Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

“Plataforma HP NonStop Tandem”

Monografía
Para obtener el grado de
Lic. en Sistemas Computacionales

Presenta:

Martha Angélica Hernández Rodríguez

Asesor:

M. C. C. Luis Islas Hernández

Pachuca, Hgo., a 28 de Noviembre de 2007

AGRADECIMIENTOS

Agradezco antes que a nadie, a mis padres, de quienes siempre recibí todo el apoyo y dedicación que podría haber esperado. A mi padre cuyo esfuerzo hizo posible mi formación profesional y quien desde donde se encuentre sigue cuidando de nosotros, a mi madre cuyo ejemplo en la vida ha servido de guía en mi vida, dedico el presente trabajo.

A mis hermanos, tíos y primos cercanos, quienes directa o indirectamente me ayudaron a concluir mi carrera y continúan apoyándome en mi vida profesional, agradezco su cariño y apoyo en los momentos mas difíciles y que siempre tendré incondicionalmente en cualquier momento de mi vida.

A mi asesor, por su tiempo, esfuerzo y dedicación, para poder concluir y perfeccionar el trabajo que estoy presentando y sin cuya guía hubiera sido difícil alcanzar el objetivo deseado.

A mi jurado, por darse tiempo en leer el presente trabajo, el cual estoy conciente es un poco amplio y a pesar de ello, dedicaron tiempo y esfuerzo para poder hacerme ver mis errores.

OBJETIVO

Dar a conocer el funcionamiento, desempeño y constitución de la plataforma HP NonStop Tandem. Proveer al lector de información, que hasta el momento solo se tiene acceso a través de las grandes empresas que emplean dicha plataforma para el manejo y explotación de los grandes volúmenes de información que en ellas se manejan.

Poco se conoce acerca de esta plataforma, no es un tema común y de fácil acceso por esto, a fin de poder aprender su manejo y explotación, se presenta el siguiente trabajo a cualquier persona o estudiante que puedan tener acceso a él y poder emplear la información que en él se presente de la mejor manera en que le convenga para su vida profesional.

JUSTIFICACIÓN

Ante la diversidad de lenguajes de computación y nuevas plataformas que en día van apareciendo, es necesario tener conocimiento de ellas y poder ser apto en el ambiente laboral, cada vez más competitivo. Para ello, se presenta el siguiente documento, a fin de ayudar al lector con este problema actual, dando a conocer la plataforma HP NonStop Tandem para el manejo de información.

Hasta el momento la mayoría de la información acerca de esta plataforma, no es de fácil acceso, o se encuentra en idioma ingles. Con la información obtenida a través de la experiencia laboral en una de las grandes organizaciones que emplean dicha plataforma se redacta el presente trabajo y se comparte dicha información con cualquier persona interesada en aprender un nuevo conocimiento y poder tener mas armas dentro del mundo competitivo del área de sistemas.

ÍNDICE

INTRODUCCION

La anatomía de una típica gran falla del sistema, es interesante, asumiendo, como es común, que una operación o falla del sistema, causó una pausa en el sistema. Puede tomar un par de minutos para alguien reparar que hay un problema y que un reinicio del equipo es la única obvia solución. Después, toma al operador cerca de 5 minutos para recuperar el estado del sistema para un análisis posterior, entonces el reinicio puede comenzar. Para un sistema grande, el sistema operativo toma un par de minutos para ser reiniciado. Entonces la base de datos se reinicia por completo, dentro de un par de minutos, pero, puede tomar una hora reiniciar una gran terminal de red. Una vez que la red esta arriba, los usuarios retoman las tareas que estaban desarrollando. Después de reiniciar, mucho trabajo ha sido guardado para el desempeño del sistema [1]

Existen equipos tolerantes a fallos, tales, como el descrito en el párrafo anterior, esto no quiere decir que sean equipos que jamás van a fallar, ya sea una falla en el sistema o una falla de origen humano, el beneficio de dichos equipos, esta en la respuesta o acción de dicho equipo a para la tolerancia a estas fallas, así como aminorar los daños originados por las mismas.

Uno de estos equipos es HP NonStop Tandem, cuya plataforma de tolerancia a fallos puede hacerlo que no necesite ser reiniciado en un largo periodo de tiempo. Así mismo, esta orientado a manejo de grandes volúmenes de información y a cubrir la necesidad de cualquier empresa de aminorar los riesgos de una posible falla del sistema.

El presente trabajo da a conocer la plataforma Tandem, describiendo su funcionalidad, arquitectura, arquitectura del sistema operativo, base de datos, así como, el manejo y explotación de la misma, seguridad del sistema, niveles de seguridad de los usuarios dentro del sistema, entre varios temas que se describen a detalle en los próximos capítulos.

1 ARQUITECTURA PLATAFORMA TANDEM

En este capítulo se darán a conocer la arquitectura de la plataforma Tandem, conceptos básicos, fundamentos de Tandem, procesos en línea, base de datos empleada, interfase SQL, protección a la base de datos, trabajo en red, seguridad, entre otros temas necesarios para la manipulación de datos almacenados en el sistema.

Lo anterior con el fin dar conocer la base de la plataforma HP NonStop Tandem, una de las empleadas en la actualidad las empresas para el almacenamiento y manipulación de información de las empresas.

1.1 Introducción

La plataforma Tandem NonStop fue fundada en 1974 con el objetivo de diseñar soluciones tecnológicas confiables para negocios en cuya operación se requieren continuamente, las 24 horas del día, sistemas de procesamiento en línea. Dichos sistemas están diseñados para proporcionar confiabilidad absoluta y escalabilidad casi ilimitada. Pueden ser utilizados en cualquier compañía donde se requiere procesar una gran cantidad de información por segundo y almacenarla en el mismo instante.

Los sistemas Tandem se han diseñados específicamente para transacciones en línea, es decir, las transacciones se procesan como van sucediendo. Con esto, las bases de datos están siempre actualizadas. En los procesos “batch” la integridad de las bases de datos no se ve afectada seriamente pues los trabajos se ejecutan de manera programada.

Los requerimientos de un Sistema de Procesamiento en Línea (OLTP)[1] son:

- Disponibilidad.- Base de datos accesible en el momento
- Integridad.- Resultados exactos
- Desempeño.- Respuesta rápida sin importar el momento

Debe ser tolerante a fallas, lo que significa proveer acceso normal a la base de datos y a las aplicaciones, incluso si existe un fallo en el hardware. Esto se logra con la replicación de procesos, procesadores, rutas y componentes en general.

1.2 Fundamentos de TANDEM para el ambiente OLTP

1.2.1 Ambiente de procesos de transacciones en línea

Los procesos de transacciones en línea (OLTP) son operaciones realizadas por computadoras procesando datos acerca de transacciones de negocios a medida que éstos ocurren. Las personas que hacen uso de los OLTP, utilizan críticamente las transacciones en línea para obtener información oportuna que refleja el estado real de los negocios.

Los sistemas OLTP son utilizados donde es necesario acelerar el flujo de recursos, capturar las transacciones en el punto de venta y ayudar en la fabricación automática en el almacén, entre

[1] Online Transaction Processsing

otros usos. Se diseñan sistemas específicamente para OLTP, que se expanden al mismo ritmo que las empresas.

Beneficios

Los datos son actuales y se puede acceder a ellos de rápidamente. El ambiente OLTP puede permanecer disponible todo el tiempo, a diferencia de la cantidad de ambientes tradicionales que deben ser detenidos periódicamente para permitir la actualización de las bases de datos.

1.2.2 Requerimientos para el ambiente OLTP

Los sistemas Tandem NonStop están diseñados para dirigir las necesidades del ambiente OLTP proporcionando:

- **Continua disponibilidad,** para confiabilidad y operaciones continuas.
- **Alto rendimiento,** para soportar una elevada cantidad de datos y rápidas respuestas.
- **Posibilidad de expandir**, para expandir sistemas modulares con líneas incrementales en rendimiento.
- **Integridad de datos,** para asegurar que la base de datos es correcta.
- **Datos distribuidos,** con la facilidad de colocar los datos cerca de los usuarios más frecuentes, mientras permite a otros usuarios acceder a ellos desde cualquier lugar de la red.
- **Sistema de red,** para soportar las ligas a los sistemas en cualquier ubicación geográfica
- **Seguridad,** para controlar el acceso a los datos, programas y otros recursos del sistema**.**

1.2.3 Continua disponibilidad

Si el sistema no puede ser tolerante a las fallas no podrá ser continuamente disponible para el negocio que requiere de éste. El fundamento para lograr el éxito de Tandem en OLTP, es que es una maquina NonStop, el primer sistema comercialmente disponible tolerante a fallas.

La arquitectura del sistema Tandem conoce las necesidades evitando el gasto innecesario de redundancia o pérdida de información. Los productos Tandem son diseñados a partir del principio de "tolerancia a fallos"[2]. La tolerancia a fallos provee continua disponibilidad (24 horas al día, 7 días a la semana) a usuarios y clientes.

Una variedad de factores de diseño contribuyen a la tolerancia a fallos de la plataforma Tandem entre ellos:

- Los componentes y las unidades centrales de procesamiento están siendo ejecutados en paralelo, compartiendo la carga de trabajo.

[2] El sistema continúa operando a pesar del fallo de cualquier componente hardware o software

- Si uno o más de éstos llega a fallar, la operatividad del sistema automáticamente reasigna el trabajo a otras partes del sistema, así todas las transacciones pueden ser completadas sin ninguna pérdida de información.

Alto desempeño

El tiempo de respuesta es la medida del desempeño de un sistema OLTP. Las aplicaciones OLTP pueden ser dirigidas por un gran número de usuarios sin ser una carga excesiva para el sistema OLTP.

Si un proceso comienza a limitar al sistema, inmediatamente un desempeño de mejoramiento, puede ser ganado corriendo una copia de este proceso en un procesador adicional, dividiendo de esta manera la carga de trabajo entre los procesadores. (Ver *figura 1.2.3.1)*

Así, como procesadores o sistemas son agregados, se obtiene un incremento directamente proporcional en el desempeño.

Los procesadores Tandem dirigen diferentes tareas al mismo tiempo debido al alto desempeño del sistema, mientras permanecen en contacto constante a través de una alta velocidad de comunicación con el pathway.[3]

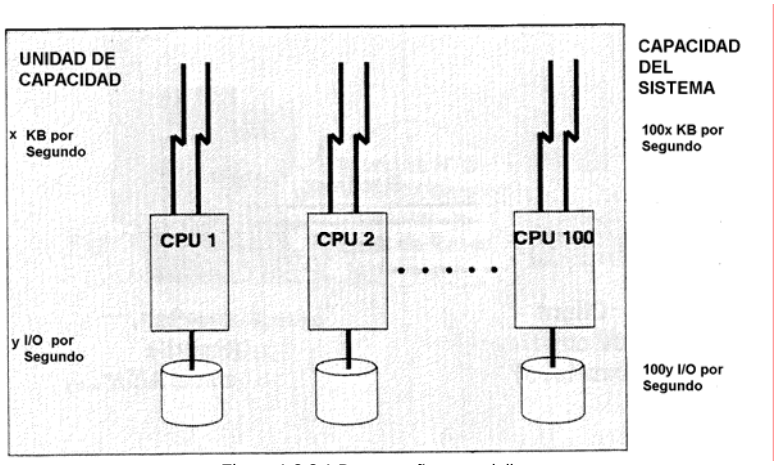


Figura 1.2.3.1 Desempeño y paralelismo

1.2.4 Expansibilidad

El sistema OLTP y sus aplicaciones pueden ser expansibles así, la arquitectura Tandem provee una respuesta incremental a través de una expansión modular, de esta forma el negocio puede construir sobre estas inversiones existentes en software y hardware.

Desde que los sistemas Non Stop están basados en un diseño modular, pueden ser expandidos agregando componentes en lugar de reemplazar sistemas más grandes.

- Los componentes que pueden ser agregados a un sistema Tandem incluyen procesadores, memoria, controladores, discos duros, terminales y otros dispositivos periféricos.

- Los componentes pueden ser agregados mientras se encuentra corriendo un sistema cuando la expansión es requerida, sin bloquear las aplicaciones críticas.

[3]Software que abastece programas y ambientes operativos para el desarrollo y ejecución de aplicaciones

La expansión es transparente a las aplicaciones. La operación del sistema Tandem está diseñada para aplicaciones que corren en dos procesadores del sistema, así también, correrán en 16 procesadores del sistema sin necesitar alterar la línea de aplicaciones. Con cada nuevo procesador que se instale, se agrega un incremento de desempeño para el sistema.

1.2.5 Integridad de datos

Los datos deben ser actualizados y precisos para reflejar la confiabilidad del estado del negocio. La integridad de datos significa que, ni el fallo de una transacción en particular, ni tampoco el fallo de componentes de hardware, como un CPU o un disco de almacenamiento, puede corromper los datos en la base de datos. Los sistemas Tandem son diseñados para mantener la integridad de una transacción y la consistencia de la base de datos.

La protección de los datos es construida en el software del sistema, dando como resultado un alto desempeño y facilidad en el desarrollo de aplicaciones. Los discos duros empleados para el almacenamiento de datos pueden ser "espejeados"[4], de tal forma que un respaldo exacto de la base de datos estará disponible en caso de un fallo del disco de almacenamiento. (Ver *figura 1.2.5.1)*

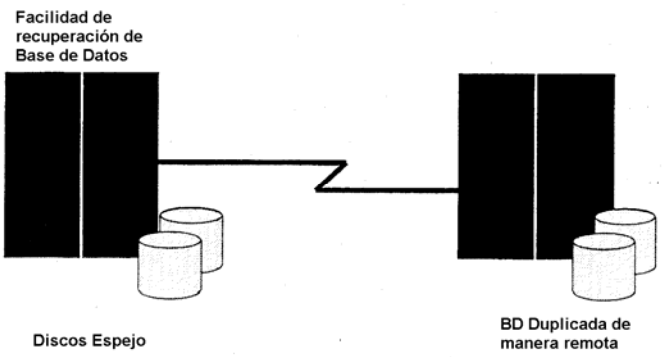


Figura 1.2.5.1 Integridad de los datos

1.2.6 Datos distribuidos

Una base de datos distribuida es uno de los objetos en los cuales residen más de un sistema en una red y a los cuales pueden ser accedidos desde cualquier sistema dentro de la red. Depositando información acerca de las personas que la usan con mayor frecuencia, reduciendo los costos de comunicación y mejorando los tiempos de respuesta del sistema. (Ver *figura 1.2.6.1)* Mientras los datos pueden ser localizados en múltiples sitios, las aplicaciones y los usuarios pueden acceder a ellos como si fuera una sencilla base de datos.

El sistema de software Tandem maneja todas las complejidades de una base de datos distribuida. Ni las aplicaciones de software ni los programadores necesitan preocuparse dónde han sido almacenados los datos específicos. Este concepto es referenciado como independencia geográfica.
Las transacciones pueden ser generadas y buscadas desde cualquier punto dentro la red a otro punto dentro de la misma, de una manera completamente transparente. El sistema

[4] Respaldados en línea

automáticamente mantiene huellas de todos los datos como registros que han sido agregados, borrados o movidos a una nueva ubicación.

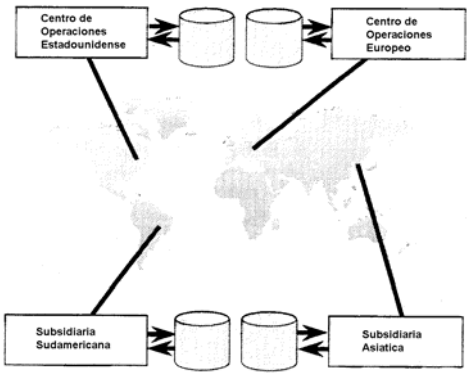


Figura 1.2.6.1 Datos distribuidos

1.2.7 Ambiente de red

Una red puede ser local, regional, nacional o internacional. Los sistemas OLTP deben ser capaces de funcionar efectivamente en redes de sistemas que son elaborados por el mismo proveedor o por distintos proveedores.

Los sistemas Tandem son diseñados para ser una red en un sistema y una red de sistemas. Los usuarios de red autorizados deben tener el acceso fácil a los recursos en cualquier parte de la red. Además, están diseñados para ser una red y ejecutarse independientemente desde cualquier ubicación geográfica. El objetivo final es interconectar múltiples proveedores, protocolos y recursos de manera transparente a usuarios y programadores.

Tandem soporta conectividad de sistemas abiertos e integración de aplicación, permitiendo integración efectiva de servidores, mini computadoras, estaciones de trabajo, computadoras personales y otros recursos de diferentes proveedores (ver *figura 1.2.7.1).* Todas las redes Tandem adecuan los fundamentos de tolerancia a fallos, así como datos e integridad de transacciones.

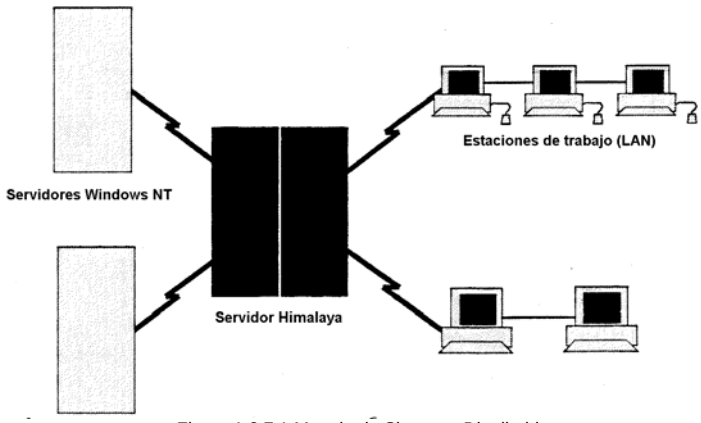


Figura 1.2.7.1 Manejo de Sistemas Distribuidos

1.2.8 Seguridad

Las aplicaciones OLTP hacen que las fuentes de datos estén disponibles a usuarios múltiples. Debido a ello es importante que:

- Tales aplicaciones deben restringir el uso de las fuentes de datos a un pequeño subconjunto del total de la población de usuarios, o el negocio pierde el control de las fuentes de datos.

- Los sistemas OLTP deben restringir a los usuarios autorizados sólo a un subconjunto de los datos.

A medida que la red crece y el número de usuarios incrementa, sucede lo mismo con el riesgo de seguridad y la necesidad para el control de la información. El sistema de seguridad Tandem tiene una gama de privilegios de acceso que le permiten controlar el acceso de los usuarios a los recursos fuentes como un archivo, periféricos y procesadores. Tandem incluye procedimientos para autorización e identificación de los usuarios, de esta forma los rastros de revisiones de cuentas pueden ser establecidos para rastrear los casos de acceso fallido. (Ver *figura 1.2.8.1)*

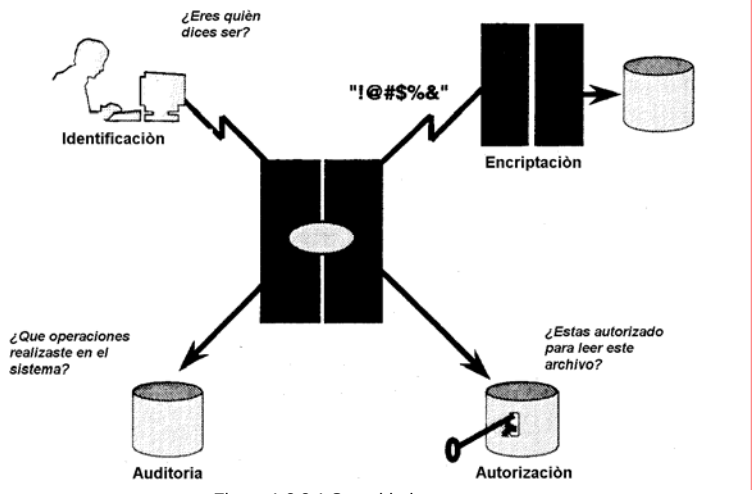


Figura 1.2.8.1 Seguridad

1.3 Arquitectura de sistema NONSTOP

1.3.1 Diseño del hardware de Tandem

Las claves principales del diseño de Tandem son:

- **Modularidad**: El hardware está compuesto por módulos, los cuales son unidades de servicio, diagnóstico, reparación y crecimiento.

- **Fallos cortos**: Cada módulo es de autocomprobación, es decir, cuando un módulo detecta un fallo, éste lo detiene.

- **Componentes de tolerancia a fallos**: Cuando un módulo de hardware o software falla, esta función inmediatamente es tomada por otro módulo.

- **Mantenimiento en línea**: El hardware y las aplicaciones de software pueden ser diagnosticadas y reparadas mientras el resto del sistema continúa proporcionando

servicio. Cuando el hardware, programas u otros datos son reparados, ellos son reintegrados sin que exista interrupción del servicio.

1.3.2 Réplica de componentes

Uno de los criterios más importantes de cualquier sistema en línea es la disponibilidad. En la mayor parte de los sistemas, un fallo de cualquier componente principal del sistema puede paralizar todas las funciones en línea. En el caso de Tandem, todos los componentes de hardware pueden ser replicados a otra parte en el sistema. Cada componente replicado realizará sus propios deberes y obligaciones en este papel de reserva, sólo durante el fallo de un componente. (Ver *figura 1.3.2.1)*

La réplica de cualquier componente debe satisfacer dos importantes objetivos de diseño:

- Debe proporcionar una reserva de tolerancia a fallos.
- Debe resaltar el funcionamiento existente.

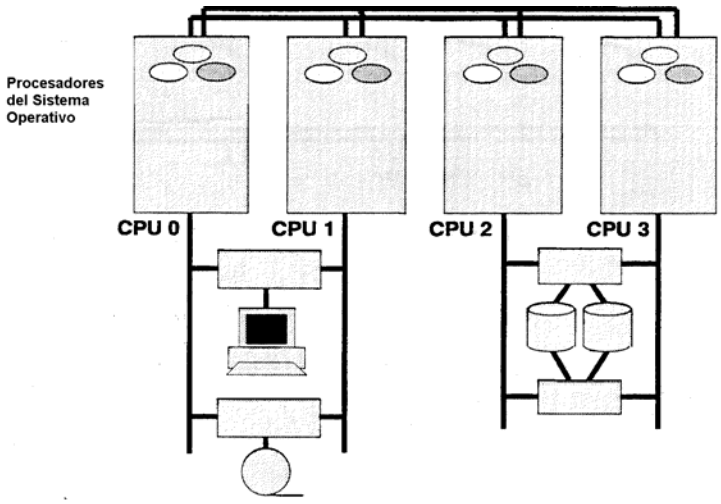


Figura 1.3.2.1 Replica de Procesos del SO

1.3.3 Expansión modular

El sistema crece por una **expansión modular** sin sustitución. Pueden agregarse procesadores uno a uno, sólo hasta un número máximo por sistema. Los dispositivos periféricos (unidades de disco, terminales, impresoras, etc.) también pueden ser agregados en cualquier parte del sistema, dependiendo cómo éste crezca. Los sistemas se pueden expandir como los procesos de transacciones lo requieran. (Ver *figura 1.3.3.1)*

El funcionamiento se amplía en una manera lineal a medida en que los procesadores son agregados. La expansión del sistema permite que la carga de trabajo se disperse equitativamente sobre todos los procesadores.

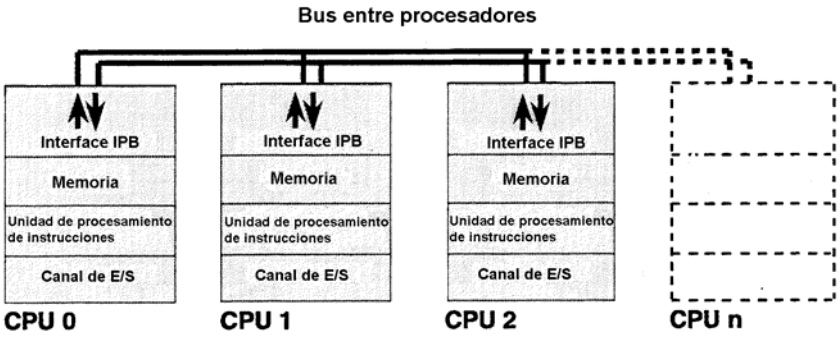


Figura 1.3.3.1 Expansión

1.3.4 Procesadores

Cada procesador es una unidad de procesamiento capaz de soportar todas las actividades de procesamiento de datos y puede ser configurado para soportar múltiples y variados dispositivos periféricos. Múltiples procesadores soportan procesamientos paralelos y proveen disponibilidad a través de la tolerancia a fallas. Cada sistema puede tener entre 2 y 16 módulos de procesadores, los cuales pueden ser agregados como sea necesario. Todos los tipos de procesadores soportan el mismo sistema operativo (GUARDIAN 90) haciendo todo el software completamente compatible a través de cadena de producción.

1.3.5 Unidades de disco espejo

Las unidades de disco espejo son dos unidades de discos físicas que comparten el mismo nombre lógico y contienen datos idénticos (Ver *figura 1.3.5.1).* Los datos son escritos simultáneamente en ambos discos y proveen el más alto nivel de integridad de datos posible, son opcionales y altamente recomendados para datos críticos. [5] [2]

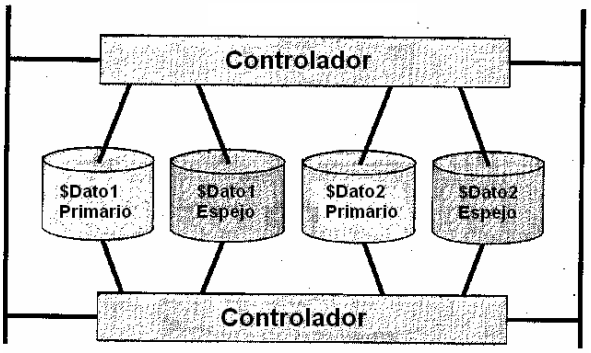


Figura 1.3.5.1 Discos Espejo

[5] El sistema sobrevivirá a fallos o contaminación de uno de los discos al usar el disco restante.

1.4 Arquitectura del sistema operativo

1.4.1 Sistema operativo GUARDIAN 90

GUARDIAN 90 es el sistema operativo Tandem. El sistema Tandem está diseñado como una entidad simple en la cual el hardware y software trabajan juntos para proveer la tolerancia a fallos, expansibilidad modular, sistema de red eficiente y otras funciones OLTP. La funcionalidad básica del sistema operativo GUARDIAN 90 es residir independientemente en cada procesador. Tanto el sistema como los procesos del usuario, se basan en el sistema operativo para proporcionar soporte de ejecución.

El sistema operativo GUARDIAN 90 proporciona soporte para cada una de estas funciones principales en cada procesador:

- **Administración de memoria**: Proporciona soporte de memoria física y virtual, responsable de controlar la asignación y utilización de memoria
- **Administración de procesos**: Comienzo, ejecución y parada de procesos.
- **Comunicación entre procesos**: Encaminamiento de mensajes entre procesos independientes.
- **Servicios de usuarios**: Muchas rutinas estándares son necesitadas, a menudo, por un proceso. Por ejemplo, éste puede necesitar convertir un dato binario a un formato ASCII o necesitar definir la fecha y hora actual.
- **Control de procesos**: La habilidad de un proceso para solicitar el inicio o fin de otros procesos.
- **Apoyo de entrada - salida (Archivos del sistema)**: La interfase[6] al archivo del sistema permite a los procesos abrir, leer, escribir y cerrar archivos lógicos.
- **Operaciones NonStop**: Un proceso de tolerancia a fallos puede ser necesario para asegurar la protección de los mismos.
- **Apoyo de producto especial**: Un proceso puede necesitar ciertos servicios fuera del GUARDIAN 90, como aquellos que son proveídos por el subsistema de integridad de base de datos o el subsistema de medición de funcionamiento del sistema.

1.4.2 Administración de memoria

La memoria en cada procesador es administrada individualmente por un proceso GUARDIAN, *administrador de memoria* localizado dentro del procesador. El ID del proceso administrador de memoria es siempre el PROCESO #1 en cada CPU. El código objeto es almacenado en disco (almacenamiento virtual) y es llamado del procesador según sea necesario.

Al inicio de un proceso, el monitoreo GUARDIAN, solicita al proceso administrador de memoria GUARDIAN asignar espacios de memoria para el código o almacenamiento de datos. Cuando se ejecuta el fin de un espacio de memoria de código, el administrador de memoria solicita más. El proceso continúa ejecutándose como tanto código sea solicitado. Esto genera que todo este

[6] Punto de conexión.

código sea residente y que la memoria de procesador disponible para él sea virtualmente ilimitada.

La memoria física usada es determinada por procesos que demandan su uso. Los espacios de memoria menos empleados por un proceso pueden ser "robados" por otro proceso que sea más activo. Cuando un proceso termina, todos estos espacios de memoria son marcados como libres y disponibles para uso posterior.

1.4.3 Estructura de proceso

Un programa es un código objeto almacenado en disco, un proceso es un programa ejecutándose. Cada proceso tendrá ciertas características:

- **Nombre del proceso:** Identificador único para cada proceso.
- **Identificación del proceso:** Una combinación del número del CPU y el número del proceso dentro de éste.
- **Prioridad del proceso:** Valor numérico designado por usuario, indicando la importancia relativa de ese proceso activo.
- **Cambio de archivo para el proceso:** El nombre del archivo en disco donde el proceso activo pueda ser reubicado.
- **Tamaño del proceso:** Cada proceso tiene un tamaño predeterminado de un máximo 128 KB de la memoria del procesador para datos y 2MB para códigos.
- **Código de usuario y espacio de datos:** Secciones del procesador de memoria física asignados para el proceso de código objeto y datos mientras éste se está ejecutando. El código objeto contiene las instrucciones actuales de la máquina para el proceso que está ejecutándose, ésta es la ejecución lógica. Los datos incluyen registros recuperados de los archivos de datos y valores calculados durante la ejecución del programa.
- **Estados del proceso:** Pueden ser activo, listo, en espera o suspendido.

Cuando un proceso llega a ser el primero en el procesador actual, éste gana acceso a las funciones del sistema operativo. De ésta manera, el sistema operativo puede soportar las actividades de este proceso ejecutándose.

1.4.4 Ejecución de procesos en el ambiente actual

Cada procesador puede soportar un máximo de 256 procesos. Sólo uno de esos procesos del sistema o un proceso de usuario se ejecuta con la interfase de GUARDIAN 90 en un momento dado. Todos los demás procesos, actualmente sin la interfase de GUARDIAN 90, están en un estado de no ejecución.

Todos los procesos que corren en un procesador son numéricamente identificables en una lista conocida como el Bloque de Control de Procesos (PCB)[7]. Este valor numérico para un proceso (0 a 255) es conocido como el Número de Identificación del Proceso (PIN) y combinado con este número de procesador con el formato (# procesador, #proceso), llega a ser conocido como ID del proceso (PID). Los posibles estados de los procesos son:

[7] Process Control Block

- **Activo:** Uno por procesador con la interfase GUARDIAN 90.

- **Listo**: Actualmente es un estado de no ejecución, guardando una posibilidad para ejecutarse.

- **En espera**: Un proceso que no puede continuar la ejecución hasta que algún evento externo (ejemplo: una operación de entrada-salida) sea completado. Cuando el evento se completa, el proceso regresa a un estado de "listo".

- **Suspendido**: Instrucciones no ejecutadas. El proceso no está activo o listo y no está en espera de un evento externo.

1.4.5 Proceso de planificación de procesos "Listos"

El sistema operativo mantiene una lista de todos los procesos, los cuales están listos para ejecutarse. Un proceso se encuentra en un estado "listo" cuando tiene algo que hacer y no está en espera del resultado de un evento anterior.

Debido a que muchos procesos pueden estar listos para realizar sus tareas simultáneamente, el sistema operativo necesita algunos criterios para colocarlos en orden. El sistema asigna una prioridad a cada proceso (entre 1 y 255) y ejecuta el proceso en estado "listo" con la prioridad más alta.

Las prioridades de los procesos de usuarios se extienden de 1 a 199. Por lo tanto, las prioridades de los procesos del sistema operativo que llegan a estar listos, siempre se ejecutarán delante del proceso del usuario. Si varios procesos entran en la lista de procesos listos con la misma prioridad, éstos serán ejecutados en el orden que llegan. Si un nuevo proceso entra al procesador, éste será insertado en la lista de procesos con estado "Listo" en la posición igual a su prioridad. Si su prioridad es más alta que el proceso actualmente ejecutándose, se colocará antes, llegando a ser el proceso en ejecución.

Hay muchas actividades que pueden causar que otro proceso llegue a ser el proceso activo en un procesador, cuando:

- El proceso activo se cicla hasta que su prioridad interna es degradada debajo de los otros procesos listos.

- Un nuevo proceso entra al procesador con una prioridad alta más que el proceso activo.

1.4.6 Estructura del proceso de los códigos objeto

El código objeto producido por todos los compiladores de GUARDIAN 90 no es modificable, por lo tanto. no pueden ser modificados durante su ejecución. Múltiples ejecuciones de un programa, incluso corriendo en diferentes procesadores, pueden usar el mismo código objeto, esto hace más eficiente el uso de memoria del procesador. Por ejemplo: tres usuarios inician el proceso de edición de textos, pero sólo es necesario un archivo de código objeto para soportar las tres distintas sesiones de edición. Esto hace más eficiente el uso de espacio libre en disco.

Cada proceso independientemente de su ubicación conservará su propio espacio de datos. Esta área no es compartida con otros procesos; también mantiene su propio archivo de intercambio (swap file) en disco donde el espacio de datos para este proceso puede ser reubicado si estos son requeridos por el proceso que se ejecuta actualmente.

1.4.7 El enfoque de Tandem al diseño OLTP

La habilidad de soportar la comunicación entre procesos es proporcionada por una combinación de hardware y un sistema de software.

- **Hardware**.- El bus entre procesadores (IPB) es la conexión física entre los módulos de procesador. Cada uno de los dos IPB's es capaz de transmitir datos entre un máximo de 16 procesadores.

- **Software**.- El sistema operativo GUARDIAN 90 fue específicamente escrito para soportar esta comunicación y hacerla con completa independencia de la ubicación de los procesos. Uno de los conceptos básicos del sistema operativo es permitir la comunicación de los procesos con cada unos de los demás mandando mensajes y recibiendo respuestas. El IPB extiende esta capacidad a través de los múltiples procesadores.

1.4.8 Sistema Operativo basado en mensajes

Una de las funciones primarias del sistema operativo, es soportar la comunicación entre procesos. Esta actividad no es una excepción ocasional al procesamiento normal, pero es la mejor forma mediante la cual las tareas son completadas en el ambiente de Tandem. El sistema de mensajes es la parte específica del sistema operativo GUARDIAN 90 que soporta la comunicación entre procesos. El sistema de mensajes (Ver *figura 1.4.8.1*):

- Permite a procesos, dentro del mismo procesador, comunicarse con cada uno de los demás. Un proceso manda un mensaje y el otro usualmente emite una respuesta.

- Permite a procesos, en diferentes procesadores, comunicarse con los demás sin el conocimiento de su ubicación.

- Realiza el rendimiento del sistema permitiendo a los procesos ser comunicados a través del sistema.

El ambiente soporta la entrada múltiple de mensajes desde diferentes procesos, cada uno mantiene un mensaje en cola conocido como $RECEIVE. Cada proceso lee mensajes y responde con mensajes a este archivo $RECEIVE.

En un escenario típico involucra dos procesos, uno conocido como "cliente" iniciará la comunicación, y otro conocido como "servidor" recibirá el mensaje y emitirá una respuesta.

- El proceso servidor encola múltiples peticiones y las procesa en una base, primero en entrar / primero en salir, a menos que esté cifrado para responder mensajes en una base de prioridades.

- El mensaje será desplazado del área de datos del cliente al área de datos del servidor. Si hay diferentes procesadores, será usado el bus entre procesos (IPB).

Debe haber una comunicación efectiva entre procesadores, a pesar de que cada uno de ellos sea funcional.

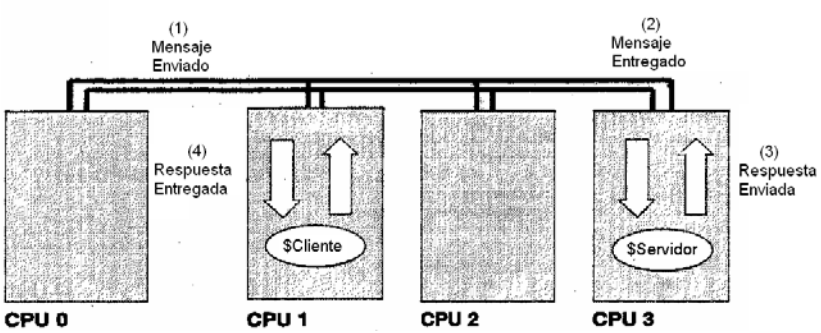


Figura 1.4.8.1 Sistema de mensajes

1.4.9 Comunicación entre procesos. Archivo lógicos

Todo aquello con lo que un proceso de usuario se comunica, tanto con un proceso o un dispositivo (impresora, disco, etc.), es tratado como un archivo. Esto es, el proceso de usuario lógicamente lo abrirá, leerá o escribirá en él y eventualmente lo cerrará. Estos procesos/archivos lógicos pueden estar subdivididos o no estarlos. Una vez abierto, el dispositivo puede leer o escribir hasta que la solicitud del proceso sea detenida.

Interfaz común

Al manejar todos los procesos y dispositivos periféricos como archivos, permite a las aplicaciones acceder a ellos de una manera consistente. La comunicación entre procesos soporta la independencia geográfica. Cuando el proceso manda un mensaje a otro proceso, el sistema de mensaje ubicará ese segundo proceso y pasará el mensaje a él. Esto permite reubicar dispositivos periféricos y procesos, como los requerimientos del sistema lo permitan, además, las aplicaciones permanecerán fácilmente localizables y accesibles.

Restricciones de nombramiento

Para cualquier proceso o dispositivo, el carácter inmediato seguido por el signo "$" debe ser alfabético. A continuación se muestran las restricciones de nombramiento de procesos y dispositivos periféricos

	Acceso local	**Acceso a través de la red**
Dispositivos:	$<7 Caracteres>	$<6 Caracteres>
Procesos:	$<5 Caracteres>	$<4 Caracteres>

1.4.10 Comunicación entre procesos. Periféricos de entrada/salida

Cuando un proceso de usuario necesita comunicarse con el archivo lógico, hace una solicitud de i/o para el segmento de sistema de archivos del sistema operativo, la cual incluye el nombre del archivo lógico. El sistema de archivos lógicos ubica el nombre del dispositivo en la Tabla de Control de Destino (DCT).

La DCT es una lista de todos los procesos llamados, es reproducida a través de todos los procesadores y continuamente actualizada como inicio y término de procesos. El propósito de esta tabla es identificar todos los procesos ejecutados en el sistema y proveer su ubicación (# procesador, # proceso). El sistema de mensajes soporta el transporte de datos, el cual se basa en el *sistema de archivos* que provee el nombre y ubicación de los procesos recibidos.

1.4.11 Procesos NonStop. Mensajes de estado del procesador

Los procesos de I/O sólo fallan cuando el procesador falla, así que, es necesario tener un método para identificar cuando el procesador ha fallado. Si un procesador reconoce que otro ha fallado, éste revisará esos procesos para determinar si alguno de ellos es un proceso backup, del procesador fallido, aquellos llegan a ser procesos primarios. Esto es realizado teniendo comprobaciones independientes de cada procesador para verificar si todos los demás procesadores están en existencia. Cada segundo, cada procesador transmite mensajes a los otros procesadores indicándoles su existencia (Ver *Figura 1.4.11.1).*

Si un procesador falla al recibir una respuesta de un mensaje de indicación de existencia, el otro procesador puede haber fallado también. Sin embargo, puede haber errores con el procesador que envía el mensaje y éste puede no estar listo para escuchar la respuesta del otro procesador.

Para determinar si el problema es un procesador fallido o la no recepción de la respuesta, este procesador también mandará un mensaje a si mismo cuando éste transmita a otros procesadores. Si este falla en recibir un reconocimiento de dos mensajes de existencia consecutivos, comprobará al verificar si recibe su propia copia. Si es así, sabrá que los otros procesadores han fallado y despertará los procesos backup. Si éste nunca recibe su propia copia del mensaje, éste se detendrá a si mismo.

Una vez que el procesador fallido es reiniciado, cada proceso que ha sido primario es reiniciado como un proceso backup. [3]

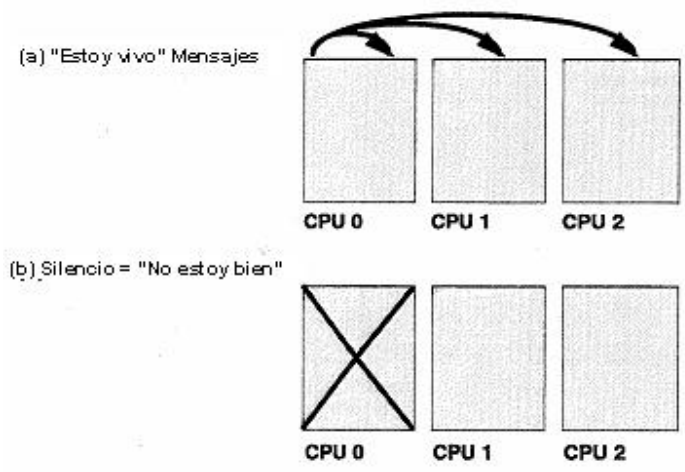


Figura 1.4.11.1 Fallas de CPU

1.5 El ambiente de aplicación OLTP

1.5.1 Software de procesamiento de transacciones en línea

El software de procesamiento de transacciones en línea debe ser flexible y soportar entrada de datos, validación, actualización, transferencia y actividades personalizadas por el usuario. Este software debe aceptar entrada de varios tipos de periféricos (ejemplo: terminales, fax) y poder transmitir estos datos a una gran variedad de periféricos de salida (ejemplo: bases de datos y a otros sistemas de computo). Para proporcionar estos servicios, el software debe ser:

- Disponible
- Fácil de usar
- De alto rendimiento
- Seguro
- Expandible

1.5.2 En un inicio

El procesamiento de transacciones en línea (OLTP), en un principio, tenía su filosofía orientada a archivos. Un simple programa fue estructurado y creado para realizar todas las tareas necesarias del OLTP como permitir la entrada de datos, realizar la manipulación necesaria y el suministro de la salida deseada. Este método "monolítico" pronto presentó muchos problemas como los que a continuación se mencionan:

- **Disponibilidad**: El fallo de un solo componente remueve toda la funcionalidad del sistema.

- **Fácil de manejar:** Un proceso grande, escrito para realizar todas las funciones, es difícil de crear y aún mas difícil de eliminar fallos y mantener el proceso.

- **Funcionamiento:** Como el proceso crece para adecuar más usuarios y soportar más funcionalidad, éste absorbe más recursos del sistema y la efectividad se degrada continuamente.

- **Seguridad/Integridad de datos**: El fallo del proceso durante la actualización de la base de datos ocasiona daños a la base de datos.

- **Expansibilidad:** El proceso aumenta su tamaño a medida que sean agregados usuarios y funcionalidades, Eventualmente, el crecimiento requiere del reemplazo de hardware y, a menudo, requiere considerables modificaciones al software que se realizan para soportar el nuevo hardware. El gran programa (método "monolítico") corre sobre el sistema Tandem, pero no toma ventaja de su hardware ni de la arquitectura de software. Este método deja la mayor parte del ambiente sin usar.

1.5.3 La solución: cliente-servidor

El método "cliente-servidor" toma los tres eventos básicos de OLTP (entrada de datos, manipulación y salida) previamente encontrados en el programa "monolítico" y divide estas actividades en dos procesos:

El proceso "cliente"

Es la actividad principal que acepta datos de entrada del usuario final. Este proceso generalmente opera de la siguiente manera:

- Despliega una pantalla.

- Acepta datos de entrada del usuario.

- Realiza edición preliminar sobre esos datos.

- Manda estos datos en forma de mensaje al proceso servidor, el cual libera la respuesta para comunicarla a otros usuarios.

- Procesa la respuesta del servidor.

Este proceso es conocido como el cliente, porque su función básica es solicitar datos del usuario final.

El proceso "servidor"

Es la segunda actividad, realiza la manipulación de datos y actividades de salida de datos, opera generalmente de la siguiente manera:

- Recibe mensajes del proceso cliente.
- Realiza los servicios deseados, como la manipulación de datos (consulta de tablas, validación, cálculos numéricos), y la salida de datos (actualización de bases de datos, solicitud de creación).
- Emite una respuesta al cliente.

En la actualidad, aproximadamente la mitad de las aplicaciones que se lanzan al mercado están basadas en el modelo Cliente/Servidor, tomando en cuenta que dichos conceptos están fundados en otros que son extensiones de los originales. El concepto "Cliente" se extiende no sólo a la terminal del usuario, sino al programa ejecutándose en dicha terminal. El término "Servidor" no sólo se refiere a la computadora donde se ejecuta la otra parte de la aplicación, sino, también, al proceso de la aplicación específica en sí. La conversión de aplicaciones Pathway (basadas en terminales) a la arquitectura Cliente/Servidor basadas en una Estación de trabajo, es bastante simple, pues la parte "cliente" emigra a la Estación y la sección "servidor" permanece en el equipo Tandem.

Ventajas de la solución "cliente-servidor"

Las ventajas que pueden atribuirse a la estructura "cliente-servidor" son: *(Ver figura 1.5.3.1)*

- **Disponibilidad.-** La distribución de procesos permite al cliente o al servidor continuar funcionando, incluso con la falla de otros procesos despliega una pantalla.
- **Fácil de uso**.- La creación o modificación de código de los programas son ahora realizadas sobre una fuente, catalogando cada uno (aproximadamente la mitad del tamaño) del listado monolítico.
- **Funcionamiento.-** Cada proceso se ejecuta en un CPU separado, permitiendo el procesamiento en paralelo de alta velocidad.
- **Seguridad/Integridad de datos.-** El facilidad para realizar el codificado de NonStop permite continuar la ejecución de una función fallida
- **Expansibilidad.-** El crecimiento, ya sea del cliente o del servidor, no afectará directamente el funcionamiento del otro. El servidor puede ser fácilmente reubicado en un nodo diferente de la red. [4]

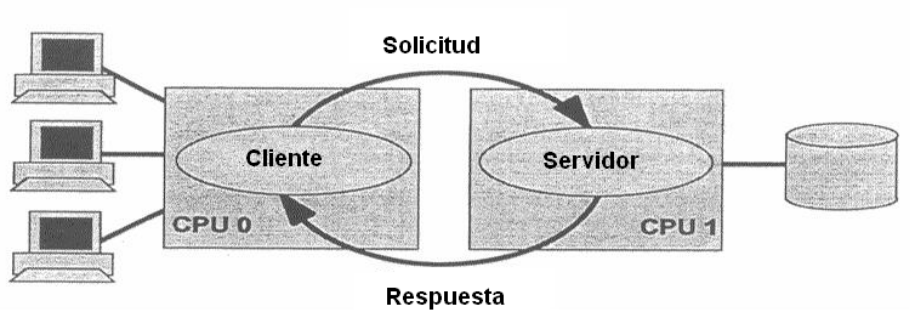


Figura 1.5.3.1 Modelo Cliente/Servidor

1.5.4 Introducción al PATHWAY

El PATHWAY no es un programa, pero si un ambiente consistente en los siguientes códigos objeto:

- $SYSTEM.SYSTEM.PATHMON
- $SYSTEM.SYSTEM.PATHCOM
- $SYSTEM.SYSTEM.PATHTCP2
- $SYSTEM.SYSTEM.SCOBOLX
- $SYSTEM.SYSTEM.SCUP

A través del PATHWAY se monitorean los procesos, todos los componentes pueden ser iniciados, terminados, monitoreados y modificados. El ambiente Pathway es la base actual del ambiente cliente-servidor interno de Tandem. Se basa en la idea del procesamiento cooperativo, es decir, la distribución de la carga de trabajos entre los diferentes procesadores. De esta manera, las aplicaciones Tandem conservan la tolerancia a fallos y escalabilidad para la cual fueron diseñados. La integridad de las BD manipuladas por el Pathway son protegidas por el NonStop Transaction Manager/Massively Parallel (NonStop TM/MP).

1.5.5 Configuración del PATHWAY

El monitor de procesos PATHWAY: PATHMON
El monitor de procesos PATHWAY (nombrado $PM1 en el ejemplo) está formado por códigos proporcionados por Tandem en $SYSTEM.SYSTEM.PATHMON. Proporciona el proceso "cliente" con el nombre del miembro más óptimo de una clase del servidor. Éste es el primer proceso iniciado y el último proceso en terminar en el escenario "cliente-servidor", ahora llamado "Ambiente Pathway".

En el orden del administrador del sistema, PATHMON controla el inicio y término de los clientes, servidores y terminales. También proporciona los status y la información de configuración de los clientes, servidores y terminales. Funciona como un par de procesos NonStop con el fin de proporcionar amplia disponibilidad. Un proceso PATHMON puede soportar múltiples aplicaciones.

Proceso de comunicación PATHWAY: PATHCOM

El administrador del sistema proporciona comandos para PATHMON a través del código $SYSTEM.SYSTEM.PATHCOM. A través del PATHCOM, el administrador del sistema puede iniciar, terminar, modificar y monitorear la terminal de control de procesos pathway (TCPs),

terminales y servidores. Este proceso es activado únicamente cuando esta funcionalidad es necesaria. (Ver *figura 1.5.5.1)*

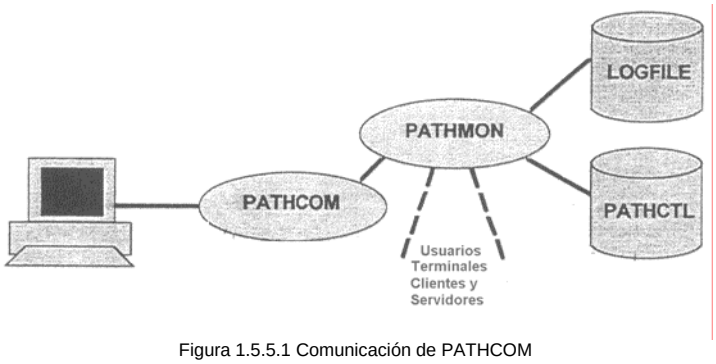


Figura 1.5.5.1 Comunicación de PATHCOM

Terminal de Control de Procesos (TCP)

La Terminal de Control de Procesos (TCP)[8] consiste en códigos proporcionados en $SYSTEM.SYSTEM.PATHCTP2. Se ejecuta como un proceso par NonStop con el fin de proporcionar amplia disponibilidad.

Las aplicaciones en línea incluyen varios dispositivos de entrada y salida. El proceso "cliente" del Pathway es conocido como TCP (Terminal Control Process). El TCP es un proceso par tolerante a fallos, puede manejar a varios usuarios concurrentemente y aceptar peticiones de diferentes dispositivos. (Ver *figura 1.5.5.2)*

Cuando se realiza una petición, ésta no se atiende directamente por el TCP, sino que se canaliza por medio del Pathmon y se le asigna un nombre a dicho proceso. El Pathmon administra los procesos Pathway y es el responsable de iniciar todos los demás procesos, incluyendo los procesos TCP.

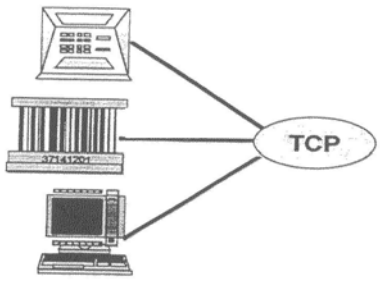


Figura 1.5.5.2 Pathway TCP

La biblioteca de COBOL (POBJDIR y POBJCOD)

El código escrito por el usuario en lenguaje COBOL es guardado en dos archivos librerías, POBJDIR y POBCOD. Y un tercer archivo POBJSYM es usado para depurar errores del programa. Este código es compilado usando el código ubicado en $SYSTEM.SYSTEM.SCOBOLX

[8] TCP: Terminal Control Process

El TCP lee programas de estas librerías y los ejecuta a nombre de las terminales unidos lógicamente a el. Cada uno de estos programas está escrito en Cobol, como si fuera para un solo usuario, y generalmente proporciona las siguientes actividades.

- Despliega una pantalla.
- Acepta datos del usuario final.
- Manda estos datos a un servidor.
- Procesa una respuesta.
- Llama otros programas escritos en Cobol (opcional).

POBJ interpreta para “Pseudo-OBJect code”, indicando que el resultado de la compilación no es el código verdadero compilado. Reconociendo que esta pseudo-compilación se ejecuta más lento que la actual compilación del código y debe ser usado sólo para simples actividades. Ésta es una “compilación” que puede ser interpretada por el TCP.

POBJDIR es el directorio archivo donde el programa es almacenado.

POBJCOD es el código archivo donde el pseudo código objeto es almacenado.

Los archivo librerías contienen copias de varios programas escritos por diferentes usuarios.

PATHWAY, Control de archivos (PATHCTL)

El PATHMON es el primer proceso iniciado en el ambiente PATHWAY. Este proceso genérico, no tiene ningún conocimiento de su aplicación en el ambiente. A través del PATHMON el administrador ordena leer el archivo especialmente formateado con la configuración del PATHWAY conocido como PATHCTL. El administrador del sistema almacena la información detallada y especifica en este archivo lo referente a: TCPs, servidores y terminales involucradas. El proceso de lectura de PATHCTL, realizado cada vez que el PATHMON es iniciado, le proporciona toda la información necesaria para manejar el ambiente en línea.

1.5.6 RSC

Debido a que la lógica de Pathway se encuentra ahora “dividida” entre una estación de trabajo y el propio Tándem, se desarrolló un producto que hiciera las funciones de cliente/servidor (RSC: Remote Service Call), el cual incluye un esquema sencillo y eficiente de TDP[9] para proveer los servicios hacia el “Cliente”. RSC también incluye controladores del lado de la estación de trabajo y aplicaciones corriendo en Windows, DOS, OS/2 y UNIX. (Ver *figura 1.5.6.1)*

El proceso conocido como LINKMON maneja las comunicaciones TDP con los procesos del servidor. El TDP es considerado la contraparte de los controladores del RSC en el servidor. Cuando hay una solicitud, el TDP pide al LINKMON identificar el proceso y dirigirlo hacia el PATHMON apropiado. [5]

[9] TDP: Transaction Delivery Process

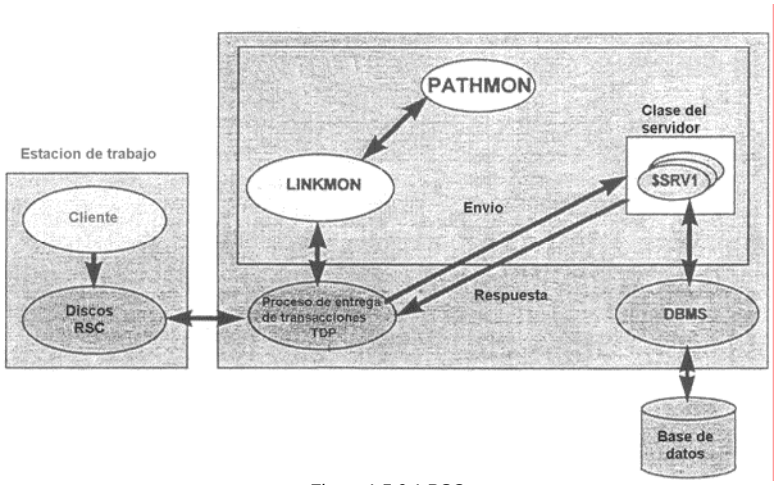


Figura 1.5.6.1 RSC

1.6 Base de datos relacional

Los sistemas de procesamiento de transacciones en línea, involucran grandes bases de datos, grandes volúmenes de actualizaciones diarias. Estos sistemas han forzado a diseñadores a adoptar los mejores principios de ingeniería en colocar ambos sistemas es simple confiable y productivo.
Sistemas de procesamiento de transacciones en línea son similares a sistemas de tiempo real, a tal grado que su objetivo es proveer una respuesta predecible a peticiones predecibles dentro de un tiempo aceptable. El tiempo de respuesta necesita no ser tan corta como es necesario en los sistemas de tiempo real. La razón: los usuarios u operadores, que interactúan con la computadora son normalmente parte de una gran transacción que abarcan a otra persona para el cual el operador esta desarrollando ese servicio, en algunos casos el usuario es un consumidor.

Aplicaciones de procesamiento de transacciones en línea en tales áreas como finanzas, seguros, manufactura, distribución y comercio son típicamente grandes en cualquier medida, tamaño de base de datos, numero de usuarios, numero actualización de base de datos, complejidad y volumen de procesamiento, incluso extensión geográfica (algunos sistemas son distribuidos sobre redes internacionales). [6]

Las bases de datos Tandem permiten acceder a datos locales o remotos utilizando comandos estándares, sin necesidad de preocuparse cómo se encuentran distribuidas las bases de datos. De ésta manera, los datos pueden ser consultados y/o actualizados desde cualquier parte de la red.

Las metodologías de almacenamiento/recuperación usadas en el sistema Tamdem son: Enscribe y NonStop SQL, que son sistemas de bases de datos relacionales. El concepto de bases de datos relacionales se basa en tablas que contienen una serie de filas y columnas para mantener la información. Dichas tablas no contienen punteros hacia ciertas direcciones y en su lugar se crean relaciones entre las diferentes columnas de las mismas, pudiendo tenerlas repetidas. La desventaja de las BD Relaciones es que, podrían consultarse demasiados registros para reportar una búsqueda y, por lo tanto, son lentas y recomendables para bases pequeñas. (Ver *figura 1.6.1)*

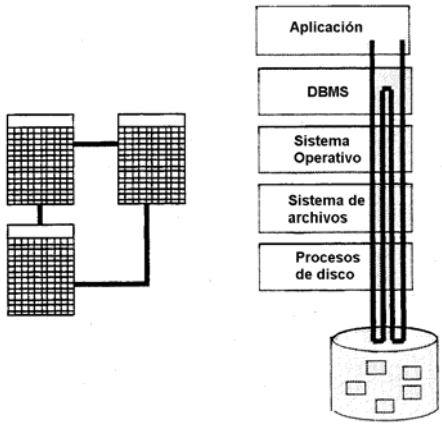
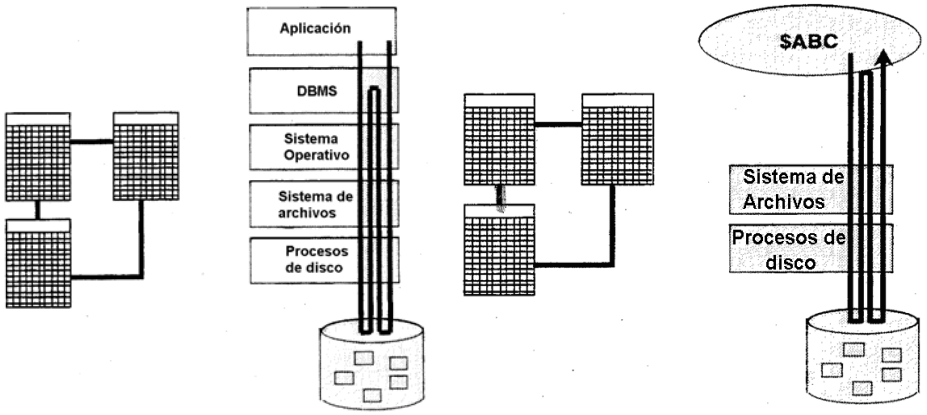


Figura 1.6.1 Bases de Datos Relacionales

1.6.1 Subsistema de archivo en disco "Enscribe"

Enscribe provee un alto nivel de acceso para la manipulación de registros en una base de datos relacional. Como una parte integral del sistema operativo GUARDIAN 90, distribuye a través de dos o más procesadores. Enscribe ayuda a asegurar la integridad de datos si un módulo, canal de I/O, o disco sufre un fallo.



Figuras 1.6.1.1 Comunicación en BD Relacionales y Archivos Enscribe

En la figura 1.6.1.1 se observa una evolución en la manera en que se consulta del disco cierta información en el sistema de las bases de datos relacionales y el subsistema de archivos "Enscribe" propio de Tandem. En éste último se ofrece un mejor desempeño y mayor velocidad debido a que, la petición por parte de la aplicación (en éste caso $ABC) no tiene que pasar por tantas capas de Software, sino que se dirige directamente hacia el archivo del sistema y de ahí al proceso del disco.

1.6.2 Lenguaje Estructurado de Consulta (SQL)

El SQL ha sido adoptado para el manejo de BD de manera estándar para mainframes y mini-computadoras. Sin embargo, debido a que SQL se utiliza para el manejo de BD relacionales, sigue contando con la limitante de ser muy lento. Tandem desarrolló el NonStop SQL/MP para

proporcionar los beneficios las BD relacionales y el SQL, pero evitando el bajo desempeño que se asocia con otras implementaciones del modelo relacional.

Esto se logra haciendo "más inteligente" el proceso del disco (llamado DP2), pues tiene la lógica para manejar varios grupos de registros a la vez (como se observa en la figura 1.6.2.1), debido a esto, las consultas a disco son mucho más veloces.

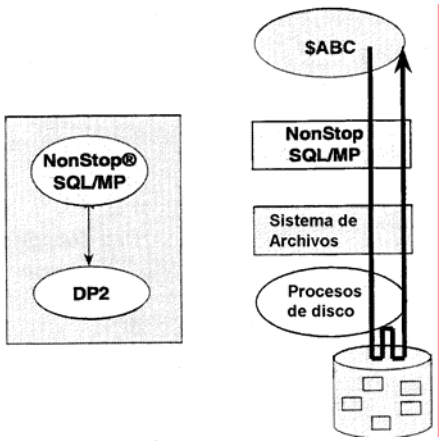


Figura 1.6.2.1 SQL

1.6.3 Terminología Tandem para BD

Archivos tipo Enscribe: compuestos de Campos y registros, los cuales forman archivos.

Archivos NonStop SQL: compuestos de filas y columnas, los cuales forman Tablas. (Ver *figura 1.6.3.1)*

Los archivos Enscribe siempre existirán físicamente en el disco, los campos siempre se encuentran combinados para formar registros y estos a su vez crean archivos en disco. Esta condición no se cumple siempre para las tablas SQL, pues las filas y columnas se unen de tal manera, que crean estructuras virtuales que consisten en datos con una localidad física y no existen como entidades separadas. Estas estructuras son conocidas como *vistas*

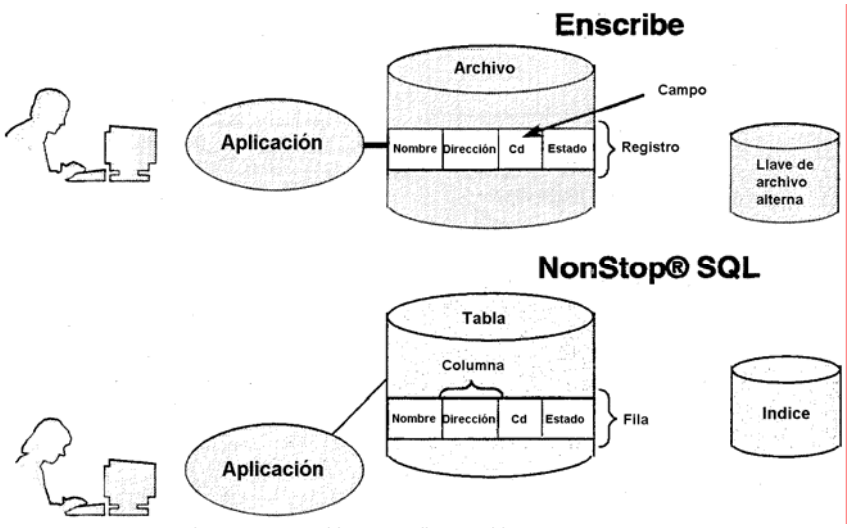


Figura 1.6.3.1 Archivos Enscribe y Archivos NonStop SQL

1.6.4 Tipos de archivos

El tipo de archivo determina cómo se ordenan los registros en el archivo. Los tres tipos son:

- **Ordenado por entrada**: Cuando una aplicación utiliza este tipo de archivo, significa que almacena en él la información a medida que va ocurriendo, es decir, se insertan los registros al final del archivo.
- **Relativo**: Se utiliza por aplicaciones donde se requiere acceso aleatorio y en donde el número de registro puede funcionar como la llave del mismo. Los registros se insertan en cualquier localidad en el archivo y pueden ser actualizados o borrados (espacio de almacenamiento reutilizable).
- **Ordenados por llave**: Son las más comunes en las bases de datos comerciales. La llave única es un campo único o combinación de campos en el registro, asimismo, los registros se almacenan en orden por la llave primaria.

1.6.5 Archivos y tablas particionados

Un archivo lógico o tabla puede ser particionado(a) en varios discos (ver *figura 1.6.5.1)*, dicha partición es transparente a la aplicación; también puede hacerse en diferentes nodos. Un archivo o tabla particionado(a) provee:

- Un procesamiento de archivos grandes con mayor velocidad, a través de técnicas de procesamiento paralelo.
- Operaciones de recuperación más rápidas y menos problemáticas.
- Almacenamiento para archivos grandes (relacionado con el término Integridad).

El NonStop SQL y el Subsistema Enscribe manejan la extracción y acomodo de registros en las particiones.

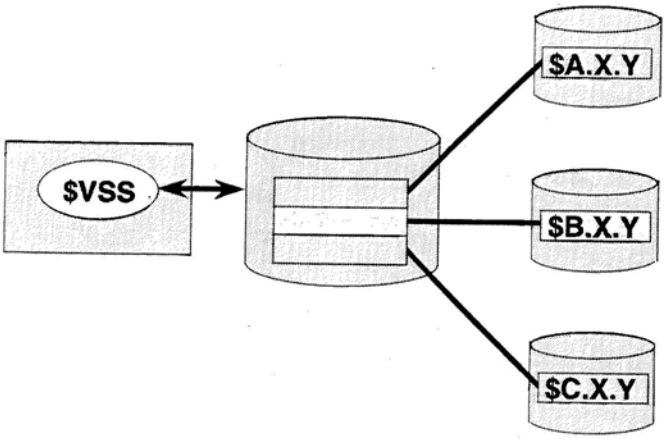


Figura 1.6.5.1 Archivos y tablas particionados

1.6.6 Acceso a datos en BD Enscribe

Existen tres formas de acceder a los registros de una base de datos Enscribe:

- **Programa Utilitario del Archivo** (FUP)[10].- El usuario puede copiar registros de archivos utilizando el comando FUP COPY. Este comando generalmente copia todo el contenido de un archivo de su inicio o de algún lugar especificado por el usuario.

- **Acceso Programático**.- Programas creados por un desarrollador en un lenguaje disponible para accesar los registros de la base de datos.

- **Software Enform**.- Se utiliza para realizar búsquedas a través de archivos Enscribe utilizando el "Enform Query Language". Permite hacer búsquedas en archivos Enscribe y tiene permite el Lenguaje de Búsquedas parecido al lenguaje natural, además, pueden compilarse y almacenarse búsquedas con la finalidad de tener una aplicación más completa.

  Dicho software accede archivos Enscribe creados utilizando el Lenguaje de Definición de Datos (DLL: Data Definition Language). Sumado a esto, las búsquedas pueden realizarse en archivos sencillos, archivos particionados y archivos distribuidos en diferentes nodos.

1.6.7 Acceso a una BD NonStop SQL/MP

Están presentes dos vías para acceder a este tipo de Bases de Datos:

- SQLCI (SQL Conversational Interface). (Ver *Figura 1.6.7.1)*

- Declaraciones propias del SQL.

Esta interfase es utilizada para crear, alterar y borrar tablas; así como insertar, borrar, actualizar y seleccionar tablas. Los comandos se introducen de manera interactiva y los resultados se arrojan en la terminal. SQLCI utiliza el optimizador SQL para determinar el método más eficiente para realizar la búsqueda.

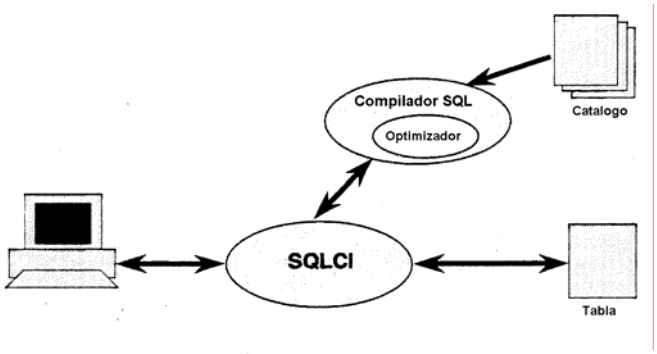


Figura 1.6.7.1 SQLCI

[10] FUP: File Utility Program

1.6.8 Procesamiento distribuido del SQL/MP NonStop

El NonStop SQL puede utilizarse en el procesamiento de BD distribuidas. En este caso, la división entre el SQL y el proceso del disco es muy importante.
Una consulta funcionaría de la siguiente manera: se realiza la consulta al proceso del disco (DP2), éste selecciona los registros que se utilizan para responder a dicha petición y los envía al NonStop SQL. Así, el resultado de la búsqueda, puede ser enviado para que el usuario lo observe. (Ver *figura 1.6.8.1)* [7]

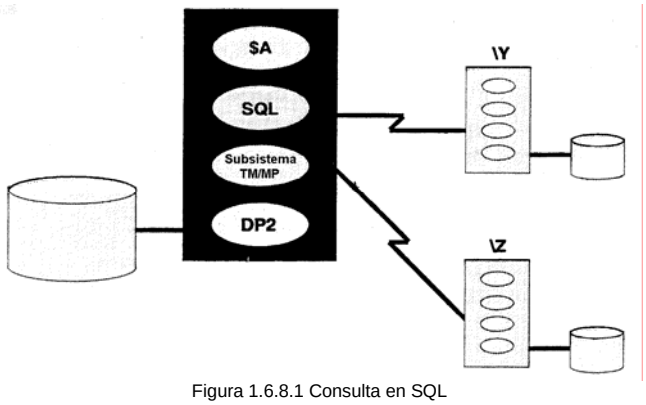


Figura 1.6.8.1 Consulta en SQL

2 SERVICIOS

La plataforma Tandem posee servicios como la protección a la base datos, recuperación de dicha la base datos al punto anterior al fallo, encriptación de la información, trabajo en red, respaldo de manera remota de los sistemas. Estos y mas servicios, serán descritos en el presente capitulo y se tocarán temas como respaldo de la base de datos en línea, es decir base de datos espejo, con tolerancia a fallos, disponibilidad, tolerancia a fallos, misma que se puede emplear en sistemas remotos. También se tocarán temas como el manejo en la seguridad del sistema, identificación de usuarios, usuarios y grupos con privilegios en el manejo de la base de datos. Mantenimiento remoto al sistema, respaldo remoto del sistema, entre otros.

Esto a fin de proporcionar al lector el conocimiento de los servicios proporcionados por la plataforma Tandem y así explotar al máximo los beneficios que nos proporciona.

2.1 Protección de la base de datos

2.1.1 Protección de base de datos TMF[11]

TMF es un software, a nivel de sistema, que proporciona protección y consistencia a la base de datos, soporta archivos de bases de datos tanto ENSCRIBE como NonStop SQL. Frecuentemente es utilizado para sustituir procesos de usuario NonStop, debido que es más competente en situaciones de recuperación.

TMF protege únicamente archivos de base de datos auditados especificados por el usuario.
Todos los demás archivos (listados de fuente, código objetos, archivos no auditados de base de datos) deben ser respaldados por el usuario.

2.1.2 Características de TMF

- **Retiro de transacciones.-** TMF protege contra alguna inconsistencia de la base de datos retirando transacciones incompletas, restaurando la base de datos al estado en que se encontraba antes de que la transacción comenzara.

- **Recuperación autorollback.-** Esta característica permite recuperación automática de volumen de disco para archivos TMF auditados. Esto ocurre después de una falla multicomponente, por ejemplo, una pausa de energía extendida. Automáticamente retirará todas las transacciones en proceso al momento de la falla.

- **Recuperación rollforward.-** Esta característica provee de recuperación manual de archivo para los que son del tipo TMF auditados en el caso de fallas técnicas o daño de los archivos auditados (por ejemplo: daño en una cabeza de disco). Entonces, pueden ser recuperados dentro de la transacción más reciente por el usuario. Los archivos borrados accidentalmente también se pueden recuperar desde el punto en el que fueron perdidos.

- **Control de Concurrencia.-** TMF implementa un protocolo de bloqueo de registro que protege contra los intentos concurrentes, por más de una transacción, de actualizar el mismo registro. Los bloqueos de registros son mantenidos hasta que la transacción entera es completada.

[11] TMF: Transaction Monitoring Facility

- **Copiado de la base de datos en línea.-** TMF soporta la capacidad de copiar la base de datos mientras el procesamiento de transacción en línea continúa, incluyendo las actualizaciones a la base de datos.

2.1.3 Conceptos principales de TMF

Archivos de base de datos auditados

Los archivos de la base de datos deben ser "auditados" por TMF para utilizar sus características de protección. Esto se hace estableciendo la bandera de revisión o auditoria al momento de crear el archivo o alterar la bandera de revisión usando el programa de utilidad de archivos (FUP). Únicamente estos archivos de base de datos identificados por el usuario (usando FUP) serán protegidos por TMF.

Rastreo de auditoria de TMF

Las características de Retiro de Transacción, Recuperación Autorollback, y Recuperación Rollforward requieren el soporte de un archivo de rastreo de auditoria TMF y los diferentes tipos de registros de auditoria que contiene. El rastreo de auditoria contiene los registros de auditoria necesarios para recrear o volver atrás una transacción. Esto se logra mediante el uso de entradas de registro de transacción Begin y End, además de entradas de registro de las imágenes del "Antes" y "Después" del registro de la base de datos que fue agregado, actualizado o borrado. Existen también otras entradas en el rastreo de auditoria.

Una transacción no es totalmente completada y los bloqueos de registro no son liberados hasta que el registro de Liberación de Transacción ha sido escrito en el rastreo de auditoria. Los archivos de base de datos deben ser auditados por TMF para utilizar sus características de protección. Esto se logra estableciendo la bandera de auditoria en tiempo de creación de archivo o alterando la bandera de auditoria utilizando FUP. Sólo esos archivos de base de datos identificados por el usuario (usando FUP) serán protegidos por TMF

Los rastreos de auditoria deberían mantenerse en unidades de disco separadas de los archivos de base de datos auditados. Los volúmenes en los que se ubican los archivos protegidos TMF deberían respaldarse en discos espejo para proporcionar procesamiento de base de datos continuo en el caso de falla de un disco único.

2.1.4 Control de concurrencia

En el ambiente OLTP, es posible que varios usuarios intenten realizar actualizaciones simultáneas de un único registro. Los resultados pueden ser fácilmente caóticos y en el mejor de los casos, impredecibles. Tanto el sistema de administración de base de datos ENSCRIBE como el NonStopSQL proporcionan soporte de bloqueo de registros para evitar que esto ocurra. Los procedimientos de bloqueo de registros son muy diferentes para ambientes TMF – protegidos y TMF – no protegidos.

Sin TMF

Sin protección TMF, los bloqueos de registros son liberados entre operaciones de E/S. Una segunda aplicación puede acceder al registro escrito en la primera operación antes que la segunda sea completada. Esto puede dar información errónea si ocurre una situación de falla o reintentar para la aplicación inicial.

Con TMF

TMF establece el Control de Concurrencia mediante la imposición de técnicas de bloqueo de registros. Únicamente una transacción puede actualizar un registro a la vez.

Con la protección TMF, los bloqueos de registros son liberados únicamente después de que TMF registra que el fin del registro de la transacción ha sido escrito en el rastreo de auditoria. El proceso de fin de transacción empieza, ya sea con una sentencia de transacción “abort” o “end”, emitida desde la aplicación. Ciertas situaciones, tales como una falla del software de aplicación, causarán un “abort” automático finalmente finalizando la transacción. La protección del bloqueo de registro TMF es establecida en transacciones de red y locales.

2.1.5 Volcados en línea y volcados de rastreo de auditoria

Volcados en línea

TMF soporta opcionalmente la capacidad de realizar respaldos de la base de datos durante procesamientos activos de transacciones en línea. Esta operación implica la creación de un volcado de memoria en línea (OLD – online dump) para grabar.

Los volcados en línea son copias de cinta de los archivos de base de datos auditados. El volcado en línea puede ser generado mientras los archivos de base de datos son abiertos para procesamiento de transacción. Rollforward hace uso de estas cintas para reestablecer la(s) base de datos en el caso de una recuperación. Todos los archivos no auditados por TMF deben ser respaldados por el usuario.

Volcados de rastreo de auditoria

Al mismo tiempo, TMF detecta todos los cambios que están siendo hechos a la base de datos en su rastreo de auditoria, los cuales son subsecuentemente almacenados en cintas de rastreo de auditoria. Ambas, almacenamiento de cintas de volcado en línea y de rastreo de auditoria son actividades opcionales de TMF.

Los volcados de rastreo de auditoria son copias en cinta de los rastreos de auditoria después de que han “llenado” y pasado al siguiente rastreo de auditoria en la secuencia. Durante el proceso de recuperación, rollforward hace uso de estas cintas de volcado de rastreo de auditoria y todos los rastreos de auditoria en el disco.

2.1.6 Programa de base de datos remota duplicada (RDF)

El programa de base de datos remota duplicada (RDF[12]) monitorea la actividad de transacción TMF en una base de datos que reside en un sistema primario y aplica actualizaciones duplicadas a una copia idéntica de la base de datos residiendo en un sistema de respaldo. La base de datos en el sistema primario debe ser protegida por TMF. La base de datos de respaldo es actualizada continuamente por RDF tan pronto como las modificaciones en el sistema primario puedan ser transmitidos y aplicados. RDF proporciona una copia actualizada de una base de datos a un sitio remoto (sitio de respaldo) y otorga permiso de disponibilidad máxima al sitio primario. Cada sitio primario requiere de su propio sitio de respaldo.

[12] RDF: Remote Duplicate Database Facility

Ciertas funciones que pueden ser realizadas en el sitio de respaldo reducen la carga en el sistema primario, proporcionando un mejor tiempo de respuesta para aplicaciones OLTP. Estas funciones incluyen:

- Consultas a la base de datos.
- Reporte en lote.
- Respaldo de la base de datos.

RDF puede ser un producto importante en un plan de recuperación de desastre. Sin RDF, la fuente principal de recuperación de desastre es el almacenamiento de cinta fuera del sitio.

Beneficios de RDF

- **Disponibilidad.-** El sistema de respaldo puede continuar las funciones de procesamiento de datos primarios poco después de que el sistema primario falle. Las consultas de la aplicación y reporte en bloque pueden hacerse en el sistema de respaldo. Los respaldos de la base de datos pueden realizarse en el sistema de respaldo incrementando la disponibilidad del sistema primario.
- **Tolerancia a fallas.-** RDF corre como procesos persistentes. En el caso de la falla de un proceso, el subsistema RDF iniciará otro proceso.
- **Flexibilidad.-** Las configuraciones del sistema pueden ser diferentes entre los sistemas primarios y de respaldo. El número y tipo de procesadores puede ser diferente. El número y nombres de los volúmenes puede ser diferente también.
- **Integridad de los datos.-** Las actualizaciones a los archivos de la base de datos son transmitidas al sistema remoto en tan sólo unos segundos, reduciendo dramáticamente el número de transacciones que podrían perderse si el sistema primario se viera envuelto en un desastre catastrófico.

Revisión operacional de RDF

TMF registra los resultados de cada transacción en el Rastreo de Auditoria Maestro (MAT[13]). Dependiendo de el resultado de la transacción (completada exitosamente o abortada) un registro de commit o abort es escrito en el MAT.
El proceso de RDF en el sistema primario lee del MAT (sólo hay uno por sistema) y extrae los registros auditados incluyendo registros commit/abort para volúmenes RDF-protegidos. RDF formatea y transmite estos registros al sistema de respaldo sobre líneas de comunicación de datos. Un proceso RDF en el sistema de respaldo recibe los datos transmitidos y escribe a un archivo de disco temporal, el archivo de Imagen RDF.

Otros procesos RDF, ejecutándose en el sistema de respaldo, leen el archivo de imagen para registros auditados, revisan los registros commit/abort y aplican imágenes-después para transacciones completadas a la base de datos de respaldo. [8]

2.2 Seguridad del sistema

Los datos son una parte irremplazable y vital de todo negocio. No sólo se deben proteger los datos, sino también todo lo que permita a las personas acceder a ellos, incluyendo: equipo de cómputo, medios de almacenamiento, sistema operativo, software de aplicación.

[13] MAT: Master Audit Trail

Las amenazas a la seguridad incluyen: Catástrofes naturales y humanas, actos intencionales tales como acceso al sistema inválido o ilegal y alteración de la información por daños o destrucción, errores debido a procedimientos débiles o faltantes, inatención o errores honestos y fallas de hardware.

La seguridad del sistema es un problema administrativo que tiene implementaciones técnicas como acceso restringido al software del sistema. La política de seguridad de su instalación debería establecer las necesidades de seguridad y las metas de su compañía, indicar quién debe y quién no acceder a los datos, describir los procedimientos de protección que emplea y los departamentos que deben seguirlos.

Las herramientas de seguridad que incluye Tandem son: Identificación, autorización, auditoria, encriptación. Se enfoca en la protección que proveen: el NonStop Kernel, software safeguard, CA-Unicenter y los Productos Atalla. (Ver *figura 2.2.1)*

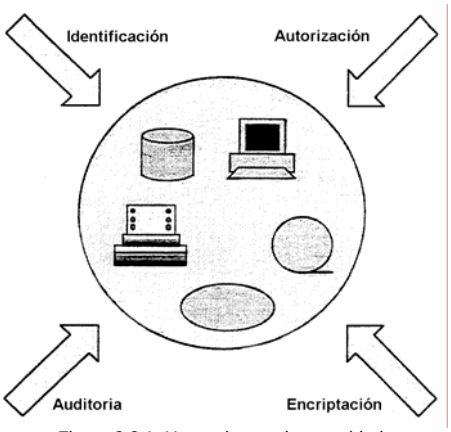


Figura 2.2.1 Herramientas de seguridad

2.2.1 Capas de seguridad de Tandem

El sistema operativo GUARDIAN 90 y sus utilerías ofrecen protección básica del sistema. El producto de seguridad del sistema, SAFEGUARD, amplía las características de seguridad de GUARDIAN 90 para incluir, auditoria, control de acceso extendido y aspectos de identificación. Las capas de seguridad de Tandem incluyen:

- **Aplicación.-** La definición de aplicación depende completamente de las necesidades de seguridad de la aplicación como se perciben por los diseñadores de la aplicación y los que las implementan. Las elecciones de seguridad del desarrollador son soportadas por los mecanismos subyacentes de la protección del sistema.

- **Software.-**
  - Los archivos de disco han asumido siempre protección, RWEP[14] (leer, escribir, ejecutar, eliminar).
  - Los procesos tienen protección implícita basada en “propiedad”
  - Otras entidades en el sistema pueden ser protegidas basadas en listas de acceso.

[14] RWEP: Read, Write, Execute, Purge

- **Hardware.-** Las instrucciones de la máquina son implementadas en micro código que implementan los constructores de protección de memoria.

2.2.2 Identificación de usuarios

La forma de identificar usuarios es por medio del inicio de sesión *(Ver figura 2.2.2.1).* Cada usuario debe tener un nombre de usuario y password válidos para acceder al sistema. Existen 256 grupos, formado cada uno por un máximo de 256 usuarios. El usuario específico es identificado por el número del grupo del que es miembro y el número de usuario (x,y). En la ilustración vemos al usuario 20 del grupo 10 (10,20), cuyo login (usuario con el cual inicia sesión) es escribe con el nombre del grupo y del usuario, respectivamente, separados por un punto (MFG.JAN).

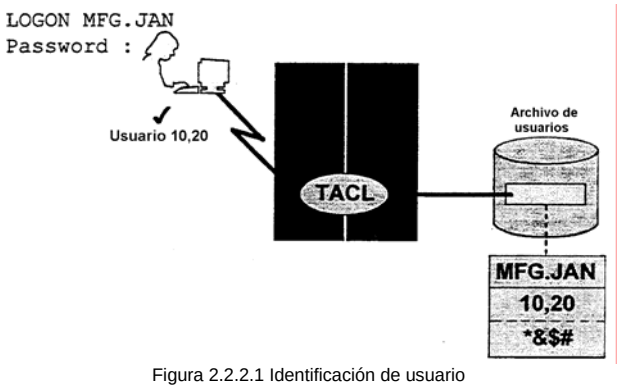


Figura 2.2.2.1 Identificación de usuario

Password para equipos remotos

Los passwords del usuario deben ser establecidos en el sistema local. En el nodo local debe existir un password, al igual que en el nodo remoto; ambos se almacenan en el archivo USERID. Posteriormente, deben establecerse ambos passwords, pero ahora desde el sistema remoto *(Ver figura 2.2.2.2).*

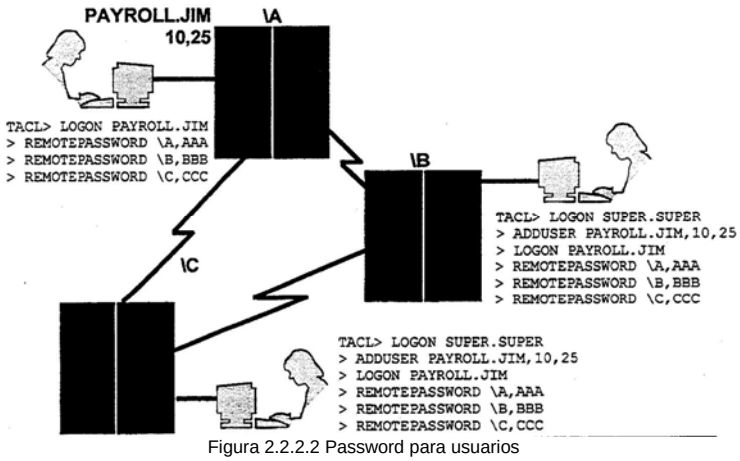


Figura 2.2.2.2 Password para usuarios

2.2.3 Usuarios y grupos

Un usuario 0 al 254 de los mismos grupos (0 a 254) no tienen privilegios especiales. El usuario 255 del grupo 0 al 254 puede agregar o borrar usuarios y acceder cualquier archivo de un usuario de su grupo, por eso se conoce como Administrador del grupo.

El grupo 255 es conocido como "Súper". Los usuarios 0 al 254 de este grupo con usuarios del "Supergrupo" pueden introducir ciertas instrucciones en algunas utilidades propias de Tandem. El usuario 255 del grupo 255 es conocido como súper, es decir, el superusuario del supergrupo y tiene control total del sistema, es decir, crear o borrar nuevos administradores de grupos, acceder y/o borrar cualquier archivo. (Ver *figura 2.2.3.1)*

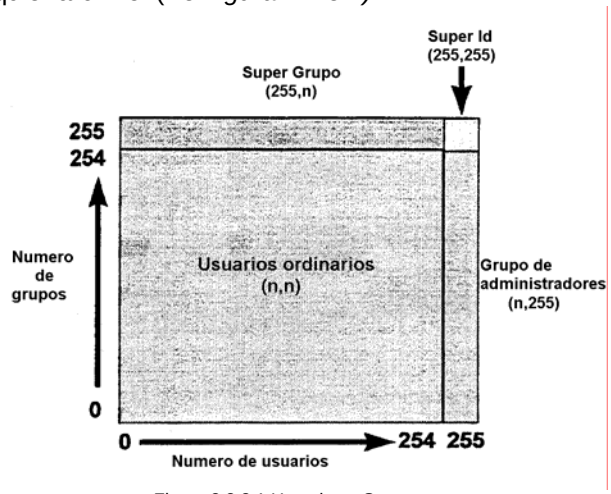


Figura 2.2.3.1 Usuarios y Grupos

2.2.4 Permisos de archivos

Los usuarios acceden a los archivos de disco en 4 modos: Lectura, Escritura, Ejecución, Eliminación[15]. Cada archivo tiene un dueño (O) y él puede dar permisos al grupo (G) o a todos los usuarios (A) a sus archivos. En la figura 2.2.4.1, el sistema verifica que el usuario tenga permisos para TEDIT, si los tiene, verifica lo mismo para el archivo que se desea.

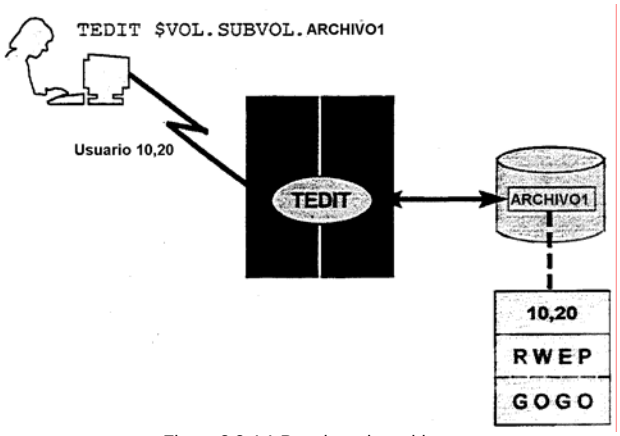


Figura 2.2.4.1 Permisos de archivos

[15] READ, WRITE, EXECUTE y PURGE

2.2.5 Programas PROGID

Los programas PROGID permiten a un usuario usar temporalmente un subconjunto controlado de privilegios de otro usuario. Cuando un usuario ejecuta un programa PROGID, el programa opera utilizando los privilegios del propietario del programa y accede únicamente a esos recursos a los que el propietario del programa ha accedido.

Los programas PROGID son empleados para controlar el acceso a las operaciones del sistema, por ejemplo, si los operadores de sistema necesitan respaldar archivos a los que no tienen acceso y si no tienen el súper ID, un programa PROGID proporciona una solución conveniente y segura. Los programas PROGID también son usados para controlar el acceso a una base de datos. Su política de seguridad debe proporcionar directrices para el uso y monitoreo del uso de los programas PROGID para prevenir:

- Accesos no autorizados a datos restringidos.
- Transmisión de los privilegios de un programa PROGID a cualquier proceso.
- Que los programas innecesarios del sistema sean habilitados como programas PROGID. Esto puede proporcionar habilidades excesivas y fácilmente derrocadas.

2.2.6 SAFEGUARD

Este software agrega funcionalidad de seguridad a los recursos propios del Kernel: Incrementa control sobre usuarios y password, permite al administrador de seguridad proteger procesos y dispositivos individuales e imponer límites a la capacidad el Super ID y provee recursos para auditar los intentos de acceso al sistema.

SAFEGUARD es un software de control de acceso que proporciona ampliaciones de seguridad significativas al sistema operativo GUARDIAN 90. Las características claves de SAFEGUARD son:

- **Identificación.-** SAFEGUARD añade mecanismos de protección tales como contraseña y datos de expiración del ID de usuario y suspensión temporal y restauración del acceso del usuario.
- **Auditoría.-** SAFEGUARD puede monitorear la actividad de usuarios específicos o monitorear toda la actividad del sistema completamente, registrando excepciones y eventos significativos para su revisión.
- **Autorización.-** Con SAFEGUARD, los administradores de la seguridad pueden ejercer control sobre todos los recursos del sistema creando listas de control de acceso para los recursos compartidos, tales como archivos de disco, volúmenes y sub-volúmenes de disco, dispositivos, líneas de comunicación y aplicaciones.

Manejo de password en Safeguard

El Control de Identificación se incrementa en gran medida utilizando Safeguard. Ahora, los administradores de seguridad pueden:

- Forzar cambio de password en determinado tiempo.
- Tener un historial de password y prevenir la reutilización de los mismos.

- Imponer límites de tiempo.
- Suspender usuarios o congelarlos después de cierto número de intentos

Administración de accesos en Safeguard

El administrador de seguridad debe construir una lista de control de accesos (**ACL**) para todos los objetos a controlar: volúmenes, sub-volúmenes, archivos, procesos ó dispositivos. (Ver *figura 2.2.6.1)*

El ACL indica qué usuarios pueden acceder qué objetos y en qué forma. Con ACL puede restringir el acceso a un sólo usuario de todo el grupo, utilizando la instrucción DENY. Si algún objeto no se incluye en la lista (ACL), no se tendrá ningún control sobre él.

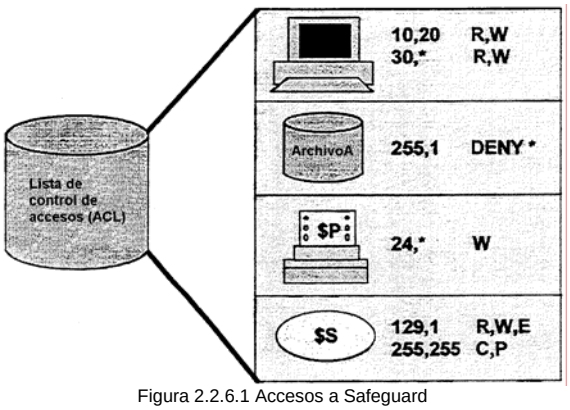


Figura 2.2.6.1 Accesos a Safeguard

GUARDIAN 90 sólo protege el acceso a los archivos de disco. SAFEGUARD protege el acceso a todas las demás entidades del sistema referidas a SAFEGUARD como “objetos”.
Los objetos de SAFEGUARD incluyen archivos de disco, sub-volumenes, dispositivos (como controladores de disco, controladores de cinta, impresoras y terminales) y procesos específicos. Utiliza Listas de Control de Acceso (ACL) para controlar el acceso a los objetos del sistema. A cada usuario o grupo se le pueden conceder autoridades específicas y/o denegar autoridades específicas. (“denegar” tiene prioridad sobre “conceder”). Los permisos especificados dependen del tipo de objeto. Por ejemplo:

- Los permisos del archivo de disco son: Lectura, Escritura, Ejecución, Eliminación, Creación y Pertenencia.
- Los permisos de dispositivo son: Lectura, Escritura y Pertenencia.
- Los permisos de proceso son: Lectura, Escritura, Eliminación, Creación y Pertenencia.

Control de terminales con Safeguard

Safeguard es capaz de identificar al usuario y posteriormente ejecutar cierto programa para ese usuario. Además, safeguard incluye una característica de “Auditing” para objetos, accesos locales, accesos remotos, intentos fallidos, etc. Todos estas características se almacenan en un audit file o archivo de auditoria.

2.2.7 CA-Unicenter

Es un software de seguridad que protege a Tandem controlando accesos, datos y recursos. Los alcances de este software son:

- Controla acceso a los recursos.
- Forzar estándares para nombre de archivo
- Previene el borrado accidental de archivos.
- Limita accesos a archivos por fecha y hora.

CA-Unicenter almacena la información de seguridad en una BD relacional común que es fácilmente accesible para los administradores del sistema

2.2.8 Encriptación por medio de Atalla

La encriptación permite a los usuarios de cierto sistema intercambiar información, previniendo que otros la lean. Tal procedimiento enmascara la información utilizando algoritmos matemáticos que la hacen difícil y costosa de desenmascarar. De esta manera, la información se enmascara y desenmascara utilizando una "llave" que sólo conocen el origen y el destino. (Ver *figura 2.2.8.1)*

Tandem tiene un dispositivo (hardware) que se conecta en ambos equipos para enmascarar y desenmascarar toda la información que viaja en una red (entre nodos), asegurando de ésta manera la información en un grado aún mayor. El encriptamiento por hardware es mucho más seguro que por software. El dispositivo en sí, está protegido contra violaciones. [9]

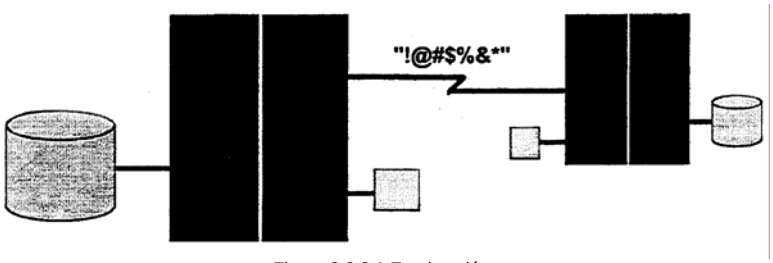


Figura 2.2.8.1 Encriptación

2.3 Administración del sistema

2.3.1 Nueva instalación del software del sistema

Mejoras del sistema operativo
Tandem publica nuevas revisiones del software para mejorar el sistema operativo. La publicación o liberación de un software contiene productos del nuevo software y versiones revisadas de los productos del software existente. Para llevar un sistema hasta el nivel actual de la revisión del programa, el usuario necesita instalar la nueva revisión del software y crear una imagen nueva del SO (sistema operativo).

Install (Instalar)

Install es un proceso de Tandem usado para crear una versión de sitio específico para el sistema operativo. A las peticiones del usuario, Tandem proporciona una Cinta de Actualización de Sitio (SUT- Site Update Tape) que contiene software en forma de procedimientos compilados y archivos informativos. Durante el INSTALL esta información es cargada a $SYSTEM y otros discos, entonces está montado con parámetros proporcionados por el usuario (por un proceso llamado SYSGEN), dentro del sistema operativo, el cual es escrito a una cinta de imagen del sistema (SIT [16]) que más tarde es copiada al disco.

INSTALL también es usada para colocar el programa de tira en el disco y actualizar el software de la interfase de mantenimiento remoto (RMI[17]), si es necesario. Después de que el sistema operativo y la tira están en el disco, el sistema puede estar "cargado en frío" con el nuevo sistema operativo. Esta fase "carga en frío", ejecutada después de que se completa el INSTALL, requiere paralizar toda la actividad del sistema.

SYSGEN

SYSGEN es una fase de INSTALL usada para:

- Agregar nuevo hardware.
- Implementar una nueva versión del sistema operativo
- Actualizar algún módulo(s) del sistema operativo.

Para construir procesos del sistema y crear procedimientos de código del sistema y tablas de datos del sistema las cuales son escritas a la SIT, SYSGEN usa lo siguiente:

- Un archivo de configuración proporcionado por el usuario.
- Un archivo de datos que contiene información acerca de Tandem y del software de Sitio-específico.
- Procedimientos proporcionados por Tandem que se cargaron durante el proceso de INSTALL.

2.3.2 Interfase de mantenimiento remoto

La Interfase de Mantenimiento Remoto (RMI) es el centro de control del subsistema de mantenimiento y diagnóstico (MDS[18]). El MDS es una red distribuida de componentes basados en microprocesadores que continuamente recopilan información de error y estatus de cada procesador y fuente de poder, así como datos de temperatura del procesador y los gabinetes de E/S-únicamente.

Es monitoreada por el Sistema de Mantenimiento y Diagnóstico de Tandem (TMDS[19]), una herramienta de diagnóstico que constantemente monitorea los componentes de hardware de un Sistema Tandem y diagnostica los problemas del sistema antes de que éstos afecten el rendimiento del mismo.

[16] SIT: Sistem Image Tape
[17] RMI: Remote Maintenance Interface
[18] MDS: Maintenance and Diagnostic Subsystem
[19] TMDS: Tandem Maintenance and Diagnostic Subsystem

La RMI proporciona monitoreo, diagnóstico y capacidades de control, ya sea remota o localmente, de todos los dispositivos del MDS. Proporciona acceso operativo y de mantenimiento tanto a la interfase de comando (TACL) como a las funciones de diagnóstico en /fuera de línea. Tiene suficiente inteligencia remanente para operar el sistema cuando se cae, incluyendo la habilidad para enfriar la carga desde el panel de control del sistema. Puede comunicarse a través de la Terminal de Operador RMI, con un alcance limitado, a través del panel de control del sistema y, remotamente, a través de un módem.

2.3.3 Utilidades de recuperación y respaldo

El proceso de respaldo, copia los archivos seleccionados del disco a una cinta. El proceso de recuperación copia los archivos seleccionados de una cinta a disco. El respaldo y la recuperación proporcionan lo siguiente:

- **Protección de la información almacenada en disco**.- Si los archivos originales son dañados o destruidos, pueden ser recuperados de las copias en las cintas.

- **Ahorro de espacio en disco.-** Puede respaldar y recuperar un archivo para quitar la fragmentación del disco; también puede respaldar archivos ocasionalmente utilizados del disco a una cinta y borrarlos del disco. Cuando los archivos sean necesitados, puede reintegrarlos al disco.

- **Mover archivos desde un sistema a otro**.- Para sistemas no conectados en red, las utilidades de respaldo y recuperación proporcionan una forma conveniente de respaldar archivos en un sistema y reintegrarlos en otro.

2.3.4 Análisis de espacio en disco y compresión

El DSAP[20] analiza si se utiliza el espacio en disco en un volumen especificado. Describe el espacio libre, el espacio utilizado y mucha más información acerca de los archivos almacenados en disco. Produce reportes resumidos acerca del uso del espacio del volumen del disco. El reporte resumido tiene tres secciones: una descripción física del disco, un resumen de uso del espacio en disco y un reporte de condición excepcional.

El DCOM[21] analiza la asignación de espacio actual en un disco, mueve regiones de un lugar a otro. Los beneficios obtenidos de correr el DCOM incluyen:

- Reducir el número de regiones de espacio libre, de tal manera que se reduce el esfuerzo requerido para asignar o liberar espacio.

- Combinar espacio libre dentro de regiones más grandes, de esta forma permite la asignación de archivos con regiones más grandes.

Tanto el DSAP como el DCOM son aplicaciones que pueden ejecutarse en línea mientras el disco está arriba y corriendo. Sin embargo, pueden no estar corriendo durante las horas pico de operación. Para asegurar la compatibilidad con interfaces de sistemas de bajo nivel, así como con utilidades se requiere que sus versiones correspondan con la versión actual del sistema operativo.

[20] DSAP: Disk Space Análisis Program
[21] DCOM: Disk Compression Program

El comando PUP LISTFREE enlista el número de páginas libres, los tamaños de bloques contiguos de páginas libres y el número de archivos actualmente en un volumen especificado.

2.3.5 Utilidades de comunicación

Programa Utilitario de Monitor de Red (NETMON – Network Monitor Utility Program)

El NETMON coloca en bitácora los cambios en el estatus de la red, muestra los cambios en el tráfico de la red, registra en bitácora y despliega los cambios en el estatus del procesador en un sistema remoto. Es la interfase del operador a la red EXPAND de Tandem.

Centro de control del subsistema (SCF-Subsystem Control Facility)

El SCF es usado para configurar, controlar y recolectar datos acerca de los subsistemas de comunicación. Es un intérprete de comandos que permite al operador administrar las líneas de comunicación, los dispositivos y subsistemas de comunicación de datos, tales como MULTILAN, EXPAND y soporte de dispositivos en general. Es la interfase del operador a un proceso intermedio llamado el Punto de Control del Subsistema (SCP-Subsystem Control Point).

El SCP lleva a cabo las peticiones hechas por el SCF para realizar operaciones en objetos particulares y envía mensajes de finalización al SCF cuando éste ha realizado las operaciones.

2.3.6 NETBATCH

Netbatch es un sistema de administración de tareas automatizado que planifica y envía tareas en bloque. El Netbatch programado es un proceso del servidor que permite:

- Especificar el tiempo en días en el que el trabajo debería correr, ceder las tareas y permite al planificador empezar las tareas automáticamente y enviar la salida al sitio correcto.
- Distribuir la ejecución de las tareas a través de los diferentes procesadores.
- Descubrir qué tan eficientemente corre cada proceso de la tarea.
- Reiniciar tareas automáticamente si un procesador falla.
- Levantar las dependencias de tareas para incrementar la eficiencia de procesamiento (ejemplo: si la tarea 3 depende en su resultado de las tareas 1 y 2 por sus datos de entrada, el planificador retrasará el comienzo de la tarea 3 hasta que se completen las tareas 1 y 2)

2.3.7 VIEWSYS Y PEEK

Utilidad VIEWSYS

La utilidad VIEWSYS permite al usuario monitorear el uso de los recursos del sistema. A través del uso no alternado de las teclas de función, puede mostrar la actividad en el procesador, la memoria, mensajes del sistema, el disco y el uso del espacio de datos del sistema.

Utilidad PEEK

El programa utilitario PEEK reporta información estadística acerca del uso de recursos en un procesador (a menos que se especifique, el PEEK le proporciona información sobre cualquier procesador que elija). La salida producida por el PEEK puede dirigirse a cualquier archivo estándar, tales como la terminal local, una impresora o un archivo existente en disco.

Utilidad MEASURE

MEASURE es una herramienta de medición y rendimiento del sistema de Tandem, recopila y examina estadísticas de desempeño necesarias para optimizar aplicaciones OLTP para sistemas NonStop. El proceso MEASCOM inicia el subsistema MEASURE.

MEASCTL es un proceso en cada procesador, el responsable de recopilar información de su procesador, dependiendo de la configuración generada por el usuario. Los datos generados por MEASCTL son almacenados en un archivo de datos. Estas entradas pueden ser vistas usando reportes o esquemas usando MEASCOM. MEASURE proveerá de muchos datos detallados en los siguientes objetos:

- Actividad del procesador.
- Desempeño de los procesos del usuario y del sistema.
- Actividad de E/S del disco.
- Acceso a archivos lógico y físico.
- Uso de la línea de comunicación (local y de red).
- Utilización de dispositivos generales y de Terminal.
- Desempeño del subsistema SQL.

Actividades especificadas del usuario dentro de un proceso.

2.3.8 Administración de Sistemas Distribuidos (DSM)

La administración de sistemas distribuidos (DSM) es un grupo de productos que permite al usuario administrar grandes redes de éste tipo de sistemas, consolidando elementos como procesadores distribuidos geográficamente, subsistemas de software, dispositivos periféricos, aplicaciones, software de comunicación de datos, complejos de líneas y dispositivos de terminal. Esta consolidación proporciona una vista general de control para una red distribuida completa y le permite mantener niveles altos de servicio para los usuarios de la red.

Las aplicaciones DSM automatizan las funciones de administración. Las siguientes aplicaciones son parte de la línea de productos DSM:

VIEWPOINT.- Es una aplicación del sistema de procesamiento de transacción PATHWAY de Tandem, que permite al operador acceder vía TACL al intérprete de comandos del subsistema o a través de una pantalla en modo bloque del PATHWAY acceder a los archivos del sistema y al subsistema MEASURE.

- **Sistema de Estadísticas de la Red (NSS).**- Es una herramienta de administración de la red a alto nivel que habilita a los operadores para poder recopilar, monitorear y reportar estadísticas de la red desde múltiples nodos en una red de Tandem a uno o más lugares en esa red.

- **Servicio de Nombres Distribuidos (DNS):** Es un subsistema del software que proporciona un método automatizado de rastreo permanente de los componentes de la red. Una base de datos DNS sirve como fuente única de información acerca de los nombres de todos los componentes de la red y el sistema y de sus relaciones.

- **Soluciones de Administración de Sistemas Distribuidos (DSMS)**: Es un paquete de programas que simplifica el monitoreo y control de los ambientes de Tandem. Puede darle una visión general del sistema completo desde una terminal y resume la información de administración en grupos de objetos.

- **DSM / Administrador de Problemas (DSM / PM):** Es una aplicación de software de alto nivel que tiene la función de establecer y administrar un ambiente de administración de problemas en un sistema autónomo o desde un punto único en una red multi-nodos. Proporciona un medio para que todos los problemas encontrados sean registrados, rastreados y resueltos de acuerdo a estándares predeterminados de precisión, puntualidad y consistencia.

- **Administrador de Red Programático (PNA):** Es un ambiente basado en reglas para operaciones automáticas. Utiliza estas reglas para analizar rápidamente mensajes de evento, tomar decisiones y llevar a cabo acciones. [10]

2.4 Trabajo en red y conectividad

La conectividad se refiere a las conexiones físicas o eléctricas de líneas y dispositivos. Sin embargo, la conectividad de red, típicamente, incluye los protocolos de transmisión de datos debido a los diferentes tipos de dispositivos disponibles. Los sistemas de Tandem soportan una amplia variedad de alternativas de conectividad, incluyendo líneas dedicadas, líneas conmutadas (dial up), enlace directo, redes de área local, redes conmutadas de paquetes y redes digitales. Los métodos de acceso y controladores de Tandem soportan todos los protocolos comúnmente usados y ofrecen un servicio para desarrollar protocolos a medida para aplicaciones únicas

2.4.1 Software EXPAND

Este software permite a los sistemas Tandem **interoperar** en redes grandes y distribuidas de hasta 255 nodos (servidores). Tal subsistema (llamado subsistema Expand), junto con el NonStop Kernel (SO GUARDIAN 90), utilizan una interfaz uniforme basada en mensajes para acceder los recursos locales y remotos. Así, tanto las aplicaciones como los dispositivos se comunican como si estuviesen en el sistema local. Expand elimina las complejidades de las comunicaciones remotas. (Ver *figura 2.4.1.1)*

Se encarga de hacer llegar los mensajes de un nodo a otro utilizando “best-path routing”. Esto se hace examinando la velocidad de las líneas de comunicación y el número de nodos por donde la

información debe acceder. La mejor ruta es la que tiene tiempo de tránsito más veloz. Cuando la información debe pasar por un nodo distinto al destino, ésta se transmite al siguiente, en una acción conocida como "pass-through routing". Esto elimina costos, debido a que no debe haber una conexión directa entre 2 nodos específicos.

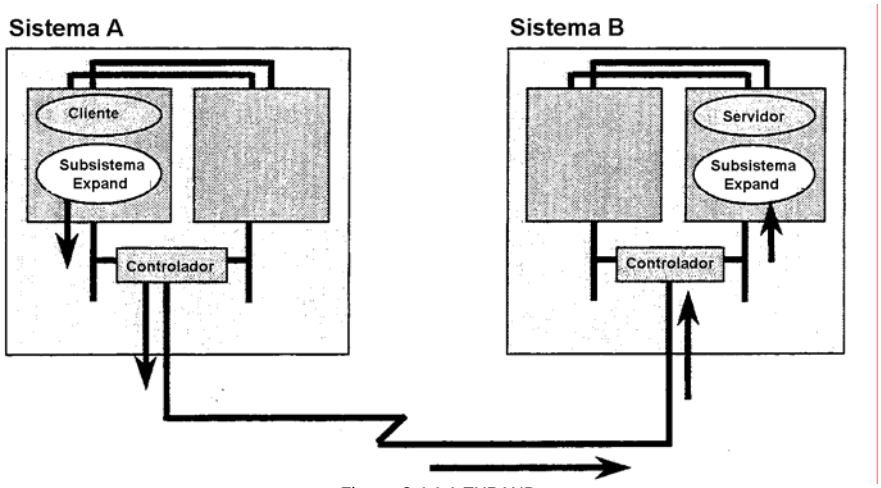


Figura 2.4.1.1 EXPAND

Ventajas de EXPAND

- **Transparente a los usuarios.-** Los usuarios en sistemas locales pueden acceder a otros recursos en la red sin conocer acerca de las conexiones de red fundamentales. Para esto, el usuario debe estar registrado como tal en ambos sistemas o nodos, en el mismo grupo y con el mismo número de usuario y grupo.

- **Operación tolerante a fallas.-** EXPAND hace uso de tantos enlaces como haya disponibles. Si una ruta falla, EXPAND automáticamente redirige los datos a la siguiente mejor ruta disponible. Cuando la ruta original está disponible otra vez, EXPAND regresa a ella automáticamente.

- **Enrutamiento automático de mensajes.-** La capacidad de enrutamiento automático de mensajes de EXPAND asegura que un mensaje enviado desde cualquier sistema en la red llegará al destino deseado en cuanto haya al menos una ruta disponible.

- **Enrutamiento de la mejor ruta.-** EXPAND transmite datos sobre la mejor ruta (la más rápida). Los mensajes pueden dirigirse a través de muchos sistemas intermediarios, si es necesario para llegar a cualquier sistema destinatario.

- **Seguridad.-** El acceso a procesos, dispositivos o archivos en un sistema remoto EXPAND puede restringirse a usuarios de ese sistema.

- **Administración.-** EXPAND soporta Administración de Sistemas Distribuidos (DSM). La DSM consolida la administración de todos los componentes de hardware y software en la red EXPAND, incluyendo recursos individuales del sistema.

2.4.2 Subsistema TorusNet

Se utiliza para expandir las capacidades de procesamiento en Paralelo propios de Tandem. Consiste de una serie de ligas de fibra óptica dispuestas de manera horizontal y vertical, creando una LAN de alta velocidad de procesadores Tandem. En TorusNet, una sección está constituida de 2 a 4 procesadores. Un nodo contiene 4 procesadores y un dominio 56 secciones (256 procesadores). Dicha arquitectura soporta hasta 18 dominios, lo cual permite un promedio de 4000 procesadores trabajando juntos como un solo servidor con procesos en paralelo. (Ver *figura 2.4.2.1)*

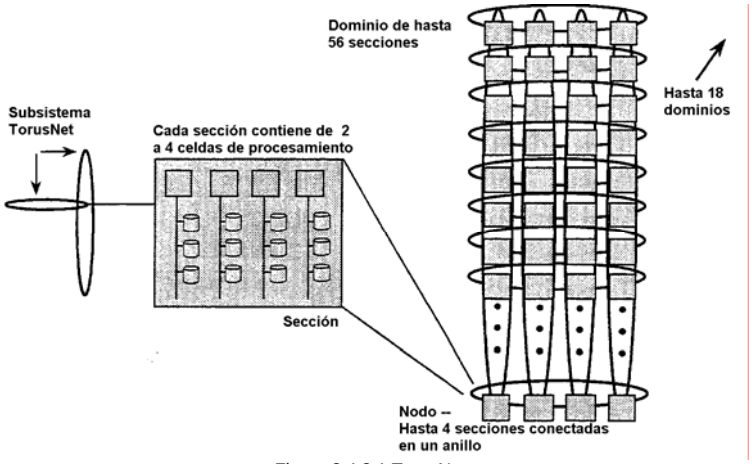


Figura 2.4.2.1 TorusNet

2.4.3 Controladores de comunicaciones

Los controladores de comunicaciones proporcionan medios para conectar redes de área amplia (WANs – Wide Area Networks), redes de área local (LANs – Local Area Networks) y dispositivos a sistemas de Tandem.

Un controlador de comunicaciones es, básicamente, un microprocesador con inteligencia para transferencia de datos entre el canal de E/S del sistema y el enlace de la red. La inteligencia es descargada del sistema computacional. Algunos controladores realizan más funciones que otros, incluyendo conversión de protocolo, manejo de errores y, posiblemente, otras tareas de red. Los controladores de comunicaciones de Tandem son diseñados para protocolos y configuraciones específicos para proporcionar varias velocidades, capacidades y niveles de tolerancia de fallas.

Soporte Asíncrono

Los protocolos asíncronos son usados comúnmente para conectar un sistema central con dispositivos que tienen espacio limitado de buffer y memoria. La mayoría de estos protocolos soportan asíncrono, transmisión de datos half-dúplex a través de enlaces punto a punto únicamente.

El objetivo general del soporte de terminal de Tandem es permitir la selección del equipo más adecuado para el trabajo o proteger una inversión previa en el equipo de la terminal. Si el cliente tiene una terminal con un único protocolo está disponible un método de acceso programático para definir soporte para éste.

Soporte Síncrono

En transmisión síncrona, los datos son transmitidos en bloques. El inicio y fin del bloque se determinan por mecanismos de tiempo sincronizados, tales como módems en los dispositivos de envío y recepción. Los protocolos de byte-síncrono típicamente usan transmisión síncrona, aunque algunos protocolos orientados a byte; también, pueden trabajar con transmisión asíncrona. La mayoría de los protocolos byte-síncronos son half-dúplex.

Los protocolos de byte-síncrono son los más eficientes y flexibles, normalmente, se usan para transmitir datos entre sistemas o entre un controlador del cluster (sectores de disco) y un sistema central. Al igual que con el soporte asíncrono, el soporte de Tandem permite una amplia latitud en la selección del mejor equipo para un trabajo en particular, en lugar de restringir las terminales que pueden ser usadas. Nuevamente, una inversión previa en equipo podrá protegerse, en lugar de requerir adquisiciones de terminales nuevas. [11]

3 INTERFACE SQL NONSTOP

En el siguiente capítulo se mostrará la manera de explotar el intérprete de comandos de NonStop SQL (SQLCI) empleando los diversos tipos de comandos, configurando un ambiente propio de trabajo, manipulación de comando recientemente empleados (ejemplo: reutilizo de los mismo o conocimiento o conocimiento de comandos empleados anteriormente) acceso, modificación, inserción, eliminación de información en la base de datos.

Lo anterior con el fin poder manipular la base de datos contenida en el sistema.

3.1 Establecer una sesión SQLCI

3.1.1 Introducción

Iniciando SQLCI

Para iniciar una sesión SQLCI se realiza el siguiente procedimiento:

- Iniciar sesión a través del emulador de Tandem con el servicio a utilizar.
- Lo primero que se visualizará sobre el monitor será el prompt: TACL>[22].
- Firmarse con el ID de usuario mediante el comando LOGON. Ejemplo: TACL> LOGON <identificador del usuario> presionar la tecla ENTER y teclear el password correspondiente. Se visualizará el prompt con el número de comando correspondiente, ejemplo 1>, 2>, etc.
- Teclear SQLCI y presionar ENTER. El monitor despliega el prompt del tipo: >>, el cual indica que se encuentra dentro de SQLCI y se encuentra listo para la ejecución de comandos.

Ejemplo:
SERVICIO
TACL1> LOGON GRUPO.USER
Password:
1SQLCI
>>

Finalizar SQLCI

Para finalizar la sesión de SQLCI, teclear sobre el prompt el comando EXIT, E o utilizar las teclas CONTROL + Y.

[22] TACL: Tandem Advanced Command Language

3.1.2 Ejecución de comandos

Aquí se describen algunas formas para el uso de los comandos de SQLCI.

- El prompt: >>, indica que el sistema esta listo para el próximo comando.
- Todos los comandos deben de terminar con punto y coma (;), excepto los comandos EXIT y FC donde es opcional.
- Un sólo comando puede abarcar más de una línea de largo. Presionando la tecla ENTER se continúa con el comando en la siguiente línea.
- Si el punto y coma no es utilizado o el comando es más largo que una línea, el prompt aparece como: +>, indicando que SQLCI está listo para aceptar más información. Teclear el punto y coma cuando se ha terminado de teclear el comando.

3.1.3 Comando SQLCI

Existen muchos tipos de comandos con SQLCI. Diferentes tipos desarrollan diferentes funciones, como recuperar información de las tablas, vistas especificas de las tablas, descripción de las tablas y muchas otras selecciones y/o cambios a la información de la base de datos.

Dentro de SQLCI existen cuatro tipos generales de comandos:

Comandos de lenguaje NonStop SQL

- *Comandos DDL*(Lenguaje de definición de datos).- Son usados para crear o modificar la definición de catálogos de una tabla, columna, índice, vista o particiones. Algunos ejemplos de estos tipos de comandos son: ALTER, DROP, CREATE y COMMENT.
- *Comandos DML* (Lenguaje de manipulación de datos).- Son usados para acceder al contenido de tablas dentro de una base de datos. Por ejemplo: SELECT, UPDATE, INSERT Y DELETE.
- *Comandos DCL* (Lenguaje de control de datos).- Son usados para controlar procesos de recursos, así como bloquear y cursores. Como ejemplos de estos comandos DCL se incluyen: LOCK o UNLOCKTABLE, FREE, RESOURCES y CONTROL TABLE.
- *Comandos TMF* (Habilidad de monitoreo de transacciones).- Controla la habilidad de monitoreo de transacciones. Habilidad que ayuda a mantener la consistencia de la base de datos y protege la base de datos contra daños del sistema. Algunos ejemplos son: BEGIN, WORK COMMIT, WORK y ROLLBACK WORK.

Comandos de reporte escrito

Permiten dar formato y crear reportes. Se puede establecer opciones que controlen el estilo default de un reporte (así como el formato de fecha y hora), o establecer opciones que controlen los márgenes de establecidos, espacio entre líneas y otros rasgos que afecten la apariencia del reporte, ya sea proyectado en la pantalla o en forma impresa.

Comandos de utilidad

Los comandos de utilidad desarrollan operaciones en objetos SQL y archivos ENSCRIBE, como copiar, cargar datos, duplicar archivos y desplegar información acerca de objetos y características de archivos. Ellos proveen herramientas las cuales ejecutan las siguientes funciones:

- Desplegar los datos de descripción de tablas y vistas (utilidad INVOKE).
- Desplegar las características físicas de tablas, índices, vistas y archivos ENSCRIBE (utilidad FILEINFO).
- Verifica la integridad de definición de objetos (utilidad VERIFY).
- Convertir definición de archivos ENSCRIBE a definición de tablas NonStop SQL (utilidad CONVERT).
- Carga de datos (utilidad LOAD)

Comandos generales

Los comandos generales de SQLCI controlan el funcionamiento de SQLCI. Estos comandos ejecutan las siguientes funciones:

- Establecen y despliegan el ambiente SQLCI, incluyendo el sistema, volumen y sub-volumen establecidos, volumen del catálogo establecido y archivos actuales de entrada y salida.
- Mostrar, establecer y reiniciar opciones de una sesión, tal como la tecla BREAK funciona y el despliegue de las advertencias.
- Mostrar, establecer y reiniciar reportes y opciones de parámetros.
- Interfase con los archivos para propósito de edición (comandos BREAK, EDIT y TEDIT) o uso de otras utilidades Tandem como son: PERUSE, FUP y PUP dentro de SQLCI.

Comandos de ayuda

Se usa el comando HELP para obtener ayuda con algún comando con el que se tenga problemas. Si se requiere obtener información global se utiliza HELP ALL. Este comando despliega todos los comandos SQL previamente planteados. El uso del comando HELP DETAIL proporcionará una visión general de cómo usar el comando HELP. Esta información también incluye la explicación de las opciones SYNTAX DETAIL y EXAMPLE:

- SYNTAX.- Despliega la sintaxis de un comando específico o elemento de lenguaje.
- DETAIL.- Despliega la sintaxis y una descripción del tema con ejemplos.
- EXAMPLE.- Despliega ejemplos de cómo usar el comando o elemento de lenguaje. Si se requiere de información acerca de un comando especifico (Por ejemplo UPDATE) se teclea HELP UPDATE.

```
>> HELP UPDATE;
        UPDATE {<Nombre-Tabla>}
               {<Nombre-Vista>}
        SET <lista-establecida> [WHERE <condición de búsqueda>]
        [ [FOR] {STABLE} ACESS ];
              {REPEATABLE}   ];
        <lista-establecida> is:
                <nombre-columna>=<expresión>, [, <nombre-columna>=<expresión>]
>>
```

3.2 Establecer un ambiente

El establecer correctamente un ambiente SQLCI, permitirá acceder a la base de datos. También se podrá grabar la información recolectada, la cual podrá ser editada e impresa como sea necesario. Los comandos que establecen el ambiente incluyen:

- Comando VOLUME
- Comando LOG
- Comando CATALOG
- Comando SYSTEM
- Comando OUT
- Comando OUT_REPORT

Todos estos comandos marcan el lugar donde diferentes tipos de información son almacenadas y son establecidas como predeterminadas. Dos ejemplos de esta información predeterminada son el volumen y subvolumen. Si no se especifica la ubicación de un archivo, SQLCI asume que el archivo es guardado en el volumen y subvolumen predeterminados.

El nombre del volumen ($VOLUME) debe ser precedido por el signo de pesos ($). Este nombre puede contener de 1 a 6 letras o dígitos, esto para permitir su acceso en la red. El primer carácter debe ser una letra.

El nombre del subvolumen (SUBVOL) puede contener de 1 a 8 letras o dígitos. El primer carácter del nombre de un subvolumen debe ser una letra.

El nombre de objetos (OBJECT) se refiriere a objetos de NonStop SQL. Estos nombres pueden contener de 1 a 8 letras o dígitos. El primer carácter del nombre de un objeto debe ser una letra. Los nombres de los objetos pueden identificar una tabla, una vista, índice, partición o programa.

Los nombres de los archivos (FILES) pueden contener de 1 a 8 letras o dígitos. El primer carácter debe ser una letra.

Comando VOLUME

Para hacer la operación de SQLCI más fácil, se pueden establecer o cambiar valores que controlan el ambiente. Definir un volumen y subvolumen hace la entrada de comandos más fácil.

Un volumen en una base de datos marca la ubicación de tablas, archivos de salida. Ejemplo:

```
1> SQLCI
>>
>> VOLUME $DATA1.SALES ;
>> SELECT * FROM PARTS;
```

En el ejemplo, como primer paso se inicia una sesión de SQLCI y se establece el volumen $DATA1.SALES como el volumen y subvolumen predeterminados respectivamente. Con el comando SELECT * FROM PARTS se accede a la tabla PARTS ubicada en el volumen DATA1 y en el subvolumen SALES.

Comando CATALOG

Es necesario especificar un catálogo cuando se está trabajando con los comandos DDL, los cuales permiten crear tablas o vistas de tablas. Cuando se trabaja con comandos DDL, el programa necesita el nombre de un catálogo para registrar los cambios a la tabla correspondiente. Si el catálogo no está definido, el sistema busca un catálogo predeterminado en la ubicación especificada por el volumen actual.

Comando SYSTEM

El comando SYSTEM sólo es utilizado cuando se accede a un sistema diferente al actual. Para cambiar el predeterminado a otro sistema diferente al que se está trabajando se teclea el siguiente comando: SYSTEM *nombre del nodo*

Para regresar al sistema predeterminado se teclea: SYSTEM

Comando LOG

El comando LOG comienza a grabar una historia de cualquier comando que se introduzca durante una sesión de trabajo. Ésta incluye cualquier información que se despliegue en pantalla. El comando LOG es iniciado tecleando: LOG *nombre-del-archivo* y es finalizado al teclear LOG;
El archivo LOG contiene información en un formato modificable de Tandem, incluyendo las correcciones que se realicen con el comando FC. Para poder acceder y/o modificar el contenido del archivo se realiza a través de los comandos TEDIT o EDIT.

Comando OUT

El comando OUT dirige la salida de los comandos SQL a un archivo o algún otro destino, el cual se especifica anteriormente. Por default, el archivo OUT es la terminal actual. Si se está trabajando en SQLCI y se requiere un reporte del comando y que el resultado de éste se imprima por medio de la impresora de la Terminal se debe teclear:

```
>> OUT $S.FASTPRT;
>> SELECT * FROM EMPLOYEE;

Cuando el reporte se haya terminado de imprimir, teclear en el prompt de SQLCI lo siguiente:
>> OUT;
```

Comando OUT_REPORT

El comando OUT_REPORT dirige sólo la salida formateada de un comando SELECT a un archivo reporte en específico. El reporte solicitado puede incluir información de una tabla o vistas de estas. Éste no incluirá el comando actual o errores que puedan ocurrir.

Cuando se usa el comando OUT_REPORT, se debe incluir el nombre del archivo, el cual es el destino de la salida, éste se especifica de la siguiente manera: >> OUT_REPORT *nombre-archivo*;

En el siguiente ejemplo, el reporte de salida es dirigido al archivo de disco llamado $DRAFT, el cual está ubicado en el volumen actual: >> OUT_REPORT DRAFT CLEAR;

Nota: cuando se incluye la palabra CLEAR en el comando, el archivo es limpiado antes de que el reporte sea escrito. Si no se incluye la palabra CLEAR, el nuevo reporte se anexará a los datos incluidos en el archivo existente. Para cerrar el reporte se efectúa de la siguiente manera: >> OUT_REPORT;

Cuando se solicita un archivo OUT_REPORT, la información de salida, de un comando SELECT específico, no se enviará al archivo OUT, sin embargo si se enviará al archivo OUT_REPORT.

Ubicación de archivos

Para localizar archivos en la base de datos se podrá usar cualquiera de los comandos: FILES o FILESNAMES.

Comando FILES

Se puede usar el comando FILES para desplegar los nombres de los archivos contenidos en uno o más volúmenes y subvolúmenes. Para el uso del comando FILES debe contar con la siguiente estructura: >> FILES [plantilla de subvolumenes];

Plantilla de subvolúmenes es una lista de una o más especificaciones de subvolúmenes en disco, separadas por comas o espacios. Cada especificación de subvolumen debe ser formateada de la siguiente manera: *[\sistema.][$volumen.] subvolumen*

Comando FILENAMES

Para listar los archivos en uno o más subvolúmenes dentro de un volumen, se usará el comando FILENAMES. El comando FILENAMES puede ser usado para desplegar los nombres de todos los archivos que satisfacen la plantilla de nombres de archivos. La estructura correcta es: >> FILESNAMES [plantilla de nombres de archivo];

La *plantilla de nombres de archivos* es una lista de una o más especificaciones de archivos en disco. Ejemplo: \NY.$WH2.SALES.EMPLOYER.

Nótese que el comando FILENAMES difiere del comando FILES en que FILES lista todos los archivos bajo un volumen.subvolumen especificado, mientras FILENAMES lista el/los archivo/archivos especificados. Por ejemplo. Cuando se especifica FILES SALES, se obtendrá un listado de todos los archivos contenidos en el subvolumen SALES; mientras, con el comando FILENAMES SALES, únicamente se obtendrá el archivo llamado SALES en el volumen y subvolumen predeterminados.

Caracteres comodines

El resultado de los comandos FILENAMES o FILES puede ser manipulado usando caracteres comodines. Los siguientes caracteres pueden ser usados con cualquiera de los comandos FILES o FILENAMES:

- ***** Un asterisco puede compararse con uno o más caracteres de cualquier tipo.

- **?** Un signo de interrogación se compara con un solo carácter.

Ejemplo > CUST* puede compararse con CUSTLIST y CUSTOMER; sin embargo, mientras PART? se compara con PARTS, no lo hará con PARTNS porque PARTNS tiene más de un carácter siguiendo a la parte “PART”.

3.2.1 Manipulación de comandos

Comando HISTORY

Se usa el comando HISTORY cuando se requiere desplegar los últimos 10 comandos realizados durante la actual sesión de SQLCI. Si no se ha realizado ese número de comandos, SQLCI lista los comandos realizados hasta ese momento.

```
Ejemplo:
>>HISTORY;
1>SELECT * FROM SALES;
2>VOLUME $DATA.SALES;
3>HISTORY;
>>
```

También, se puede usar el comando HISTORY para mostrar un número específico de líneas previas. Ejemplo: HISTORY 4:

Sólo pueden ser mostrados un número máximo de 25 comandos. Así, el comando HISTORY 99 mostrará sólo los últimos 25 comandos y no los 99.

Comando !

El comando signo de exclamación (!) es usado para volver a ejecutar un comando especifico, e inmediatamente re-ejecutarlo. Cuando se usa este comando se puede escoger cualquier comando que se quiera re-ejecutar especificándolo de la siguiente manera:

- El número del comando que se muestra en la pantalla cuando se usa el comando HISTORY. Por ejemplo, para re-ejecutar inmediatamente el comando en la línea 2, usar **!2;**

- La ubicación del comando que se quiera ejecutar (esto es, contando hacia atrás cuantos comandos previos a el actual). Esto es hecho usando el signo de menos y el número (-#) para indicar cuántos comandos contar hacia atrás del comando actual. Para ejecutar el cuarto de los últimos comandos, escribir: **! -4;**

- El último comando de algún escrito particular (“SELECT por ejemplo). Para ejecutar el último comando SELECT se escribe: **!SELECT;**

- El último comando ejecutado. Para volver a re-ejecutarlo escribir únicamente: **!;**

Alternativamente, se puede escribir una porción de texto de una cadena del comando que se quiera ejecutar. Por ejemplo, si se escribe: **!HI**, el último comando HISTORY será ejecutado. El comando **!S** ejecutará el último comando que comience con S, el cual es el comando SELECT.

Comando FC

Se usa el comando FC cuando se ha cometido un error al escribir un comando. Permite recuperar, editar y re-ejecutar cualquier línea de comando que se haya escrito previamente. A continuación se mencionan cuatro formas de volver a llamar un comando usando FC:

- Escribir **FC** para el último comando que se escribió.
- Dar el número de línea especifico del HISTORY (Esto es: **FC 2**).
- Usar el signo de menos para indicar cuántas líneas atrás está el comando partiendo de la línea actual (Ejemplo: FC -4).
- Escribiendo una cadena de una texto que identifique el comando (Ejemplo: FC SELECT o FC SEL o FC S).

El comando indicado se desplegará por línea a la vez. Por cada línea aparecerá una segunda línea en donde se corregirá o modificará el comando desplegado, esto se realiza moviendo el cursor bajo el carácter a cambiar, realizar el cambio y continuar con la siguiente línea hasta ejecutar el comando.

Para BORRAR una o más letras del comando:

- Escribir **D** bajo la letra o letras que se desea borrar.
- Presionar la tecla enter para revisar los cambios realizados.
- Presionar la tecla enter para ejecutar el comando.

Para INSERTAR una letra faltante:

- Escribir **I** bajo la letra a partir de la cual serán insertados el/los caracteres siguientes a la I.
- Presionar la tecla enter para visualizar los cambios.
- Presionar la tecla enter para ejecutar nuevamente el comando.

Para REEMPLAZAR una letra:

- Escribir una **R** bajo la letra y entonces escribir la letra con la cual se reemplazará.
- Presionar la tecla enter para visualizar los cambios.
- Presionar la tecla enter para ejecutar el comando.

```
Ejemplo:
>>VOLUME $DATA.SALES;
>>SELECT * FRM PARTS;
>>FC
>>SELECT * FRM PARTS;
..                IO
>>SELECT * FROM PARTS;
..
>>(En esta línea el comando muestra la información solicitada)
```

COMANDO SAVE

Se puede usar el comando SAVE para llamar instantáneamente cualquier comando que se ha usado y que se puede usar después en la sesión. Para los comandos usados frecuentemente, éste puede ser un método más rápido que usar el comando HISTORY. El comando HISTORY solamente guarda los últimos 25 comandos de la sesión actual SQLCI.

Se necesitará asignar un nombre de archivo para cualquier comando que se quiera guardar, así éste se podrá recuperar fácilmente más tarde. Por ejemplo, si se usa el comando SAVE para guardar *SELECT * FROM PARTS,* se leerá:

SAVE COMMAND 2 TO *nombre-de-archivo* ó SAVE COMMAND -4 TO *nombre-de-archivo* ó SAVE COMMAND ¨SELECT¨ TO *nombre-de-archivo*

Se puede reemplazar el contenido de un archivo Obey existente colocando la palabra CLEAR después del nombre del archivo.

Comando OBEY

Ahora que se ha guardado un comando, este se puede llamar fácilmente, cuantas veces se desee, tecleando:

OBEY *nombre-de-archivo;*

Nota: Un archivo OBEY puede contener varios comandos.

3.3 Recuperar datos usando SQLCI

3.3.1 Acceso a datos

Permitir el acceso a información sobre tablas SQL permite determinar como las tablas y archivos son organizados y nombrados. Esta ventaja es usada especialmente para recuperar datos de una tabla SQL cuando no se tiene la certeza de cómo las tablas están estructuradas. Estos comandos, en parte determinan a qué tablas acceder para recuperar datos. Hay que recordar que en NonStop SQL, todas las tablas son guardadas como archivos. Para encontrar información en un archivo, se accederá el nombre de la tabla usando los comandos que a continuación se mencionarán.

Listando características de archivos

Con el comando FILEINFO se puede obtener información y desplegar las características físicas de tablas, índices, vistas y archivos ENSCRIBE. Este comando es una utilidad comparable con el

comando FUP DUP. Si se quiere obtener información de la tabla CUSTOMER por ejemplo, se tecleará el siguiente comando FILEINFO:

```
>>FILEINFO SALES.CUSTOMER;
       CODE    EOF       LAST MODIF    OWNER RWEP  TYPE    REC   BLOCK
$DATA1.SALES
  CUSTOMER     A    2288   8Mar04 11:57   8,100    AAAA      KTa        80      4096
```

Esta respuesta muestra información acerca de la tabla CUSTOMER. Nótese que esta información son datos descriptivos de un archivo y no de la información contenida en el archivo. FILEINFO permite el uso de varias opciones cuando se solicita información. La opción DETAIL del comando FILEINFO desplegará un bloque de información detallada para cada archivo.

```
                >>FILEINFO SALES.CUSTOMER, DETAIL
                $DATA1.SALES.CUSTOMER                          7 SEP 1988, 19:22
A               SQL BASE TABLE
 B              CATALOG $DATA1.SALES
C               TYPE K
                EXT ( 16 PAGES, 64 PAGES, MAXEXTENTS 160 )
                REC 80
                PACKED REC 60
                BLOCK 4096
   D            KEY ( COLUMN 0, OFFSET 0, LENGTH 2, ASC )
 E              INDEX (   1, $DATA1.SALES.XCUSTNAM,
                COLUMN 1, OFFSET 2, LENGTH 18, ASC )
                NOT UNIQUE )
                 AUDIT
                BUFFERED
                OWNER 88, 255
                SECURITY (RWEP): A000
                MODIF: 13 May 1988, 9:27
                CREATION DATE: 13 May 1988, 9:19
                REDEFINITION DATE: 20 May 1988, 16:25
                LAST OPEN: 7 Sep 1988, 14:01
                EOF 12288 (0.1% USED)
                EXTENTS ALLOCATED: 1
                INDEX LEVELS: 1
                >>
```

De la información detallada del archivo se marcan los puntos que a continuación se describen:

A. Éste indica si es una tabla, vista, índice o programa. En este caso, es una tabla bajo SQL. (Una tabla que contiene datos).

B. **CATALOG** identifica el catálogo en el cual el objeto es definido.

C. **TYPE** muestra como el archivo es reconocido. Esta organización puede ser Secuencia-Llave, secuencia-llave, relativa, no estructurada. En este caso, K indica que ésta es una tabla de secuencia-llave (Key-Sequenced)

D. La línea KEY muestra la siguiente información: COLUMN indica la posición de la columna llave en la fila (la primera columna es numerada 0); OFFSET indica la dirección cero-relativa de la columna llave; LENGTH indica la longitud en bytes de la columna llave; y ASC indica el orden ascendente.

E. La línea INDEX muestra la siguiente información acerca de los índices de la tabla: El primero indica el número de archivos llave alternativos; el segundo dato es el nombre del archivo llave alternativo; COLUMN, OFFSET, LENGTH y ASC son la misma descripción para lo descrito en la línea KEY y NOT UNIQUE indica que el valor de la llave puede ser el mismo en diferentes registros.

Describiendo columnas de tablas

El comando INVOKE produce una definición de una tabla o vista. El nombre de la columna y tipo de datos son listadas para la tabla o vista.

```
>>INVOKE PERSONAL.EMPLEADO;
    -- Definition of table \EXT25.$DATA1.PERSONAL.EMPLEADO
    -- Definition current at 14:00:02 – 03/31/88
        (
        , NUMEMP                NUMERIC( 4,0 ) UNSIGNED
        , NOMBRE                CHAR( 15 )
        , APELLID                       CHAR( 15 )
        , NUMDEP                NUMERIC( 4,0 ) UNSIGNED
        , CODTRAB               NUMERIC( 4,0 ) UNSIGNED
        , SALARIO               NUMERIC( 8,2 ) UNSIGNED
        )
>>
```

Tipo de datos

Tipo de dato	Denominación NonStop SQL	Descripción
Caracteres de longitud fija	CHAR (n) PIC X (n)	Cadena de carácter de longitud fija ASCII
Caracteres de longitud variable	VARCHAR (n)	Cadena de carácter de longitud variable ASCII
Número binario entero	NUMERIC (n) SMALLINT	n = 1 a 4 dígitos Signo Opcional
	NUMERIC (n) INTEGER	n = 5 a 9 dígitos, Signo requerido
	NUMERIC (n) Signed LARGEINT PIC 9(n) COMP	n = 10 a 10 dígitos. Signo requerido
Escalar (Punto decimal implícito)	NUMERIC (n,s) PIC 9 (n - s) V9(s) COMP	n = 1 a 18 dígitos: Signo opcional para 1 a 9 dígitos; Signo requerido para 10 a 18digitos
Número decimal entero	DECIMAL (n) PIC 9 (n)	N = 1 a 18 dígitos: Signo opcional para 1 a 9 dígitos; Signo requerido para 10 a 18 dígitos
Escalar (Punto decimal implícito)		

	DECIMAL (n,s) PIC 9 (n - s) V9(s)	n = 1 z 18 dígitos: Signo opcional para 1 a 9 dígitos: signo requerido para 10 a 18 dígitos

Nota: n = longitud, un entero positivo; s = escala de un número, un entero positivo, el cual representa el número de posiciones a la derecha del punto decimal implícito.

Tipos de dato numérico

NUMERIC, PIC 9, SMALLINT, INTEGER, LARGEINT Y DECIMAL indica que la columna es un dato numérico. Los contenidos son representados externamente como una combinación de 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.

Un dato numérico puede ser un entero o tener una fracción decimal. Un dato numérico, puede ser almacenado de dos diferentes formas: DISPLAY: es un conjunto de caracteres ASCII; NUMERIC o PIC(n) COMPUTATIONAL es un suplemento binario. La diferencia entre estas dos formas está en como los datos son almacenados internamente por la computadora. Binario toma menos espacio, lo que beneficia a columnas más grandes.

Tipos de dato carácter

CHARACTER, PIC X O VARCHAR indica que la columna es un dato alfanumérico. El contenido del dato puede tener cualquier combinación de caracteres ASCII, cuando una columna es definida como alfanumérica no se pueden ejecutar funciones aritméticas usando dicha columna, incluso si el dato almacenado está compuesto únicamente por dígitos (0-9).

Pueden almacenarse datos alfanuméricos en una tabla en alguna de estas formas: carácter fijo, (el tipo CHARACTER o PIC X), o carácter variable (tipo VARCHAR). La diferencia es como el dato es almacenado internamente, en los tipos de datos numéricos no hay diferencia en cómo usarlos.

La forma CHARACTER o PIC X reserva la cantidad indicada de espacio en columna para cada registro en la tabla, tanto todo el espacio sea usado o no. En contraste, la forma VARCHAR usa únicamente el almacenamiento interno que sea necesario

3.3.2 Recuperar datos

Una ventaja principal de usar base de datos relacional es que todo tipo de dato (incluso las respuestas de un query) se puede encontrar dentro de las tablas. Este tipo de tablas (en este caso, una respuesta de la computadora) es referenciado como una tabla resultado.

Comando SELECT

El propósito del comando SELECT es recuperar un conjunto de valores de uno o más tablas o vistas. Un comando sencillo SELECT puede regresar un o más registros de datos. Las diversas opciones en el comando permiten especificar como se quiere desplegar la información.

El comando SELECT tiene la forma general: **SELECT * FROM *nombre-de-la-tabla***. Ésta es la forma más simple del comando y proporciona todos los datos de una tabla. El asterisco (*) es el método taquigráfico para seleccionar todo de las columnas, como se muestra a continuación:

```
>> SELECT * FROM PERSONAL.EMPLEADO;
NUMEMP      NOMBRE      APELLIDO    NUMDEP      CODTRAB     SALARIO
---------   ---------   ---------   ---------   ---------   ---------
      1     ROGER       GREEN       9000        100         175500.00
     23     JERRY       HOWARD      1000        100         137000.10
. . . .
     29     JANE        RAYMOND     3000        100         136000.00
--- 57 row(s) selected
>>
```

La cláusula **FROM** siempre es requerida en el comando SELECT para especificar la tabla de la cual el dato es seleccionado, una cláusula es un segmento de un comando SQL que comienza con una palabra clave.

Seleccionar columnas específicas

Con el comando SELECT se puede encontrar la información específica que se necesita. La forma más simple del comando SELECT proporciona todas las columnas y filas en una tabla, pero puede producir mucho más información de la que se necesita.

El comando SELECT permite desplegar un grupo específico de columnas para proporcionar información adicional, así como los nombres de las columnas específicas que se quieren desplegar. La vista resultante es llamada una proyección. Para seleccionar columnas especificas de todas las filas (registros) de una tabla, se debe teclear el comando siguiente: SELECT columna1, columan2 FROM nombre-tabla

Nótese que los nombres de las columnas están separados por coma. Se puede listar el nombre de columnas en cualquier orden, pero éstas deben seguir la palabra clave SELECT, precedidos por la palabra clave FROM. Las columnas se despliegan en el orden que se especifica.

Información en secuencia ordenada

Los registros pueden ser desplegados en cualquier orden ya sea ascendente o descendente, simplemente usando la cláusula ORDER BY en el comando SELECT. Por ejemplo, el comando: SELECT * FROM tabla ORDER BY columna;

Despliega los datos contenidos en “columna” ordenados de manera ascendente (el orden ascendente es establecido). Si se quiere cambiar a orden descendente, simplemente se teclea DESC al final de la cláusula ORDER BY de la siguiente manera: SELECT * FROM tabla ORDER BY columna des;

3.3.3 Uso de condiciones para limitar la selección de datos

Usando Cláusulas

Las cláusulas ALL/DISTINCT, WHERE y la cláusula HAVING son herramientas que permiten obtener información más precisa de la base de datos. A menudo las tablas contienen más información que la necesitada. Cuando estos registros sólo son conocidos bajo ciertas condiciones, sólo pueden ser seleccionadas a través de un proceso llamado selección o restricción.

Seleccionar todos los registros

Todos los registros de datos de una tabla pueden ser seleccionados usando la palabra clave ALL, la cual también está por default en los queries. Si no se especifica la palabra ALL el resultado seríael mismo si no se usara. En cualquier caso, si hay registros duplicados en la tabla, estos son listados. Por ejemplo, si se quiere conocer todos los registros de una columna se teclea:

SELECT ALL columna FROM tabla

Eliminar registros duplicados

Si se quiere eliminar los registros duplicados del listado se usa la cláusula DISTINCT, como se muestra: SELECT DISTINCT campo FROM tabla

Cuando la cláusula DSTINCT es agregada al comando SELECT, SQL primero produce la tabla resultado. Después, la tabla resultado es ordenada y todos los registros duplicados son eliminados. La sintaxis de la expresión SELECT relacionada con la cláusula DISTINCT es:

SELECT [ALL/DISTINCT] *lista-a-seleccionar* FROM *nombre-tabla*

Al usar la palabra DISTINCT en el comando SELECT, se pueden remover los registros seleccionados de la lista de salida. Otro método para eliminar registros listados, es usar las cláusulas WHERE y HAVING.

Condiciones de búsqueda

Una parte de las cláusulas WHERE y HAVING es una condición de búsqueda. Una condición de búsqueda es definida como uno o más predicados que definen el criterio para datos seleccionados. Un predicado es un elemento de una condición de búsqueda que retorna un valor de verdadero o falso cuando aplica a un registro. Si retorna un valor de verdadero, el registro es incluido en la lista de salida. Si retorna un valor de falso, éste no será incluido en la lista de salida.

Una condición de búsqueda puede ser compuesta de uno o más predicados. Si está compuesta de más de un predicado, son combinados usando AND o OR. Los paréntesis son usados para indicar el orden de evaluación. Como un resultado de ejecutar la cláusula WHERE y/o la cláusula HAVING, algunos registros serán eliminados de la tabla resultado.

Cláusula WHERE

```
>> SELECT * FROM PERSONAL.EMPLEADO
+> WHERE CODTRAB = 100;
NUMEMP      NOMBRE      APELLIDO    NUMDEP      CODTRAB     SALARIO
----------  ----------  ----------  ----------  ----------  ----------
         1  ROGER       GREEN       9000        100         175500.00
        23  JERRY       HOWARD      1000        100         137000.10
        29  JANE        RAYMOND     3000        100         136000.00
. . .
--- 11 row(s) selected
>>
```

Cláusula HAVING

```
>> SELECT * FROM PERSONAL.EMPLEADO
        WHERE CODTRAB > 250
        HAVING SALARIO  > 2500;
NUMEMP      NOMBRE      APELLIDO    NUMDEP      CODTRAB     SALARIO
---------   ----------  ----------  ----------  ----------  ----------
       75   TIM         WALKER      3000        300         32000.00
       87   ERIC        BROWN       4000        400         89000.00
       89   PETER       SMITH       3300        300         37000.00
. . .
      568   JESSICA     CRINER      3500        300         39500.00
--- 43 row(s) selected
>>
```

La cláusula HAVING provee un método alternativo para expresar restricción. La mayor diferencia entre usar la cláusula WHERE y la cláusula HAVING es cuando HAVING es evaluada en comparación con otras cláusulas SELECT.

Predicados de comparación

Una cláusula WHERE o una cláusula HAVING tienen una condición de búsqueda compuesta de uno o más predicados. Hay varios tipos de predicados: Comparación, BETWEEN, IN, LIKE, EXISTS y cuantificación.

Un predicado de comparación es usado en un comando SELECT al comparar los valores de dos expresiones. Los operadores de comparación que son usados en NonStop SQL son:

Operadores de comparación	
= Igual	**> Mayor que**
<> Diferente	**< = Menor que o igual**
< Menor que	**> = Mayor que o igual**

El orden de evaluación en una condición de búsqueda del primero al último es el siguiente:

- NOT
- AND
- OR

Predicado BETWEEN

El predicado BETWEEN es útil cuando se necesita seleccionar registros que contienen información, la cual es ubicada dentro de un rango de dos valores de una columna.

```
>> SELECT * FROM PERSONAL.EMPLEADO
        WHERE CODTRAB BETWEEN 200 AND 260;
NUMEMP      NOMBRE      APELLIDO    NUMDEP      CODTRAB     SALARIO
----------  ----------  ----------  ----------  ----------  ----------
        75  TIM         WALKER      2000        250         27000.00
        87  ERIC        BROWN       2000        200         24000.00
        89  PETER       SMITH       2000        250         29000.00
--- 3 row(s) selected
>>
```

Nótese que en este comando la selección WHERE…BETWEEN todos los datos entre los valores son seleccionados incluyendo el valor inicial y el final. También puede observarse que el comando WHERE mostrado anteriormente es lo mismo que: CODTRAB >= 200 OR CODTRAB <=260

Si se quiere excluir un rango de valores, se puede agregar la frase **"NOT"** al comando Ejemplo:

```
>> SELECT * FROM PERSONAL.EMPLEADO
        WHERE CODTRAB NOT BETWEEN 200 AND 260
        ORDER BY APELLIDO;

NUMEMP      NOMBRE      APELLIDO    NUMDEP      CODTRAB     SALARIO
----------  ----------  ----------  ----------  ----------  ----------
        231 HERB        ALBERT      3300        300         33000.00
        210 HERB        ALBERT      3300        300         33000.00
. . .
        321 BILL        WINN        2000        900         32000.00
         43 PAUL        WINTER      3100        100         90000.00
--- 54 row(s) selected
>>
```

La columna y variable con las cuales se especifica la cláusula **WHERE...BETWEEN** deben ser del mismo tipo de dato. Ejemplo, si la columna es numérica la variable debe ser numérica.

Predicado IN

En algunas ocasiones es necesario encontrar registros que contienen ciertos valores en una columna particular. Ejemplo:

```
>>SELECT * FROM PERSONAL.EMPLEADO
+> WHERE CODTRAB IN ( 200, 250, 260 );
NUMEMP      NOMBRE      APELLIDO    NUMDEP      CODTRAB     SALARIO
----------  ----------  ----------  ----------  ----------  ----------
        219 DAVID       TERRY       2000        250         27000.00
        230 ROCKY       LEWIS       2000        200         24000.00
        233 TED         MCDONALD    2000        250         29000.00
- - - 3 row(s) selected.
>>>
```

Cada registro es calculado, el valor guardado en la columna nombrada es comparada con la lista IN y el registro es seleccionado si el valor de la columna se iguala con uno de los valores de la lista. El tipo de valor de la columna y de la lista de valores debe ser el mismo.

Nuevamente, cuando se desee excluir datos de la respuesta, se puede incluir la frase **"NOT"** en el predicado IN.

```
>>SELECT * FROM PERSONAL.EMPLEADO
+> WHERE CODTRAB NOT IN ( 200, 250, 260 );
NUMEMP      NOMBRE      APELLIDO    NUMDEP      CODTRAB     SALARIO
----------  ----------  ----------  ----------  ----------  ----------
         1  ROGER       GREEN       9000        100         175500.00
        23  JERRY       HOWARD      1000        100         137000.10
. . .
       557  BEN         HENDERSON   4000        400          65000.00
- - - 54 row(s) selected.
>>>
```

Una expresión que incluye una variable y es numérica no necesita comillas. Sin embargo, si se realizará la búsqueda en la columna las variables deberán ser encerradas entre comillas.

Predicado LIKE

Un predicado LIKE es usado en el comando SELECT para buscar por cada carácter cadena que se iguale con un modelo especificado después de la palabra LIKE en el comando. Nuevamente, los registros son seleccionados basados en el valor de una columna. Ejemplo:

```
>>SELECT * FROM PERSONAL.EMPLEADO
+> WHERE APELLIDO LIKE "K%";

NUMEMP      NOMBRE      APELLIDO    NUMDEP      CODTRAB     SALARIO
----------  ----------  ----------  ----------  ----------  ----------
       214  JULIA       KELLY       1000        500         50000.00
       223  HERBERT     KARAJAN     3200        300         29000.00
       235  MIRIAM      KING        2500        900         18000.00
- - - 3 row(s) selected.
>>>
```

Sólo las columnas con datos de tipo carácter (CHARACTER o PIC X) pueden ser usadas con el predicado LIKE. Nótese que en el comando contiene un nuevo elemento que no ha sido mencionado. Este carácter llamado **comodín** permite establecer condiciones de comparación. También, pueden manipularse los siguientes comodines:

% Un sigo de porcentaje indica que cero o más caracteres de cualquier tipo pueden ser comparados.

_ El guión bajo indica que cualquier carácter sencillo es aceptado en esta posición.

Por ejemplo:

%WAR% se compara con WARN o HARDWARE y SOFTWARE pero no con software porque está en minúsculas.

WAR_ se compara con WARE pero no con WAR o WARNING porque estos tienen algunos números de caracteres mayores a cuatro y tampoco se compara con ware porque éste se encuentra en minúsculas.

Predicado EXITS

El predicado EXISTS es usado para determinar si algunos registros satisfacen la condición especificada en un subquery.

Predicado de cuantificación

Los predicados de cuantificación (ALL, ANY, SOME) son usados para comparar el valor de una expresión a todo, algunos o algún, con el valor de una columna que resulta de un subquery.

Orden de evaluación

Las cláusulas en un comando SELECT son procesadas en un orden específico de evaluación. El orden de procesamiento es: FROM, WHERE, GROUP BY, HAVING, (ALL/DISTINCT), lista-select, ORDFER BY.

La tabla resultado de cada es la entrada para el siguiente paso. Mientras SQL procesa comando en este orden. Aunque SQL procesa los comandos en este orden, no es necesario introducir las cláusulas del comando SELECT en el mismo orden.

3.3.4 Empleo de valores calculados

Expresiones Aritméticas

NonStop SQL permite el uso de expresiones aritméticas en la expresión SELECT. Estas expresiones aritméticas pueden aparecer en la lista-select, la cláusula WHERE o en la cláusula HAVING. Un ejemplo de ello es una expresión aritmética usada en la lista-select y muestra a contiuación:

```
>>SELECT ARTICULO, PRECIO * ART_DISPONIBLE
+> FROM PARTES;
ARTICULO                  (EXPR)
------------------------  ---------------------------
PC SILVER 20G                            95000.00
PC GOLD 30G                             198000.00
PC DIAMOND 60G                           84000.00
```

Otro ejemplo en donde la expresión aritmética se encuentra en la cláusula WHERE es:

```
>>SELECT ARTICULO, PRECIO, ART_DISPONIBLE
+> FROM PARTES
+> WHERE PRECIO * ART_DISPONIBLE > 1000000;
ARTICULO        PRECIO          ART_DISPONIBLE
--------------- ----------      ------------------
LASER PRINTER,          4200.00                 300
```

Uso de Funciones Agregadas

NonStop SQL provee un número especial de funciones integradas para realzar este poder de recuperación de información. Las funciones disponibles son COUNT, SUM, AVG, MAX Y MIN. Cada una de estas funciones opera en la colección de valores en una columna de alguna tabla y produce un valor como resultado. Este valor es definido como sigue:

COUNT Contabiliza el número de registros que resultan de un query o de un número de registros contenidos en un valor distinto en un columna especifica.

SUM Suma los valores en la columna.

AVG Promedia el valor en la columna.

MAX Determina el valor máximo en la columna.

MIN Determina el valor mínimo en la columna.

Ejemplo: Desplegar el mínimo, máximo y salario promedio para cada empleado introduciendo el siguiente comando:

```
>>SELET MIN (SALARIO), MAX (SALARIO), AVG (SALARIO)
+> FROM EMPLEADOS;

(EXPR)          (EXPR)          (EXPR)
---------------  ---------------  ---------------
    12000.00        175500.00        48784.65
```

Se puede observar que el nombre de la columna se encuentra entre paréntesis después de cada operación a ser ejecutada. Ésta es llamada el argumento y dice al query en qué columna ejecutará la función particular. El argumento también puede mostrarse como una expresión que se refiere al valor de cada registro de la tabla resultado, como: AVG (ART_RECIBIDOS * COSTO).

DISTINCT

El argumento de la función puede ser opcional y ser precedida por la palabra clave DISTINCT, para indicar los valores duplicados que serán eliminados antes de que la función sea aplicada. Si DINSTICT es especificada, el argumento debe consistir en una sola columna. Si DISTINCT no es especificada, el argumento puede consistir en una expresión aritmética como PRECIO * ART_DISPONIBLES. Las funciones agregadas pueden ser muy útiles para sumas columnas, obtener promedios de las columnas, etc. Estas funciones permiten la extracción de datos de varias columnas.

Agrupando registros de información

El operador GROUP BY reconfigura conceptualmente la tabla representada por la cláusula FROM dentro de grupos o colecciones tal que dentro de un grupo o colección, todos los registros tienen el mismo valor para el campo GROUP BY. En el siguiente ejemplo, la tabla EMPLEADOS, es agrupada a fin de que cada grupo contenga todos los registros para CODIGO 200.

Ejemplo: Se quiere conocer el mínimo, máximo y promedio de los salarios de los empleados para cada código de trabajo.

```
>>SELECT CODIGO, MIN (SALARIO), MAX (SALARIO), AVG (SALARIO)
+> FROM EMPLEADOS
+> GROUP BY CODIGO;
CODIGO (EXPR)          (EXPR)          (EXPR)
-----------  ----------  ----------  ----------
100          56000.00    175500.00    105954.59
200          24000.00     24000.00     24000.00
...
500          25000.75     50000.00     34666.91
```

La lista select del comando SELECT es aplicada a cada grupo en la tabla. Cada expresión en la lista select debe ser un valor sencillo para cada grupo, este valor puede tomar una o varias formas:

- Una forma puede ser la columna GROUP BY misma. En el ejemplo, la columna GROUP BY, CODIGO, es listada para cada grupo.
- Otra forma, es una expresión aritmética involucrando una columna GROUP BY.
- Otra forma, es una función, como MAX, que corre sobre todos los valores de una columna dada dentro de un grupo y reduce estos valores a un simple valor.

3.3.5 Obtener datos de múltiples tablas

Producto Cartesiano

Un producto cartesiano es el resultado de todas las posibles combinaciones que serán seleccionadas de cada registro o cara tabla las cuales han sido especificadas en el query SELECT. Suponiendo que se quiere unir la tabla DEPTO y la tabla EMPLEADO, el resultado de esta unión será un Producto Cartesiano.

```
Ejemplo:
>> SELECT * FROM DEPTO
Obtiene la información contenida en la tabla DEPTO

>> SELECT * FROM EMPLEADO
Obtiene la información contenida en la tabla EMPLEADO

>> SELECT * FROM DEPTO, EMPLEADO

Une la información contenida en ambas tablas DEPTO y EMPLEADO
```

JOIN

En la mayoría de los casos no se requiere de todas las combinaciones de información que un producto cartesiano produce. Por lo tanto, se quiere acortar la salida creando una unión (join) usando la cláusula WHERE. Una unión es el producto de dos o más tablas relacionadas, combinadas, así que toda la información aparece de inmediato. En este proceso, todos los registros que no satisfacen el predicado join serán eliminados del producto cartesiano.

Una cláusula WHERE para obtener datos, se puede eliminar todos los registros de datos innecesarios Con frecuencia se usará el signo igual (=) en una cláusula WHERE para indicar

cuales columnas se quieren unir. Cuando se usa el signo igual en una consulta, el resultado de la unión será llamada *EquiJon*

EquiJon

Para crear un equijon, se debe tener una columna, la cual es común con dos o mas tablas base o vistas. Un equijon será creado cuando la tabla resultado contenga todas las combinaciones de informaciones de ambas tablas, así que los registros en la primera tabla son unidos con los registros del asegunda igualando la columna en común dentro de ambas tablas. El EquiJon, entonces es la expresión de una relación entre dos tablas, cuando la liga está hecha entre una llave extranjera de una tabla y la llave primaria de otra tabla.

Para crear una unión, la cual contiene información de dos tablas, se preparará un comando SELECT. En la formulación de una consulta primero se especifican las tablas que se quieren unir dentro de la cláusula FROM, expresando la conexión entre ellas con un signo igual dentro de la cláusula WHERE.

Ejemplo:

```
>> SELECT * FROM DEPTO, EMPLEADO
+> WHERE DEPTO.NUMDEPTO = EMPLEADO.NUMDEPTO;
```

Todas las columnas de cada tabla son combinadas dentro de una nueva tabla resultado. Como una consecuencia hay dos columnas NUMDEPTO, una dentro de la tabla EMPLEADO y otra de la tabla DEPTO.

Consulta Join con condiciones adicionales

Ocasionalmente, se requerirá crear una unión en la cual se es muy específico acerca de la información que se requiere. En lugar de unir dos tablas y buscar visualmente la información especifica, se puede obtener únicamente la información que se requiere. Por consiguiente, cuando se crea una unión, se pueden especificar condiciones adicionales usando predicados, otros más que el predicado unión dentro de la cláusula WHERE del comando SELECT. Usando **AND** después de la cláusula WHERE, se puede agregar tantas condiciones adicionales se desee.

Ejemplo:

```
>> SELECT * FROM DEPTO, EMPLEADO
+> WHERE DEPTO.NUMDEPTO = EMPLEADO.NUMDEPTO
+> AND CODIGO = 100;
```

Nombrar columnas

El colocar prefijos a los nombres de las columnas con los nombres de las tablas puede ser útil, pero esto puede ser tedioso al introducir repetidamente el nombre completo de la tabla. Afortunadamente esto no es necesario, ya que un *alias* se refiere a algo similar a un nombre relacional, puede ser definido para cada tabla usada en el comando SQL y el nombre relacional puede ser usado como prefijo.

Se puede definir un nombre relacional en la cláusula FROM de cualquier comando SELECT o en la cláusula SELECT de un comando INSERT o subconsulta.

Ejemplo:

```
>> SELECT C.NOMBRE, C.CIUDAD, C.ESTADO, O.NUM
+> FROM CONSUMIDOS C, ORDENES O
+> WHERE O.NUM = C.NUM
+> AND C.ESTADO = "CALIFORNIA"
+> ;
```

3.3.6 SELECT empleando subconsultas (subqueries)

SubConsultas

Una subconsulta es una expresión SELECT…FROM…WHERE que está anidada dentro de otra expresión. La diferencia entre una expresión SELECT y una subconsulta, es que el comando SELECT obtiene valores que serán desplegados o reportados. Una subconsulta regresa valores a usar en comparaciones de las condiciones e búsqueda de un SELECT. Una expresión que contiene una subconsulta puede tener dos componentes:

- Un query externo.-Es la parte de la expresión precediendo la primera expresión WHERE.
- Un query interno.-Es parte de la expresión siguiente la expresión WHERE (ésta es la subconsulta).

Desde que un subquery es así mismo, de la forma SELECT…WHERE, puede contener un subquery en turno, así que el query interno (subquery) de un SELECT tiene sus propios componentes interno y externos. El límite de subqueries anidados in NonStop SQL es 16.

Subqueries con el predicado IN

Subqueries son típicamente usados para representar el grupo de valores ser buscados por medio de un predicado IN.

```
>>SELECT NOMBRE, CIUDAD FROM  CLIENTES
+>   WHERE CLIENTES.NUMERO IN
+>    (SELECT ORDENES.NUMERO
+>        FROM ORDENES
+>        WHERE FECHA > 870801);
```

Nótese que en este ejemplo el subquery se encuentra entre paréntesis y realiza comparaciones. El sistema evalúa el query completo, primero procesa el subquery, después regresa un grupo de valores para una columna sencilla y entonces el query exterior evalúa el predicado IN.

Subqueries con predicados de comparación

Un operador de comparación puede ser usado en lugar del predicado IN. Cuando se usa un operador de comparación, el resultado de un subquery debe ser una columna sencilla.

Ejemplo:

```
>> SELECT NOMBRE, APELLIDO, SALARIO
+> FROM EMPLEADOS
+> WHERE SALARIO >
+>    ( SELECT MAX(SALARIO)
+>      FROM EMPLEADOS WHERE NUMDEPTO = 1500);
```

El subquery entre paréntesis determina el salario máximo de empleados en el departamento.

Subqueries con predicado ALL

El predicado ALL puede ser usado con un operador de comparación para incluir todos los posibles valores en una selección de un subquery.

Ejemplo:

```
>> SELECT * FROM EMPLEADOS
+> WHERE SALARIO > ALL
+>   (SELECT SALARIO FROM EMPLEADO WHERE CODIGO = 400) ;
```

Subqueries con predicados ANY/ALL/SOME

Los predicados ANY/ALL/SOME son usados con operadores de comparación y son colocados en el comando SELECT y después el subquery. Algunos puntos que nos determinan el porqué utilizar estos predicados son:

- Cuando el predicado ALL es usado, el resultado global es verdadero si el subquery es verdadero para cada valor seleccionado por el subquery o si el subquery seleccionado no tiene valores.

- Es usado para evaluar cualquier valor procesado por el subquery, donde este resultado es verdadero para cada uno de los valores seleccionados en el subquery

- El predicado SOME es equivalente de ANY, pueden ser usados de forma intercambiable.

Ejemplo:

```
>> SELECT * FROM EMPLEADO
+> WHERE SALARIO > ANY
+>    (SELECT SALARIO FROM EMPLEADO WHERE CODIGO = 400) ;
```

3.4 Creación de tablas

3.4.1 Catálogos

Un catálogo es un grupo de tablas NonStop SQL las cuales contienen información acerca de objetos SQL en la base de datos, como el número de veces que sea hecho un cambio a la base de datos, como la agregación de una columna a una tabla, dicha información se refleja en el catálogo. Esto pasa automáticamente cuando un comando DDL es usado para cambiar la base de datos El acceso a la base de datos también usa información contenida en el catálogo.

Un objeto SQL es una entidad de base de datos el cual es creado, manipulado o dado de baja usando comandos SQL y es descrito en un catálogo SQL. Esto incluye tablas, índices, vistas archivos, columnas, particiones y restricciones.

El beneficio de tener un catálogo estructurado, como una tabla, es que el administrador de la base de datos puede consultar las tablas catálogo, usando SQLCI y localizar información acerca del estado actual de la base de datos. Por ejemplo, el administrador de la base de datos puede listar los nombres de todas las tablas a partir de información disponible en el catálogo.

Otro beneficio de tener catálogos es el hecho que ellos contienen información estadística acerca de las tablas. Esta información es usada por el optimizador para determinar el mejor acceso estratégico para expresiones DML.

Tablas catálogo

Cada catálogo NonStop SQL es comprendido por un grupo de tablas catálogo. Cada tabla en un catálogo describe un tipo particular o algún aspecto de un objeto. Aún los objetos que definen el catálogo (ejemplo tablas, vistas, etc.) son descritos en el catálogo Cuando se crea un catálogo NonStop SQL automáticamente se crea todas estas tablas e índices.

Nombre de la tabla	Función de la tabla
BASETABS	Describe los atributos de las tablas base
COLUMNAS	Describe las columnas de las tablas.
COMMENTS	Guarda comentarios en columnas, restricciones, índices, tablas o índice.
CONSTRNT	Describe las restricciones definidas en tablas.
FILES	Describe los atributos de archivos que contienen tablas e índices.
INDIXES	Describe los índices definidos en las tablas
KEYS	Describe las columnas llaves de índices
PARTNS	Describe particiones de tablas e índices
PROGRAMS	Describe archivos de programa objetos SQL
TABLES	Describe tablas y vistas
TRANSIDS	Guarda IDs de transacciones TMF para operaciones DDL en el catálogo.
USAGES	Describe dependencias entre objetos SQL
VERSIONS	Provee una copia respaldo de la información de versión en las tablas catálogo
VIEWS	Describe los atributos de las vistas

Cada catálogo tiene un nombre asignado por el sistema. Aún cuando el nombre de la tabla sea el mismo en todos los catálogos, el nombre completo de cada tabla catálogo es único en la red. Por ejemplo, el nombre completo de un catálogo tabla es:

\EXT25.$DATA1.SALES.BASETABS.

Cada sistema en el cual NonStop SQL es usado tiene al menos un catálogo. Cada objeto NonStop SQL en un sistema (tabla, vista, índice, partición, etc.) puede ser descrito en el mismo sistema.

Cada volumen en un sistema puede contener uno o más catálogos. Cada subvolumen de un volumen puede tener sólo un catálogo NonStop SQL Este catálogo puede describir objetos en el mismo subvolumen, en otro subvolumen del mismo volumen u otro volumen del mismo sistema. El nombre del catálogo es el mismo del nombre del subvolumen en el cual esté ubicado.

Consultando catálogos

Ejemplo: si se requiere ver todas las tablas en el catálogo se puede realizar mediante el comando:

```
>> SELECT * FROM SALES.BASETABS;

TABLENAME                              CONSTRAINT    ALLINDEXESHERE
FILENAME                      STATISTICSTIME         ROWCOUNT      ROWSIZE
VALIDDEF       VALIDDATA

\EXT25.$DATA1.SALES.BASETABS           N             Y
\EXT25.$DATA1.SALES.BASETABS  0                      -1            90
Y              Y
\EXT25.$DATA1.SALES.BASETABS           N             Y
\EXT25.$DATA1.SALES.BASETABS  0                      -1            253
Y              Y
\EXT25.$DATA1.SALES.BASETABS           N             Y
\EXT25.$DATA1.SALES.BASETABS  0                      -1            202
Y              Y
\EXT25.$DATA1.SALES.BASETABS           N             Y
\EXT25.$DATA1.SALES.BASETABS  0                      -1            3068
Y              Y
. . .
\EXT25.$DATA1.SALES.BASETABS           N             Y
\EXT25.$DATA1.SALES.BASETABS  0                      -1            3041
Y              Y
```

Cuando se usa este comando, se podrá observar que todos los nombres de las tablas tienen los nombres restringidos como GUARDIAN 90 lo requiere. Si se quiere consultar una tabla para encontrar todos los índices, por ejemplo en la tabla PARTS, se puede tomar ventaja con el comodín usando el siguiente comando:

```
>> SELECT TABLANAME.INDEXNAME FROM INVENT.INDEXES
+> WHERE  TABLENAME.INDEXNAME LIKE "%PARTS%"
NOMBRETABLA                                          NOMBREINDICE
-----------------------------------------------      -----------------------------------------------
\EXT25.$DATA1.INVENT.PARTSUPP                        \EXT25.$DATA1.INVENT.PARTSUPP
\EXT25.$DATA1.INVENT.PARTSUPP                        \EXT25.$DATA1.INVENT.XSUPORD
```

Ambos comodines son necesarios en este comando porque el primero de ellos califica el nombre y el segundo se encarga de buscar los espacios en blanco. En el primer registro listado, TABLENAME e INDEXNAME son el mismo. Esto es cierto para cualquier tipo de tabla y graba la existencia de la llave primaria (o tal vez el índice primario) para la tabla.

Catálogo maestro

SQL puede conocer acerca de todos los catálogos que hay en el sistema. Esta información es alamcenada en un Catálogo Maestro, el cual se encuentra en un subvolumen de sistema SQL. El subvolumen de sistema SQL está señalado por el administrador de la base de datos cuando SQL es iniciado y siempre tiene un nombre de subvolumen de SQL. Comúnmente es el volumen $SYSTEM.SQL, pero es forzoso que el catálogo maestro se encuentre en $SYSTEM. (Ver *figura 3.4.1.1)*

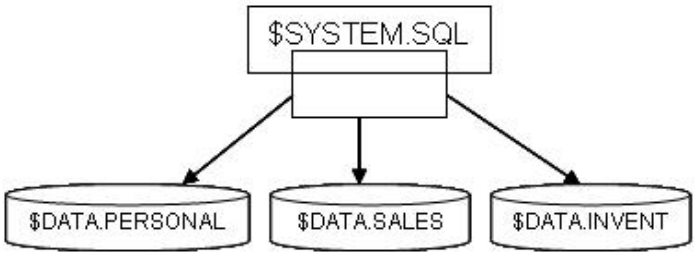


Figura 3.4.1.1 Catálogo Maestro

Para ver todo los catálogos que se encuentran en el sistema se usa el siguiente comando:
SELECT * FROM $SYSTEM.SQL.CATALOGS;

Para ver la tabla SQL,CATALOGS se debe tener los permisos de lectura. También se debe tener permisos de lectura y escritura a la tabla SQL.CATALOGS para poder crear un nuevo catálogo.

Creando catálogos

Se pueden crear catálogos utilizando el comando CREATE CATALOG (el cual es un comando DDL). Cuando el sistema crea un catálogo, el grupo completo de tablas catálogos aparecen en el subvolumen específico (indexes, partns, files, columns, views, basetabs, tables, versions, comments, transids, usages, programs, constrnt, keys).

Un mínimo de un catálogo debe existir para cada sistema o nodo. Sin embargo, se puede decidir el tener más de un catálogo en un sistema. La decisión de crear más de un catálogo se debe basar sobre cómo se desea administrar la base de datos.

La división de datos dentro de dos catálogos que se encuentran dentro de la misma base de datos, es lógica, basada en la división de datos para la organización. Cualquier entrada escrita para el catálogo es también escrita catálogo maestro SQL:CATALOGS, en el cual el catálogo se encuentra.

La estructura del comando para crear un catálogo es: CREATE CATALOG *nombre-catálogo*;
La parte del comando *nombre-catálogo* identifica el subvolumen en el cual el catálogo será guardado. El nombre del catálogo deber único para el subvolumen del volumen en el que se encontrará. Si se omite *nombre-catálogo*, el sistema determinará el catálogo default.

Eliminar un catálogo

Un catálogo no puede ser modificado o particionado. Cualquier catálogo creado con el comando CREATE CATALOG puede ser dado de baja con el comando DROP CATALOG.

Nota: Cualquier usuario autorizado puede leer un catálogo, pero sólo un proceso SQL autorizado puede actualizar un catálogo.

3.4.2 Tipos de tablas y descripción de tablas

Tipo de tablas

Existen tres diferentes tipos de tablas en SQLCI. Esto es muy importante ya que se tiene que especificar en el momento en que se haga uso del comando CREATE TABLE. Sus diferencias están basadas en como los datos son almacenados en la base de datos y como ellos son accedidos por el SQLCI. Los tres tipos de tablas son:

- Ordenada por llave
- Ordenada por entrada
- Relativa

Tablas ordenadas por llave

Es un grupo de registros de longitud variable, lógicamente almacenados en orden del valor de la llave primaria. Una llave primaria es una llave especificada por el usuario (compuesta por una o más columnas de la tabla) o una llave generada por el sistema (un timestamp de 8 bytes en tiempo de greenwich). La llave primaria no puede ser modificada cuando se esté actualizando una entrada.

Cualquier registro de una tabla puede ser eliminado o modificado. Ellos automáticamente son insertados dentro de la ubicación que sea definida por el valor de la llave primaria Se pueden agregar columnas a tablas ordenadas por llave y las columnas del tipo VARCHAR pueden ser disminuidas o alargada en longitud.

Tablas ordenadas por entrada

Es un grupo de registros que pueden variar en longitud de cero bytes (vacío) a longitud máxima, la cual es especificada cuando la tabla es creada (RECLENGTH). Las tablas ordenadas por entrada no tienen llave primaria. Cada registro es identificado por una llave creada por el sistema (dirección de 4 bytes), conocida como dirección de byte relativo. A diferencia de la tabla ordenada por llave, la llave creada por el sistema no contiene datos, ésta sirve como identificador para realizar cambios a la tabla.

Todos los registros son automáticamente insertados al final del archivo (por ello el nombre de ordenado por entrada). Los registros pueden ser modificados, pero una vez almacenados no podrán ser borrados. No se puede agregar columnas a una tabla ordenada por entrada y el valor de las columnas VARCHAR no puede ser incrementado.

Tablas relativas

Las tablas relativas (acceso directo) consisten en un grupo de registro de longitud fija. No tienen llave primaria. El identificador único es creado por el sistema (Número de registro relativo de 4 bytes), el cual es especificado por el sistema o por el usuario haciendo referencia a un registro en particular. El primer registro en una tabla relativa es señalada por el registro cero (0); los registros sucesivos serán identificados por un número ascendente en incrementos de uno (1). Los registros son insertados automáticamente dentro de la ubicación, que está implícita por el valor de la llave creada por el sistema. Ellos pueden ser eliminados o modificados.

Comando CREATE TABLE

El comando CREATE TABLE es un comando DDL usado para definir los elementos de una nueva tabla. Todos los datos en una base de datos NonStop SQL residen en tablas que se crean al usar este comando. Los datos son descritos por definiciones de columna dentro de la tabla.

Para crear una tabla se deben poseer permisos de escritura al catálogo en el cual se encontrará la descripción de la tabla (y cualquier partición).

Para identificar las columnas componentes, la definición de una tabla incluye el nombre asignado a la tabla, el nombre del catálogo que soporta la información de la tabla, la información de seguridad de la tabla y las características del archivo que soporta los datos de la tabla. Se pueden modificar muchos aspectos de la tabla una vez que ésta ha sido creada; pero no se puede modificar la ubicación del catálogo, la organización o la descripción de la llave primaria (de una tabla ordenada por llave).

Muchos atributos la tabla tienen valores default[23], por lo tanto, en la definición de la tabla no se especifican. El siguiente ejemplo, muestra cómo usar el comando CREATE TABLE, para crear una tabla llamada EMPLEADO, en el subvolumen actual.

```
>> CREATE TABLE SUBVOLUMEN.EMPLEADO
+> (
+> NUMDEPTONUMERIC(4) UNSIGNED      DEFAULT SYSTEM,
+> APELLIDO   CHARACTER (20)        NO DEFAULT,
+> NOMBRE     CHARACTER (15)        NO DEFAULT,
+> NUMEMP     NUMERIC (4) UNSIGNED  NO DEFAULT,
+> CODTRAB    NUMERIC (4) UNSIGNED  DEFAULT 100,
+> SALARIO    NUMERIC (4) UNSIGNED  DEFAULT SYSTEM,
+> PRIMARY KEY NUMEMP
+> )
+> CATALOG NOMBRE-CATÁLOGO
+> ORGANIZATION KEY SECUENCED
+> ;
```

La tabla empleado es registrada en el catálogo indicado (en un subvolumen) y éste reside en otro subvolumen. Esta contiene seis columnas en el orden dado.

Cualquier tabla que se haya creado puede ser borrada usando el comando DROP TABLE. La tabla y cualquier dependencia son removidas del catálogo y los datos contenidos son perdidos.

Valores default de las columnas

Se deben especificar las columnas que son permitidas para tomar un valor default y definir dicho valor. Si no se define el valor default en el comando CREATE TABLE, cada vez que se inserte un registro a la tabla se tendrá que especificar el valor que contendrá este registro. El siguiente cuadro muestra las diferentes opciones default con la correspondiente acción del sistema cuando se inserta un registro

Opción DEFAULT	Acción del sistema al inserta un registro
DEFAULT *literal*	*literal* es colocada en la columna
DEFAULT SYSTEM	NUMERIC columna llena con ceros CHARACTER columna llena con espacios en blanco VARCHAR columna cadena de longitud cero (vacia)
NO DEFAULT	Los datos deben ser dados cuando se inserte

[23] Valores asumidos a pesar de la explícita anulación de ellos como parte de la sintaxis del comando. Valores predeterminados.

	un registro un mensaje de error es mostrado.

Si la opción default no es especificada, DEFAUL SYSTEM se asume como tal.

Seguridad de la tabla

Para ejecutar cualquier operación de base de datos, se deben poseer los permisos necesarios para ello. Para encontrar si se está autorizado para ejecutar una operación, el sistema hace referencia al ID del usuario quien se identifica e ingresa al grupo que pertenece. Basado en la identificación y si es un usuario local o de red, el sistema verifica la seguridad de los objetos que se quiere usar. Para verificar la seguridad actual de cualquier tabla, se utiliza el comando FILEINFO

Definiendo la seguridad de tablas

La seguridad de una tabla se puede establecer usando la cláusula SECURE *cadena-seguridad* en el comando CREATE TABLE Éste establece la cadena de seguridad que controla el acceso al archivo.

La fracción de la cadena de atributos es una cadena de caracteres de longitud de 4 con la forma "rwep" (Read, Write, Execute, Purge). Este atributo específica los usuarios que son autorizados con permisos de lectura escritura, ejecución y eliminación. Los caracteres que pueden aparecer en *rwep* son los siguientes:

- Solo super ID local
O Propietario local o super id local
G Cualquier miembro del grupo local o propietario local o super id local
A Cualquier usuario local o super id
U Cualquier miembro de la clase del usuario propietario (usuario local o remoto con el mismo id del usuario propietario del archivo)
C Cualquier miembro de la comunidad del usuario (usuario local o remoto con el mismo numero de grupo del propietario)
N Cualquier usuario local o remoto.

El usuario que crea la tabla es el propietario, por lo tanto, automáticamente se tiene autorización para ejecutar en ella cualquier operación, a menos que la tabla esté asegurada únicamente para el super id. La "*e*" (ejecución) en la fracción de acceso de *rewp* únicamente afecta a programas ejecutables u objetos de archivos.

Si no se especifica la seguridad de una tabla o vista cuando se crea (usando los comandos CREATE TABLE o CREATE VIEW), SQLCI la define con la seguridad default actual. No se puede cambiar la seguridad default actual durante una sesión de SQLCI. También, cualquier partición, índice y restricción que se ha creado, tendrá la misma seguridad que la tabla o vista con la cual se han asociado.

3.4.3 Tablas particionadas

Particionar tablas es un término de base de datos que puede ser usado con dos diferentes significados. Uno de ellos es simplemente el proceso de separar las tablas de una base de datos en alguna forma.

Tablas distribuidas

Esto llega a ser comprendido cuando se da cuenta que la arquitectura de Tandem permite a estas tablas estar separadas en dos diferentes sistemas ligadas por líneas de comunicación con toda la demás base de datos.

Particionar también significa que se tienen los datos de una tabla particular físicamente dividida en dos o más piezas. Como un usuario, sin embargo, se usa como una sola tabla. Para poder partir una tabla se puede hacer:

- Proporcionar más espacio en disco que el disponible en un volumen
- Incrementar los caminos de acceso a la base de datos, de esta manera mejorar el desempeño.
- Ubicar los datos en un nodo de la red el cual es cerrado a donde éste sea usado.

Particionar una tabla es un simple método para distribuir los datos a través de dos o más volúmenes lógicos, de tal manera que las partes estén conectadas. NonStop opera sobre ellos como si estuvieran ubicados en un sólo lugar. Ellos también pueden estar conectados a la misma computadora o a diferentes computadoras Tandem las cuales están ligadas en distintas ubicaciones geográficas. Desde que la computadora es ligada dentro de la red, NonStop SQL puede acceder fácilmente a los datos de una ubicación a otra.

Particiones locales

Las particiones locales son a menudo necesarias cuando hay muchos datos para acomodar en un sólo volumen. Este tipo de situación puede requerir más de un dispositivo de almacenamiento para mantener los datos.

Configuración de particiones

Existen básicamente dos configuraciones posibles cuando se particiona una tabla.

1. Se puede particionar una tabla entre varios volúmenes en el mismo sistema. Cada partición puede ser administrada por diferentes catálogos, aunque esto no sea un requisito.
2. Se puede dividir una tabla entre volúmenes diferentes en varios sistemas (nodos). La partición en un sistema particular puede ser administrada un catálogo o catálogos en ese mismo sistema.

Para una tabla particionada, se aplican las siguientes condiciones:

1. Se debe especificar la cláusula PARTITION en el comando CREATE TABLE, para cada partición otra que la primera, la cual es entendida para empezar con los valores limitados de la llave primaria.
2. Cada partición tiene el mismo subvolumen y nombre de tabla, pero diferente nombre de volumen.
3. Para desarrollar cualquier operación DDL se requiere que se posean privilegios de escritura a todas las particiones y sus catálogos referenciados.

Ejemplo:

```
>> CREATE TABLE \SYS1.$DATA1.SUBVOLUMEN.EMPLEADO
+> (
+> NUMDEPTONUMERIC(4) UNSIGNED        DEFAULT SYSTEM,
+> APELLIDO   CHARACTER (20)          NO DEFAULT,
+> NOMBRE     CHARACTER (15)          NO DEFAULT,
+> NUMEMP     NUMERIC (4) UNSIGNED    NO DEFAULT,
+> CODTRAB    NUMERIC (4) UNSIGNED    DEFAULT 100,
+> SALARIO    NUMERIC (4) UNSIGNED    DEFAULT SYSTEM,
+> PRIMARY KEY NUMEMP
+> )
+> CATALOG NOMBRE-CATÁLOGO1
+> MAXEXTENTS 160
+> ORGANIZATION KEY SECUENCED
+> PARTITION (\SYS1.$DATA2.SUBVOLUMEN.EMPLEADO
+>                          CATALOG \SYS1.$DATA2.CATÁLOGO2
+>                          FIRST KEY 5000
+>                          MAXEXTENTS 90 ) ;
+> ;
```

Nótese los volúmenes diferentes $DATA1, para la primera partición y $DATA2 para la segunda. También, se observa que los catálogos diferentes y FIRST KEY 500 parte de la cláusula PARTITION, que indica que todos los registros tienen un valor NUMEMP de 500 o más irán a la segunda partición. Por implicación un NUMEMP entre 0000 y 4999 irán en la primera partición.

Consideraciones de Acceso Remoto

Desde la perspectiva de un usuario SQLCI, incluso si una tabla puede estar particionada entre varios nodos, el usuario la visualiza como una sola tabla. Si se usa el comando SELECT se obtendrá una respuesta sin alguna advertencia sobre la fuente de los datos que se están visualizando. Esto significa que esta partición es transparente al usuario; sin embargo, desde la perspectiva del administrador de la base de datos existen varios puntos a considerar.

El primer punto es que los datos, primeramente usados por un usuario local, pueden estar en el nodo local. Esto redunda en reducción de costos de comunicación de datos. Para asegurar que datos locales están en el nodo local, un indicador de división debe ser elegido para provocar que esto suceda. Por ejemplo, seria buena idea dividir la tabla acorde al numero de plantas.

Un segundo a punto considerar es cuando índices son frecuentemente usados ya que la partición física de la tabla son menos importantes. La razón de esto es que, la tabla será dividida basada en el valor del índice primario y el índice alternativo será probablemente particionado de modo distinto del índice primario y la tabla.

3.4.4 Modificar la estructura de una tabla

Permisos necesarios

Los cambios a algunas características de una tabla (estructura de la base de datos) los usuarios comunes no están permitidos a hacerlo, Usualmente, los gerentes de sistemas o administradores

de base de datos realizan estos cambios, aunque a menudo son relaizados por programadores cuando se trabaja en un ambiente de desarrollo.

Para cambiar las características de una tabla o de cualquier otro objeto SQL, se debe ser el propietario local, el usuario SUPER.SUPER local o un propietario remoto con permisos de borrado a la tabla. Además, se pueden tener permisos de lectura y escritura a los catálogos asociados

Cambiando atributos de tablas

El comando ALTER es un comando DDL usado para modificar los atributos de seguridad (y archivo físico) de un objeto SQL. Uno de los usos más comunes es modificar los atributos de tablas, vistas, índices, catálogos y programas. La estructura del comando es la siguiente:

>> ALTER *objeto nombre-objeto spec*

Donde:
>> : Prompt del sistema
ALTER: Comando SQL
Objeto: Puede ser, CATALOG, TABLE, VIEW, PROGRAM, INDEX.
nombre-objeto: Nombre del objeto
spec: Puede ser: security-spec, attribute spec.

Comando ALTER… SECURE

Si se desea modificar la seguridad de una tabla con el propósito de limitar el acceso a ciertos usuarios. Por ejemplo: si se quiere modificar la seguridad de la tabla EMPLEADO a manera que cualquier usuario local o remoto tenga permisos de lectura y escritura, pero únicamente el propietario local o el super Id local tengan permisos de ejecución o borrado, se estructura el comando de la siguiente manera:

```
>> ALTER TABLE EMPLEADO
+> SECURE "NN00";
```

Agregando columnas a tablas existentes

El comando ALTER TABLE ADD COLUMN es un comando DDL, usado para agregar nuevas columnas a tablas existentes. Cuando se agrega una nueva columna, ésta se visualiza en la tabla como la última columna y aparece en todas las particiones, si existen.

Al agregar una columna no provocará que los registros existentes sean leídos o reescritos. Si se actualiza alguna otra columna en el registro, la nueva columna no estará todavía agregada a la columna que fue anexada. Sólo cuando se ejecuta un UPDATE a la columna que fue agregada, esta columna será integrada a los registros.

Los registros existentes tomarán el valor default hasta que la nueva columna sea actualizada. Si se ejecuta un comando SELECT * en la tabla, la nueva columna será desplegada con el valor default siendo mostrado como si éste hubiera sido ya agregado. Esto quiere decir que si se agrega una columna a una tabla con registros ya existentes, dicha columna tomará el valor default para los registros contenidos en la tabla. El comando ALTER TABLE ADD COLUMN utiliza el siguiente formato:

>> ALTER TABLA *nombre-tabla* ADD COLUMN *nombre-columna tipo-de-dato*

Donde:

>> Prompt del sistema
ALTER: Comando SQL
TABLE: Identifica el comando SQL
nombre-tabla: Identifica la tabla que se quiere modificar
ADD COLUMN: Identifica el comando SQL
nombre-columna tipo-dato: Identifica la columna que se quiere agregar y su tipo de dato.

El nombre de la tabla identifica la tabla que será modificada. Deberá reunir los siguientes requisitos para que el comando sea válido:

- Si la tabla es particionada, todas las particiones deben ser accesibles.
- La tabla no debe ser ordenada por entrada.

El nombre de la columna es simple y debe ser único para identificar las columnas en la tabla.

Ejemplo:

```
>> ALTER TABLE EMPLEADO
+> ADD COLUMN ANIOS_SERVICIO SMALLINT
+> DEFAULT SYSTEM;
```

Se puede observar la estructura de la tabla EMPLEADO usando el comando INVOKE.

Ejemplo:

```
>> INVOKE EMPLEADO
-- Definition of tabla \EXT25.$DATA1.EMPLEADO
-- Definition current at 13:29:46 – 06/06/88
(
NUMDEPTO                NUMERIC (4,0) UNSIGNED
, LAST_NAME             CHAR (20)
, FIRST_NAME            CHAR (15)
, NUMEMP                NUMERIC (4,0) UNSIGNED
, CODIGO                NUMERIC (4,0) UNSIGNED
, SALARIO               NUMERIC (8,2) UNSIGNED
, ANIOS_SERVICIO        NUMERIC (5,0)
```

Esto muestra que la columna que se anexó existe en la tabla; sin embargo el espacio en disco será asignado para la nueva columna sólo para registros que fueron actualizados para guardar valores en la columna, por lo tanto el comando ALTER TABLE cambia únicamente la definición del catálogo de la tabla; después un comando DDL colocará datos dentro de la columna.

Agregando particiones a tablas existentes

El comando ALTER TABLA ADD PARTITION es un comando DDL, usado para agregar una nueva partición a una tabla base. Para agregar una partición a una tabla ordenada por llave, se debe incluir la cláusula FIRST KEY para definir el punto de inicio de los registros en los cuales residirá la partición.

Limitaciones de agregar particiones

No se pueden agregar particiones que incluirían un registro que actualmente existe. Se debe asegurar que el valor FIRST KEY define una partición para la cual no existen registros. Si se requiere reconfigurar particiones existentes, deberá ser creada una nueva tabla con los valores deseados por FIRST KEY y entonces la nueva tabla es cargada para la tabla existente, después de la cual la tabla anterior es removida. Las particiones y catálogos asociados deben residir en el mismo sistema y todas las particiones estar accesibles cuando el comando se ejecuta.
Se puede agregar una partición a cualquier tabla ordenada por llave, que tiene una llave primaria especificada por el usuario (no una asignada por el sistema). En tablas relativas y ordenadas por secuencia las particiones dependen de la cantidad de espacio permitido para la tabla en cada volumen.

Eliminar columnas y particiones

No se puede eliminar una partición o columna de una tabla una vez que ha sido creada. Para borrar una columna o partición de una tabla, se debe crear una nueva tabla sin la partición/columna y cargar los datos que se quieren mantener dentro de la nueva tabla.

3.4.5 Características adicionales de tablas

Restricciones

Una restricción (Constraint) es una expresión que ayuda a proteger la integridad de los datos en una tabla especificando las reglas que deben satisfacer los valores en columnas particulares de la tabla. La restricción afectará una tabla fundamental como sigue:

- Las restricciones no podrán contener una función agregada o join. Ellas no acceden a columnas o registros fuera del registro individual bajo consideración.
- Todos los registros existentes en una tabla deben satisfacer la restricción o la restricción no podrá ser creada. Por consiguiente, crear una restricción causará que columnas existentes sean leídas.
- La tabla debe ser accesible cuando el comando se ejecuta. Si la tabla es particionada, todas las particiones deben ser accesibles.
- Crear una restricción no afecta a una restricción existente.
- Si restricciones múltiples existen para una tabla, todas ellas deben ser satisfechas. Se debe tomar cuidado de no crear restricciones conflictivas que pueden impedir la actualización de la tabla.

Para crear una restricción se debe ser el propietario de la tabla fundamental, el super.super local o un propietario remoto con autoridad de eliminación al catálogo en el cual esta descrita la tabla.

Revisando restricciones existentes

Se pueden examinar las restricciones que existen actualmente en una tabla dada, consultando la tabla CONSTRNT del catálogo que contiene la descripción de la tabla.

```
Ejemplo:
Se quiere examinar las restricciones de la tabla ORDERS, el comando es el siguiente:
>>SELECT * FROM SALES.CONSTRNT
+> WHERE NOMBRETABLA  LIKE "%ORDERS%";
NOMBRETABLA                              NOMBRESTRICCION          NUMSEC
---------------------------------------  ------------------------  ----------
TEXTRESTRICCION
------------------------------------------------------------
\EXT25.$DATA.SALES.ORDERS                RESTRIC_FECHA             0
FECH_ENTREG >= FECHA_ORDENADA
>>
```

Estableciendo restricciones

Se ejecuta el comando CREATE CONSTRAINT usando la siguiente sintaxis:

CREATE CONSTRAINT *nombre-restricción*
ON *nombre-tabla* CHECK *condición-búsqueda*

La cláusula nombre-restricción define un nombre de restricción el cual es un simple nombre y debe ser único para asociarlo con la tabla. La cláusula ON *nombre-tabla* define la tabla asociada con la nueva restricción. CHECK *condición-búsqueda* especifica la condición de la restricción. Las siguientes limitaciones son colocadas encima de la condición de búsqueda cuando son aplicadas al comando CREATE CONSTRAINT.

- El texto de la condición debe ser menor de 3000 caracteres.
- Funciones no son permitidas.
- Subqueries no son permitidas.
- Para cualquier registro del *nombre-tabla*, la *condición-búsqueda* debe ser resuelta por la búsqueda sólo para esta tabla.
- Una llave creada por el sistema (Syskey) no es permitida.

Para asegurarse que cualquier número de empleado asignado en la tabla EMPLEADO es menor que 2000 utilizar:

```
>>CREATE CONSTRAINT NUMEMP_LONG
+> ON  EMPLEADO
+> CHECK NUMEMP < 2000;
```

Limitaciones de restricciones

Si otros usuarios tienen una tabla abierta al mismo tiempo que se está usando el comando CREATE CONSTRAINT, el sistema esperará hasta que la tabla esté disponible. Si hay varios registros en la tabla, otros usuarios pueden esperar mientras la restricción está siendo verificada.

Índices

Un índice es definido como un camino de acceso alternativo a una tabla diferente al camino de acceso normal (llave primaria) es definido para la tabla cuando ésta es creada.

Estableciendo un índice

Se puede usar el comando CREATE INDEX para crear un índice en la tabla base. El índice puede especificar una o más columnas de la tabla y ellas no tienen que ser contiguas. El orden indica que puede ser ascendente o descendente.

Documentar Objetos SQL

El comando COMMENT es un comando DDL y es usado para escribir comentarios acerca de un objeto (una tabla, vista, columna, índice o restricción) en un catálogo. Comentando un objeto SQL ayuda a recordar porqué ésta es creada. Un ejemplo de una estructura del comando COMMENT es:

```
>> COMMENT ON CONSTRAINT NUMEMP_LONG
+> ON EMPLEADO
+> IS "SI NUMEMP EXCEDE 4 DIGITOS SE NECESITARÁ ORDENAR DE OTRA MANERA";
```

Nota: La misma estructura podría aplicar cuando se comenta cualquier objeto

Eliminar restricciones, índices y comentarios

Para cancelar una restricción se aplica el comando DROP como sigue: DROP CONSTRAINT *nombre-constraint* ON *nombre-tabla.* En una forma similar, un índice puede ser eliminado: DROP INDEX *nombre-índice* ON *nombre-tabla*. Se puede borrar un comentario de la siguiente manera: COMMENT ON *objeto* IS “” CLEAR

3.4.6 Generar vistas de tablas de datos

Una vista es derivada de un subconjunto de columnas o registros de una o más tablas o de otras vistas. Una vista es conocida como una tabla virtual porque actualmente no existe como una tabla de almacenaje. Una vista no está soportada por sus datos almacenados; sin embargo, una vez que ha sido creada, se puede hacer referencia a ella como si fuera una tabla. Una vista es un simple resultado de un comando SELECT al cual se le asigna un nombre para uso posterior. El uso de vistas permite optimizar el uso de la base de datos para satisfacer las necesidades individuales.

NonStop SQL proporciona dos tipos de vistas:

- Vistas protegidas.- Son derivadas de una tabla sencilla por tomar una proyección de las columnas en la tabla o una selección de los registros en la tabla, o ambas.

- Vistas Abreviadas.- Son derivadas de una o más tablas o de otras vistas.

Si se tiene una tabla que contiene diferentes tipos de información, y alguna de ellas no es necesario tener disponible para todos los usuarios, se requiere crear una vista protegida. Este tipo de vista “esconde” columnas específicas y registros de ciertos usuarios. Por otro lado, si tienen tablas excepcionalmente largas, que contiene más información de la que comúnmente se usa, se podrá crear una vista abreviada, la cual contiene un resumen de todas las tablas.

Ventajas de vista

El uso de vistas para la lectura de datos seleccionados tiene las siguientes ventajas, en lugar de recuperar los datos de la tabla:

- Para proporcionar un comando SELECT simplificado para usuarios, quienes constantemente usan múltiples tablas, se puede eliminar la necesidad de teclear comandos JOIN complejos haciendo uso de las vistas.

- Para eliminar la posibilidad de que otros usuarios modifiquen las tablas, se puede crear Vistas abreviadas que proporcionan acceso de sólo lectura sin afectar a las tablas fundamentales. Cuando es necesario actualizar una tabla, donde las columnas específicas a actualizar son un subconjunto de la tabla completa, puede ser usada una vista protegida para restringir el acceso.

- Limitar a usuarios casuales a un grupo de vistas previamente definidas es una manera de asegurar que sólo un grupo conocido de consultas pueden ser aplicadas a la base de datos. Reduciendo la oportunidad de ejecutar una consulta impropiamente construida.

Comando DISPLAY USE OF

Muestra las vistas que actualmente se usan en una tabla, ejemplo:

```
>>DISPLAY USE OF EMPLEADO;

Object Name                                   Type      S       P       Owner Name    Secure
--------------------------------------------- -------   ---     ----    ------------  ----------
          Catalog Name
          ---------------------------------

1 \ext25.$DATA.PERSNL.EMPLIST                 PV                        SQLED    .NGR   A000
          $DATA.PERSNL
..........
>>
```

El ejemplo anterior muestra que hay una vista protegida, EMPLIST. Ésta siempre será una buena idea revisar las vistas protegidas para observar las restricciones que éstas imponen. Se puede observar cuáles son las vistas descritas en una tabla catálogo llamada VIEWS de la siguiente manera:

```
>>INVOKE PERSNL.VIEWS
-- Definition of tabla \EXT25.$DATA1.PERSNL.VIEWS
-- Definition current at 13:29:46 – 06/06/88
(
  VIEWNAME                CHAR(34)
, PROTECTION              CHAR(1)
, VALIDDEF                CHAR(1)
, AUDIT                   CHAR(1)
, WITHCHECKOPTION         CHAR(1)
, INSERTABLE              CHAR(1)
, VIEWTEXT                VARCHAR(3000)
)
>>
```

Crear Vistas

Para crear cualquier tipo de vista, es necesario tener permisos de escritura para el catálogo que contendrá la descripción de la vista. También, se necesitan permisos de escritura a la tabla USAGES de estos catálogos, los cuales describen las tablas y vistas fundamentales. El comando CREATE VIEW es un comando DDL y tiene la siguiente estructura:

>> CREATE VIEW *nombre-vista list-col* AS *comando-select* option ;
Donde:
>> Es el prompt del sistema
CREATE VIEW: Comando SQL
Nombre-vista: Nombre de archivo Tandem
list-col: Puede ser: nombre-columna, -- o omitido por completo
AS *comando-select:* Puede tener más de una cláusula SELECT
option: Puede ser CATALOG SECURE, FOR PROTECTION WITH CHECK OPTION

Si se omite la lista nombre-columna, la columna de la vista toma el nombre de la columna que se está especificando en el comando SELECT. Una vista puede tener un máximo de 200 a 400 columnas dependiendo del tamaño de la definición de cada columna.
La fracción AS *comando-select* del comando define las columnas y el criterio de selección para la vista. Éste debe llenar todos los criterios para un comando regular SELECT junto con los siguientes requerimientos:

- El texto de las condiciones de búsqueda deben ser menor de 3000 caracteres.
- El comando no puede contener cualquiera de las siguiente cláusulas: ORDER BY BROWSE ACCESS, STABLE ACCESS o REPEATABLE ACCESS

CATALOG *nombre-catálogo* especifica el nombre del catálogo que almacenará la descripción de la vista. El catálogo y la vista deben residir en el mismo sistema. El nombre del catálogo es el mismo como el nombre del subvolumen en el cual reside el catálogo. El catálogo debe ser el mismo que el usado para crear la vista protegida. Si se omite la cláusula CATALOG, el sistema asumirá el catálogo default.

SECURE *string* define la seguridad, la cual ha sido asignada a la vista. La fracción de la cadena es de 4 caracteres “rwep” (Read, Write, Execute and Purge). La diferencia en la seguridad a vistas y la seguridad de tablas radica en que, la primera proporciona una audiencia más amplia con acceso para limitar la cantidad de datos contenidos en la tabla. Si se omite la cláusula SECURE, la seguridad establecida es la default para el ID del usuario actual de la sesión actual.

FOR PROTECTION es la cláusula que identifica una vista protegida. WITH CHECK OPTION define el nivel de integridad requerido de datos colocados dentro de la base de datos con una vista protegida.

Vistas Abreviadas

Para crear una vista abreviada se debe contar con permisos de escritura al catálogo que contendrá la descripción de la vista. El creador de la vista es automáticamente el propietario. Después de crear una vista de este tipo, pueden ser leídas, pero no modificadas. Y cualquier usuario teniendo permisos de lectura a la tabla original (usada para crear la vista), automáticamente tendrá permisos de lectura a la vista.

```
>>CREATE VIEW TABL.FOUNDERS
+> AS SELECT NUMEMP, APELLIDO, NOMBRE, NUMDEPTO, CODIGO
+>     FROM EMPLEADO
+>         WHERE  NUMEMP < 100
+>     CATALOG NOMBRECAT
+>;
```

Las ventajas de una vista abreviada llegan a ser más aparentes cuando se usan en lugar de una tabla join. Recuperar información de una vista abreviada es muy fácil. El comando SELECT contiene una simple cláusula FROM con el nombre de la vista. Una vez que la vista ha sido creada, puede ser leída como si fuera una tabla. Esta vista es también definida como parte de la base de datos y proporciona información como una de ellas.

Vistas Protegidas

Las vistas protegidas proporcionan seguridad para bloquear datos de acceso general a usuarios no autorizados. Una tabla puede estar asegurada para prevenir el acceso a cierta información confidencial. Se puede crear una vista de esta tabla la cual omite cierta información y esta vista puede quedar disponible para los usuarios en general.

Para crear una vista protegida se debe ser el propietario local de la tabla fundamental o un propietario remoto con permisos de borrado a la tabla. La razón por la que sólo el propietario crear una vista protegida es que, por crear una vista protegida se está autorizando a un UPDATE potencial de la tabla fundamental. Cuando se crea una vista protegida se está usando el comando CREATE VIEW, se puede especificar cualquiera de las siguientes cláusulas en la sintaxis FOR PROTECTION, CATALOG, SECURE O WITH CHECK OPTION en cualquier orden siguiente AS *comando-select*

Cláusula FOR PROTECTION

FOR PROTECTION especifica que es una vista protegida, por omitir esta cláusula, la vista será abreviada por default.

Check Option

Cuando se crea una vista protegida se puede usar WITH CHECK OPTION, que especifica que ningún registro de datos podrá ser colocado en la base de datos a través de la vista, a menos que satisfaga la definición de la vista. Esta opción es utilizada sólo con una vista protegida. Si se omite WITH CHECK OPTION, un registro recién insertado o actualizado no necesita satisfacer la definición de la vista. Esta opción tampoco afecta el comando SELECT. A continuación se muestra un ejemplo del comando CREATE VIEW empleando WITH CHECK OPTION.

```
>> CREATE VIEW EMPLISTA
+> AS SELECT
+>          NUMEMP, APELLIDO, NOMBRE, NUMDEPTO, CODIGO
+>          FROM EMPLEADO
+>          FOR PROTECTION
+>           WITH CHECK OPTION
+>          CATALOG NOMCAT
+> ;
Esta vista puede ser empleada para insertar datos en la tabla empleado. Sólo si se está autorizado para ello.
>> INSERT INTO EMPLISTA
+> VALUES (99, "EDWARD", "MICHAEL", 1000, 100 );
```

Eliminando Vistas

Para eliminar una vista se usa el comando DROP como sigue: DROP VIEW *nombre-vista* Para eliminar una vista es necesario tener permisos de lectura y escritura a todos los catálogos los cuales describen la vista. Todos los índices asociados, vista y objetos SQL se deben encontrar accesibles.

3.5 Cargar la base de datos

3.5.1 Insertar datos

Autoridad para cambio de contenidos

Es necesario tener la autoridad apropiada para cambiar los datos contenidos de una tabla. Basado en el código de seguridad de tablas, es necesario ser un individuo o miembro de un grupo con permisos de escritura a la tabla. Los cambios a la estructura de la tabla o la actualización del contenido de los datos de una tabla no requieren de permisos de escritura al catálogo el cual alberga la descripción de la tabla.

La seguridad de las tablas está bajo el control del administrador de la base de datos o administrador del sistema, quienes proporcionan de una combinación de códigos de seguridad de tablas e identificadores a los usuarios.

Insertando datos a una tabla

NonStop SQL proporciona el comando INSERT para agregar datos a uno o más registros en una tabla o vista protegida. La forma general del comando INSERT es la siguiente:

>> INSERT INTO *nombre-tabla lista-objetivo valores-fuente*;

Donde:
>> Prompt del sistema
INSER INTO: Comando SQL
Nombre-tabla: Puede ser una tabla o vista protegida
Lista-objetivo: Puede ser lista de nombres de columnas, *, o puede ser omitido
Valores-fuente: Puede ser la cláusula VALUES (valor1, valor2…) o cláusula (SELECT…)

Se debe tener permisos de escritura a las tablas para poder manejar este comando. Cuando un INSERT es ejecutado, todas las restricciones para una tabla deben ser satisfechas. Si no es así, el nuevo registro no será insertado dentro de la tabla.

```
Ejemplo:
>>INSERT INTO EMPLEADO
+> (APELLIDO, NUMEMP, NOMBRE)
+> VALUES ("HARRISON", 507, "JOAN");
```

Insertar un registro especificando las columnas

En esta forma del comando INSERT, los nombres de las columnas y los valores de cada columna son listados y especificados. Los valores deben corresponder al tipo de valor de la columna. La secuencia de las columnas puede ser diferente de la secuencia en la cual fueron definidas.

Cuando una columna de la tabla no es incluida en la lista, la columna asume el valor default, que es definido cuando la tabla fue creada. Si no hay valor default determinado para la columna ocurrirá un error durante la ejecución. Una razón del porqué emplear esta forma es con propósitos de documentación pues es más claro observar qué ha pasado si se emplean los nombres de las columnas.

Insertar un registro sin especificar las columnas

Se puede preparar un comando INSERT en el cual no se especifiquen las columnas, siempre y cuando se proporcione un valor para cada columna de la tabla. De esta manera los valores son insertados dentro de las columnas en el orden en el que la tabla fue definida.

```
Ejemplo:
>>INSERT INTO EMPLEADO (*)
+> VALUES (3100, "PAVEN", "MATHEW", 999, 100, 25000, 00);
```

El uso de cualquiera de estas dos formas no altera el resultado en la base de datos. Sin embargo, si fue agregada una columna anteriormente, el comando INSERT, sin especificar las columnas, no se llevará a cabo dado que no corresponden las columnas definidas para dicha tabla.

Puntos Clave

Cuando se inserta un registro hay que recordar estos puntos:

- La lista de columnas puede ser omitida, excepto la llave primaria definida por el sistema (Syskey), si se está asignando un valor para cada columna.
- Si no se usan los nombres de las columnas entonces el orden de los valores deben corresponder con el orden implícito de las columnas, éste es determinado por el orden en el cual ellas fueron definidas cuando se usó el comando CREATE TABLE.
- Cuando se inserta un registro, se proporciona un valor para cada columna que no tiene un valor default definido.
- Un registro insertado a través del comando INSERT debe satisfacer cualquier restricción de la tabla o de la tabla fundamental de la vista.

Insertando múltiples registros

Para insertar más de un registro dentro de una tabla o vista protegida, se requiere el uso de una cláusula SELECT como parte de un comando INSERT.

```
Ejemplo:
>>INSERT  INTO EMPL1
+> (SELECT NUMDEPTO, APELLIDO, NOMBRE, NUMEMP, CODIGO SALARIO
+>  FROM EMPLEADO
+>  WHERE SALARIO > 100000.00
+>  ORDER BY NUMEMP );
```

La cláusula SELECT en este ejemplo es ejecutado como un ordinario SELECT, pero el resultado es copiado dentro de una tabla. Éste reemplaza a la cláusula VALUE usada en un comando sencillo INSERT.

Especificar tablas fuente y destino

Se debe especificar las columnas en cualquiera de las siguientes maneras:

- Definir los nombre de las columnas fuente, en el orden de la definición de la definición de la tabla destino. En este caso, los valores deben ser proporcionados para cada columna de la tabla destino.

- Especificar las columnas de la tabla destino, en orden de la definición de la tabla fuente (el orden en el cual serán proporcionada por una cláusula SELECT *…).

- Declarar las columnas de las tablas fuente y destino en el mismo orden. En este caso, no todas las columnas de la tabla destino necesitan ser nombradas; únicamente para las cuales la tabla fuente proporciona un valor. Estas columnas que no son nombradas deben tener un valor default definido.

Colocación de un registro en una tabla

NonStop SQL controla cómo lo registros son colocados dentro de la tabla y cómo esto depende del tipo de tabla.

INSERT…*lista-columnas*	Tabla ordenada por llave	Tabla relativa	Tabla ordenada por entrada
(…,…) Referencia a syskey no permitida	Basada en valores dado o valores default para las columnas de llave primaria	Opción APPEND.- Agrega el registro al final de La tabla. Opción ANYWHERE.- Coloca el registro en el primer espacio disponible	Agrega el registro al final de la tabla.
(*)	Basada en valores dados o valores default para las columnas de llave primaria.	Opción APPEND.- Agrega el registro el final de la tabla. Opción ANYWHERE.-	Agrega el registro al final de la tabla.

		Coloca el registro en el primer espacio disponible	
(*, Syskey) o (Syskey, *)	Operación no permitida	Coloca el registro en el número relativo indicado por la syskey	Operación no permitida.

3.5.2 Utilidades de la base de datos

Permisos de Acceso

Los tipos de permisos que se deben de poseer para trabajar con las utilidades son las listadas a continuación.

- Permiso de lectura a objetos o archivos que se está copiando.
- Permiso de lectura a los catálogos en los cuales los objetos son descritos (únicamente DUP y CONVERT).
- Permiso de escritura al catálogo en el que los objetos serán descritos (únicamente DUP y CONVERT).

Sintaxis de la expresión

Cada uno de los comandos usados para transferencia de datos de una ubicación en la base de datos a otra, comparten una estructura de comando que se muestra a continuación:

>> Nombre-Utilidad archivo-origen, archivo-salida Opción-Utilidad

Donde:
>>.- Es el prompt del sistema
Nombre-Utilidad.- Puede ser DUP, COPY, CONVERT, LOAD
Archivo-origen.- Nombre del archivo origen
Archivo-destino.- Nombre del archivo destino
Opción-utilidad.- Puede ser: Para DUP: CATALOG ALLOWERRORS, LISTALL, SAVEALL, SOURCEDATE, TARGET, INDEX, VIEW. Para COPY: Control-Option, Un-Option, Out-Option, Display-Option, Move-Option. Para CONVERT; MOVENAME, CATALOG, COMMENT, DICTIONARY, FILE IS, LOAD/NO LOAD, PART/NO PART, SOURCE, VARCHAR/NO VARCHAR. Para LOAD: Control-option, In-option, Key-sequence-option, Move-option.

Utilidad DUP

La utilidad DUP es empleada para crear copias idénticas de grupos de tablas, vistas, archivos de programas y archivo ENSCRIBE. El comando DUP se asemeja al comando FUP DUP usando el sistema de archivos en disco ENSCRIBE. Aunque el comando DUP se asemeja al comando FUP DUP en función y sintaxis, no se puede utilizar FUP DUP para trabajar con objetos SQL.

La razón por la cual no se puede aplicar FUP DUP para copiar objetos NonStop SQL es que, NonStop SQL tiene un diccionario de datos activo. Cuando un comando DUP es ejecutado en SQLCI, éste se comunica con el administrador de catálogos, cuya función es actualizar las tablas catálogo. FUP no se comunica con el administrador de catálogos, debido a esto FUP DUP

no puede ser usado para copiar objetos NonStop SQL. Cuando se duplica una tabla base, esta descripción, particiones e índices son duplicados, tan bien como la vista protegida en la tabla.

```
Ejemplo:
>> DUP $DATA1.SALES.*,  $DATA2.*.*
+>          CATALOG $DATA2.CATL;
```

La palabra CATALOG especifica un catálogo existente donde el objeto destino será descrito. Los asteriscos son usados cuando el objeto destino tenga los mismo archivos que el objeto fuente.

Es importante recordar las siguientes reglas al maneje la utilidad DUP:

- Usar un asterisco si los objetos destino tendrán el mismo nombre que los objetos fuente.
- A cada archivo destino le son otorgados los mismos atributos físicos que los correspondientes al archivo fuente.
- Se debe especificar un grupo de archivos destino o la opción MAP NAMES.
- Cualquier COMMENTS, CONSTRAINS o estadística aplicada al archivo fuente también serán aplicadas al archivo destino.
- No hay protección TMF durante una operación DUP.

Formatos Opcionales

Muchos elementos se pueden agregar a la utilidad DUP para ayudar a especificar cómo se quiere realizar la operación de duplicar.

```
Ejemplo:
>>DUP ($DATA1.SALES.* FROM CATALOG $DATA1.SALES),
+>          MAP NAMES ($DATA1.SALES.* TO $DATA.SALES.* )
+>          CATALOG $DATA2.CATL;
```

FROM CATALOG es una alternativa para especificar la ubicación de objetos fuente o destino. Esta opción restringe la lista de los archivos que son descritos en los catálogos especificados. MAP NAMES es una alternativa para especificar la ubicación de objetos recibidos. Éste reemplaza la lista de grupo de archivos destino.

Ejemplo
>> DUP ($DATA1.PERSNL.*), $DATA2.TESTTABS.* CATALOG $DATA2.TESTCAT;
En este ejemplo se especifican ambos volúmenes y subvolumenes. El formato copia el volumen completo.

Utilidad LOAD

La utilidad LOAD es empleada para cargar datos de una tabla o archivo a otra tabla o archivo. La utilidad LOAD se asemeja al comando FUP LOAD usado con ENSCRIBE. Sin embargo, no se puede utilizar FUP LOAD para trabajar con objetos SQL. Esta utilidad es utilizada para introducir nuevos datos a una tabla o archivo destino vacío. La utilidad LOAD carga datos de:

- Un archivo ENSCRIBE a una tabla, escribiendo cada registro fuente al archivo destino como un registro.

- Una tabla NonStop SQL en un archivo ENSCRIBE, escribiendo cada registro fuente al archivo destino como un registro.

- Una tabla NonStop SQL a otra tabla NonStop SQL con cada registro fuente insertado a la tabla destino.

Cuando los datos son cargados a la tabla, los índices para esta tabla son automáticamente cargados. Cuando una tabla es cargada a otra tabla, ésta sobrescribe los datos existentes.

Los archivos ENSCRIBE son definidos en su propio diccionario, que no es el diccionario activo NonStop SQL y puede ser diferente para cada archivo (un diccionario puede describir varios archivos).

Cuando la carga un archivo ENSCRIBE, se usa la cláusula DICT SOURCE para identificar el subvolumen del diccionario y la cláusula SOURCE REC para nombrar rl registro de definición en el diccionario que será usado. Para cargar a un archivo ENSCRIBE, son cláusulas TARGET DICT y TARGET REC equivalentes.

Si son omitidas SOURCE DICT o TARGET DICT, la utilidad LOAD asume que el diccionario ENSCRIBE existe en el subvolumen default. Si las opciones SOURCE REC o TARGET REC son omitidas la utilidad LOAD asume que el archivo fuente/destino está en formato explotado, esto significa que los campos del archivo son colocados en el mismo orden que las columnas de la tabla, en las cuales toda la longitud variable de las cadenas de caracteres son expandidas al máximo y llenadas con espacios en blanco.

Una cláusula *opcion-move* puede ser definida al cargar datos entre un archivo ENSCRIBE y una tabla SQL. *Opcion-move* asocia el registro origen con los registros destino, así que LOAD transfiere datos de cada registro origen a su correspondiente registro destino. Las opciones son:

- MOVEBYORDER ON.- Carga datos por posición. Los datos son cargados del primer campo del registro origen al primer campo del registro destino, después el segundo y así, hasta que la operación sea completada.

- MOVE BY NAME ON.- Carga datos del origen al destino por campos con el mismo nombre.

- MOVE *nombre-origen,* TO *nombre-destino.-* Carga datos de cada campo origen nombrado a su campo destino especificado. Éste es usado cuando el campo destino tiene diferentes nombres del campo origen.

MOVEBYORDER es asumido por default si no se especifica la cláusula.

```
Ejemplo:
>>LOAD $DATA1.INVENT.PARTLOC, $DATA.*.PARTINV,
+>        MOVEBYORDER ON;
```

Aunque las columnas sean nombradas de diferente manera en las dos tablas, tienen la misma posición relativa, tipo de datos y tamaño. Para cada registro, los datos son movidos de la tabla PARTLOC a la tabla PARTINV, basadas en la posición de la columna (la cual es la misma) en cada tabla.

Utilidad COPY

La utilidad COPY es utilizada para copiar datos para y de archivos ENSCRIBE y tablas NonStop SQL, agregando el nuevo dato a cualquier dato existente en un archivo o tabla. También usada

para desplegar el contenido de una tabla o archivo. La utilidad COPY es diferente de la utilidad LOAD en dos aspectos:

- COPY puede ser usada con una tabla auditada, por lo tanto COPY puede ser empleada dentro de una transacción TMF. La consecuencia de cada transacción es que si una tabla puede llegar a ser corrompida.

- COPY retiene los registros existentes y agrega registros a la tabla. En contraste con la utilidad LOAD, la cual elimina todos los datos en la tabla y entonces inserta los registros.

Los índices para la tabla son automáticamente actualizados cuando los datos son copiados a la tabla. La opción move disponibles para LOAD también están disponibles para COPY.

```
Ejemplo:
>> COPY EMPLEADO, EMP2, MOVEBYNAME ON, ALLOWERRORS 99;
```

La cláusula ALLOWERRORS permite al comando continuar con el proceso, aún cuando ocurran errores. Ésto es usado cuando se requiere transferir grandes volúmenes de datos.

Utilidad CONVERT

La utilidad CONVERT es utilizada para convertir un archivo ENSCRIBE descrito dentro de un diccionario DDL a una tabla SQL, descrita en un catálogo. Cuando CONVERT es utilizada genera:

- Un comando CREATE TABLE de una definición de registro DDL y lo escribe un archivo EDIT.

- Un comando CREATE INDEX de cada archivo de llave alternativa y escrito en un archivo EDIT.

- Un comando de la utilidad LOAD para cargar los datos del archivo ENSCRIBE a la tabla y asociarla con el índice y escribir este comando en un archivo EDIT.

```
Ejemplo:
>> CONVERT RECORD EMPREC TO TABLE $DATA.PERSNL.EMPLEADO
+> MAP NAMES ( EMP0 TO XEMPNOM, EMP1 TO XEMODEP )
+> DICTIONARY PERDDL;
```

RECORD EMPREC es la definición en el diccionario del archivo ENSCRIBE. El diccionario se encuentra en el subvolumen default o en el subvolumen especificado por la opción DICTIONARY. MAP NAMES es para particiones y nombres de índices más que para columnas y campos.

3.6 Modificar el contenido de la base de datos

3.6.1 Actualizar registros en la base de datos

Si se quiere actualizar la base de datos en cualquier momento tendrá que ejecutarse un UPDATE. El comando UPDATE es un comando DML y es utilizado para actualizar valores en

uno o más columnas de una tabla, directa o indirectamente (a través de una vista protegida). La estructura del comando es:

>> UPDATE *nombre-objeto* SET *lista-valores* [WHERE *condición-búsqueda]*;

Nombre-objeto identifica la tabla o vista que se quiere actualizar.

Cada elemento de la lista-valores es una columna apareada con una expresión que puede ser evaluada (nombre-columna = expresión). El nombre de la columna identifica una columna a ser actualizada. No se puede repetir una columna. La expresión significa cualquier expresión SQL válida que no incluya cualquier función. En una expresión se puede referir a cualquier columna en el registro a ser actualizada, incluyendo la llave definida por el sistema. El valor el cual es el contenido antes de la actualización es usado para evaluar la expresión y determinar el nuevo valor.

WHERE condición-búsqueda especifica el criterio para buscar los registros a ser actualizados. Cuando se usa el comando UPDATE…SET…WHERE, los registros que satisfacen la condición de búsqueda son actualizados acorde con la especificación en la cláusula SET. Ésta es la forma general, utilizada para actualizar cualquier registro(s). Si la cláusula WHERE es omitida, todos los registros de la tabla son actualizados.

En un subquery de la condición de búsqueda no se puede hacer referencia a la tabla, vista o cualquier tabla fundamental de la vista que se esta actualizando. Para actualizar únicamente un registro en especial, se debe especificar en la condición de búsqueda de la cláusula WHERE un identificador, el cual es único. La selección de este identificado es muy importante ya que seleccionará el registro que deseamos.

```
Ejemplo:
>> UPDATE EMPLEADO SET SALARIO = SALARIO *1.05
+> WHERE NUMEMP = 225;
```

Cuando se hace un cambio global a la base de datos, se pueden actualizar múltiples registros al mismo tiempo. La cláusula opcional WHERE puede ser usada para especificar el grupo o grupos de registros a ser actualizados. Es importante notar que aunque se actualicen múltiples registros de una sola tabla, únicamente se puede actualizar una tabla por expresión.

```
Ejemplo:
>> UPDATE  EMPLEADO, SET CODIGO = 555, SALARIO = (SALARIO * 1.10)
+> WHERE APELLIDO LIKE "H%";

El siguiente ejemplo muestra cómo actualizar registros utilizando un subquery:

>> UPDATE PARTSUPP
+>              SET
+>                          QTY_RECEIVED = 0
+>              WHERE
+>                          PARTSUPP .SUPPNUM =
+>                                      (SELECT SUPPLIER.SUPPNUM
+>                                      FROM
+>                                                  INVENT.SUPPLIER
+>                                      WHERE
+>                                                  CITY LIKE "%DALLAS%");
```

Actualizando múltiples tablas
Cuando se necesita actualizar más de una tabla, hay que recordar que no es posible insertar o borrar más de una tabla en una expresión UPDATE sencilla. Esto está en contraste con expresiones join SELECT, las cuales permiten hacer referencia a más de una tabla. Se puede actualizar más de una tabla pero se debe usar más de una expresión.

3.6.2 Borrar registros de la base de datos

Se puede utilizar el comando DELETE…FROM…WHERE para borrar cualquier registro de una tabla. Este comando trabaja de manera similar que el comando UPDATE. La sintaxis es la siguiente: >> DELETE FROM *nombre-objeto* [WHERE *condición-búsqueda*]

Donde:
>> Es el prompt del sistema
DELETE FROM: Es el comando SQL
Nombre-objeto: Puede ser nombre de la tabla o vista
WHERE: Es la cláusula WHERE

```
Ejemplo:
Para eliminar un solo registro.
>> DELETE FROM EMPLEADO WHERE NUMEMP = 507;
```

Múltiples registros pueden ser eliminados de una tabla usando la cláusula WHERE, así como se emplea en el comando UPDATE.

```
Ejemplo:
>> DELETE FROM EMPLEADO WHERE SALARIO > 120000.00;
```

Empleo de la utilidad PURGEDATA

Se puede emplear la utilidad PURGEDATA para limpiar los datos de cualquier tabla no auditada. No se puede limpiar una tabla auditada, programas archivos SQL, vistas o cualquier catálogo. A fin de usar PURGEDATA para limpiar una tabla, se deben tener permisos de escritura a la tabla

y catálogo afectados. Para limpiar un archivo, se deben tener permisos de escritura sobre dicho archivo. Tampoco se puede incluir un comando PURGEDATA dentro de una transacción TMF. La sintaxis para el comando es la siguiente: >> PURGEDATA *lista-grupo-archivos opciones*;

La fracción lista-grupo-archivos especifica los archivos o tablas a ser limpiados. Cuando se limpia una tabla, PURGEDATA también limpia datos de los índices definidos en la tabla. Si se incluye el nombre de una partición secundaria en la lista, la partición no será limpiada. Sin embargo, si se incluye el nombre de la partición primaria en la lista, todas las particiones secundarias serán limpiadas.

Para poder ejecutar un PURGEDATA en una tabla auditada, la propiedad audit, tendrá que ser eliminada. Se puede emplear el comando ALTER TABLE para colocar el valor OFF a la propiedad AUDIT de la tabla a afectar. Después de eliminar los datos, se debe usar nuevamente el comando para cambiar el valor a ON de la propiedad AUDIT. Este paso es importante, ya que cuando la propiedad audit se encuentra con valor “on”, TMF considera que la tabla será una nueva tabla.

3.7 Control de transacciones

3.7.1 Transacciones en NonStop SQL

El término de transacción ha sido empleado para describir como NonStop SQL procesa el trabajo. Una transacción ha sido definida como una unidad lógica de trabajo que consiste en ejecutar una secuencia específica de operaciones. La secuencia de la transacción puede contener una o más acciones UPDATE. Sin embargo, una transacción no es necesariamente una sencilla operación de SQL, más bien, es una secuencia de varias operaciones que transforma un estado consistente de la base de datos a otro estado consistente, sin preservar necesariamente la consistencia de todos los puntos intermedios.

Una transacción es referenciada como un ciclo de vida, ya que cada transacción inicia, tiene un punto intermedio productivo y un fin. El ciclo de vida de una transacción es a menudo seguida de una nueva generación de transacciones. En cada transacción el usuario debe garantizar que cada una de ellas debe:

- Ser ejecutada exitosamente en su totalidad o
- No debe haber efecto alguno sobre la base de datos.

El control de este control de transacciones proviene de la cooperación entre NonStop SQL y el TMF.

Transacciones TMF

NonStop SQL hace uso de TMF para asegurar que todas las transacciones resultan en una base de datos consistente. Si se quiere asegurar la consistencia de los datos, hay que realizar los cambios dentro de una transacción TMF. Este tipo de transacciones verifica que una transacción finalice exitosamente y si no, regresa la base de datos al estado anterior que la transacción haya iniciado.

Cuando varias transacciones ocurren al mismo tiempo, es importante que cada transacción finalice sin interferencia de alguna otra transacción simultánea. Para asegurar que TMF proteja

la transacción hay que: auditar las tablas la cuales serán afectadas por transacciones y definir el inicio y termino de cada transacción.

Todas las tablas NonStop SQL son auditadas a menos que se indique lo contrario. Si no se inicializa la transacción, cada comando que modifique la base de datos es automáticamente definido en una transacción. Por consecuencia, TMF protege las tablas en la base de datos de transacciones incompletas, a menos que se especifique que se realice sin administración TMF de actualizaciones a las tablas.

Comandos DDL

Durante una sesión de SQLCI, los comandos DDL son ejecutados en una transacción TMF. Las utilidades COPY, PURGE y SECURE también son ejecutadas dentro de una transacción TMF cuando operan en objetos NonStop SQL auditados. El inicio y término de una transacción de comando DDL es definida por SQLCI.

Comandos DML

Todas las operaciones de actualización a tablas auditadas o vistas de tablas auditadas, deben ser ejecutadas dentro de una transacción TMF. No es necesario definir las transacciones a menos que se quiera ejecutar una transacción consistente en varias operaciones.

Si la opción AUTOWORK se encuentra con valor ON, SQLCI automáticamente definirá una transacción para cada comando DDL. Si no se quiere que SQLCI defina las transacciones automáticamente, se puede establecer la opción AUTOWORK con valor OFF. Las transacciones TMF no afectan a operaciones DML en tablas no auditadas.

Transacciones definidas por el usuario

Si se quiere asegurar que varios comandos se ejecuten exitosamente o ninguno de ellos lo haga, se puede definir una transacción para varios comandos usando los comandos BEGIN WORK Y COMMIT WORK. Se puede cancelar una transacción con el comando ROLLBACK WORK. SQLCI no define una transacción TMF para un comando DML que se introduzca durante una transacción TMF definida.

No se puede emitir los siguientes comandos en una transacción TMF definida por el usuario:

- Comandos DDL operando en objetos no auditados
- DUP, LOAD y PURGEDATA
- PURGE, cuando se esta eliminando sobre objetos no auditados.

Comando BEGIN WORK

El comando BEGIN WORK inicia una nueva transacción TMF. Habilita los comandos INSERT, UPDATE y DELETE para ser ejecutados bajo el control de una transacción que el usuario controla.

Comando COMMIT WORK

El comando COMMIT WORK indica el fin de una transacción TMF y realiza que todos los cambios a la base de datos sean permanentes. Establece el fin de la transacción en progreso y establece un nuevo estado consistente para la base de datos.

Comando ROLLBACK WORK

El comando ROLLBACK WORK también marca un fin de una transacción. Eliminará los cambios realizados desde donde el comando BEGIN WORK haya sido establecido. Éste aborta la transacción en progreso y regresa la base de datos a un estado consistente.

```
Ejemplo:
>>BEGIN WORK;
>>UPDATE EMPLEADO SET CODIGO = 200 WHERE NUMEMP = 5689;
--- 1 row(s) updated.
>> UPDATE EMPEADO SET NUMDEPTO = 3000 WHERE NUMDEPTO = 3500;
--- 3 row(s) updated.
>>ROLLBACK WORK;
--- Successful SQL operation
>>
```

En este ejemplo, ninguna de las actualizaciones realizadas a la tabla EMPLEADO fueron guardadas, indicándolo con el comando ROLLBACK WOK, para que estos cambios permanecieran, en su lugar se ejecutaría el comando COMMIT WORK.

Tablas Auditadas

Una tabla que es marcada para ser auditada por TMF es referenciada como una tabla auditada. Para determinar si una tabla es auditada, ejecutar el comando FILEINFO. Un código de archivo "A", indica que la tabla es auditada por TMF. Todas las tablas NonStop SQL son auditadas por default a menos que un usuario, específicamente, modifique esto cambiando la función audit al valor OFF.

```
Ejemplo:
Para modificar el valor de la función audit, se ejecutan los siguientes comandos.
>>ALTER TABLE EMPLEADO NO AUDIT;
>>ALTER TABLE EMPLEADO AUDIT;
```

3.7.2 Control de concurrencia

Cuando se usa un sistema de procesamiento en línea (OLTP) a la base de datos, es necesario que múltiples usuarios estén permitidos a acceder a ella. Es necesario también que dos usuarios diferentes no traten de realizar cambios al mismo registro, al mismo tiempo. Esto es referenciado como: Control de Concurrencia. Para complementar esta tarea, se debe usar un bloqueo para asegurar que otros usuarios no pueden obtener el dato que se está cambiando.

A través de este mecanismo de bloqueo, NonStop SQL coordina las operaciones a la base de datos que uno ejecuta con las operaciones ejecutadas por otros usuarios, para proveer un grado especifico de concurrencia. En operaciones DDL todos los bloqueos ocurren automáticamente. Para operaciones DML, se pueden aceptar las características default o en su defecto codiciarlas.

Características de Bloqueo:

Tamaño del bloqueo (granulosidad)

El tamaño o granulosidad de un bloqueo, se refiere a qué tanto de la tabla ha sido bloqueado, toda la tabla, un subgrupo de registros o un simple registro. Esto puede ser influenciado por el

comando LOCK TABLE y la propiedad LOCKLENGHT de la tabla. Un comando LOCK TABLE limita el acceso a toda la tabla o la tabla fundamental de una vista, hasta que se libere la tabla bloqueada.

La propiedad LOCKLENGHT de la tabla es usada para incrementar la granularidad bloqueada, esto es, reemplaza los bloqueos a registros individuales (los cuales comparten un valor en común para una partición de la llave primaria) con un bloqueo simple para el grupo. La propiedad LOCKLENGHT es una característica que únicamente puede ser modificada por el comando ALTER TABLE sobre la tabla afectando a todos los usuarios.

```
Ejemplo:
>>FILEINFO PERSNL.CONSTRNT, DETAIL;
$DATA1.PERSNL.CONSTRNT  9 Aug 2004, 19_03
            SQL CATALOG BASE TABLE
             CATALOG $DATA1.PERNSL
             TYPE K
...
             KEY (     COLUMN 0, OFFSET   0, LENGHT 34, ASC,
             COLUMN 1, OFFSET 34, LENGHT 30, ASC)
             LOCK LENGHT 34
```

Por definición de LOCKLENGHT como 34 caracteres, esta tabla puede tener todos los registros compartiendo el mismo valor TABLENAME bloqueado como uno solo, incluso que la llave primaria de la tabla está definida como TABLENAME y CONSTRAINTNAME.

Tipo de bloqueo (modo)

El modo de bloqueo tiene dos valores: SHARE y EXCLUSIVE. Bloquear una tabla o registro con un modo SHARE significa que se quiere permitir a cualquier otro a tener acceso al mismo tiempo. El bloqueo en el modo EXCLUSIVE significa que se quiere ser el único en tener acceso a la tabla o registro, para el momento se está garantizado el bloqueo hasta el momento en que se elimine este bloqueo.

El modo de bloque es influenciado por el uso de las cláusulas: IN SHARE MODE O IN EXCLUSIVE MODE del comando SELECT y el comando LOCK TABLE. Cualquier comando DML que afecte al contenido de la base de datos (INSERT, DELETE o UPDATE) automáticamente buscará un bloqueo exclusivo.

Tipo de bloqueo de duración

Duración es el bloqueo de características que se refiere al tiempo en que un bloqueo es sostenido. Existe una cláusula en expresiones DML (INSERT, SELECT UPDATE, UPDATE) para especificar si son usadas BROWSE ACCESS, STABLE ACCESS O REPEATABLE ACCESS. Estas expresiones determinan la duración del bloqueo.

Browse Access:

Permite a cualquier usuario acceder a la tabla y leer cualquier dato, de hecho puede leerlos a través de cualquier bloqueo, únicamente se puede utilizar para leer datos. Los cambios (actualizaciones, inserciones o eliminaciones) no son permitidos. No adquiere un bloqueo mientras está accediendo a los datos, ni tampoco verifica si existen bloqueos antes de leer los datos. Los datos obtenidos pueden ser inconsistentes ya que existe la posibilidad de que estén siendo cambiados al mismo tiempo que se están leyendo.

Repeatable Access:

Permite a una transacción pretender como si fuera la única en tomar lugar en el sistema. Una vez que un registro está disponible, se puede actualizar y leer la información, sin importar que otras transacciones estén intentando modificar o leer el dato al mismo tiempo. Esto garantiza que el dato que se está accediendo no puede ser cambiado por otro usuario hasta que la transacción termine. Las transacciones pueden proseguir como si éstas fueran procesadas secuencialmente. Impide a otras transacciones o usuarios de actualizar, insertar o eliminar cualquier registro en el rango que se ha seleccionado.

Stable Access:

Garantiza que ningún otro usuario puede acceder a los datos que han sido actualizados desde que la transacción inició. Mantiene un candado en cualquier dato que se esté leyendo hasta que el próximo registro sea leído. Esta mantiene un candado en un registro que se esté modificando hasta que se complete la transacción. De todas estas opciones, stable access proporciona un nivel medio de consistencia y un nivel medio de concurrencia, cayendo entre browse y repeatable access.

Diferencia entre opciones

La diferencia entre usar stable access y repeatable access, para una operación de actualización, es que, en stable access otro proceso puede insertar un registro dentro del rango especificado por la cláusula UPDATE… WHERE. Usando repeatable access, otro proceso no puede insertar un registro dentro del rango especificado por la cláusula UPDATE…WHERE.

Controlando los candados

El comando CONTROL TABLE proporciona la habilidad de controlar los candados y abrirlos en tablas SQL seleccionadas o vistas. También afecta las decisiones del compilador al momento de ejecutar otros comandos SQL. El valor de CONTROL TABLE permanece hasta que se ingresa otro comando CONTROL TABLE que especifica la misma opción y cambia el valor previo.

La expresión de entrada del comando CONTROL TABLE esta dividida en dos partes, como se muestra en el ejemplo.

```
Ejemplo:
>> CONTROL TABLE nombre-objeto opción
```

Donde:
>>: Es el prompt del sistema
CONTROL TABLE: Es el nombre del comando SQL
Nombre-objeto: Puede ser nombre de una tabla, vista o *
Opción: Puede ser: RETURN/WAIT IF LOCKED, SYNCDEPTH =, TABLELOCK =, TIMEOUT =

Nota: El uso del asterisco indica todo, especifica todas las tablas que subsecuentemente se refiera en la sesión SQLCI

3.8 Ejecución de comandos SQLCI

3.8.1 Definiendo nombres

Un nombre definido proporciona una etiqueta conveniente a un valor o grupo de valores con características especificas. Cuando se crea un nombre definido, se debe escribir el tipo o clase del nombre y los valores que el nombre representará. Estos valores deben conformar condiciones que son descritas para cada clase de nombre definido. Típicamente para NonStop SQL los nombres definidos son creados para dos clases: para identificar un catálogo (donde el valor asociado es el nombre del subvolumen) y para identificar la tabla de una base o vista (donde el valor es el nombre completo de un archivo Tandem). Un nombre definido comienza con un signo de igual (=) y puede ser usado en la mayoría de los comandos.

Ejemplo: SELECT * FROM =NOMBTABLA;

Usando nombres definidos

Existe un par de casos en los cuales los nombres definidos son benéficos para SQLCI. Primero, cuando un archivo está siendo escrito, el autor puede no conocer dónde se ubicarán físicamente las tablas, cuando éste sea ejecutado. Por consecuencia, el autor emplea un nombre lógico para la tabla contenida en el archivo obey. Entonces, antes de que el archivo obey sea ejecutado, la ubicación de la tabla es establecida.

Los nombres definidos también son benéficos cuando un autor de un archivo obey quiere escribir comandos genéricos. En otras palabras un archivo obey será usado para diferentes tablas en varias ubicaciones físicas. Por consecuencias, antes el usuario reejecutará el archivo obey, el ejecuta cualquiera de los comandos SET DEFINE o ALTER DEFINE para apuntar a la tabla física a acceder.

En una forma similar, el programador puede usar el nombre definido en declaraciones SQL embebidas dentro de programas. Esta forma el programador puede apuntar a la base de datos de prueba durante el desarrollo y entonces, puede ser apuntada a la base de datos real, cuando el programa se instale. Esto puede ser hecho sin la necesidad de cambiar los nombres de las tablas codificados, dentro del programa y recompilar el código fuente del programa.

Controlar Definidos (DEFINEs)

Se puede elegir entre habilitar o deshabilitar los nombres actuales definidos. Esto es que, se puede desear parar el uso de nombres definidos y la creación de más nombres (los cuales pueden oponerse con el uso personal de un nombre particular). Para hacer esto, se emplea el comando SET DEFMODE, estructurado como se muestra:

```
SET DEFMODE [OFF] ;
            [ON]
            [ALL]
```

Donde:
OFF: Deshabilita todos los definidos (defines). No puede agregar defines, pero permite modificarlos. Este modo es empleado como configuración inicial.
ON: Habilita defines lógicos. Si se encuentra esta opción cuando se inicia sesión, ésta hereda los defines contenidos en la sesión TACL.
ALL: Lo mismo que se especifica en el modo ON

Se emplea el comando DEFMODE con ON o ALL, cuando se desea usar el comando ADD DEFINE o ejecutar comandos que contienen nombres definidos

Creando nombres definidos

Se emplea el comando SET DEFINE para establecer un grupo de trabajo o nombre clave y sus valores. El comando ADD DEFINE es empleada para asociar un nombre lógico (simbólico) con estas palabras claves y sus valores. El uso principal de nombres definidos en NonStop SQL es para establecer nombres simbólicos para las tablas, vistas y nombres de catálogos.

Para crear un nombre definido que puede ser usado donde un catálogo sea requerido, se emplea el comando SET y ADD tal como:

```
>>  SET DEFINE CLASS CATALOG, SUBVOL $DATOS.PERSONAL;
>> ADD DEFINE =CAT;
```

Para introducir un nombre definido que pueda ser usado dondequiera que un nombre de archivo pueda ser usado (esto es para una vista, ENSCRIBE o programa), se emplean los comandos SET y ADD tal como:

```
>>SET DEFINE CLASS MAP, FILE $DATOS.INVENTARIO.PARTLOC;
>> ADD DEFINE =INVENTARIO;
>>
```

En este ejemplo se ha creado un nombre definido (define name) de una clase particular (CATALOG en un caso y MAP en otro). El tipo de clase identificador (Ejem MAP) tiene características correspondientes (SUBVOL y FILE respectivamente) a las que corresponde el valor definido. Y finalmente, se encuentra el valor actual para el cual el nombre definido es únicamente un símbolo. El nombre definido, ahora puede ser usado en un comando SQL, por ejemplo:

```
>> SELECT * FROM =INVENTARIO
+> WHERE CODIGO LIKE "%P%";
```

El grupo de acción

El comando SET DEFINE establece un grupo de acción de atributos y valores que pueden ser cambiados únicamente por otro comando SET DEFINE. Esto significa que múltiples nombres definidos pueden ser creados con estos mismos atributos.

```
Por ejemplo:
>>SET DEFINE  CLASS MAP,
+> FILE $DATOS.INVENTARIO.PARTLOC;
>> ADD DEFINE =WAREHOUSE_INVENTORY_NY;
>> ADD DEFINE =WAREHOUSE_INVENTORY_MTL;
>>
```

3.8.2 Comando ALTER DEFINE

Modificar "defines"

Se puede emplear el comando ALTER DEFINE para cambiar los atributos de uno o más defines existentes. Usar este comando es equivalente a ingresar el comando DELETE DEFINE para eliminar un define lógico y entonces manipular el comando ADD DEFINE para agregar el DEFINE con nuevos valores. Considerar la siguiente secuencia de comandos:

```
>> ADD DEFINE =EMPTAB, CLASS MAP,
+> FILE PERSONAL.EMPLEADO;

>>SELECT * FROM =EMPTAB
+> WHERE LASTNAME LIKE "H%"
+> ORDER BY APELLIDOP;
- - - - - - - - - - - - - -
>>ALTER DEFINE =EMPTAB, CLASS MAP, FILE <nombre> TABL.EMPLEADO;
```

Para cambiar el valor asociado con un nombre definido, se usa el comando ALTER DEFINE como a continuación se detalla:

>> ALTER DEFINE nombre especificado

Donde:
>> Prompt del sistema
alter define Comando SQL
nombre Indica un nombre definido existente o =*, o **, especifica todos los nombres definidos.
spec especificación de atributo

El atributo spec es una palabra clave para una característica (como CLASS CATALOG O CLASS MAP) y este asocia valores

3.8.3 Comando INFO DEFINE

Se puede emplear el comando INFO DEFINE para desplegar los atributos y valores asociados con uno o más nombres definidos existentes. Esto significa que se debe especificar un nombre definido en el comando INFO DEFINE para ver sus atributos o usar uno de los nombres patrones (=*, o **) para especificar todos los nombres definidos.

```
>> INFO DEFINE ** ;

DEFINE NAME                 =CAT
CLASS                       CATALOG
SUBVOL       \EXT25.$DATOS.PERSNL

DEFINE NAME                 =EMPTAB
CLASS                       MAP
FILE                        $DATA1.<nombre_estudiante>TABL.EMPLEADO
>>
```

La sintaxis correcta del comando INFO DEFINE es: INFO DEFINE *lista_nombre_definido*

La porción *lista_nombre_definido* del comando se refiere al nombre definido o todos los nombres definidos (=*, **).

3.8.4 Controlando archivos OBEY

Se pueden establecer valores para parámetros que hacen referencia a comandos SQLCI usando el comando SET PARAM. Un parámetro es un nombre especificado en un comando para el cual se sustituye un valor cuando se está ejecutando el comando. En otras palabras, un parámetro es usado para especificar, hasta ahora, un valor desconocido en un comando SQLCI. Éste puede ser usado en cualquier parte que una cadena o literal numérica es normalmente usada en un comando SQL.

Típicamente, los parámetros frecuentemente usados en comandos SQL son almacenados en un archivo obey. Usando parámetros permite pasar un valor a un archivo obey, el cual se puede cambiar en el momento que se requiera.

Comando	Propósito
SHOW PARAM…	Despliega el valor del parámetro actual
SET PARAM…	Establece valores para parámetros
RESET PARAM…	Limpia el valor de uno o más parámetros
SAVE PARAM	Guarda todos los parámetros definidos por el usuario, como comandos “SET PARAMETER” a un archivo obey.

Cómo hacer uso de los parámetros

La forma de un nombre de parámetro es la siguiente: ?[nombre]

Para crear un archivo obey llamado SALRNG, el cual contenga el siguiente comando SELECT:

```
SELECT APELLIDO_M, NUMEMP FROM PERSNL.EMPLEADO
WHERE SALARIO BETWEEN ?SALARIO1 AND ?SALARIO2
```

Los signos de interrogación en el comando son usados para establecer valores para los parámetros, antes de ejecutar el comando.

```
Ejemplo:
>>SET PARAM ?SALARIO1 10000.00, ?SALARIO2 20000.00;
>> SHOW PARAM * ;
SALARIO1                    10000.00
SALARIO2                    20000.00

>> OBEY SALRNG ;
>> SELECT APELLIDO, NUMEMP
+> FROM PERSNL.EMPLEADO
+> WHERE SALARIO BETWEEN ?SALARIO1 AND ?SALARIO2;
APELLIDO                              NUMEMP
HERMAN                          201
DAY                                   203
CRAMER                          208
CHAPMAN                               209
JOSEF                                 224
KING                                  235
>>
```

Guardando valores de parámetros

Se pueden guardar los parámetros en su propio archivo obey, usando el parámetro SAVE PARAM *nombre-parámetro* TO *nombre-archivo*. En este caso, el comando SAVE se leería:

```
>> SET PARAM ?SALARIO 0;
>> RESET PARAM ?SALARIO2;
>> SHOW PARAM *;

SALARIO1      0

>> OBEY SALVALS;

>> SET PARAM ?SALARIO1    10000.00 < ---- almacenado como set command
>> SET PARAM ?SALARIO2    20000.00 < ---- almacenado como set command

>> SHOW PARAM*;

SALARIO1      10000.00
SALARIO2      20000.00
>>
```

Cuando se requiera de limpiar valores de uno o más parámetros, se empleará el comando RESET PARAM. La sintaxis de éste es: RESET PARAM *nombre-parámetro*

También se puede usar * para limpiar los valores de todos los parámetros. Después de limpiar un parámetro, éste no contiene valor alguno. Si se ejecuta un comando que haga referencia a limpiar el parámetro, un mensaje de ERROR puede aparecer. [12]

4 LENGUAJE DE COMANDOS AVANZADOS TANDEM (TACL)

El siguiente capítulo muestra la manera en que se podrá tener comunicación con el Sistema Operativo, es decir, empleando el Lenguaje de Comandos Avanzados Tandem (TACL) y la utilidad FUP, esto en un nivel básico, aunque más detallado que los capítulos anteriores.

Se mostrará al lector la forma de iniciar sesión al ambiente, generar un ambiente propio, trabajar con archivos, accesos directos a comandos, definición de teclas de función, control de procesos, desplegar información de un archivo en pantalla comandos de ayuda y otros comandos empleados.

Lo anterior con el fin de proporcionar a los usuarios las herramientas necesarias que muestran las características y capacidades que podrán ser empleadas en cualquier momento dentro del ambiente.

4.1 Introducción al TACL

Propósito de los comandos TACL

Los comandos TACL permiten la comunicación con el Sistema Operativo NonStop Kernel. El sistema operativo NonStop Kernel internamente guarda o vigila el sistema entero para asegurar protección contra usuarios no autorizados, seguridad de archivos y proporcionar validaciones contra entradas erróneas.

Todos los comandos TACL son introducidos después del símbolo prompt TACL (>). Este prompt aparece en la pantalla cuando se enciende la terminal. Un número aparecerá después del símbolo prompt TACL como se muestra a continuación: TACL 1 >

Inicio de sesión

Al iniciar sesión al sistema permite comunicar con TACL a la terminal. El propósito del comando LOGON es identificar a un usuario válido en el sistema. El comando LOGON es ilustrado a continuación.

TACL 1 > LOGON nombregrupo.nombreusuario

Los números en el prompt de cada comando inician desde el comando LOGON. Cuando se cierra sesión del sistema el prompt regresa al paso 1.

Nombre del grupo y nombre de usuario

Un grupo y nombre de usuario son asignados para cada usuario, los cuales están compuestos por ocho caracteres alfanuméricos (sin espacios) comenzando con una letra. Ambos nombres son usados como parte del comando LOGON. El nombre del grupo es típicamente el departamento o grupo de trabajo y el nombre de usuario es normalmente el nombre de la persona. Cuando se emplea el comando LOGON se captura el nombre del grupo y nombre de usuario como se mostró anteriormente.

Alias

Un usuario alias puede ser creado para comparar el formato del nombre de usuario en otros sistemas, o simplemente para recordar fácilmente el nombre de usuario. El usuario alias puede tener más de 32 caracteres de longitud. Se pueden agregar mayúsculas, minúsculas y números, el nombre también puede contener los caracteres punto (.), guión (-) y guión bajo (_). Un alias comenzará con un número o carácter. El comando LOGON con un usuario Alias se muestra a continuación:

```
TACL 1> LOGON Nombre_Alias_124
```

Passwords

Un password es una cadena de más de ocho caracteres alfabética, numérica o una combinación de ambos. Los espacios o símbolos no están permitidos. Después de capturar el comando LOGON, el sistema despliega un prompt para password como se muestra a continuación:

```
TACL 1 > LOGON Nombregrupo.Nombreusuario
Password:
```

Si aún no ha sido asignado un password, presionando la tecla enter, permitirá acceso al sistema. Si un password ha sido asignado anteriormente, el prompt del password aparecerá. Por razones de seguridad, el cursor no se moverá y el password no será desplegado en la pantalla.

Comando PASSWORD

Después de iniciar sesión exitosamente, se puede establecer un password empleando el comando PASSWORD, seguido de una cadena de más de ocho caracteres. Los passwords son sensibles a mayúsculas y minúsculas, por lo tanto, cuando se haga uso de él se tendrá que capturar como se estableció, con mayúsculas y minúsculas. El siguiente ejemplo, establece la palabra SECRETO como un nuevo password.

```
2 > PASSWORD SECRET
```

Una vez que se ha cambiado el password, este tendrá efecto al uso del siguiente comando LOGON. Se puede cambiar el password en cualquier momento por medio del comando PASSWORD. Algunos usuarios de ambiente pueden tener passwords previamente definidos y protegidos, en esos casos no se podrá hacer uso del comando.

Cerrando sesión del sistema

Cuando se deja la estación de trabajo se debe cerrar sesión del sistema a fin de evitar que alguien obtenga acceso no autorizado al sistema. El comando LOGOFF se ilustra a continuación

```
7 > LOGOFF
SEE YOU LATER, USER
16 NOV 20004, 18:04
TACL 1 >
```

4.2 Terminología de archivo

Componentes de archivos

Existe una terminología básica de archivos que cada usuario de TACL[24] debe conocer. Una manera de describir como son usados los archivos en el Sistema Tandem es compararlos con los archivos de un sistema no computarizado.

Un archivo es un grupo de registros de datos relacionados o de información. En el sistema estándar de archivos, la información es colocada en un fólder. El fólder es colocado dentro de un cajón específico de un gabinete de archivos. Por lo tanto, típicamente se tienen tres principales componentes cuando se archivan datos: el gabinete de archivos, el cajón y el fólder. Para encontrar información se debe conocer en qué gabinete y cajón buscar el nombre del fólder.

En comparación con el sistema de archivos Tandem, los cuales consisten en tres partes: el volumen (volume) se refiere al disco físico y puede ser comparado con el gabinete de archivos; el subvolumen (subvolume) es un área específica dentro de un volumen de disco y es similar al cajón; el nombre del archivo, funciona como un fólder dentro del cajón.

Con el Sistema de Archivos Tandem, se debe conocer el nombre del volumen, subvolumen y archivo con el fin de encontrar información. El nombre del volumen es asignado por el administrador del sistema y siempre comienza con un signo de pesos ($). Se puede crear volúmenes y archivos como se necesiten. Un ejemplo de un archivo se muestra a continuación:

$Gabinete.Cajon.Folder
(Volumen) (Subvolumen) (Nombre de archivo)

Nótese que están separados por un punto. A menudo se puede hacer referencia al archivo únicamente con el nombre del archivo. Un ejemplo que hace referencia a un archivo usando un editor de texto es: 3 > TEDIT NombredeArchivo

Cuando se captura el nombre de un archivo sin el nombre del volumen y subvolumen, el sistema busca el archivo en el volumen y subvolumen actuales.

Reglas para nombrar archivos

Las reglas para nombrar un archivo son las siguientes:

1.- Cada parte del nombre de un archivo debe estar compuesto, por lo menos, de un carácter, pero no más de ocho caracteres alfabéticos o números. Los espacios y caracteres especiales no son permitidos (excepto el primer carácter en el sistema y nombre de volúmenes).

2.- El primer carácter debe ser una letra. El nombre de volumen comienza con un signo de dólar ($) y nombres del sistema comienzan con una diagonal (\).

3.- No se pueden tener dos archivos con un mismo nombre en un mismo subvolumen.

4.- Los nombres de archivos no son sensibles a mayúsculas y minúsculas. Si un archivo ha sido creado con todos los caracteres en mayúsculas, se puede acceder a él usando caracteres en minúsculas.

[24] TACL: Tandem Advanced Command Language

Comando FILES

El Comando FILES es usado para visualizar los nombres de los archivos contenidos en uno o más volúmenes o subvolumenes. Ejemplo:

1.- Para listar todos los archivos en el subvolumen actual, se captura el siguiente comando en el prompt del TACL:

```
1 > FILES
```

Si no se indica el nombre del volumen o subvolumen en el comando FILES, se visualizarán los archivos en el volumen y subvolumen actuales.

2.- Para listar todos los archivos de todos los subvolumenes del volumen actual, se captura:

```
2 > FILES *
```

Defaults

Cuando se inicia sesión, se está asignado automáticamente a un sistema, volumen, subvolumen y archivo de seguridad default. Estos pueden ser modificados con el comando DEFAULT y tomarán efecto la próxima ocasión en que se inicie sesión.

El siguiente ejemplo modifica el volumen y subvolumen default:

```
6> DEFAULT $TRAIN.JUAN
The DEFAULT sys-vol-svol HAS BEEN CHANGED TO $TRAIN.JUAN
7>
```

Comando VOLUME

Los archivos que se generan son colocados dentro del volumen y subvolumen actuales. Los usuarios tienen ciertos volúmenes en el sistema para almacenar cierto tipo de archivos. El comando VOLUME es empleado para cambiar a un nuevo o diferente volumen y subvolumen o regresar al volumen y subvolumenes default.

Nota: Es posible desplegar el volumen y subvolumen actuales capturando el comando SETPROMPT como se muestra a continuación:

```
2>SETPROMPT BOTH
$TRAIN PAT 3 >
```

El comando SETPROMPT despliega en el volumen y subvolumen actual hasta que se cierra sesión. Se puede regresar al prompt original de TACL en cualquier momento capturando el comando SETPROMPT NONE.

En el siguiente ejemplo se observa como son cambiados el volumen y subvolumen actuales:

```
$AUDIT.ALAN:
$TRAIN PAT 3 > VOLUME $AUDIT.ALAN

$AUDIT.ALAN 4>
```

El volumen ALAN es el volumen actual en el sistema.

```
$AUDIT NEWSUB 5 > VOLUME
$TRAIN PAT 6>
```

En el siguiente ejemplo regresa al usuario el volumen y subvolumen iniciales.

Nota: En caso de capturar algún comando de manera incorrecta, el sistema responderá con un mensaje de error. Este mensaje de error indica que no ha reconocido como valido el comando.

4.3 Ambiente TACL propio

Para conocer los recursos que han sido asignados, se puede emplear el comando. Permite visualizar la información detallada sobre el ambiente actual. El siguiente es un ejemplo de lo visualizado al momento de capturar el comando WHO:

Home Terminal: $Z214A1
TACL PROCESS: \CASG.$D170
Primary CPU 3 (NSR-M) Backup CPU: 2 (NSR-M)
Default Segment File: $SOFTED.#3534
    Pages allocated : 40 Pages Maximum: 1024
    Bytes Used: 68348 (3%) Bytes Maximum: 2097152
Current Volume: $DEVEL2.LBUC
Saved Volume: $SOFTED.BROWNING
Userid: 24,206 Username: APPLIC.GEOFF Security: "OOOO"

Lo visualizado por el comando WHO es lo siguiente:

Home Terminal:	La terminal en donde se inició sesión
TACL Process:	Nombre del sistema y proceso del proceso TACL
Primary CPU:	Número de CPU en donde el proceso primario TACL está siendo ejecutado y el tipo de CPU.
Backup CPU:	Número de CPU en donde el proceso backup TACL está siendo ejecutado
Default Segment File:	Archivo de disco creado temporalmente por TACL para almacenar las variables generadas.
Pages Allocated:	Número de páginas y bytes de memoria que TACL ha empleado para almacenar información
Bytes used:	Estimado de memoria empleada
Current Volume:	El volumen y subvolumen actuales
Saved Volume:	El volumen y subvolumen defaults cuando se ha iniciado sesión
Userid:	Id de usuario para Guardian (Nombre del grupo, número de usuario)
Username:	Nombre del usuario para Guardian (nombre del grupo, nombre de usuario)
Security:	Opciones default de seguridad

Opciones de seguridad

Existen cuatro opciones de seguridad (RWEP) disponibles para cada archivo. Estos son definidos como sigue:

- **READ.-** Indica quién puede leer el archivo.

- **WRITE.-** Indica quién puede escribir en el archivo.
- **EXECUTE.-** Indica quién puede ejecutar el archivo del programa.
- **PURGE.-** Indica quién puede eliminar el archivo.

Niveles de Seguridad

El propietario del archivo puede especificar uno de siete niveles de seguridad para cada opción de seguridad (RWEP):

- Únicamente super ID local.
O Únicamente el propietario local.
G Miembros locales del grupo Guardian al que pertenece el propietario.
A Usuario local.
U Usuario de red quién es el propietario (Éste es el usuario de un archivo local quién requiere tener acceso al archivo de manera remota).
C Miembros de una comunidad de usuarios de red (Por ejemplo, el personal de un departamento que estén localizados en diversas oficinas remotas y deban tener acceso a archivos almacenados en sus oficinas centrales).

Sistema de Usuarios

Los usuarios deben tener un ID para acceder al Sistema Tandem. Este ID está compuesto por un nombre del grupo, nombre de usuario y número de usuario. Los accesos a todos los archivos en el sistema son controlados por estos identificadores. La estructura del grupo y usuarios es jerárquica.

Un usuario (Ejemplo, 10,100) puede trabajar bajo un grupo de administradores (ejemplo, 10,255). Esto permite al administrador tener acceso a cualquier archivo del cuyo propietario está en el mismo grupo. El ID del usuario más poderoso del sistema es el 255,255, conocido como super.super o el super ID. El super ID puede abrir o eliminar cualquier archivo del sistema.

Comando USERS

El programa USERS provee información acerca de un simple usuario o de un grupo de usuarios en el sistema.

El siguiente ejemplo ilustra el uso del programa USERS.

1.- Para visualizar información acerca del usuario que inició sesión, se captura:

```
2>USERS
GROUP.USERS  I.D. #                  SECURITY          DEFAULT VOLUMEID
BILLS .PAT                 024,026   NUNU              $ACCTS.SMITH
```

La información visualizada es la siguiente:

Group: El nombre del usuario del grupo Guardian
User: El nombre de usuario
ID#: Número del grupo y usuario
Default Volume: Volumen y subvolumen default configurados

2.- Otra opción para listar los nombres de usuarios y ID de usuarios del grupo al que se pertenece, se observa a continuación:

```
3>USERS *
GROUP.USERS    I.D. #    SECURITY      DEFAULT VOLUMEID
APPLIC  .MARY  024,002   AAAA          $SOFTED.MARY
APPLIC  .SUE   024,003   NNNN          $USERS.SUE
APPLIC  .PAT   024,255   NUNU          $USERS.PATMGR
```

Se puede localiza un usuario por medio de su ID.c Si se conoce el ID del usuario y se desea conocer el nombre del usuario, se captura lo siguiente:

```
4>USER 25,206
GROUP.USERS    I.D. #    SECURITY      DEFAULT VOLUMEID
DEPTA.LYNN     025,206   CUCU          $WORK1.PARKS
```

4.4 Trabajando con archivos

El comando FILEINFO

Este comando permite visualizar la información detallada de los archivos, tal como: quién es el propietario del archivo, fecha de última modificación y tamaño del archivo.

Para obtener la información de todos los archivos almacenados en el volumen y subvolumen actual, se escribe:

```
>> FILEINFO
$TRAIN.JANE
          Code    EOF    Last Modification      Owner   RWEP        PExt    Sext
FILE      101     5324   2-Jan-04 16:05:18      24,206  "OOOO"  2   2
NEWS      101     3060   3-Dic-04 8:45:21       24,206  "OOOO"  2   2
PGM1      101     2251   9-Feb-04 10:14:16      24,206  "OOOO"  2   2
```

FILEINFO lista los nombres de los archivos actuales con la siguiente información:

- ***Code*.-** Los archivos de programas y archivos editables tienen asignados diferentes códigos. Los archivos editables tienen un código 101.
- ***EOF.-*** El tamaño del archivo en bytes.
- ***Last Modification.-*** La fecha y hora de la ultima modificación del archivo.
- ***Owner.-*** El ID del usuario propietario del archivo.
- ***RWEP.-*** La seguridad asignada al archivo. Identifica quién esta permitido a leer, escribir, ejecutar y eliminar los archivos.

- ***PExt, SExt.-*** Los bloques primarios y secundarios en páginas.

El comando FILENFO permite listar archivos que comiencen con la misma cadena de caracteres los cuales son almacenados en múltiples volúmenes y subvolumenes. El comando FILEINFO es similar al comando FILES, pero permite la búsqueda en múltiples subvolumenes de archivos con nombre similares. El siguiente ejemplo muestra el uso del comando FILEINFO usando el símbolo asterisco (*): para localizar los archivos llamados CAP1, CAP2 y CAP3. Se puede visualizar todos los archivos en el sistema cuyo nombre comience con CAP usando el comando FILEINFO como se muestra: 1> FILEINFO $*.*.CAP*

El signo de pesos y el asterisco indica que la búsqueda se realizará en todos los volúmenes; el segundo asterisco indica la búsqueda en todos los subvolumenes, el nombre del archivo, seguido con un asterisco indica que el nombre del archivo comienza con CAP.

Comando RENAME

El comando RENAME permite cambiar el nombre de un archivo. Se puede cambiar el nombre del subvolumen, nombre de archivo o ambos. Sin embargo, no se puede renombrar un archivo en un volumen distinto.

Para renombrar un archivo en el volumen y subvolumenes actuales, se captura el RENAME seguido del nombre que se va alterar, un signo de coma y después el nuevo nombre del archivo. Ejem: 7> RENAME ARCH1, ARCH2

Para renombrar un archivo desde el volumen y subvolumen actuales a un subvolumen y nombre diferentes, se realiza como se muestra a continuación:

8>RENAME ARCH1, SUBVOL2.ARCH2

Para renombrar únicamente elsubvolumen se captura lo siguiente:

9> RENAME SUBVOL.FINAL, SUBVOL2.FINAL

Eliminando Archivos

Este comando permite borrar archivos del sistema de uno o más discos. Una vez que el archivo ha sido borrado, no se puede recuperar a menos que se haya realizo una copia del mismo. Para eliminar un archivo del volumen y subvolumen actual, se ejecuta el comando como sigue:

10> PURGE ARCHIVO.

Para eliminar múltiples archivos de manera simultánea se captura:

11> PURGE ARCH1,ARCH2,ARCH3,$SAVEIT.CLASS.SCHED

Todos los archivos son eliminados del disco. Nótese que los nombres de los archivos están separados por comas. También se puede eliminar un archivo de un volumen y subvolumen distinto al actual.

4.5 Accesos directos con TACL

Comando HISTORY

El comando HISTORY permite visualizar los comandos empleados recientemente, identificados por número o abreviación de comandos. Una vez desplegado el historial de comandos se puede volver a ejecutar cualquiera de ellos. El siguiente ejemplo muestra el uso del comando HISTORY.

Para visualizar los últimos 10 comandos empleados se captura:

```
12> HISTORY

3> FILES
4> WHO
5> PURGE FILEX
6> FILES
7> FILEINFO
8> RENAME arch1, arch2
9> USERS *
10> PURGE arch1, arch2
11> FILEINFO book
12> HISTORY

13>

Para visualizar únicamente los tres últimos comandos, se captura:

13> HISTORY 3

12> FILEINFO book
13> HISTORY
13> HISTORY 3
```

Comando !

El comando! (signo de exclamación) ejecuta comandos previamente empleados.
En el siguiente ejemplo el comando ! de la línea 3 vuelve a ejecutar el comando de la línea 1.

```
1>FILES
2>PURGE notas
3> ! 1
3> FILES
        FILES1              FILES2          FILES3          FILES4
4>
```

Nota: después de presionar la tecla enter en la línea 3, el comando de la línea uno es visualizado nuevamente. El comando **!** recuerda y vuelve a ejecutar comandos previos.

Puede ser empleado para recordar con pocos caracteres comandos largos, como se muestra a continuación:

```
4>FILEINFO book
5> WHO
6> ! FI
7> FILEINFO book
```

Comando FC

El comando FC es empleado para modificar la línea de un comando capturado anteriormente. Cuando se está empleando el comando FC se puede utilizar la barra espaciadora con el fin posicionar el cursor bajo el carácter a ser modificado.

```
Ejemplo:

1) 8> HITTORY (Presionar Enter)
2) 9> FC   (Presionar Enter)
3) 9> HITTORY
4) 9> . . S(Presionar Enter)
5) 9> HITORY
6) 9 . .              (Presionar Enter para ejecutar el comando correcto)
```

1.- Para cambiar un comando erróneo, presionar enter (una respuesta normal es un mensaje de error)
2.- Teclear FC en la línea siguiente y presionar ENTER
3- Nótese que el comando precedente contiene los errores capturados (comando erróneo)
4.- En adición otra línea también aparece, la cual contiene el número TACL (sin el símbolo del prompt) Esta línea es un prompt para capturar las correcciones del comando y presionar ENTER
5.- La corrección es desplegada para verificar si alguna otra línea requiere de alguna corrección más
6.- Al presionar enter en esta línea la cual no contiene correcciones, ejecuta el comando indicado.

Si se requiere realizar otra corrección en la ultima línea (9...) se puede volver a capturar el comando correctamente o se pueden emplear subcomandos FC los cuales se describen a continuación.

Los subcomandos FC

Los cuatro subcomandos que pueden ser capturados en conjunto con el comando FC son:

I = Inserta texto
D = Borra texto
R = Remplaza texto
// = Deja de remplazar, insertar o borrar

Si los subcomandos I, D o R no son usados, el reemplazo se asume automáticamente. Por ejemplo:

```
1) 10> FILEINSO
2) 11> FC
3) 11> FILEINSO
4) 11..         F
```

Los subcomandos pueden ser capturados en minúsculas o mayúsculas. Los subcomandos FC son capturados en la línea que comienza con puntos. Se puede usar la barra espaciadora o tecla de retorno para posicionar el cursor donde es encuentran los errores.

El siguiente ejemplo muestra el uso del subcomando FC **I**:

```
1) 12> VOLME
2) 13> FC
3) 13> VOLME
4) 13..      IU  (Presionar Enter)
5) 13> VOLUME
6) 13. .            (Presionar enter para ejecutar)
```

En el punto cuatro, el subcomando I, indica que los caracteres que le siguen serán insertados al comando en la posición indicada.

El siguiente ejemplo muestra el uso del subcomando FC **D**

```
1) 14> HISTORY 20
2) 15> FC
3) 15> HISTORY 20
4) 15..           DD
5) 15> HISTORY
6) 15>..                (Presionar enter para ejecutar el comando)
```

En el ejemplo el número 20 será eliminado. En la línea 15, el cursor es posicionado bajo el 20 y los subcomandos DD son capturados, indicando que esos serán los caracteres a eliminar.

El siguiente ejemplo muestra el uso del subcomando FC **R**

```
1) 16> USERS *
2) 17> FC
3) 17> USERS *
4) 17..          R22,001          (Presiona Enter)
5) 17> USERS 22,001
6) ...                            (Presiona enter para ejecutar el comando)
```

El ejemplo anterior no contiene errores pero puede ser empleado como base para una variación del comando USERS. El curso es posicionado hasta el carácter * , donde el subcomando R es capturado para remplazar los caracteres siguientes por los caracteres que le siguen a dicho comando

Se puede emplear más de un subcomando en una sola línea, siempre y cuando se usen las diagonales (//) para separar cada comando. El siguiente ejemplo ilustra el uso de los tres comandos D,R,I mientras se realizan múltiples correcciones a la línea.

```
1) 18> COMMMENT esto son un comentario
2) 19> FC
3) 19> COMMMENT esto son un comentario
4) 19..     D//              DRis//       Ie
5) 19> COMMENT esto es un comentario
6) 19. .
```

Programando Teclas de Función

TACL permite crear teclas de función simples y macros para ejecutar comandos TACL. Esto es muy útil para el uso frecuente de comandos como WHO, FILES, VOLUME, TEDIT, etc.

Definición de Teclas de Función

Las teclas que son usadas para ejecutar comandos TACL están numeradas por F1 a F16. También, las teclas pueden ser usadas en combinación con la tecla shift (SF1 a SF16), por lo tanto, hay 32 funciones posibles que pueden ser invocadas desde estas teclas. Nótese que la tecla F16 esta predefinida como la tecla de ayuda.

Para definir una tecla, se debe crear un archivo PS TEXT EDIT (TEDIT). Para crear un archivo nuevo, teclear TEDIT nomarch! En el prompt del TACL. Por ejemplo: 2> TEDIT mykeys!

Nota: Cuando se presiona la tecla enter, será presentada una pantalla en blanco en el editor de TEDIT. En este punto se puede hacer uso de las teclas de dirección y la tecla de retorno, para posicionar el cursor para la entrada de texto. Después de haber completado de editar texto se usa las teclas shift y la tecla de función F16 para salir del TEDIT.

Una vez que el archivo ha sido nombrado, se comienza a definir cada tecla de función. En el siguiente ejemplo, en la primera línea captura (en la columna 1 del archivo TEDIT) *?SECTION* seguido del nombre del a función elegida a usar y por la palabra MACRO. En la segunda línea, se captura el comando completo, el cual la macro ejecutará.

```
Ejemplo:
?SECTION nombre-funcion MACRO
```

Nota: El carácter **?** debe estar posicionado en la columna 1 y cada carácter debe estar separado de otro por un espacio.

La Definición de MACRO

Una macro es una predefinición de una serie de comandos. Está permitida para múltiples comandos y para el uso de variables (encerrado entre el carácter %). Por ejemplo, para definir la función F1 como el comando WHO, F2 como el comando FILES y F3 como el comando STATUS *, TERM, se captura lo siguiente en el archivo mykeys:

```
?SECTION F1 MACRO
WHO
?SECTION F2 MACRO
FILES
?SECTION F3 MACRO
STATUS *, TERM
```

Con una definición de una macro, se pueden indicar las variables que las teclas acepten. Estas variables pueden ser presentadas dos maneras:

1.- Se puede incluir el comando especifico en la definición, ejemplo:

```
?SECTION F4 MACRO
TEDIT
```

2.- Se puede incluir una variable en la definición, la cual es remplazada con un valor especificado cuando se ejecuta el comando. Cada variable es indicada por un número encerrado por signos de porcentaje (%). Por ejemplo:

```
?SECTION F5 MACRO
TEDIT %1%
```

Para ejecutar esta macro desde el prompt de TACL, se debe introducir el nombre del archivo que se desea editar y se presiona la tecla de función F5

Cargando archivo de definición de teclas

Una vez que se han definido las teclas de función, se sale del TEDIT, y se captura el siguiente comando en el prompt del TACL

```
3> LOAD / KEEP 1 / mykeys
```

KEEP 1, indica a TACL borrar cualquier versión anterior de la definición de teclas de función cargadas de la ultima versión del archivo mykeys. Si no se incluye KEEP 1, no se podrá hacer uso de la correcta definición de macros.

Visualizar definición de teclas

Una vez que se han definidos todas las teclas de función, se puede desplegar una lista de teclas de función y su definición usando el comando KEYS, como se muestra:

```
9> KEYS
F16  = (Tecla de ayuda)
F1   = WHO
F2   = FILES
F3   = STATUS *, TERM
F4   = TEDIT %1%
SF16 = LOGOFF
10>
```

Archivos Macro

Se puede definir comandos de una macro, de la misma manera en que se definen las teclas de función. La diferencia es que estos comandos son invocados tecleando algo más que solo presionar una tecla.
Se emplea el TEDIT para generar el archivo de la macro. Para los ejemplos se hace referencia al archivo mymacs

Dentro del archivo mymacs, se puede definir la letra “P” como una macro para el comando PERUSE, la letra “T” para el comando TEDIT:

```
?SECTION P MACRO
PERUSE
?SECTION T MACRO
TEDIT %1%
```

Una vez que se ha creado la definición que se requiere, se carga la macro de la misma forma en que se cargo en memoria las funciones para las teclas: 12> LOAD / KEEP 1 / mymacs

Carga automática de archivos macro

Para hacer uso de las macros, se debe emplear el comando LOAD cada vez que se inicie sesión, o use un archivo especial que automáticamente carga las macros cuantas veces se inicie sesión.

Para realizar esto se edita un archivo (con el comando TEDIT) que debe ser nombrado, TACLCSTM, en el volumen y subvolumen configurados como default: TACLCSTM es un archivo que ha sido creado automáticamente para este propósito, así que lo único que es necesario realizar en capturar la información dentro del archivo. Por ejemplo:

```
?TACL MACRO
==TACL created this file for your protection
LOAD / KEEP 1 / mykeys
LOAD / KEEP 1 / mymacs
```

Con esto, cada vez que se inicie sesión, TACL revisa un archivo nombrado TACLCSTM y ejecuta los comandos indicados en el archivo. Si se realizan cambios a cualquier definición una vez iniciada la sesión, se puede emplear el comando LOAD o en su defecto cerrar sesión y volver a iniciarla a fin de que se actualicen las nuevas definiciones.

Desde el archivo TACLCSTM se pueden incluir cualquier comando que se requiera tome efecto al momento de iniciar sesión.

4.6 Control de procesos con TACL

Comando STATUS

El comando STATUS que proporciona información acerca de procesos, incluyendo el nombre del proceso, ID del usuario asociado con el proceso, la ubicación del archivo del programa y la terminal en la cual el proceso fue iniciado.

El comando STATUS un número de opción de comando. Por ejemplo, para desplegar información acerca de todos los procesos ejecutándose en la terminal, se usa asterisco (*) para indicar todos los procesos y la palabra TERM para indicar la terminal:

```
1> STATUS *, TERM
Process  Pri      PFR   %WT   Userid   Program File                  Hometerm
$Z102    B 3,31   150         001      178,1   $SYSTEM.SYS06.TACL    $ZTNT.#T01
$Z102      5,31   150    R    001      178,1   $SYSTEM.SYS06.TACL    $ZTNT.#T01
```

La primera columna en lo desplegado por el comando STATUS es el nombre del proceso
La "B" después del nombre del proceso, indica un proceso respaldo para el proceso $Z102.
La próxima columna muestra el número de CPU donde el proceso se está ejecutando y el número de identificación del proceso (PIN) para el proceso en ese CPU (31)

Las otras columnas mostradas son:

Pri: Ejecución prioritaria del proceso
FR: Código que indica que el proceso contiene el código privilegiado (P), está esperando en una búsqueda errónea (F), o esta en listo(R).
%WT Valor octal que indica el estado de espera del proceso. Un valor de 000 indica que el proceso no está en espera.
Userid ID del usuario asociado al proceso
Program File Nombre del archivo del programa del cual el proceso fue ejecutado
Hometerm Nombre de la terminal en donde el proceso fue iniciado

El comando STATUS emplea las siguientes opciones para encontrar información acerca de los procesos:

- Iniciados por el usuario que inicio sesión: STARTED *, USER
- Iniciados por un usuario especifico: STATUS *, USER sales.mike
- Ejecutándose en una terminal especifica: STATUS*, TERM $T14
- Ejecutándose bajo una prioridad asignada: SATUS *, PRI 150
- Ejecutándose con un nombre especifico: STATUS $acctg
- Ejecutándose con ID de proceso especifico; STATUS 1,34
- Ejecutándose en un CPU específico: STATUS 2

El comando PPD

El comando PPD despliega nombres de procesos, ID de procesos y antecesores del proceso nombrado (procesos creadores). Se emplea el comando PPD, si se conoce el nombre de un proceso y se quiere encontrar su CPU y PIN (numero de identificación del proceso). PPD lista el nombre del proceso, el CPU y PIN del proceso de respaldo primario y el nombre o CPU y PIN de procesos antecesores. Si no se especifica un proceso PPD despliega información acerca de todos los procesos en el sistema.

Por ejemplo, supóngase que un proceso llamado $S está ejecutándose en el sistema, se puede encontrar la siguiente información capturando el comando:

```
1> PPD $S
Name        Primary     Backup      Ancestor
$S          1,26        0,24        $SPLS
```

La columna de antecesores indica $SPLS en proceso que generó $S. Con esta información se pude emplear el comando STATUS usando el nombre del antecesor $SPLS.

Comando STOP

El comando STOP se emplea para detener un proceso que fue iniciado incorrectamente o para detener un proceso que ya no es necesario. Este ejemplo detiene el proceso que tiene los números de CPU y PIN 2,29:

```
4> STOP 2,29
5>
```

Si no se especifica un proceso por el número con el comando STOP, TACL detiene el último proceso que se inicio. Se puede emplear el comando STATUS para verificar que el proceso ha sido detenido.

4.7 Programa de Utilidad de Archivos (FUP)

El programa de utilidad de archivos FUP[25] permite desempeñar muchas funciones involucrando archivos y dispositivos como una sola terminal. FUP es empleado para obtener "Información" de archivos, "DUPlicar" archivos, proporcionar (GIVE) archivos de un propietario a otro usuario y asegurar (SECURE) el acceso a los archivos propios,

Captura de comandos FUP a través de TACL

Se puede emplear FUP y el nombre de un comando FUP válido en el prompt TACL, como se observa a continuación:

```
2> FUP INFO SUBVOL.FILE
```

Al Capturar FUP se accede a la utilidad FUP. INFO es un comando FUP usado para desplegar información acerca de un archivo, y subvolumen. Información desplegada por el comando INFO es:

```
CODE   EOF    LAST MODIF     OWNER RWEP   TYPE   REC    BLOCK
$VOLUME.SUBVOLUME
FILE
3>
```

El comando INFO ha sido ejecutado en el archivo asignado. La ventaja de este método es que después de acceder al FUP y capturar el comando, el sistema automáticamente regresará al prompt del TACL

Captura de comandos FUP interactivamente a través de FUP

Otra manera de acceder a la utilidad FUP es capturar FUP en el prompt TACL:

```
3> FUP
```

Después de presionar ENTER, la utilidad FUP, despliega el encabezado del mensaje de la siguiente forma:

[25] FUP: File Utility Program)

```
File Utility Program – T6553D30 – (31OCT94) SYSTEM \XYZ
Copyright Tandem Computers Incorporated 1981, 1983, 1985-1995
-
```

El símbolo guión (-) es el prompt FUP que espera cualquier comando FUP válido.
Cuando un comando es capturado, ejecutado y concluido, la utilidad FUP permanece activa y a la espera de otro comando FUP. El siguiente ejemplo muestra dos comandos FUP

- INFO MIO
Este comando muestra la información del archivo MIO.
- SUBVOLS $SYSTEM
Muestra un lista de subvolumenes en el volumen $SYSTEM

La ventaja de este método es que la utilidad FUP permanece activa, así que se pueden capturar los comandos que sean necesarios sin tener que regresar al prompt del TACL después de cada comando. Para poder salir del prompt de FUP se captura el comando EXIT para poder regresar al prompt del TACL

4.7.1 Comando INFO

El comando INFO despliega las características de uno o más archivos. Para desplegar las características de todos los archivos en un volumen específico llamado $DATA y subvolumen llamado MISC, se captura lo siguiente: - INFO $DATA.MISC.*

Como parte del comando INFO, se puede introducir un volumen, subvolumen y un nombre de archivo. Un símbolo asterisco (*) puede ser usado para representar todos los archivos para el volumen y subvolumen asignados. Si el volumen o subvolumen no han sido asignados con el comando INFO, los datos de los archivos contenidos en el subvolumen y volumen actuales son desplegados.

La respuesta del comando, puede ser lo siguiente:

```
                CODE   EOF     LAST MODIF      OWNER RWEP   TYPE    RFC     BLOCK
$VOLUME.SUBVOLUME
DES             101    3424    13JUL04 2:21    24,10 AAAO
LABS            101    6$$0    1FEB$$ 1:0$     24,10 AAAO
```

Para desplegar las características de un archivo llamado MIARCH en el volumen/subvolumen actuales, se captura: - INFO MIARCH

Nota: Si se introduce el nombre de un archivo no existente, un error ocurrirá:

WARNING -$<VOL>.<SUBVOL>.<FILE>: ERR 11

Éste es un mensaje de advertencia, indicando que FUP no reconoce un volumen, subvolumen o nombre de archivo válido. Cuando se obtiene un ERR y un número agregado a él, se puede obtener su significado de la siguiente manera:

2> ERROR XX (Presionar enter)
(xx representa el número en el mensaje de error)

El comando INFO, despliega nueve columnas de información:

Columna 1	Nombre del archivo, especificado en el comando INFO
CODE	Código de archivo. Un archivo en formato modificable es un código 101. Un programa de archivo es un código 100.
EOF	Longitud actual del archivo en bytes
LAST MODIF	Fecha y hora de la última modificación del archivo. Si es el día actual, únicamente aparece la hora
OWNER	ID del usuario propietario del archivo
RWEP	Seguridad asignada al archivo,
TYPE, REC, BLOCK	Muestra información sobre archivos estructurados.

4.7.2 Comando COPY

Si se requiere de visualizar los datos o texto en un archivo, se puede emplear el comando COPY para desplegar su contenido en pantalla. Para copiar un archivo llamado MIO en pantalla se captura el siguiente comando en el prompt de FUP:

```
- COPY MIO
```

Al presionar enter, registro por registro son desplegados en pantalla. Si es un archivo grande, el contenido es desplegado continuamente, página por página hasta el fin del archivo. Para copiar el contenido de un archivo a una impresora se captura el siguiente comando:

```
- COPY MINE, $S.#PRINT3
```

4.7.3 Comandos DUP y GIVE

Comando DUP

El comando DUP realiza una copia de disco a disco de un archivo para propósitos de respaldo o para uso de mantenimiento general. Para duplicar un archivo se introduce DUP en el prompt de FUP, seguido por el nombre del archivo original (ARCH1 en el ejemplo) y el nombre para la copia (ARCH2), separados por una coma:

```
- DUP ARCH1, ARCH2
```

Después de presionar la tecla ENTER, una copia de ARCH1 será ubicado en el volumen y subvolúmenes actuales y este será nombrado ARCH2. Cuando se crea una copia de un archivo en un volumen y subvolumen distintos a los actuales, se debe especificar el volumen y subvolumen origen del archivo:

```
- DUP $VOL1.SUBVOL1.ARCH1, ARCH2
```

Si el archivo a duplicar se encuentra en el mismo volumen, pero en subvolumen distinto, únicamente se introduce el nombre del subvolumen y el nombre del archivo:

```
- DUP SUBVOL2.ARCH1, ARCH2
```

ARCH2 es creado en el subvolumen actual.
Múltiples archivos del subvolumen actual pueden ser duplicados dentro de diferentes subvolúmenes. Por ejemplo, en el prompt de FUP, introducir DUP y la lista de nombres de archivos a ser suplicados en paréntesis. (Separando con una coma los nombres de archivos y nombre del subvolumen destino). Para conservar los mismos nombres de archivos, se agrega un asterisco después del nuevo subvolumen: - DUP (ARCH1, ARCH2) , NVOSUBV.*

ARCH1 y ARCH2 son ubicados en NVOSUBV

El siguiente ejemplo duplica todos los archivos en el subvolumen actuales a otro subvolumen.

-DUP * , SUBVOL.*

Comando GIVE

El comando GIVE permite al usuario reasignar el propietario a otro usuario. El archivo permanece en la misma ubicación. El archivo no se mueve al volumen default del nuevo propietario, únicamente el ID del propietario cambia. Sin embargo, únicamente se debe emplear el comando GIVE cuando se este completamente seguro de que se quiere asignar la propiedad del archivo a otro usuario.
Después del prompt FUP, introduce GIVE seguido por el nombre del archivo cuyo propietario será modificado. Después introduce el ID del nuevo propietario, el cual consiste en numero del grupo y ID del usuario (la coma separa el nombre del archivo del ID del nuevo propietario.

Para reasignar un archivo al usuario cuyo ID es 24,206
- GIVE ARCH1 , 24,206

Otro ejemplo consiste en, después del prompt FUP introducir GIVE, e introducir el asterisco si todos los archivos será modificados de propietario (la coma separa el nombre del archivo del ID del nuevo propietario)I

Para reasignar todos los archivos del subvolumen actual al usuario con el ID 24,206 se emplea:
- GIVE * , 24,206

Para reasignar los archivos: MIARCH, MIPROG en el subvolumen actual al usuario cuyo ID es 24,206 se emplea: - GIVE (MIARCH, MIPROG) , 24,206

Nota: Únicamente el propietario actual del archivo, el administrador del grupo o súper ID pueden reasignar propietario de un archivo a otro usuario. Si un proceso tiene un archivo abierto cuando a este archivo se esta reasignado el propietario, el comando GIVE será ineficaz.

4.7.4 Comando SECURE

El comando SECURE permite a los usuarios, definir las opciones de seguridad para cualquiera de sus archivos. Como se explicó anteriormente existen niveles de seguridad (RWEP). El propietario del archivo puede especificar uno de los siete niveles de seguridad para cada una de las cuatro opciones de seguridad: - , O, G, A, U, C, N,

Cambiando Opciones de Seguridad

Para asegurar un archivo local llamado EJEMPLO permitiendo a cualquier usuario de la red leerlo y ejecutarlo y únicamente el usuario local pueda escribir sobre él y eliminarlo, se estructura de la siguiente forma:

- SECURE EJEMPLO, “NONO”

Después de introducir SECURE en el prompt, el nombre del archivo que se quiere asegurar. Una coma debe ser colocada después del nombre del archivo. Introduce las cuatro opciones, las cuales deben ser marcadas entre comillas (“ “). Para verificar el cambio introduce el comando INFO para este archivo y revisa la columna RWEP.

Para asegurar un archivo local llamado ARCH1, admitiendo a cualquier usuario para leerlo y únicamente un grupo de la comunidad o miembro de un grupo pueda escribir sobre él, el usuario de red (propietario) puede ejecutarlo, eliminarlo, se ejecuta lo siguiente:

- SECURE ARCH1, “NCUU”

Para cambiar la seguridad de dos archivos (ejemplo: DATOS1 y DATOS2) simultáneamente, de tal forma que cualquier usuario en el sistema local, pueda leer y ejecutar y cualquier miembro puede escribir o eliminarlo, se introduce:

- SECURE (DATOS1, DATOS2), “AGAG”

Para asegurar todos los archivos en un volumen llamado $TRAIN y subvolumen llamado JANE, de tal manera que el propietario local pueda leer, escribir, ejecutar y eliminar todos los archivos:
- SECURE $TRAIN.JANE.* , “OOOO”

4.7.5 Comando HELP

El comando HELP despliega una lista de todos los comandos FUP con su sintaxis correspondiente.

Ejemplos:
Para listar los nombres de todos los comandos FUP, en el prompt FUP introduce:

```
- HELP ALL
ALLOCATE        CREATE ALTER          COPY
CREATE HELP           DUP             EXIT
FILES           GIVE            REVOKE INFO
LISTOPENS       LOAD            SYSTEM PURGE
RENAMERESET           SECURE VOLUME
```

Para listar la sintaxis de un comando en particular, se introduce HELP seguido por el nombre del comando como se muestra:

```
- HELP INFO
INFO [ / OUT <list file/] [< fileset list > [ , <info option> ] . . . ]

<info-option>
  { DETAIL                              }
  { EXTENS                              }
  { STAT[ISTICS] [, PARTONLY        ]   }
  {                [, PARTIAL <pct>     ]   }
  { USER [<USERID>  | <USERNAME>]   }
```

[13].

CONCLUSIONES

El presente trabajo muestra información básica y avanzada de la plataforma Tandem, dicha información puede ser en partes demasiado técnica, sin embargo, debido a que este trabajo esta orientado a personas con conocimiento de sistemas, se parte desde un punto en que no debería ser necesario enseñar cuestiones básicas de computación.

Así mismo, este trabajo, es presentado al lector, como una oportunidad de aprender el manejo de una plataforma cuyo uso es mayormente a nivel empresarial para la administración de grandes volúmenes de información.

La visión presentada, no es únicamente desde el punto de vista del administrador del sistema, si no también, como usuario del mismo, y así poder tener el conocimiento y sensibilidad necesaria para poder hacer uso correcto de la explotación y manejo de la información y poder cumplir de una manera más fácil los objetivos del negocio.

Se mostró la información en los anteriores capítulos de tal manera, que el lector entienda de primera instancia la arquitectura de la plataforma, desde seguridad de la misma, arquitectura del sistema operativo, procesamiento de transacciones en línea al entendimiento de las bases de datos relacionales en un primer capitulo. Dentro del segundo capitulo se muestran los servicios que nos proporciona así como el manejo de la seguridad del sistema dentro del manejo de los grupos de usuarios, así como los perfiles de dichos usuarios y poder así controlar el acceso a la información sensible. En el tercer capitulo se muestra la manera en que se podrá explotar la información contenida en la base de datos a través de la herramienta SQLCI y sus distintos comando. Como ultimo capitulo se muestran los distintos comando avanzados de Tandem cuyo empleo adecuado nos podrá permitir la comunicación con el sistema operativo, manejo y explotación de los distintos archivos y tipos de archivos así como la seguridad de los mismos según el perfil del usuario que tiene acceso a ellos.

La Información alrededor de la plataforma Tandem es demasiado extensa, a pesar de que el presente trabajo, es amplio, no se puede cubrir toda la información, y tampoco es el objetivo del mismo. Por esto mismo, se intenta cubrir toda la información necesaria, para el entendimiento general de la plataforma y proporcionar así, las herramientas necesarias al lector, para una ampliación del presente trabajo y continuar con la investigación que se inicia con el presente trabajo.

Existen demasiados temas con los cuales se puede continuar, entre los que se encuentra la administración y monitoreo de procesos, programación con SQL embebido en lenguajes como C, Java, Cobol. Cada uno de estos temas, son bastante amplios si se tocan a fondo, ya que existen empresas que emplean uno o varios de estos lenguajes, para la creación de programas cuyo uso automatiza la explotación de la información de la organización.

GLOSARIO

Lista de Control de Acceso (ACL – Access Control List).- Habilidad de SAFEGUARD que permite la restricción al acceso de objetos del sistema.

Asíncrono.- El protocolo asíncrono está orientado al carácter, cada byte que se transmite se rodea de un grupo de bits necesarios para la comunicación. Cuando no hay datos que transmitir la línea o salida del transmisor se encuentra en el estado lógico 1.

Backup.- La utilidad BACKUP copia los archivos de disco a una cinta.

COBOL (Common Business Oriented Languaje).- Un standard para la industria del lenguaje, diseñado mas para uso comercial que para aplicaciones científicas.

Programa de Compresión de disco (DCOM) Una utilidad que automáticamente se consolida fragmentando espacio del disco moviendo archivos asignados a un espacio libre de diferentes ubicaciones en otro volumen de disco.

Programa de análisis de espacio libre en disco (DSAP) Una utilidad que analiza como el espacio en un volumen de disco es usado.

ESNCRIBE.- Archivo del sistema que permite el acceso y manipulación de os registros de una base de datos relacional.

EXPAND.- Ligas geográficamente distribuidas de sistemas tandem para crear una red que tenga la misma fiabilidad, capacidad para conservar la integridad de la base de datos y potencial para expansión modular como un solo sistema.

Programa Utilitario de Archivo (FUP).- Permite realizar una variedad de operaciones en archivos de disco.

GUARDIAN 90.- El sistema operativo de los 90's para sistemas Tandem NonStop.

IEEE.- Instituto de ingenieros eléctricos y electrónicos.

INSTALL.- Programa utilitario que brinda el sistema hasta el nivel actual de la revisión del programa, crea una nueva imagen del sistema operativo y mantiene una instalación de la base de datos.

LAN.- Red de área local: Sistema que interconecta computadoras, procesadores de texto y otras maquinas electrónicas de oficina para crear una red entre oficinas

MULTILAN.- Un producto de hardware y software que permite a los usuarios a conectar su red de área local (LAN) al sistema Tandem NonStop.

NonStop SQL.- Lenguaje de consulta estructurado. Sistema administrador de base de datos relacional que promueve el acceso eficiente en línea en base de datos distribuida grande.

NonStop VLX.- Una computadora mainframe de Tandem.

OLTP.- Procesamiento de transacciones en línea.- Procesamiento de datos que implica acceso inmediato y actualización de base de datos mientras el usuario espera para los resultados. La base de datos es constantemente actualizada y refleja el estado actual del negocio.
PATHCOM.- Comando de interfase del PATHWAY.

PATHMON.- Proceso que supervisa y controla todos los procesos y dispositivos de red en un sistema PATHWAY.

PATHWAY.- Software que abastece los programas y ambiente operativo requerido para el desarrollo y ejecución de aplicaciones de procesamiento de transacciones en línea.

PUP.- Utilidad que permite a un usuario observar y modificar el estado de un dispositivo periférico del sistema Tandem.

Programas PROGID.- Un programa que permite a un usuario que utilice temporalmente los privilegios de otro usuario

SQLCI.-Interprete de comandos de NonStop SQL.

TACL. - Tandem Advanced Command Language: Interfase de Tandem al sistema operativo GUARDIAN 90. Este también tiene muchas otras características incluyendo lenguaje de programación.

VIEWSYS.- Utilidad interactiva que monitorea recursos del sistema mientras el sistema de ejecuta.

BIBLIOGRAFÍA

[1] http://www.hpl.hp.com/techreports/tandem/TR-85.7.html Why Do Computers Stop and What Can Be Done About It? - Gray, Jim

[2] Disk and Tape Utilities Reference Manual, part no. 15355-C20, Tandem Computers, Incorporated.

[3] Distributed System Management (DSM) Installation and Operation ISP, part no. 18486, Tandem Computers, Incorporated.

[4] Data Communications Concepts Course, part 12851, Tandem Computers Incorporated.

[5] Introduction to PATHWAY, part no. 82339-B10, Tandem Computers, Incorporated.

[6] http://www.hpl.hp.com/techreports/tandem/TR-81.5.html Relational Data Base Management for On-Line Transaction Processing - Schuster, Stewart A.

[7] Relational Database Desing Course, part no. 13997, Tandem Computers, Incorporated.

[8] Introduction to the Transaction Monitoring facility (TMF), part no. 15996-C20, Tandem Computers, Incorporated.

[9] Security Administration Guide, part no 11535-C1, Tandem Computers, Incorporated.

[10] System Management Course , part no. 38721, Tandem Computers, Incorporated.

[11] EXPAND/FOX Architecture Course, part no. 18071, Tandem Computers, Incoporated.

[12] Interface SQL NonStop 2.0 Edition Part Number First 105517 Second 117323. Tandem Computers Incorporated

[13] Basic Guardian Utilities. Independent Study Program Training, Parte 113496, Primera Edición 1995, Tandem Computers Incorporated