejadores de recursos, permitiéndoles ligar transacciones, e informes significantes de eventos de transacciones (commit, rollback). Esto resuelve transacciones inciertas para reiniciar. Esto coordina el arranque del sistema, caídas y la terminación de la transacción.

JORNADAS

CICS usa jornadas para recuperación. Consistentes con el diseño abierto de CICS, las aplicaciones pueden crear nuevas jornadas y pueden leer y escribir jornadas. Para reiniciar, CICS opcionalmente invoca un programa, pasando la jornada de transacción abortada. Esto permite a las aplicaciones ejecutar la recuperación para reiniciar.

RECUPERACION

CICS soporta tres formas de recuperación: (1) abort de transacciones, (2) caída del sistema (shutdown) y reinicio, (3) y rompimiento del sistema y reinicio. CICS recientemente en (1991) agregó soporte para recuperación desde una copia de archivo, que permite medios de recuperación si la versión en línea del sistema está dañado. Anterior a esta suma, las colas CICS y los archivos fueron ocasionalmente perdidos, CICS soporta un sistema en hot standby, XRF, en el estilo de IMS (ver V.1.6).

MANEJADOR DE PROGRAMAS

Los programas de aplicación de transacciones son registrados con CICS; después, éstos pueden ser invocados por clientes. CICS maneja la ligación, cargado y ejecución de tales programas. Los programas son escritos para que muchos threads puedan ejecutar un programa concurrentemente. Los programas pueden invocar (llamar) uno a otro directamente o pueden poner mensajes en las colas para invocar un segundo programa asíncronamente (transacción).

THREADS

Cuando un nuevo requerimiento llega (de un cliente o una cola), CICS primero autoriza el requerimiento y entonces crea una ejecución de thread para procesar el requerimiento. Cada thread tiene una prioridad (suma de operador y prioridad de programa). CICS planea threads con un planeador de no prioridad. Si el requerimiento llega vía LU6.2, esto podría llevar un TRID, y el nuevo thread será parte de la transacción llamada.

AUTENTIFICACION Y AUTORIZACION

Tanto el password y una respuesta obligatoria, son mecanismos ofrecidos por CICS para autentificar clientes. Una vez que un cliente es autentificado, esto da un subgrupo de 24 clases de seguridad (la clase 1 es público), es especificada en una tabla de seguridad. Por ejemplo, un cliente en particular tiene posibilidad de obtener un grupo de clase de seguridad, 1,5,9,11,22. Cada uno de los programas de aplicación y cada uno de los otros objetos tienen una clase de seguridad. El cliente deberá tener esta clase en su grupo en órden para accesar el objeto. CICS opcionalmente permite una tercera parte para ejecutar la autentificación.

Estas son las características de programación nativas de CICS. La mayoría de todos estos manejadores de recursos permíten a los clientes agregar lógica al manejador de recursos en la forma de salidas. Por ejemplo, los sistemas de archivos tienen salidas para permitir al cliente agregar nuevos algoritmos hash para cierres asociativos y para agregar encriptación para seguridad de datos. En suma CICS ofrece una interface de operaciones para configurar y operar el sistema, así como una pista y facilidad debugg para permitir a los programadores probar sus programas. También, la interface del manejador de recursos conecta CICS a inumerables sistemas de B.D. (notablemente DL/1, SQL, DB2 de IBM).

V.2.3.- FLUJO DE TRABAJO CICS

CICS es un sistema de PT. Cuando un mensaje llega desde una terminal, CICS examina el encabezado de mensajes y el descritor de terminal para determinar el nombre del programa de la aplicación de la transacción para correr; ya sea el header o el descriptor de terminal puede especificar el nombre. Entonces CICS verifica si el cliente es autorizado para correr el programa de la aplicación de la transacción. Sino, la violación de la seguridad se queda en la bitácora, y el requerimiento es rechazado con mensaje de error. Si el cliente tiene autoridad para correr el programa aplicación transacción, un thread es creado para correr el programa asociado, y la terminal (cliente) es marcado como un "propio" por el thread. El thread, que pasó el mensaje de entrada y nombre de terminal, corre sobre la clase de seguridad del usuario autentificado. Todos los accesos a objetos son verificados nuevamente en la clase de seguridad de los clientes.

El programa de aplicación puede actuar sobre el requerimiento y la respuesta, interactúa con el cliente como una transacción conversacional, o simplemente encola el requerimiento para procesarlo más tarde y regresarlo al cliente. Si el programa aplicación accesa objetos protegidos (transaccional), entonces la transacción deberá eventualmente commit (llamados puntos de sincronía en CICS) o abort. Todos los manejadores de recursos involucrados en la transacción pueden usar jornadas CICS o pueden mantener un jornada privada para la bitácora. Todos los manejadores de recursos que han ligado la transacción serán invocados por el manejador de recuperación CICS para commit y abort. Cuando el programa de aplicación sale a CICS, los mensajes de salida son liberados.

El programa aplicación hace requerimientos a servicios CICS y a otros manejadores de recursos. Todos estos llamados caen en la pantalla de una transacción, llamada unidad lógica de trabajo en CICS. Estas transacciones ofrecen integridad de B.D. e integridad de mensajes.

La integridad de mensajes CICS trabaja como sigue: Si una transacción es protegida, entonces el primer mensaje de entrada y todos los mensajes de salida son de bitácora. Enviando el último mensaje de salida es diferido a la fase 2 de commit. Para reiniciar, el último mensaje de salida de la transacción committed serán resentida si esto no ha sido reconocido. En suma, el primer mensaje de entrada será reprocesado si la transacción fue aborted y si el mensaje de entrada en la bitácora sobrevivió al reinicio. Esta lógica es comparable al mecanismo de integridad de mensajes para IMS. Las colas CICS están fuera de interés. Una cola puede ser dirigida ya sea un programa o una terminal.

Las colas relacionadas a terminal almacenan mensajes de salida dirigidos a terminales ocupadas. Las colas relacionadas a programas causan un programa de transacción para ser invocadas para cuando una cola venga muy llena o cuando un cierto tiempo haya transcurrido (*colas de intervalo de control*). Las colas relacionadas a programas pueden también ser usadas para invocar programas batch. Los programas de aplicación CICS pueden llamar servicios del sistema operativo en host directamente, pero estos llamados deben todos ser sin bloqueo (ellos no deben hacer llamados a la rutina de espera del sistema operativo). Si el llamado did-block, que debe bloquear CICS y el espacio de dirección entero. CICS esta multiplexando este proceso a lo largo de todos los diferentes threads. Ya que la mayoría de los sistemas operativos ofrecen muy pocos servicios de sin-bloqueo, la aplicación deberá llamar CICS y CICS, simulará el servicio del sistema operativo. Cualquier espera involucrada en el servicio permitirá otros threads para ejecutar. Si el sistema operativo ofrece un multi-threading las facilidades del sistema operativo puden se usadas, pero a la fecha el performance de las interfaces thread de propósito general han sido inaceptables.

V.2.4.- PROCESAMIENTO DE TRANSACCIONES DISTRIBUIDAS CICS

Cada uno de los sistemas CICS maneja una partición de la B.D. y la red de terminales. Las transacciones en un sistema CICS ya sea que puedan ser ruteadas a otros sistemas o pueden hacer llamadas a procedimientos remotos a servicios o aplicaciones en otro sistema.

La configuración más simple es tener múltiples sistemas CICS corriendo como procesos sobre un sistema operativo. Estos procesos CICS se comunican vía llamados al sistema operativo basados en memoria compartida. La comunicación memoria-a-memoria es típicamente mucho más eficiente que la correspondiente comunicación LAN o WAN a través de una red de SNA (5,000 contra 25,000 instrucciones por par de mensajes). Esta configuración de memoria compartida es llamada Operación Multi-regional *"MRO"* (Multi-Región Operativo).

MRO permite un gran sistema CICS para ser descompuesto en partes pequeñas, resultando en mejor contenimiento de falla. Como una regla, cada organización obtiene un sistema CICS separado. El debug y la prueba de programa obtienen un sistema separado. Los programas de aplicación en línea y batch son segregados en sistemas separados para que las aplicaciones en línea puedan ser dadas con alta prioridad (Sistema Operativo); mientras aplicaciones batch están dadas con baja (sistema operativo) prioridad.

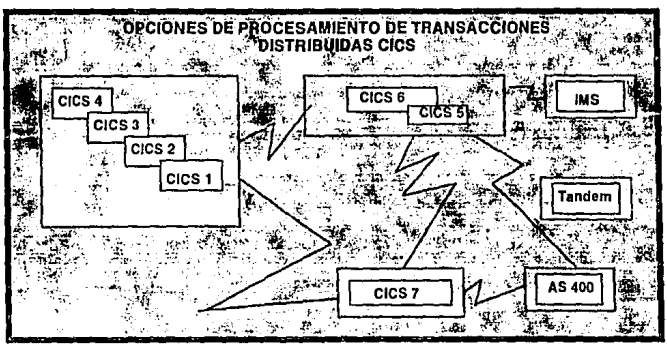

FIGURA V.6.- OPCIONES DE PROCESAMIENTO DE TRANSACCIONES DISTRIBUIDAS CICS. Múltiples sistemas CICS pueden cohabitar la misma máquina y comunicar mensajes vía memoria-a-memoria proporcionadas por el sistema operativo; esto es llamado Operación Multiregión (OMR). Alternativamente, los sistemas CICS que están en múltiples computadoras conectadas por el protocolo WAN de IBM (SNA LU6.2) permite a CICS accesar sistemas remotos y sistemas IMS; esto es llamado Comunicación Inter-Sistemas (CIS).

El principal problema con la descomposición de un sistema CICS en muchas regiones es que el manejo y operación de dos regiones es alrededor de dos veces más difícil que manejar y operar una región. En suma el programador debe estar consiente de que los datos están en la región ya que CICS no provee transparencia de localización. Más aún, si un objeto es remoto (archivo, cola , aplicación), entonces el administrador local deberá darle un alias local y el programa local deberá accer el objeto remoto vía el alias local.

Si los sistemas CICS no son co-localizados, están en un protocolo de comunicaciones LAN o WAN que deberán ser usados. El CICS usa el protocolo SNA de LU6.2 de IBM para comunicarse con otros sistemas CICS. Porque la definición de LU6.2 es pública, a cualquier otro sistema y puede pasar como un sistema CICS. En la figura V.6., diferentes sistemas están particionados en la red CICS. Cuando los sistemas CICS se comunican usando LU6.2, la configuración es llamada (Inter System Comunication *"ISC"*). Las interfaces de programación aplicación MRO e ISC son idénticas y son genéricamente llamadas Procesamiento de transacciones Distribuidas (Distibuted Transaction Processing (DTP)). La aplicación obtiene accesos transparentes a recursos remotos (colas, archivos) accesando objetos locales que están actualmente apodados a objetos remotos. Los accesos a programas remotos es un modelo de envío/recepción, y por lo tanto, es diferente del modelo local de llamada/retorno de programas locales. Esta diferencia da lugar a una tricotomia de procesamiento de transacciones distribuidas:

RUTEO DE LA TRANSACCION

Basado sobre un código de transacción (o colas, o procedimientos escrito por el usuario), un mensaje de entrada puede ser ruteado para una aplicación sobre otro sistema para el procesamiento. Una aplicación de remplazo actúa como sustituto para pasar mensajes entre el cliente y el servidor de aplicaciones.

ENVIO DE FUNCION

La respuesta de sobrenombres, las operaciones CICS sobre datos temporales, colas, terminales y archivos son convertidos a llamados a procedimientos remotos transaccionales (TRPC), que son ejecutados para sistemas remotos. Los llamados son enviados a un espejo, un programa de transacción suministrado por CICS especifica el requerimiento local (ver figura V.7.). El hecho de que el espejo es remoto es oculto del llamador. La comunicación con el espejo es vía a sesión LU6.2.

COMUNICACION PROGRAMA A PROGRAMA

También se conoce como *Comunicación Punto-a-Punto* (peer-to-peer). Si una aplicación quiere invocar un programa de aplicación remoto como parte de esta transacción, esto deberá ALLOCATE el programa de aplicación remoto y entonces ya sea SEND este dato, RECEIVE de datos, o CONVERSE con este (un combinado de SEND y RECEIVE). Una vez que la aplicación está completa es llamado SYNCPT, y la aplicación remota podría votar sobre el punto de sincronización. Este modelo de comunicación programa-a-programa está basado en LU6.2 (ver subsección V.2.5). Si el sistema remoto no soporta el protocolo 2PC, entonces CICS es forzado a usar el último truco del manejador de recursos para permitir que el sistema llegue al coordinador commit. Esto significa que solamente uno de estos sistemas "cerrados" pueda participar en una transacción. CICS usa esta técnica para permitir accesar sistemas IMS. Una transacción CICS puede actualizar atómicamente muchos sistemas CICS y para la mayoría de un sistema IMS como parte de una transacción.

CICS usa un mecanismo de commit heurístico para resolver transacciones dudosas. En caso de que los bloques commit debido a un nodo o red fallen, cada transacción tiene la opción para heurísticamente commit, abort, o esperar. El deafault es para abort heurísticamente. Una segunda opción para heurísticamente commit si la transacción es dudosas. Una tercera opción es para dar derecho, es decir, esperar para que el coordinador resuelva la transacción. Esto requiere mantener actualizado locked uncommitted hasta que la sesión es recuperada y el coordinador anuncia la salida.

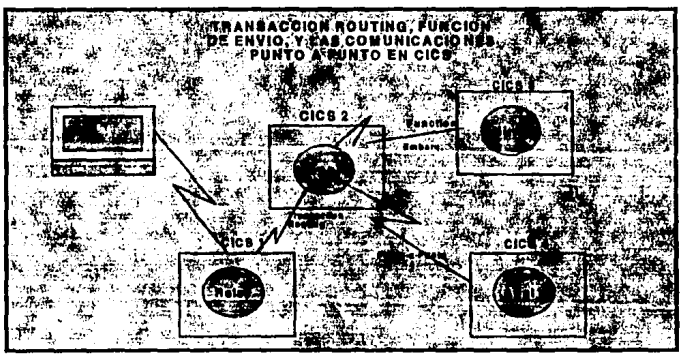

FIGURA V.7.- TRANSACCION ROUTING, FUNCION DE ENVIO, Y LAS COMUNICACIONES PUNTO-A-PUNTO EN CICS. El routing de transacciones usa un transmisor de transacción como medio entre la aplicación y la terminal si la aplicación es remota. La función de envío usa una aplicación espejo para ejecutar operaciones de CICS en objetos remotos. El espejo recibe un TRPC desde la aplicación y invoca la operación localmente. La Comunicación punto-a-punto es usada cuando la aplicación consiste de dos o más programas escritos por el usuario que cooperan para ejecutar la transacción.

Dicha opción está disponible en un caso especial: involucrando transacciones no MRO y consistiendo enteramente para insertar y borrar colas de almacenamiento temporal. El trabajo encolado espera al coordinador para resolver la transacción. Como esta en pie el default de la lógica del commit de CICS no proporciona atomicidad en caso de fallas; ya que estos simplemente detectan inconsistencia para reiniciar si la decisión heurística no es igual que la decisión del coordinador.

V.2.5.- LU6.2

LU6.2 provee Comunicación Transaccional Punto-a-Punto entre aplicaciones corriendo bajo dos manejadores de transacción (típicamente CICS). En terminología de redes de IBM, éstas son unidades físicas (PUs, o Cajas), y unidades lógicas (LUs, o programas). CICS, por ejemplo, es una unidad lógica. Algunas unidades lógicas no tienen estándar; ellos hablan en común, el protocolo de nivel de sesión, y el también llamado LU.0. Otras unidades lógicas tienen una personalidad específica. Por ejemplo, LU1 es una impresora, y LU2 es un monitor IBM 3270 con este programa de entrada-salida orientado a formas. El original LU6. fue un protocolo inventado por CICS hace mucho tiempo para describir esta personalidad como un monitor TP. El protocolo fue revisado en 1980, y el resultado es llamado LU6.2.

Cada Monitor de PT y terminal en la red tiene un nombre único, es llamado con el nombre de LU. El nombre tiene el formato general red.nodo. Un nombre típico para un sistema CICS corriendo sobre una workstation AIX (UNIX) Bruece Lindsay tiene que ser IBM-Almaden Lindsay. Un nombre típico para un sistema CICS corriendo sobre una máquina servidor para Yorktown tiene que ser IBM_Yorktown.TestCICS.

Los programas de transacción son registrados con el Monitor de PT. Suponga que la transacción débito/crédito fue registrado bajo el nombre de programa transacción DebitCredit en el servidor sistema CICS. Entonces un programa corriendo bajo CICS para una workstation de Bruce Lindasy´s podría invocar este programa ejecutando el verbo CICS.

```
EXEC CICS ALLOCATE DebitCredit  IBM_Yorktown.testCICS.
          DATA " este es el startup string"
          USERID "Bruce Lindsay"
          PASSWORD...
          RETURNING sessionid;
```

Esta operación tiene muchos efectos. Este enviá un mensaje desde Almaden al LU. nombrado para Yorktown para establecer una sesión para este servidor. El mensaje lleva la autorización del cliente (un password, o el primer paso de un protocolo de respuesta de desafío), el nombre del servicio deseado (DebitCredit), y muchos otros parámetros.

El servidor autentifica el cliente, observa el nombre de servicio (nombre del programa de la transacción), verifica la autoridad del cliente para accesar el servicio, forza un proceso (thread) ejecutando este servicio, registra el trid que está entrando (pasado por el cliente) con el manejador transacción local (un componente CICS), liga el servidor a la transacción y entonces despacha el servidor con un parámetro punteado para el primer mensaje y otro punteado a la sesión LU6.2. conectando el servidor thread al cliente. Sobre el lado del cliente, el ALLOCATE coloca al cliente a la mitad de la sesión y regresa a una sesión ID para el cliente, tal que el cliente pueda enviar mensajes al servidor.

Cuando un programa de transacción invoca SYNCPT, este mensaje es enviado sobre cada conversación anexada al programa de transacción. Así cada vecino obtiene un mensaje TAKE SYNCPT. Este se repite hasta que todos votan "sf". En suma, CICS esta colectando votos desde los manejadores de recursos locales ligados a la transacción.

Las sesiones SNA son expansivas salidas para repartir, como los procesos que son expansivas salidas para crear. Consecuentemente, CICS nunca trata de durar para distribuir ALLOCATE una sesión. Ya que, ésta mantiene pools de sesiones predistribuidas para varios LU's. Cuando un requerimiento ALLOCATE llega, CICS trata para customize una sesión preexistente (afinando el servidor y algunos parámetros) ya que reparte una nueva sesión completa. Esto es para hacer distinción entre sesiones, que son pipes de comunicación bidireccional entre LUs, y *conversaciones*, que son sesiones transaccionales entre un cliente y un thread de programa de aplicación servidor. Estas sesiones de bajo costo y alta función son llamados *diálogos* en OSI-TP.

Los servidores LU6.2 usualmente soportan algunos servicios estándar; creación de procesos, operaciones de cola, operaciones de archivos planos, operaciones DL/1 y operaciones SQL. Los sistemas de PT particulares actúan en otra forma. Por ejemplo, CICS tiene el espejo y los servicios de relevo. De especial interés es un servicio llamado ***"CNOS"*** (Cambio de Números de Sesiones). Este pequeño programa arrancó fuera como una forma para que el sistema CICS negocie con el vecino sobre el tamaño y polaridad del pool de sesión predistribuido usado para conversaciones. Gradulamente, CNOS creció en función para incluir casi todos los parámetros de configuración CICS. En particular, esto soporta las siguientes operaciones:

Instalar un nuevo programa de transacción.
Leer y alterar la lista del control de acceso sobre un programa transacción.
Leer y alterar el arranque de CICS y los parámetos de control de carga.
Resolver transacciones dudosas y decisiones heurísticas para reconnexión.

Porque CNOS es un servicio, cualquier cliente que sea conveniente es autorizado para poder manejar todos los sistemas CICS en una red grande invocando programas CNOS a los nodos para ser manejados.

LU6.2 tiene muchas características y lo principal aquí es sobre transacciones, es importante mencionar que cada conversación es opcionalmente transaccional. Cada conversación tiene de 1 de 3 niveles de puntos de sincronía especificados cuando estos son distribuidos:

Nivel de Sincronía 0.- No hay integridad de mensajes más allá de la secuencia de números para detectar duplicados o perdidas de mensajes.

Nivel de Sincronía 1: Soporte para los verbos CONFIRM-CONFIRMED que permiten al cliente y servidor hacer reconocimiento fín-a-fín.

Nivel de Sincronía 2: Soporta para el verbo SYNCPT, que provee propiedades ACAD a través de transacciones distribuidas.

La mayoría de las implementaciones de LU6.2 no son soportadas en el Nivel 2 Sincronía. Las implementaciones IBM sobre CICS y VM/370 son las excepciones para esta regla: ellos soportan el verbo SYNCPT.

V.3.- X/OPEN DTP, OSI/TP CCR

La Organización Internacional de Estándares (International Standards Organization "ISO") tiene definido un protocolo estándar que permite Sistemas de Procesamiento de Transacciones para interoperar. Este protocolo tiene dos partes: Interconectar Sistemas Abiertos - Procesamiento de Transacciones (Open System Interconnect - Transacction Processing "OSI-TP") y un segundo estándar de Interconexión Sistemas Abiertos - Recuperación y Control de commits (Open Systems Interconnection Commit Control and Recovery "OSI-CCR"). Juntos definen formatos para identificadores de transacciones globales y definen estándares de formáto de mensaje 2Pc y un protocolo correspondiente [OSI-TP 1992, OSI-CCR 1990]. Este protocolo de commit opera en la parte alta del sexto arreglo de OSI pero por otro lado es muy similar al LU6.2 que opera en la parte alta del protocolo SNA de IBM.

Estos estándares ISO-OSI definen interfaces de mensajes para permitir a las computadoras interoperar. Una interface de programación de aplicación estándar es necesaria para permitir a los programas de aplicación, manejadores de recursos o manejadores de transacciones ser portados y recompilados desde un sistema a otro. X/Open tiene una definición de subgrupo para tales interfaces de programación de aplicaciones: el Procesamiento de Transacciones Distribuidas X/Open (X/Open DTP). Esto tiene definido el concepto de programa de aplicación, manejador de transacción, manejador de recursos, y manejador de comunicaciones. Esto tiene definido interfaces estándar tal que las aplicaciones portables puedan ser escritos.

El estándar X/Open tiene la intención de permitir un manejador de transacciones para usar protocolos de commit propietarios internamente y usar los protocolos OSI-TP cuando interoperen con manejadores de transacción escritos por otros vendedores. Estas interfaces de programación permiten aplicaciones portables y manejadores de recursos portables.

A la Inversa, de esos estándares capturan los algoritmos principales implementados por la mayoría de los sistemas de PT. El estándar X/Open DTP tiene algunos componentes principales que son constantes, mientras otros aspectos están rápidamente evolucionando. La mayoría de las partes controversacionales del diseño X/Open DTP son intencionados para definir un ambiente de monitor de PT (servidores y servicios) y los debate sobre los méritos relativos del llamado a procesadimiento remóto transaccional (TRPC) y las comunicaciones punto-a-punto (peer-to-peer "PTP"). En el presente, ambas aproximaciones son representadas en el estándar. Estos son debates considerables acerca de como los manejadores de comunicaciones deberían encontrarse dentro del estándar.

El modelo X/Open postula que cada manejador de recursos (RM), tiene su propia bitácora y su propio manejador de recuperación. El manejador de transacciones X/Open puede ser brevemente caracterizado por la siguiente lista de propiedades:

Protocolo Commit.- Usa un 2PC, y un protocolo de commit de presumido-abort.

Rooted.- Solamente la raíz puede commit la transacción.

Optimizado.- Soporta la optimización de lectura-solamente.

Heurístico.- Soporta decisiones heurísticas.

Branches.- Soporta subtransacciones, pero la transacción abort si cualquier branch abort.

Opción de cadena.- Las transacciones pueden arrancar encadenadas explícitamente.

Timeout.- Limitar el tiempo de ejecución transacción.

OSI/TP.- Los protocolos podrían ser usados cuando se comuniquen con diferentes manejadores de transacciones.

X/Open permite a las aplicaciones proveer servidores de aplicaciones transaccional (procesos), que ofrecen servicios transaccionales. Los servicios pueden ser llamados a procedimientos remotos o full-duplex, conexiones punto-a-punto (en la manera de diálogos OSI/TP). Las conexiones RPC y PTP pueden ser transaccionales o no.

Los RPC pueden ser asíncronos o no. X/Open no tiene una clara noción de servidor de clases. La configuración del servidor es específica del vendedor. En la figura V.8. muestra el modelo X/open DTP, en este modelo se han basado los actuales OLTP del mercado-

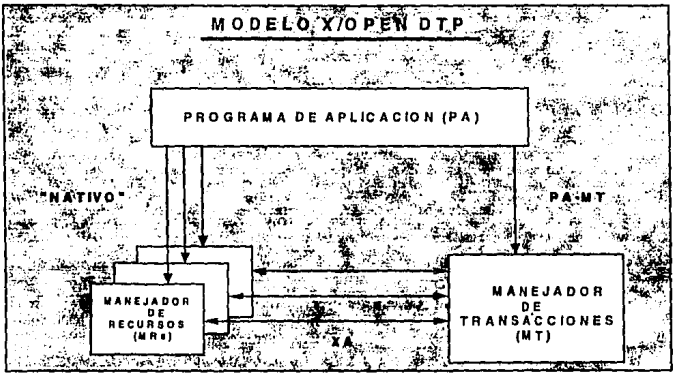

Figura V.8.- MODELO X/OPEN DTP. Este modelo incluye los tres elementos básicos: el Programa de Aplicación "AP", el Manejador de Recursos "MR" y el Manejador de Transacciones "MT". La aplicación es el componente que determina cuando una transacción distribuida comienza o termina. La aplicación una al MR para ejecutar el trabajo para hacer parte de la transacción distribuida. La aplicación simple finaliza una transacción con un comando de commit o rollback. El MT procesa la transacción de los llamados de comienzo y final desde la aplicaicón y transporta esta inforamticón en el manejador de recursos. El MT también coordina el 2PC y el procesamiento de recuperación de la transacción. el MR maneja una cierta parte de los recursos compatidos. Las redes de comunciación y B.D. son ejemplos de recursos compartidos. La interface AP-MT comunica al AP con el MT y la interface XA comunica al MR con el MT. El modelo no especifica una interface entre la AP y el MR, sino que la llama "nativo" por que no pertenece al modelo.

Las comunicaciones externas pueden ser vía un mecanismo RPC o PTP o vía un protocolo de comunicaciones específico del manejador (ejemplo DECnet, LU6.2, Encina sobre TCP/IP). La estructura de X/Open puede ser mejor explicado por primero observar el case local para un simple proceso de programa de aplicación ejecutando una aplicación que involucra varios manejadores de recursos locales. Una vez que este simple caso es entendido, la discución de una aplicación distribuida puede proceder como una extensión para el modelo básico.

El modelo de OSI TP es modelo de referencia define los principios de los sistemas abiertos. Con OSI TP una transacción puede contener muchos participantes. Cada nuevo participante en una transacción es llamada un nuevo identificador de la transacción. Una transacción puede ser representada en una estructura de árbol que represente el nodo raíz y de tres participantes que paticipan y inicializan una transacción. Cada nuevo identificador va expandiendo el árbol.

Un nodo es creado del resultado de los identificadores, y estos crean a su vez nuevos identificadores llamados nodos intermedios. Un nodo que no tenga identificador es llamado nodo de terminación (hoja).

V.3.1.- EL CASO LOCAL

El diseño X/Open en términos de programas de aplicación está destinado a un manejador de transacción y a un manejador de recursos ejecutándose en un simple proceso. El estándar X/Open habla en términos de compilación y ligación de todos estos programas juntos como parte de los procesos de aplicación creados.

Las primeras llamadas de aplicación llama tmopen( ) para conectar al manejador de transacción. Después, este puede llamar tmbegin( ) para comenzar una transacción, tmabort( ) para abort, tm commit( ) a commit, y tmsync( ) para encadenar una nueva transacción (tm commit( ) suma tmbegin ( )). La aplicación se desconecta desde el manejador de transacciones por el llamado tmterm( ). Estas rutinas comprenden los aspectos locales de la interface también llamada TX, la interface entre una aplicación cliente y los manejador de transacción local.

Cada manejador de recursos residiendo en los procesos proveen una lista de callbacks a los manejadores de transacciones. Estos callback, llamados el switch de manejador de recursos, son definidos cuando la imágen del proceso es construído (procesos dirigídos o tiempo de liga). Esto es un aprovechamiento estático para ligar procedimientos.

Las interfaces MT-RM son generéricamente llamadas la interface XA. El MR switch lleva el nombre de MR; este también tiene una bandera que indica si está automáticamente el manejador de recursos ligando todas las transacciones en los procesos, o si solo liga algunos de ellos. Esto es llamado registro estático ó dinámico.
Si el registro estático es requerido, cada nueva transacción genera en los procesos una llamada xa_start( ) al MR. Otro switch bandera indica si el MR; está dispuesto a ejecutar operaciones asíncronas que regresen inmediatamente al MT. El MT puede usar estas operaciones para incrementar en las operaciones de commit y abort paralelamente. El callback xa_complete( ) permite al MT esperar a que termine y cumpla con las operaciones asícronas del MRS.

Para xa_open( ), el MT llama al MR, preguntándole por la lista de las transacciones resueltas heurísticamente o dudosas. Para cada transacción dudosa para la cual el MT conoce el outcome, estos llaman a los callback xa_commit( ) o xa_abort( ). Para transacciones que fueron heurísticamente resueltas, el MT llama xa_forget( ) una vez que tiene trato con esta transacción. El MT ejecuta operaciones no explícitas en duda con respecto para el MT; estos permanecen dudosas con respecto al todavía MR.

Si el switch RM indica que el manejador de recursos liga dinámicamente transacciones, entonces el tmbegin( ) deberán no llamar al manejador de recursos, y el manejador de recursos debería llamar gtrid-reg( ) sobre primera invocación por cada nueva transacción. Este es el equivalente de John W( ) en el capítulo 10 y 11. Un manejador de recursos puede solicitar la optimización de sólo-lectura a un switch de time o puede solicitarlo dinámicamente como el progreso de la transacción por no registrarse, o no ligarse, la transacción.

El diseño de identificador de transacciones del X/Open DTP es heredado del mejor diseño OSI-CCR. Los trids X/Open y OSI-CCR son llamados identificadores de transacciones global, o gtrids. Lo más curioso es que cada MT está participando en una transacción que crea un diferente gtrid para la transacción, y cada manejador de recursos puede opcionalmente obtener un único gtrid (cada diferente gtrid es llamado un branch de la transacción). Los trids globales, entonces, no son ralmente globales después de todo, si cualquier gtrid abort, todos gtrids relacionados deben abort; consecuentemente, ellos no deberán no ser confundidos con puntos de salvación de la transacción o con transacciones anidadas. Uno puede pensar del gtrid de la transacción raíz como el gtrid de la transacción entera, pero generalmente ninguno de los otros MTs siempre oyen acerca de este gtrid.

Ya que los gtrids son grandes (requieren de 32 bytes para ser almacenados y de 64 bytes para ser desplegados), ellos probablemente contienen lotes de información extra (semejantes a un identificador branch). Consecuentemente, las aplicaciones no son permitidas para ver dentro gtrids. El manejador de transacción proporciona interfaces para comparar, hash y desplegar gtrids.

En realidad, X/Open DTP es poderosamente armado para procesos multithreaded. Las especificaciones X/Open usan el término thread exclusivamente, (cada thread participa en una sóla transacción en un tiempo, pero el proceso entero contiene varios threads semejantes).

V.1.2.- EL CASO DISTRIBUIDO: SERVICIOS Y SERVIDORES

X/Open define las nociones de servidores y servicios. La especificación es muy vaga de como estos están indefinidos para el monitor PT. Este es un punto de vista de las prioridades y de las funciones administrativas, pero sobre todo el diseño es incompleto. Por ejemplo, estas no son nociones de configurar un servidor de clases.

Dos tipos de servidores son reconocidos:

Peer.- Usa algún mecanismo de comunicación punto-a-punto nativo desconocido para el MT.

Servidores.- proveen un servicio, el cual es un proceso (thread) manejado por el MT. Los servidores proveen ya sea servicios TRPC, significando que este llamado es libre de contexto y tiene un mensaje de respuesta, o ellos proveen servicios conversacionales. Si un servicio es conversacional, (esto es, si lo mantiene el contexto cliente), entonces deberá usar un protocolo de comunicación puto-a-punto X/Open para comunicarse con el cliente. El protocolo LU6.2 X/Open evoca que no soporta el concepto de un TRPC de contexto-sensitivo .

Para servidores (TRPC o conversacional), el MT actúa como la rutina principal para el servidor. Este poleo de mensajes de entrada, maneja el identificador de transacción, y invoca el servicio.

Los clientes invocan servicios de TRPC usando la rutina tmcall( ) de X/Open. Esta rutina proporciona dos áreas de buffer: un mensaje de peticiones y un buffer de mensajes replica. La puesta en marcha de los parámetros no están definidos. Si el llamado cruza sistemas heterogéneos, la transacción de datos de los requerimientos y las réplicas pueden ser requeridas, pero esta no es parte especificada por el protocolo X/Open DTP. Estos puntos son mencionados para recalcar la naturaleza del diseño preliminar.

Esto parece probablemente que las aplicaciones usarán otros protocolos punto-a-punto y TRPC. Los estándares X/Open y OSI-TP están bién posicionados para permitir a los protocolos registrar trids outgoing y incoming. Este mecanismo es llamado registro branch en OSI-TP, y esta interface de programación de aplicación es llamada la interface XA + (ó interface CM) en X/Open DTP.

V.3.3.- ENCINA

Encina es un producto diseñado por la Compañía Transarc Corporation y está integrado con el *"DCE"* (Distributed Computing Environment) de *"OSF's"* (Open Software Foundation). Transarc es la primera empresa que libera soluciones OLTP basadas en el DCE. DCE de OSF establece la base para construir soluciones OLTP distribuidos, con datos compartidos. El DCE ofrece la comunicación necesaria y servicios de recursos compartidos proporcionando una plataforma común de seguridad de desarrollo y aplicaciones distribuidas. En enero de 1991 Transarc Corporation anunció la tecnología para sistemas abiertos (OLTP) basada sobre la DCE de OSF. El objetivo de la DCE de OSF es proporcionar una infraestructura común para el procesamiento distribuido. El DCE maneja primariamente la función cliente/servidor, localidad de los recursos, seguridad, escalabilidad y recursos compartidos.

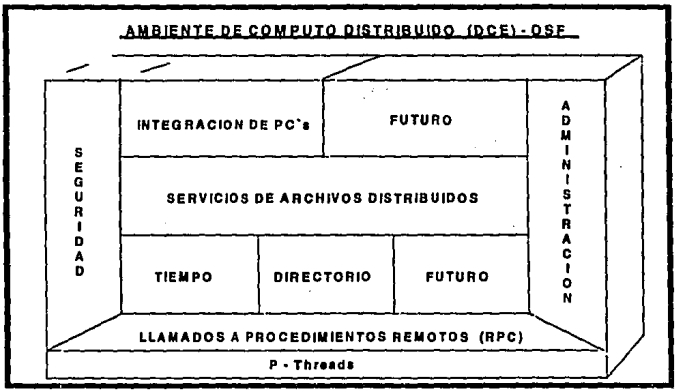

FIGURA V.9.- AMBIENTE DE COMPUTO DISTRIBUIDO DCE - OSF. El DCE incluye un servicio nombre que sirve para la localización los servidores en el ambiente, un servicio de seguridad que soporta la autenificación, autorización y encriptación servicio de tiempo distribuido para las discrepancias direccionadas entre las reglas para manejar los accesos de listas de control, un soporte de integración PC/LAN, un servicio de archivos distribuidos donde los usuarios y aplicaciones pueden accesar archivos almacenados remotamente de la misma manera que ellas sean accesadas localmente. Finalmente opciones en la arquitectura DCE para insertar en el futuro tecnologías con características, mejorando la computación distribuida base.

Transarc ha desarrollado los productos de la familia Encina para el procesamiento de transacciones que simplifican la construcción de sistemas distribuidos confiables y para proveer la garantía de la integridad requerida para Empresas de Computación con negocios críticos.

Encina introduce una serie de productos para el procesamiento de transacciones distribuidas, estas combinan avances de tecnología, adhieren estándares y modularidad. Estos productos proporcionan modernos paradigmas de programación (por ejemplo multi-threading y RPC), tecnologías avanzadas (Transacciones anidadas, interoperabilidad con LU6.2) runtime y servicios administrativos necesarios para redes de computadoras para ejecutar procesamientos complejos en datos distribuidos y heterogéneos.

Encina está diseñado para usar threading, comunicación, nombramiento, y seguridad en el Ambiente de Computación Distribuida DCE de OSF. Estos están también siendo portados a sistemas operativos que no son UNIX.

En la actualidad Oracle tiene soporte para los productos en línea Encina y la integración de Oracle RBDMS con monitor de procesamiento de trancciones. La integración de estos productos son llevadas a cabo a través de la interface XA, y con la especificación de X/Open facilita la integración de los manejadores de recursos y sistemas de procesamiento de transacciones. Estas combinaciones proporcionan un alto performance OLTP para sistemas UNIX basadas en el DCE de OSF que puedan accesar datos almacenados en ORACLE. Encina interactúa con otras B.D. como Sybase, Informix, Ingres y otros.

La configuración del sistema es ejecutado vía UNIX, DCE y herramientas Encina. Encina no viene con un repositorio o sistema de B.D. Esto ha sido "interfaseado" para cumplir con los sistemas de B.D. X/Open tales como Oracle e Informix.

El monitor Encina interopera y está integrado con herramientas diseñadas para desarrollo de aplicaciones 4GL (como por ejemplo JAM de JYACC). El modelo cliente/servidor (función de envío) es un servidor aplicación que exporta una función particular o servicio al cliente aplicación.

Transarc incluye soporte para principales vendedores de hardware como: IBM RS/600, CICS para AIX, HP9000 (HP/UX), HP3000 (MPE), Stratus Computer, Digital DECstation, SUN SPARCstation.

Transarc proporciona código fuente para vendedores de hardware. El precio del código fuente es en un rango de $25,000 a $75,000 dolares o U.S. dlls por productos individuales Encina.

Para efectuar los requerimientos anteriormente descritos, Encina introduce un set de productos y servicios principales para OLTP que incluyen:

- **Toolkit:**
  - **Executive.**
  - **Server Core.**

- **Encina Monitor.**
- **Encina Structured File Service (SFS).**
- **Encina Peer-to-Peer Comunication (PPC) Executive.**
- **Encina Peer-to-Peer Comunication (PPC) Gateway/SNA.**
- **Encina Recoverable Queuing Service (RQS).**

El Encina **Toolkit** soporta el desarrollo de aplicaciones de procesamiento de transacciones cliente/servidor, proporciona invocación remota transaccional (TRPC), procesamiento distribuido 2Pc; y la administración de datos recuperables ofreciendo integridad en transacciones; estos accesan y actualizan los datos en base a las propiedades ACAD. El **Encina Toolkit Executive** tiene una serie de componentes principales, definiciones de API's que permiten definir clientes y servidores transaccional. Estos componentes son los siguientes:

Transactional-C "*Tran-C*".- Es un API que simplifica y maneja la demarcación de la transacción, control de comunicaciones y el manejo de excepción. El Tran-C esta hecho de macros y rutinas de librerías para el C estándar/ANSI.

Transactional RPC "*TRPC*".- Es una extención del DCE RPC con semánticas de transacción. Cuando un TRPC falla, la transacción incluída es aborted. Cuando una transacción es completa satisface a todos los participantes, de lo contrario es un abort que será emitido causando alguna modificación local o remota para ser "rolled back". Cuando un RPC falla, el cliente generalmente no puede determinar si los mensajes nunca llegaron al servidor, o si el servidor falla, o cuando un mensaje fue perdido no retorna.

Ambiente de Desarrollo Base (Base Development Environment "*BDE*").- Aisla los sistemas operativos dependientes de los productos Encina. La tecnología Encina se encuentra entonces portando por reimplementación los llamados BDE (POSIX threading, archivos e/s, localización de memoria, etc.) con un nuevo sistema operativo.

El Distributed Transaction Service "*TRAN*".- TRAN soporta múltiples mecanismos de comunicación simultánea, una transacción envía peticiones a los servidores en un número de diferentes caminos y ofrece el estilo de 2Pc sobre todos los participantes.

En realidad TRAN proporciona soporte para la integridad de múltiples servidores de recuperación, en el cual cada uno proporciona los servicios rollforward/rollback para datos recuperables. Sobre un protocolo 2Pc, una transacción coordina los votos de todos los participantes. Si son conocidos entonces son preparados para commit, el coordinador hace commit la transacción pendiente. Algunos participantes fallan, entonces el coordinador aborta la transacción. El coordinador informa a todos los participantes de lo sucedido, a esto se le conoce como el protocolo 2P commit presumido abort.

Transarc es compatible del ambiente de programación Transaccional-C, es una extensión para el lenguaje de programación C el cual facilita la escritura trasacciónal de clientes y servidores. Las extensiones son en parte sintácticas, usando macros C y parte de una extensa librería de procedimientos para manipular transacciones y almacenamiento transaccional. La básica extensión sintáctica es para autorizar la declaración de transacciones, subtransacciones, y "manejo de excepción".

Esta sintáxis tiene muchas variaciones. Un manejador de almacenamiento transaccional le sigue la trayectoria al almacenamiento distribuido y libres si la transacción aborta. El almacenamiento de programación no permanente es proporcionada directamente en Transacción-C; cuando un proceso falla, este estado no es reinstalado. Las aplicaciones pueden usar archivos, colas, B.D, u otros manejadores de recursos transaccional para almacenamiento permanente. Transaccional-C es designado para programación multi-threaded. Cada thread para procesos multithreaded pueden tener un separado identificador de transacción. Los Threads pueden ser creados con una sintaxis semejante a:

```
concurrent {
        subTran <block>
         ...
        subTran <block>
        }
        onCommit <block>
        onAbort  <block>
```

Threads pueden también ser creados por llamados a un sistema de rutina de librerías. Transarc hace considerables esfuerzos para hacer las librerías thread-safe. Este permite múltiples threads para llamar la misma rutina de librería al mismo tiempo sin confundir las variables globales para esas librerías. Encina provee distintos mecanismos de sincronización para threads dentro de unos procesos. Por otro lado el manejador de seguros son mecanismos de semáforos.

El Encina Toolkit Server Core. Se compone de los siguientes servicios:

- El Servicio Lock "***LOCK***".- Proporciona un paquete lógico de locking para implementar la seriabilidad en la transacción. Se aquieren seguros transaccionales (lo seguros son sostenidos hasta que se presente un commit o abort).

- Servicio de Recuperación "***REC***".- Proporciona la lógica UNDO/REDO requerida para implementar un rollbarck después de abort y un roll-forward cuando se se presenten fallas en el sistema

- **Servicio de Volumen** "***VOL***".- Incluye un manejador de volumen que mapea segmentos de múltiples discos físicos a un disco lógico. El manejador de volumen puede ser instruído para duplicar discos físicos así que el volumen lógico esté altamente disponible. Un manejador de buffer orientado a bloques provee de operaciones de lectura y escritura sobre volumenes lógicos.

- **Servicio de Bitácora** "***LOG***".- Proporciona en la bitácora una escritura (write-ahead) para almacenar las transacciones ocurridas y actualizadas para los datos recuperables. La Interface TRAN/XA (MT/XA) permite accesos transaccional a XA soportando B.D. semejantes al RDBMSs. La interface XA es un estándar de X/Open para inicializar y coordinar transacciones en las B.D.

El Monitor Encina .- Proporciona servicios que complementan al ambiente abierto distribuido OLTP (configuración, administración, seguridad, performance garantizado y soporte a herramientas de integración). Las características del monitor Encina supera en tres áreas de soporte a aplicaciones de procesamiento de transacciones:

- **Ambiente de Desarrollo.**- Proporciona APIs y herramientas necesarias para sistemas de procesamiento de transacciones distribuidos desarrollados interconectados. Las interfaces de programación y aplicación (TRAN-C, TRPC) son portables a ambientes del monitor. El monitor incluye soporte para interactuar con herramientas front-end y 4GL´s. Ofrece librerías, extensiones, formas de editor (WYSIWYG), construcción de rutinas en transaccional-C para ambientes gráficos (MOTIF) y 4GL (JAM).

- **Ambiente de Ejecución.**- Liga transparencia ofrecida por el monitor, extiende los servicios ligados ofrecidos por el DCE para automáticamente ejecutar balanceo de cargas a través de repetir caso de un servidor y desviar el trabajo cuando un servidor falla. Adicionalmente garantiza el performance, es proporcionado a través de la prioridad de planeación de servidores monitor: sobre el modelo RPC (multi-threaded) servidores están típicamente esperando para ser disponibles de respuestas inmediatas.

Como quiera que sea a través del uso de servicio de encolado Encina las tareas pueden ser encoladas y programadas por orden de prioridad. El Encina Monitor también simplifica la seguridad comprometida para protección de cada servidor con accesos administrados centrales de listas de control para evitar ejecución de servidores aislados para un cliente en particular durante los accesos (a través de un solo cliente puede emitir estilos simultáneos de respuesta para el servidor). En este camino, sólo un cliente es activo con un servidor a tiempo, por medio de eso reduce la posibilidad de interferencias.

Ambiente Administrativo.- El administrador de este ambiente tiene constantes accesos que deberán ser autentificados para la entrada a los sistemas de procesamientos de transacciones distibuidas. Estos incluyen la capacidad para activar clientes del monitor, servidores disponibles y cargas autorizadas, información auditada. Administra los "shutdown" de la máquina. Cuando la máquina es restaurada, estos accesan la configuración de la B.D. mantenida por el monitor para determinar que servidores deberán ser inicializadas.

Un manejador de colas es prometido para una versión de Encina posterior. El Monitor de PT Encina actúa como un intermediario para llamadas a procedimientos remotos transaccional arrivando de los clientes. Los Trids son etiquetados en cada TRPC Encina rutea mensajes de clientes servidores. La seguridad está basada en el diseño Kerberos de DCE para autentificación, fín-a-fín (end to end) en encripción para privacidad y las B.D DCE usuario/grupo. El balanceo de carga esta hecho por una combinación de "randomización" y encolamiento DCE RPC.

Encina Structured File Service "SFS". Ofrece múltiples características que el sistema de archivos UNIX y el sistema de archivos distribuidos DCE sufren de numerosas limitaciones para soportar aplicaciones de procesamiento de transacciones. Estas características son: archivos de registros orientados para soportar APIs de archivos ISAM y VSAM integridad transaccional SFS soporta un número grande de concurrentes usuarios, archivos grandes para ser extendidos en múltiples discos. El SFS soporta múltiples índices secundarios con accesos transaccional y no transaccional.

El sistema de archivo estructurado de Encina provee registros con archivos. Esto es implementado usando el seguro, transacción, recuperación, y servicios de volumen ya descritos. Los archivos pueden ser de entrada secuenciada, relativa, o llaveada (B tree). Los archivos pueden tener índices secundarios. Los indices "hashed" no son directamente soportados. Cuando se abre un archivo, el cliente puede especificar el grado de aislamiento: browse, estabilidad del cursor, lectura repetida. La actualización a archivos es cubierta por técnicas de la bitácora. Encina soporta ambos, recuperación de reinicio del sistema y recuperación de medio de los archivos.

Encina Peer-to-Peer Comunication "PPC" Services.- Las aplicaciones Encina transaccionalmente accesan y actualizan datos en mainframes. Similarmente, los mainframes deberán estar disponibles para descargar la computación transacción en plataformas Encina. Encina soporta Comunicaciones Punto-a-Punto que son vistas como una alternativa para RCP y TRPC, lo conveniente de los servicios PPC Encina es la interoperabilidad con sistemas existentes. Este servicio ofrece dos productos que dan soporte a las Comunicaciones Punto-a-Punto:

Encina Peer-to-Peer Comunication "*PPC*" Executive. Contiene los APIs para emitir PPC Lu6.2, así como la implementación del LU6.2 sobre TCP/IP. Los API's de los servicios PPC consisten de bibliotecas de rutinas que pueden ser usadas por Transactional-C o C.

Encina Peer-to-Peer Comunication "*PPC*" Gateway/SNA. En esta opción el Gateway PPC proporciona interoperabilidad transaccional sobre un protocolo SNA implementación de LU6.2, semejante a CICS de IBM, que puede coordinar o ser coordinado por las plataformas Encina.

El Monitor de PT Encina soporta un gateway DCE RPC para comunicación al mundo no-IBM, un gateway SNA-LU2 para comunicación al IBM (CICS, IMS, AS/400), y un gateway X/Open para integración de otros manejadores de recursos. Estos gateway permiten sistemas transacciones heterogéneas multisite y la integración de sistemas de B.D. comerciales en un sistema de procesamiento de transacciones integrado. Heurísticos commit, commit para transferir, sólo-lectura, y muchas otras optimizaciones son soportadas. El X/Open XA accesa mapeos para los identificadores de transacciones Encina para trids X/Open. Cuando transacciones anidadas son usadas, cada substransacción recibe un nuevo branch identificador. Este no se aproxima al trabajo de LU6.2, así de esta manera transacciones anidadas no pueden viajar en estas ligas.

Encina Recoverable Queuing Service "RQS". Este servicio permite tareas transaccional para ser encoladas y después ser procesadas. El RQS proporciona múltiples niveles de prioridad y escalas para soportar numerosos usuarios y altos volumenes de datos. Provee el encolado y desencolado transaccional de los datos.

Esto da una forma más general de nombre de seguro (un espacio de nombre nivel-dos hasta para 512 caracteres) y provee necesidad-transacción asegurado con distintas formas para sucesión y anti-sucesión. Esto mantiene contadores de seguro sobre una base por-substracción pero no provee conversión de seguros automática. El manejador de seguros de Encina toma una mayor flexibilidad para archivado de bitácora. Cada tipo de registro puede ser direccionado a un flujo diferente de archivo.

Encina soporta transacciones anidadas y commit dependientes de ellos. Encina provee las interfaces y estructura fuera de línea por la evolución del procesamiento de transacciones distribuidas X/OPEN. Donde la interface es indefinida, Encina toma el interesante aproximamiento de implementar todas las alternativas y permitiendo a los clientes seleccionar el funcionamiento deseado.

El manejador de recuperación Encina ejecuta una búsqueda REDO-UNDO de la bitácora para reiniciar e invocar los "callbacks" del manejador de recursos para manejar sus REDO-UNDO para reiniciar. Este mismo mecanismo es usado para abort la transacción y para recuperar. Encina no provee soporte para puntos de salvación, sino que efectúa soporte de transacciones anidadas.

V.3.4.- TOP END

En 1963 NCR introduce el primer sistema para procesamiento de transacciones distribuidas para soportar el mercado bancario. El volumen de trabajo ejecutado en una máquina de cajero, mientras el mainframe de NCR proporciona accesos a archivos centralizados. Desde ese tiempo NCR ha ofrecido muchas generaciones de monitores de OLTP. El sistema de administración OLTP propietario es de los más recientes de NCR que es el producto MULTI-TRAN, el cual ofrece soporte para la NCR 9800. NCR tiene 25 años que comercialmente ha proporcionado tecnología distribuida para el procesamiento de transacciones "DTP" y que ahora ha culminado con la creación de TOP END.

TOP END es un sistema de administración de transacciones distribuidas diseñado especialmente para encontrar los requerimientos de sistemas de procesamiento abierto. NCR ofrece productos e interfaces para sistemas abiertos, a través de la Arquitectura de Computación Cooperativa Abierta (Open Cooperative Computing Architecture ***"OCCA"***). TOP END es la culminación de la tecnología comercial que proporciona procesamiento de transacciones desarrollada por NCR. El objetivo primario de TOP END es llevar las capacidades de clases de mainframes para procesamiento de transacciones distribuidas a los sistemas abiertos. TOP END ofrece servicios y capacidades que están asociados con ambientes de mainframe propietarios. TOP END es abierto en diferentes sentidos. La arquitectura de TOP END adhiere estándares a sistemas abiertos, por ejmeplo la adminstración transacciones distribuidas está conformado por el modelo DTP que es especificado por X/Open. Soporta una gran varidad de sistemas de B.D., es portable a otros productos. En la actualidad TOP END está actualmente extendiendo sus capacidades para más mainframes propietarios orientados a OLTP. Por ejemplo el diseño de TOP END permite a un amplio rango de opciones de conectividad para ser usado (soporte a protocolos TCP/IP y X.25). TOP END soporta usuarios workstation que generan transacciones distribuidas accesando datos en una manera segura y controlada. TOP END ha adoptado grupos de la industria de la estandarización como ISO, ANSI, X/Open y SQL. En la figura V.11 se muestra el modelo X/Open DTP de TOP END con una extensión aumentada del modelo original.

TOP END está escrito en el lenguaje de programación ANSI C y está diseñado para ser portable a través de una gran variedad de plataformas. Algunas de las características que ofrece el producto TOP END son las siguientes:

ADMINISTRACION DE TRANSACCIONES DISTRIBUIDAS.

Soporta transacciones distribuidas basadas en el modelo X/Open DTP; es decir, el trabajo es ejecutado en múltiples nodos distribuidos geográficamente, accesando múltiples manejadores de recursos heterogéneos. Esto lo hace por medio del protocolo 2PC.

SOPORTA INTERACCION PARA CLIENTE/SERVIDOR.

TOP END ofrece servicios para procesamiento cooperativo permitiendo a clientes y servidores a participar en una transacción distribuida. Cada cliente o servidor pueden comenzar una transacción. Los clientes y servidores pueden estar en nodos diferentes o en el mismo; la localización del servidor es transparente a los clientes. TOP END automáticamente genera y maneja copias paralelas (réplicas) de aplicaciones y ejecuta todo el trabajo necesario para balanceo de cargas a través de las copias para asegurar que todos ellos son utilizados.

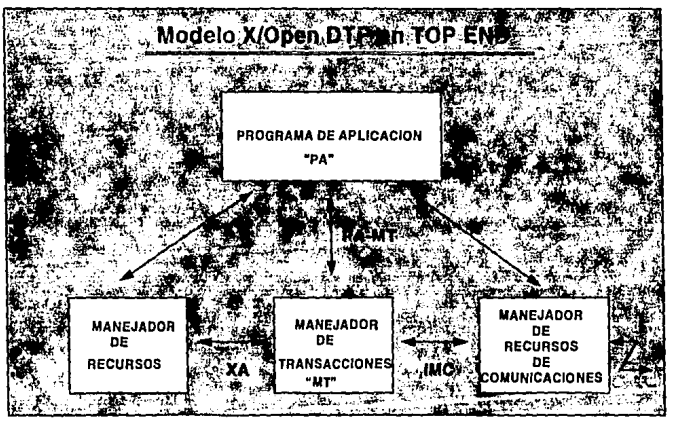

Figura V.11.- MODELO X/OPEN DTP EN TOP END. TOP END se basó del modelo X/Open y lo extendió dando soporte al manejador de recursos de comunicaciones. Esta extensión permite manejadores de comunicaciones, semejantes a un RPC. La interface IMC comunica al MT con el manejador de recursos de comunicaciones "MRC".

SOPORTE PARA SISTEMAS DE ADMINISTRACION DE BASE DE DATOS

TOP END ofrece soporte par sistemas de administración de B.D. (DBMSs) UNIX. Estas permiten aplicaciones para usar DBMS. TOP END trabaja con las B.D. para asegurar el trabajo de las transacciones ditribuidas que se están procesando efectivamente.

SOPORTE A WORKSTATION

TOP END ofrece soporte para ambos, UNIX y DOS basada en workstation. TOP END proporciona librerías de aplicación, este puede ser usado para manejo de transacciones distribuidas.

SERVICIOS DE APLICACION

TOP END proporciona muchos servicios para la aplicación. Por ejemplo, soporte para aplicaciones escritas en lenguajes de programación C y COBOL.

ADMINISTRACION DEL SISTEMA

TOP END proporciona usuarios amigables para interactuar con herramientas para sistemas de administración. Estas herramientas emplean interfaces gráficas para el usuario.

SOPORTE A UN GRAN NUMERO DE USUARIOS

TOP END está diseñado para soportar un ambiente con número grande de usuarios para accesar aplicaciones OLTP.

ALTA DISPONIBILIDAD

TOP END proporciona muchas capacidades que mejoran la disponibilidad en aplicaciones complejas. Por ejmplo, las aplicaciones pueden ser automáticamente "replicadas" para asegurar la continua disponibilidad en un evento cuando una aplicación réplica falla.

INTERNACIONALIZACION

La internacionalización es una parte importante de TOP END. La Internacionalización cubre diferentes áreas. Por ejemplo, si la entrada de los datos en las terminales son usadas para inicializar las transacciones, TOP END puede asegurar el desplegado de la pantalla con un lenguaje propio, por ejemplo, Francés, Español o Inglés. También toda la información y mensajes de error son de bitácora y pueden facilmente ser vendidos al cliente de Inglés a otros lenguajes.

SEGURIDAD

TOP END proporciona ventajas en la capacidad de asegurar áreas como la autentificación y autorización. TOP END usa el programa Kerberos que es usado como base de los servicios de autentificación, servidores de autorización, seguridad de niveles en sistemas UNIX.

Kerberos es desarrollado por el Instituto de Tecnología de Massachusetts. Kerberos está adoptado como parte de la computación distribuida ofrecida por la Fundación de Sistemas Abiertos (Open System Foundation (OSF). El servicio de ***autentificación*** valida la identidad de un componente o usuario.

El servicio de ***autorización*** valida a un cliente autentificado que es permitido para hacer accesos a servicios específicos.

SISTEMAS DE NIVELES DE SEGURIDAD

TOP END se auxilia en la seguridad de los niveles del sistema UNIX para proteger módulos de carga, archivos y colas.

SEGURIDAD AUDITADA

TOP END proporciona distintas capacidades de seguridad auditadas. Todas las peticiones con autentificación todas las peticiones con autorización y todas las transacciones desde un recuros o usuario pueden ser auditadas. Todas las bitácoras auditadas son archivos de nodo local que pueden cada uno ser combinado con bitácoras desde otros nodos antes de ser reportados individualmente o en grupo. Los accesos a las bitácoras de archivos son controlados a través de la seguridad de UNIX. El producto proporciona más sistemas de seguridad. Para el estudio de ésta tesis se tomaron algunas en cuenta, en los manuales viene más extensa la información acerca de estos sistemas.

RECUPERACION

TOP END usa un componente de nodo-local llamado el sistema monitor, Sysmon, para los procesos de UNIX. Con componentes incializados, ellos mandan la señal-prendida de Sysmon. Entonces, si los procesos arrancados fallan, Sysmon será activado e inicializado para el procesamiento de recuperación. Existen otras funciones de recuperación como: Fallas en aplicaciones servidores, fallas en manejadores de nodos, fallas a aplicaciones cliente y fallas a comunicaciones que el producto TOP END tiene funciones propias para salvar las aplicaciones en caso de que ocurra alguna.

SOPORTE DISPOSITIVOS

TOP END ofrece soporte directo a PC´s y a terminales tipo TTY.

SOPORTE A REDES DE COMUNICACIONES

TOP END permite una amplia variedad de protocolos de comunicación e interfaces para ser usados con el sistema.

V.3.5.- TUXEDO

El primer desarrollo fue alrededor de 1984, el Sistema Tuxedo tuvo una evolución sobre 9 años. A lo largo de los 1980's, las principales instalaciones del sistema Tuxedo fueron abarcados por la Compañía de Operación Regional Bell y AT&T. El sistema Tuxedo empezó a estar comercialmente disponible por AT&T en 1989. Desde entonces se han desarrollado varias aplicaciones Tuxedo en giros bancarias, manufactureras, aéreas y otras. En los principios de 1990, la versión 4.0 fue liberada. Esta versión tiene características para el procesamiento de transacciones distribuidas para los sistemas. En 1991 aparece la versión 4.2 de Tuxedo mejorando los sistemas cliente/servidor, la funciónalidad e interoperabilidad con los host, soporta Cobol, interopera con el protocolo LU6.2 para host de IBM y cuenta con un sistema de encolado confiable.

Históricamente ***"USL"*** (UNIX System Laboratories) fue un código fuente de sistema para procesamiento de transacciones proporcionado a vendedores de sistemas, sistemas integrados y Vendedores de Software Independientes ***"VSI"***. Originalmente desarrollado por AT&T/USL, par switcheo de redes en UNIX, el sistema de redes en UNIX, el sistema Tuxedo está ahora adoptando sistemas abiertos de procesamiento de transacciones con 30 licencias. El sistema Tuxedo soporta un modelo de procesamiento de transacciones distribuido implementando estándares de trabajo por DTP X/Open y otros. Novell fue adqurido por la USL, por lo tanto también fue adquirido por Tuxedo. Tuxedo reporta ahora grupos para la Appware Division de Novell. Sobre Novell, Tuxedo se ha incrementado dando soporte a la tecnología Netware incluyendo Netware Loadable Module ***"NLM"*** en las implementaciones principales.

El sistema Tuxedo es una colección de productos diseñados para soportar aplicaciones en el procesamiento de transacciones distribuidas. Actualmente es conocido como Tuxedo Enterprise Transaction Processing ***"ETP"***. Tuxedo soporta un amplio rango de plataformas desde workstation Sun hasta mainframes Amdahl. Está implementado para las interfaces XA, MT, ATMI, etc. Sus componentes principales son los siguientes:

System/T.- El System/T es el componente central del sistema Tuxedo y proporciona los servicios de Monitor de PT: cliente/servidor, balanceo de cargas, transacciones logging, recuperación, programas de aplicación e interacción con manejadores de recursos API's.

Cada Sistema/T ejecutado por el procesador lógico tiene unos procesos *puente*, estos reciben todas las peticiones y peticiones de salidas adelantadas para otros sistemas Tuxedo. El puente recibe los mensajes de entrada y envía mensajes de salida vía el UNIX (System V), facilitando los mensajes encolados.

Los mensajes son mapeados para las colas basadas en información sobre una memoria-compartida *Bulletin Board* manejado por el sistema Tuxedo. La información del bulletin board, los mensajes fuente y el contenido, determina el destino del mensaje. El dato en bulletin board registra los servidores local y remoto disponibles para ejecutar el servicio, la cola del servidor, la utilización para estos servidores y otra información estadística. Todas las comunicaciones entre cliente-servidor, servidor-servidor, y manejadores de recurso son ruteadas por el bulletin board. La unificación del System/T de estructura de datos es la memoria compartida del nombre del servidor Bulletin Board (BB). Con todos los nombres de los servidores, los servidores de aplicaciones registran servicios nombres y direcciones. Las aplicaciones de una transacción cliente envía al servicio de petición de mensajes los cuales son ruteados de acuerdo a la información mapeada por el BB.

El System/T tiene la habilidad de integrar varias computadoras de varias plataformas con la estructura de una sola aplicación. Los programadores pueden usar esta Interface de Administración de Transacciones de Aplicaciones (Application Transaction Management Interface "ATMI") para hacer aplicaciones disponibles de una variedad de plataformas y ambientes software. El System/T ofrece computación distribuida para garantizar la integridad de los datos accesados a través de distintos sites o manejados por los sistemas de administración de B.D. La pista de esta transacción participa vía la noción de una transacción global. También el system/T hace la integración para el alto-performance, sistemas de administración de B.D. relacional semejantes a Oracle, Ingres, Sybase y System/D de Tuxedo.

System/Q.- Este sistema facilita las colas de mensaje implementadas en disco, almacenamiento recuperable para mensajes de aplicación-a-aplicación. Novell tiene extendido el sistema Tuxedo ATMI API para incluir verbos asociados "tpenqueue", como "tpenqueue" para el procesamiento de mensajes confiables.

System/D.- Es un modelo de manejador de B.D. (Database) implementado por accesos relacionales ANSI SQL y accesos orientados a registros ISAM. El paquete /D incluye utilidades de DBMS semejantes a un reporte de escritura, herramientas de administración y funciones loggin y backups. El /D está específicamente diseñado para soportar aplicaciones OLTP corriendo sobre UNIX V. Este incluye un reporte de escritura, una interface SQL, una función interface la cual proporciona soporte a red y modelos de datos jerárquicos

System/DES.- El sistema Tuxedo para Sistema de entrada de Datos (Data Entry System) es una aplicación cliente front-end. Este incluye software sobre ambas formas de creación y administración.

System/Host.- El /Host es el gateway de comunicación del Sistema Tuxedo para comunicar con IBM MVS/CICS usando protocolos LU6.2. El gateway punto-a-punto incluye código para ambos nodos System/T UNIX y el Host IBM ligados por la comunicación punto-a-punto.

Algunos procesos de aplicación del sistema Tuxedo puede usar rutinas de servicio de la interface ATMI para hacer transparentes los accesos para los servicios CICS usando este gateway.

System/WS.- El gateway Workstation permite multiples workstations para comunicar con los servicios del BB. El /WS son funciones de nodo cliente soporta el control y la demarcación de la transacción (begin, end, y transacciones abort) y mensajes empaquetados, permite la presentación de funciones de procesamiento de formas para ser ligadas entre el espacio de la aplicación cliente. El /WS host su código principal es "multi-estado", permitiendo muchos nodos cliente de /WS para comunicar con el nodo System/T sin que los procesos típicamente asociados con algún dispositivo uno-a-uno estén conectados sobre una red.

V.3.6.- TABLA COMPARATIVA DE ALGUNOS SPT EN EL MERCADO

En la siguiente tabla se contemplan las características principales que conforman un OLTP y que se pueden encontrar actualmente en el mercado comercial. Cabe mencionar que no son la totalidad de características, sino las más relevantes, ya que dependiendo del fabricante varían en número y en el tipo de función que realizan.

FUNCION	IMS	CICS	ENCINA	TOP END	TUXEDO
Modelo de transacción	PLAN A	PLAN A	PLANA ANIDADA	PLANA	PLANA ANIDADA
Atomicidad/Durabilidad					
Multiples Manejadores de Recursos B.D.	SI	NO	SI	SI	SI
Integridad de Mensajes.	SI	ALGUNO	SI	SI	SI
Clientes					
Soporta Terminales	SI	SI	SI	SI	SI
Clientes Inteligentes	SI	SI	SI	SI	SI
Modelos de Procesamiento					
Directo	SI	SI	NO disponible	NO disponible	NO disponible
Encolado	SI	SI	NO disponible	NO disponible	NO disponible
Conversacional	NO	SI	NO disponible	SI	SI
Ruteo	SI	SI	NO disponible	NO disponible	NO disponible
Ejecución de Aplicación					
Protección desde fallas de otras Aplicaciones	SI	NO	SI	SI	SI
Usos de Multiprocesos	SI	NO	SI	SI	SI
Utilidad/Manejabilidad					
Integridad con Repositorio	ALGUNO	ALGUNO	NO	NO	SI
Integridad con Herramientas CASE	NO	SI	SI	SI	SI
Generadores de Aplicaciones	ALGUNO	MUCHOS	SI	SI	SI
Integridad con Herramientas Workstations	ALGUNO	ALGUNO	SI	SI	SI
Open					
Interface con Manejadores de Recursos	SI	SI	SI	SI	SI
Interface con Manejadores de Comunicación	NO	SI	SI	SI	SI
Distribuido					
Ruteo de Transacción	SI	SI	SI	NO disponible	SI
Función de Envío	NO	SI	SI	NO disponible	SI
TRPC	NO	SI	SI	NO disponible	SI
Contexto sensitivo TRPC	NO	SI	SI	NO disponible	SI
Comunicación Peer-to-peer	NO	SI	SI		SI
Commit					
Two-phase presumido aborto	SI	SI	SI	SI	SI
Optimizaciones	ANIDADAS	ANIDADAS	ANIDADAS	NO	NO
Disponibilidad					
Scalas de aplicación en un cluster	NO	NO	NO disponible	NO disponible	NO disponible
Hot standby	XRF	XRF	NO disponible	NO disponible	NO disponible

FUNCION	IMS	CICS	ENCINA	TOP END	TUXEDO
Aplicaciones Dev. Env.					
Soporta 3GL	SI	SI		SI	SI
Soporta 4GI	SI	SI	SI	SI	SI
Formato Builder	NO	NO	SI	SI	SI
Soporta Standares					
X/Open	SI	NO	SI	SI	SI
X/Open DTP	SI	NO	SI	SI	SI
ISO OSI	NO	SI	SI	SI	SI
OSF-DCE	NO	NO	SI	NO	SI
Soporta Bases de Datos					
Propietarios	SI	SI	NO	NO	SYSTEM/D
Oracle	NO	SI	SI	SI	SI
Sybase	NO	SI	NO	SI	SI
Ingres	NO	NO	SI	SI	SI
Informix	NO	NO	SI	SI	SI
Soporta Lenguajes					
COBOL	SI	SI	SI	SI	SI
FORTRAN	SO	SI	NO	NO	SI
PL/1	SI	SI	NO	NO	SI
C		SI	SI	SI	SI
C++	NO	SI	SI	SI	SI
Pascal	NO	NO	NO	NO	NO

VI.- CONCLUSION

La tabla presentada en el capítulo anterior es la parte importante de estas conclusiones, ya que ahí se muestra el resúmen y las partes tanto prácticas como comerciales, que se encuentran en todos los OLTP actualmente disponibles en el mercado.

Los diferentes productos evaluados en este estudio no son todos los que comercialmente se encuentran disponibles, pero si son los más populares internacionalmente y los que más apoyo tienen de la Orgnaización Internacional de Estándares, por ejemplo "ISO" (International Standards Organization).

Dependiendo de la Empresa de que se trate y de acuerdo a su infraestructura informática el OLTP que mejor se adapte a sus necesidades será aquel que de acuerdo a la tabla presentada anteriormente le ofrezca el mayor número de funciones para su mejor aprovechamiento.

No todas las Empresas son factibles de incorporar este tipo de herramienta ya que no está desarrollada para todos los diferentes giros de las Empresas ya que son de un uso específico aún, debido a que los OLTP básicamente manejan miles y millones de transacciones por segundo "tps" y no todas las Empresas cumplen con este requisito. Para el mejor aprovechamiento de esta tecnología es necesario que las Empresas realicen un estudio de viabilidad ya que puede ser injustificable e incosteable. Por lo tanto estos productos están enfocados a satisfacer el mercado de aquellas Empresas que manejan grandes volumenes de transacciones, como por ejemplo Bancos; Casas de Bolsa, Aerolíneas, Finanzas, etc.

Una Empresa factible de incorporar un OLTP tendrá los beneficios que se desprende del uso de esta herramienta dentro de los cuales los más importantes son:

- **Estandares basados en OLTP**
  - El modelo X/Open DTP.
  - DBMSs condescendientes con XA.
  - Protocolos de redes abiertos OSI y TCP/IP.
  - POSIX y XPG.
  - Sistema X/Windows y Interfaces de usuario graficas como Motif.

- **Alto performance.**
  - Alto rendimiento.
  - Integridad en la Información.
  - Velocidad en tiempo de respuesta.
  - Altos volumenes de transacciones.
  - Procesamiento ininterrumpido durante fallas.
  - Bajo costo de Transacciones por segundo ***"TPS"***.

- **Accesos continuos.**
  Rápida recuperación.

- **Procesamiento Cooperativo con Cliente/Servidor.**
  Soporte al protocolo 2PC.
  Clientes y Servidores Distribuidos.
  Interoperabilidad en ambientes Mainframes.

- **Procesamiento de Transacciones Distribuido.**
  Bases de Datos Heterogeneas.
  Escalabilidad en la red.

En virtud de que el software OLTP está desarrollado básicamente en lenguaje de programación "C", lo hace compatible con otros desarrollos de aplicaciones bajo el mismo código y que sean afines a dar solución a problemas transaccionales.

Un OLTP puede funcionar adecuadamente en la mayoría de los equipos de cómputo conocidos comercialmente así como en diversas plataformas de hardware ("maxis, minis y micros"), lo que lo hace portable, ya que todos los OLTP cuentan con código fuente para la mayoría de los fabricantes que lo deseen incorporar a sus desarrollos o productos propios.

Como la base de desarrollo de un OLTP es el uso del protocolo 2PC, le permite interacturar más eficientemente en ambientes de cómputo distribuido a nivel nodo, B.D. y redes de comunicación de datos; porque la base del ambiente distribuido es el compartir los datos adecuadamente y con mayor "performance".

Actualmente los Sistemas de Información requieren de un mejor manejo de los datos por lo que el uso de los OLTP brindan grandes beneficios en este rubro porque el manejo de la información es altamente confiable, tanto por la seguridad, como la eficiencia y la recuperación en casos de falla..

La tecnología de los monitores de procesamiento de transacciones en sistemas abiertos tendrán la capacidad de adoptar avances en el futuro de otras tecnologías, tales como la tecnología Orientada a Objetos; que es cada vez más aceptada y difundida, donde los usuarios deberán planear la coexistencia de objetos en los sistemas implantados en las Organizaciones. Los SPTOO (Sistemas de Procesamiento de Transaccciones Orientados a Objetos) benefician la integración con otros monitores de procesamiento de transacciones. En la actualidad varios vendedores de software están desarrollando interfaces Orientadas a Objetos para los monitores de PT con el fín de obtener beneficios de ambas tecnologías.

Otro desarrollo de la tecnología OLTP, son los nuevos modelos de transacciones como: las transacciones anidadas, en cadena, distribuidas, multi-nivel y exóticas que serán soportadas e incorporadas por el manejador de transacciones y definidas en un S.P.T.

La finalidad de esta tesis no tiene la intención de publicar algún producto, hacer un análisis competitivo o una guía de estudio de mercado de los OLTP. Se tomaron en cuenta las características más comunes de cada uno de los OLTP, por lo tanto no se profundizó en otras características técnicas propias de cada fabricante, ya que estas son muy extensas y considero que no son motivo de este estudio.

VII.- OBRAS CITADAS

🕮.- M. Ansari, L. Ness, M. Rusinkiewcz, and A. Sheth.
Using flexible transaction to support multi-system telecommunication applications.

🕮.- U. Dayal, M. Hsu, and R. Ladin. Organizing Longrunning activities with triggers and transactions.
In Proc. of ACM SIGMOD Int. Conf. on Management of Data, 1990.

🕮.- U. Dayal, M. Hsu, and R. Ladin. A transaction model for long-running activities.
In Proc. of the *17th* Int. Conf. on VLDB, 1991.

🕮.- UNIX SYSTEM LABORATORIES.
TUXEDO Enterprise Transaction Processing System Release 4.2.

🕮.- UNIX SYSTEM LABORATORIES.
The Standish Group Presents
Open System OLTP Monitors Bow Showing Novell's TUXEDO System 1993.

🕮.- TOP END Distributed Transaction Processing for Open System.
Parallel Distributed Information Systems (PDIS) Conference January 22,1993.
Randy Smerik.
NCR Corp., San Diego.

🕮.- DATAPRO Reports on UNIX System & Software.
Computing in the 1990s: UNIX, Downsizing, and Client/Server Systems.

🕮.- DATAPRO Reports on UNIX System & Software.
Transarc Corp. Encina Transaction Processing Software.

🕮.- DATAPRO Reports on UNIX System & Software.
IBM CICS Transaction Monitors.

🕮.- DATAPRO Reports on UNIX System & Software.
The Transaction Processing Performance Council.

🕮.- DATAPRO Reports on UNIX System & Software.
Transaction Monitors in Distributed Processing.

🕮.- DATAPRO Reports on UNIX System & Software.
TOP END Distributed Transaction Managemente System

🕮.- OLTP Databases and UNIX.
ORACLE Versión 6 supports new UNIX technologies to create a poweful on-line transaction processing environment.

🕮.- ORACLE MAGAZINE.
OPERATING ENVIRONMENT
MINICOMPUTERS. Implementign Fault-Tolerant OLTP.
Summer 1990. Pag.80

🕮.- IBM Open Distributed Transaction Processing for AIX/600
Presentation Guide.
G326-0275. September 1992.

🕮.- UNIX IN THE OFFICE
Unix OLTP.
Getting Ready for Commercial Prime Time.
John R.Rymer.
VOL 5. NO.8., ISSN:0887-3054.
AUGUST 1990.

🕮.- TRADE PUBLICATIONS ABOUT TOP END.
Computes Systems News.
January 14,1991.

🕮.- TOP END Product Overview.
Distributed Transaction Processing for Open Systems.
NCR March 1991.

🕮.- UNIX Open/OLTP
A technological Overview of OLTP in the 1990s.

🕮.- DATAMATION.
MIDRANGE/MAINFRAME OLTP.
Monitors Form Foundation For Open OLTP.
Pag. 109. March 1, 1993.

🕮.- UNIX International.
Distributed Transaction Processing Environments: A Competitive Analysis of UNXI System Laboratories TUXEDo System and Transarc Encina.
April 1992.

🕮.- Proceedings.
Second International Conference on Parallel and Distributed Information Systems.