

CAPITULO 20.PROCESAMIENTO AVANZADO DE TRANSACCIONES

20.1 SISTEMAS REMOTOS DE COPIAS DE SEGURIDAD

Ante la creciente necesidad de sistemas de procesamiento de transacciones que ofrezcan una disponibilidad elevada y que puedan funcionar pese a los desastres ambientales, se puede obtener ésta utilizando bases de datos distribuidas en las que los datos estén **replicados** y las transacciones actualicen todas las copias de datos.

Se puede utilizar el compromiso de dos fases, pero ante los inconvenientes que presenta, una opción alternativa es:

Realizar el **procesamiento** de la transacción en **un solo** nodo → **nodo principal**, donde se realizan las actualizaciones

Tener un nodo en el que se **repliquen** todos los datos del nodo principal → **nodo remoto copia seguridad (o secundario)**, que debe

Mantenerse **sincronizado** con el nodo principal. (Enviándose todos los registros del registro histórico desde el nodo principal al remoto)

Debe hallarse físicamente separado del principal para evitar que una catástrofe en el principal no afecte a la copia

Cuando falla el nodo principal, el **nodo remoto copia de seguridad** asume el procesamiento:

Primero lleva a cabo la recuperación utilizando su copia (tal vez anticuada) de los datos del nodo principal y los registros del registro histórico recibidos

Después, comienza a procesar transacciones

El rendimiento de los sistemas remotos para copias de seguridad es **mejor** que el de los sistemas distribuidos de dos fases

Aspectos a tener en cuenta

Detección de fallos

Para evitar que se confunda el fallo de las líneas de comunicación con un fallo del nodo principal, hay que mantener varios enlaces de comunicaciones con modos de fallo independientes entre el nodo principal y el nodo remoto copia de seguridad

Transferencia de control

La manera mas sencilla de **transferir el control** es:

→ el nodo principal antiguo recibe el registro histórico de operaciones rehacer del nodo copia de seguridad antiguo y se pone al día con las actualizaciones aplicándolas de manera local

→ entonces puede actuar como nodo remoto de copia de seguridad

si hay que devolver el control, el nodo remoto copia de seguridad antiguo puede simular que ha fallado

lo que da lugar a que el nodo principal antiguo asuma el control

Tiempo de recuperación

Para **reducir** el tiempo de recuperación del registro histórico si este se hace excesivamente grande, se puede hacer que el nodo remoto **copia** de seguridad **procese** de manera **periódica** los registros **rehacer** del registro histórico que haya

recibido y realizar un control de manera que pueda **borrar las partes más antiguas** del registro histórico

relevo caliente: Puede hacer la toma de control por el nodo copia casi instantáneamente.

El nodo remoto copia de seguridad procesa los registros rehacer del registro histórico según llegan, y aplica las actualizaciones de manera local.

Si falla el principal, el nodo copia completa la recuperación retrocediendo las transacciones incompletas y queda preparado para realizar las nuevas

Tiempo de compromiso

Para asegura que las actualizaciones de una transacción comprometida sean duraderas, no se debe declarar comprometida una transacción hasta que sus registros del registro histórico hayan alcanzado el nodo copia de seguridad
Grados de durabilidad:

Uno seguro

Las transacciones se comprometen tan pronto como sus registros de compromiso del registro histórico se escriben en un almacenamiento estable en el nodo local

Dos muy seguro

Las transacciones se comprometen tan pronto como sus registros de compromiso del registro histórico se escriben en un almacenamiento estable en el nodo principal y en el nodo copia de seguridad

Dos seguro

Las transacciones se comprometen tan pronto como sus registros de compromiso del registro histórico se escriben en un almacenamiento estable en el nodo principal y en el nodo copia de seguridad **si ambos están activos**.

Si solo está activo el principal Las transacciones se comprometen tan pronto como sus registros de compromiso del registro histórico se escriben en un almacenamiento estable en el nodo local

20.2 MONITORES DE PROCESAMIENTO DE TRANSACCIONES

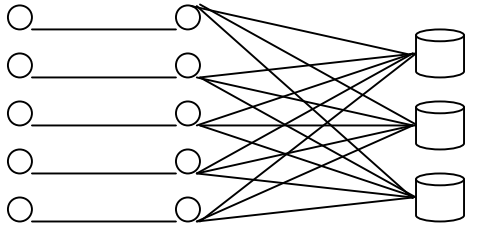
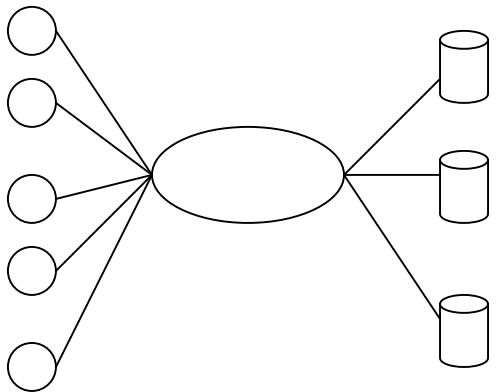
El protocolo de compromiso de dos fases constituye la piedra angular del trabajo con transacciones que abarcan varias bases de datos. Sin embargo, se necesitan niveles de software y las correspondientes arquitecturas de software para coordinar la ejecución de dichas tareas. Los **monitores de procesamiento de transacciones** (TP) son los sistemas que ofrecen ese apoyo

Se desarrollaron como respuesta a la necesidad de trabajar con un gran número de terminales remotas (Ej. Líneas aéreas) desde una sola computadora, habiendo evolucionado desde entonces.

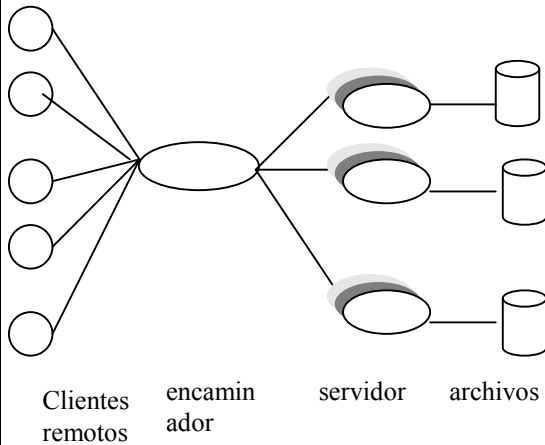
20.2.1 Arquitectura de los monitores TP

Los usuarios inician una sesión y ejecutan uno o varios procesos. El sistema operativo tiene una sobrecarga de memoria considerable con cada inicio de sesión y esta sobrecarga de memoria sería excesiva si todos los usuarios tuvieran que iniciar la sesión en el sistema informático. Además, por motivos de seguridad, no conviene permitir a demasiados usuarios remotos el acceso sin ligaduras al sistema operativo mediante los inicios de sesión.

En lugar de tener un inicio de sesión de cada terminal, tenemos las siguientes opciones alternativas.

<p>Modelo de proceso por cliente</p>  <p>Clientes remotos Servidor archivos</p>	<ul style="list-style-type: none"> • Un proceso del servidor se comunica con la terminal tratando la autenticación y ejecutando las acciones solicitadas por la terminal. • Inconvenientes: <ul style="list-style-type: none"> → Las necesidades de memoria de cada proceso son elevadas → El sistema operativo reparte el tiempo de UCP disponible entre los procesos cambiando de uno a otro (multitarea). Cada cambio de contexto de un proceso al siguiente supone una sobrecarga considerable de la UCP
<p>Modelo de proceso unico</p>  <p>Clientes remotos servidor archivos</p>	<p>Evitar problemas anteriores:</p> <ul style="list-style-type: none"> • Único proceso del servidor al que se conectan todas las terminales remotas, que envían solicitudes al proceso del servidor, que las ejecuta. • Para evitar el bloqueo de los demás clientes cuando se procesa una solicitud de gran tamaño de un cliente, el proceso del servidor es multihembrado (Tiene una hebra de control para cada cliente e implementa su propia multitarea con poca sobrecarga. Ejecuta el código en nombre de un cliente durante un periodo de tiempo, luego guarda el contexto interno y cambia a otro cliente.) • El coste de cambio entre hebras es reducido • Proporciona elevadas tasas de transacciones con recursos limitados • Inconvenientes: <ul style="list-style-type: none"> → No hay protección entre las aplicaciones al ejecutarse como un único proceso → No son adecuados para las bases de datos paralelas o distribuidas.

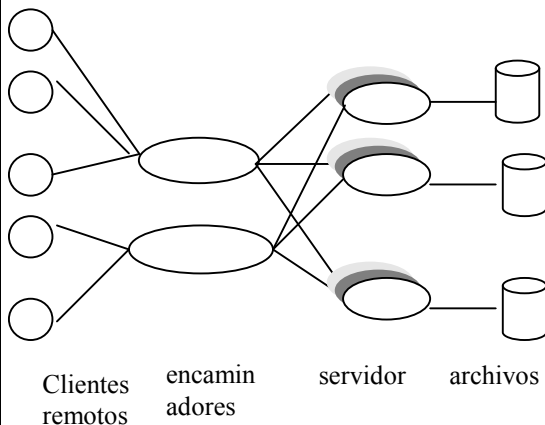
Modelo de varios servidores y un solo encaminador



Evitar problemas anteriores:

- Ejecutar varios procesos del servidor de aplicaciones que tengan acceso a una base de datos común y permitir que los clientes se comuniquen
- Soporta procesos del servidor independientes para varias aplicaciones
- Cada aplicación puede tener un grupo de procesos del servidor, cualquiera de los cuales puede tratar una sesión cliente, de forma que una solicitud puede encaminarse al servidor con menos carga del grupo.
- Cada proceso del servidor puede ser multitenantedo, de forma que pueda tratar varios clientes de forma concurrente.
- Pueden ejecutarse en diferentes nodos de una BD paralela o distribuida.
- El proceso de comunicación puede tratar la coordinación de sucesos

Modelo de varios servidores y varios encaminadores



- Los procesos de comunicaciones de los clientes interactúan con uno o varios procesos de encaminamiento que encaminen sus solicitudes al servidor apropiado
- Un proceso de control inicia los demás procesos y supervisa su funcionamiento

Estructura detallada de un monitor TP

Gestiona las colas de los mensajes que lleguen

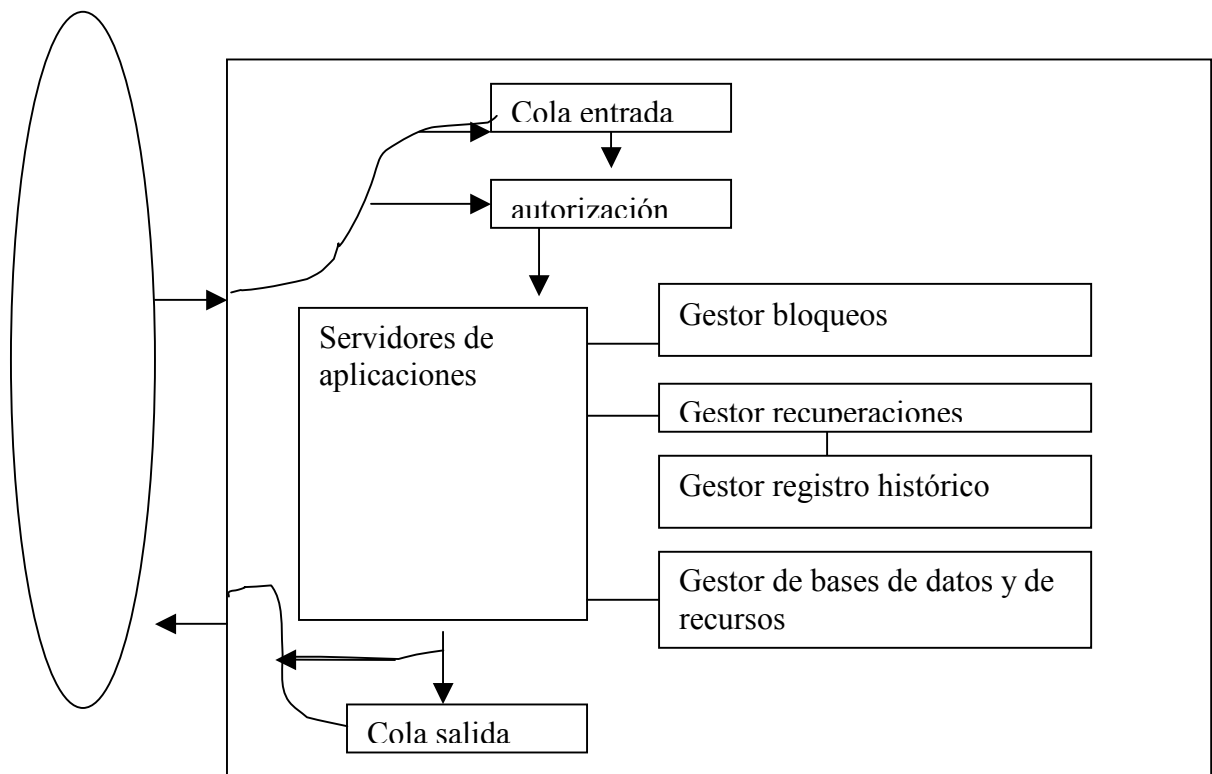
Los mensajes pueden registrarse en un almacenamiento estable

Controla la gestión de la autorización y de los servidores de aplicaciones (inicio servidor y encaminamiento de los mensajes a los servidores)

Proporcionan utilidades de registro histórico, recuperación y control de la concurrencia

Proporcionan un almacenamiento estable de colas de los mensajes salientes

Proporcionan utilidades para presentaciones para los clientes sin capacidad de proceso como las terminales, ayudando a crear interfaces para menús y formularios



red

20.2.2 Coordinación de aplicaciones utilizando monitores TP

Los monitores TP proporcionan soporte para la construcción y la administración de grandes aplicaciones formadas por varios subsistemas, como las bases de datos, los sistemas heredados y los sistemas de comunicaciones

Los monitores TP tratan cada subsistema como un **gestor de recursos**

La interfaz entre el TP y el gestor de recursos se define como un conjunto de primitivas (iniciar transacción, comprometer, abortar, preparar..)

La interfaz del gestor de recursos se define en la norma X/Open

Los monitores TP proporcionan servicios como la gestión de colas persistentes y el transporte de mensajes

Los monitores TP proporcionan una interfaz RPC (llamada a procedimientos remotos) para transacciones para los sistemas cliente-servidor

Los monitores TP también se pueden utilizar para administrar sistemas cliente-servidor complejos de varios servidores y gran número de clientes

Cada servidor se comporta como gestor de recursos y se registra en el monitor TP. Este, coordina las actividades, como los controles y cierre del sistema

Proporciona seguridad y la autenticación de los clientes

Administra los grupos de servidores añadiendo o eliminando servidores sin que el sistema sufra interrupciones

Controla el alcance de los fallos

Los monitores TP también pueden utilizarse para ocultar los fallos de las bases de datos en los sistemas replicados

20.3 SISTEMAS DE TRANSACCIONES DE ALTO RENDIMIENTO

Para permitir un índice elevado de procesamiento de transacciones hay que utilizar hardware de alto rendimiento y aprovechar el paralelismo. Sin embargo, estas técnicas, son insuficientes debido a:

La E/S de disco sigue siendo un cuello de botella

Las transacciones que se ejecutan en paralelo pueden intentar leer o escribir los mismos elementos de datos, lo que da lugar a conflictos que reducen el paralelismo efectivo

Las técnicas para reducir la influencia de estos problemas son:

20.3.1 Bases de datos en memoria principal

Generalmente, el rendimiento de los sistemas de bases de datos queda limitado por la velocidad a la que los datos se leen del disco y se escriben en él. Se puede reducir el grado de dependencia del disco de una base de datos aumentando el tamaño de la memoria intermedia de la misma (actualmente el coste es muy reducido)

Las memorias principales de gran tamaño permiten el procesamiento más rápido de las transacciones, dado que los datos están residentes en la memoria.

Limitaciones:

Los registros del registro histórico deben escribirse en un almacenamiento estable antes de comprometer las transacciones, lo que puede convertirse en cuello de botella

Hay que seguir escribiendo los bloques de memoria intermedia marcados como modificados por las transacciones comprometidas para que la cantidad de registro histórico que haya que repetir en el momento de la recuperación se reduzca

Si el sistema cae, se pierde toda la memoria principal **Oportunidades para la optimización:**

Las estructuras de datos pueden tener punteros que atraviesen varias páginas, a diferencia de las bases de datos de disco, en las que el coste de E/S de atravesar varias páginas es excesivamente elevado

No hace falta colocar las páginas de la memoria intermedia en la memoria antes de que tenga acceso a los datos, ya que éstas no se sustituyen nunca

Se deben diseñar las técnicas de procesamiento de consultas para minimizar la sobrecarga de espacio, de modo que no superen los límites de la memoria principal mientras se evalúa una consulta

Las operaciones de bloqueos y pestillos pueden transformarse en nuevos cuellos de botella. Se evita mediante mejoras en la implementación de dichas operaciones

Los algoritmos de recuperación pueden optimizarse, ya que rara vez hay que escribir las páginas para hacer sitio a otras páginas.

20.3.2 Compromiso en grupo

20.4 TRANSACCIONES DE LARGA DURACIÓN

20.4.1 Ejecuciones no secuenciables

El permitir únicamente planificaciones secuenciables no es aconsejable en las transacciones de larga duración. Todos los protocolos siguientes, por algún u otro motivo tienen efectos negativos en las transacciones de larga duración:

Bloqueo de dos fases

Si el elemento de datos está bloqueado por una transacción de larga duración, la espera será larga y los tiempos de espera largos provocan tiempos de respuesta largos y mayores posibilidades de interbloqueo

Protocolos basados en grafos

Al bloquear los elementos de datos de manera consistente con la ordenación, puede que se bloqueen más datos de los que se necesitan. Por tanto es posible que se produzcan bloqueos de larga duración

Protocolos basados en marcas temporales

Estos protocolos exigen que en determinadas circunstancias aborta una transacción, lo que haría perder mucha información en transacciones de larga duración, con el consiguiente malestar para el usuario

Validación

Estos protocolos también hacen cumplir la secuenciabilidad abortando transacciones, con lo que el problema es igual al anterior.

Por todas estas consideraciones, se estudian nuevas formas de control de concurrencia:

20.4.2 Control de la concurrencia

El control de la concurrencia puede conseguirse sin secuenciabilidad. La corrección depende de las ligaduras de consistencia específicas de la base de datos

las propiedades de las operaciones realizadas por cada transacción **Técnicas** para conseguirlo:

Utilizar las ligaduras de consistencia de las bases de datos como base para dividir la base de datos en subbases de datos en las que se pueda gestionar la concurrencia por separado

Tratar algunas operaciones, además de leer y escribir, como operaciones fundamentales de bajo nivel y extender el control de la concurrencia para tratar con ellas

Técnicas de control de concurrencia que aprovechan varias versiones

20.4.3 Transacciones anidadas y multinivel

Una transacción **anidada o multinivel** T consiste en un conjunto $Y=\{t_1, t_2, \dots, t_n\}$ de subtransacciones y en un orden parcial P para T . Una transacción t_i puede abortarse sin obligar a abortar T .

La anidación puede tener varios niveles de profundidad, lo que representa una subdivisión de las transacciones en subtareas, subsubtarear, etc. En el nivel inferior de anidación están las operaciones normales de las bases de datos leer y escribir

Si se permite que una subtransacción de T libere bloqueos al completarse T , se denomina transacción **multinivel**.

Si esta representa una actividad de larga duración, a veces se denomina **saga**.

De manera alternativa, si los bloqueos mantenidos por una subtransacción t_i de T se asignan de manera automática a T al completarse t_i , T se denomina transacción **anidada**

20.4.4 Transacciones compensadoras

El concepto de las transacciones **compensadoras** ayuda a abordar el problema de que la exposición de los datos no comprometidos cree la posibilidad de retrocesos en cascada

Sea la transacción T dividida en varias subtransacciones t_1, t_2, \dots, t_n . Después de comprometer la subtransacción t_j , libera sus bloqueos. Ahora, si hay que abortar la transacción T del nivel exterior, hay que deshacer el efecto de sus subtransacciones. Supóngase que se han comprometido las subtransacciones t_1, \dots, t_k y que t_{k+1} se estaba ejecutando cuando se tomó la decisión de abortar. Los efectos de t_{k+1} se pueden deshacer abortando esa subtransacción. Sin embargo, no es posible abortar las subtransacciones $t_1 \dots t_k$, dado que ya se había comprometido.

En lugar de eso, se ejecuta una nueva subtransacción tc_1 , denominada transacción **compensadora**, para deshacer el efecto de la transacción t_1 .

Cada subtransacción t_i , debe tener una transacción compensadora tc_i

Hay que ejecutar las transacciones compensadoras en el orden inverso tc_k, \dots, tc_1 .

Si el sistema cae en medio de la ejecución de una transacción del nivel exterior, hay que retroceder sus subtransacciones cuando se recupere

20.4.5 Aspectos de la implementación

Las transacciones de larga duración tienen muchos problemas para su implementación al complicarse el proceso de recuperación

Hay dos enfoques para la reducción de la sobrecarga de asegurar la recuperabilidad de los elementos de datos de gran tamaño:

Registro histórico de las operaciones o lógico

Solo se guardan en el registro histórico la operación realizada en el elemento de datos y el nombre de éste. Para cada operación debe haber una operación inversa. Se lleva a cabo una operación **deshacer** utilizando la operación inversa y **rehacer** utilizando la propia operación

Registro histórico y paginación en la sombra

El registro histórico se utiliza para las modificaciones de los elementos de datos de tamaño pequeño, pero los elementos de datos de gran tamaño se vuelven recuperables mediante una técnica de paginación en la sombra, donde solo hay que guardar por duplicado aquellas páginas que se han modificado realmente

Es conveniente permitir que algunos datos no fundamentales queden exentos del registro histórico y confiar en su lugar en las copias de seguridad sin conexión y en la intervención humana.

20.5 SISTEMAS DE TRANSACCIONES EN TIEMPO REAL

En algunas aplicaciones las ligaduras incluyen **tiempo límite** para los cuales la tarea debe estar completada. Se caracterizan:

Inflexible: La tarea tiene valor **nulo** si se completa **después** del tiempo límite

Flexible: La tarea tiene valor **que disminuye** si se completa **después** del tiempo límite y se aproxima a cero a medida que aumenta el retraso.

Los sistemas con tiempos límite se denominan sistemas de **tiempo real**

La gestión de transacciones debe tener en cuenta el tiempo límite

Un problema importante surge con la varianza del tiempo de ejecución de las transacciones

El tiempo de ejecución de las transacciones sólo puede estimarse de forma muy aproximada si los datos residen en el disco. Por ello, suele utilizarse las bases de datos en memoria principal, aunque también aparecen varianzas en el tiempo de ejecución

El problema principal son los límites de tiempo más que la velocidad absoluta

Diseñar un sistema de tiempo real implica asegurar que hay suficiente capacidad de procesamiento para no superar los tiempos límite sin exigir demasiados recursos de hardware.

20.6 NIVELES DEBILES DE CONSISTENCIA

Hay veces que los protocolos necesarios para asegurar la secuencialidad permiten muy poca concurrencia para algunas aplicaciones. En estos casos se utilizan los niveles más débiles de consistencia, añadiendo una nueva carga a los programadores para asegurar la corrección de las bases de datos

20.6.1 Consistencia de grado dos

Su objetivo es evitar abortar en cascada sin asegurar necesariamente la secuencialidad.

Su protocolo utiliza los mismos modos de bloqueo que se utilizan para el protocolo de bloqueo de dos fases: **compartido ©** y **exclusivo (X)**

Las transacciones deben mantener el modo de bloqueo adecuado cuando tengan acceso a un elemento de datos

Los bloqueos compartidos © pueden liberarse en cualquier momento y también se pueden establecer bloqueos en cualquier momento

Los bloqueos exclusivos (X) no se pueden liberar hasta que la transacción se comprometa o aborte

La secuencialidad no queda asegurada, lo que hace que este enfoque no sea conveniente para muchas aplicaciones

20.6.2 Estabilidad del cursor

Es una forma de consistencia de grado dos diseñada para los programas escritos en lenguajes de propósito general orientado a registros, como Pascal, C, Cobol, PL/I o Fortran. Estos programas suelen iterar sobre las tuplas de las relaciones. En lugar de bloquear toda la relación, la estabilidad del cursor asegura

- La tupla que está procesado la iteración esté bloqueada en modo compartido
- Todas las tupas modificadas estén bloqueadas en modo exclusivo hasta que se comprometa la transacción.

Estas reglas aseguran que se obtenga la consistencia de grado dos

El bloqueo de dos fases no es necesario

La secuencialidad no queda asegurada

Se utiliza para las relaciones con muchos accesos como un medio de aumentar la concurrencia y de mejorar el rendimiento del sistema

20.7 FLUJOS DE TRABAJO DE TRANSACCIONES

Un **flujo de trabajo** es una actividad que implica la ejecución coordinada de varias tareas llevadas a cabo por entidades de proceso diferentes. También se llaman **flujos de tareas** o **aplicaciones multisistemas**

Una **tarea** define un trabajo que hay que realizar y puede especificarse de varias maneras, incluyendo la descripción textual en un archivo o en un mensaje de correo electrónico, en un formulario, en un mensaje o en un programa informático. A veces también se denominan **pasos**

Una entidad de procesamiento que lleve a cabo las tareas puede ser una persona o un sistema de software. En general, los flujos de trabajo implican a una o varias personas.

Hoy en día, toda la información relacionada con los flujos de trabajo se halla guardada muy probablemente de forma digital en una o varias computadoras, y con la expansión de los sistemas de redes, la información puede transferirse con facilidad de una computadora a otra. Por tanto, es factible que las empresas **automatizen** sus flujos de trabajo. El mismo flujo de trabajo implica el paso de responsabilidad de una persona a otra, e incluso a programas que puedan reunir de manera automática la información necesaria. Las personas pueden coordinar sus actividades utilizando medios como el correo electrónico.

Problemas para **automatizar** los flujos de trabajo:

Especificación del flujo de trabajo: Detallar las tareas que hay que ejecutar y definir las necesidades de ejecución

Ejecución de los flujos de trabajo: Debe realizarse al tiempo que se proporcionan las salvaguardias de los sistemas tradicionales de bases de datos relacionadas con la corrección del cálculo, la integridad de los datos y la durabilidad

20.7.1 Especificación del flujo de trabajo

Las especificaciones JCL (Job Control Language) de los sistemas operativos por lotes, donde se permite al usuario especificar los trabajos como conjunto de pasos se pueden considerar como ejemplos de especificaciones de flujos de trabajo.

No hace falta modelar los aspectos internos de las tareas con vistas a la gestión de los flujos de trabajo.

En cualquier momento de la ejecución, el estado de un flujo de trabajo consiste en el conjunto de estados de las tareas que lo constituyen y en los estados (valores) de todas las variables de la especificación de dicho flujo

La coordinación de las tareas se puede especificar de manera

Estática: las tareas y las dependencias existentes entre ellas se definen antes de que comience la ejecución del trabajo.

Las **dependencias** entre tareas pueden ser:

Sencillas → completar cada tarea antes de que comience la siguiente.

Generalización → imponer una condición previa a la ejecución de cada tarea del flujo de trabajo, de manera que se conozcan con antelación todas las tareas posibles del mismo y sus **dependencias**:

Los **estados de ejecución** de otras tareas

Los **valores de salida** de otras tareas

Las **variables externas modificadas** por los sucesos externos.

Estas dependencias se pueden combinar con las conectivas lógicas (o, y, no)

Dinámica: Ejemplo → sistemas de distribución de correo electrónico

20.7.2 Requisitos de atomicidad ante fallos de los flujos de trabajo

Se debe permitir que el diseñador defina los requisitos de **atomicidad ante fallos del flujo** de trabajo

→ **estados de terminación aceptable** del flujo de trabajo: . El sistema debe garantizar que cada ejecución de un flujo de trabajo terminará en un estado que satisfaga los requisitos de atomicidad ante fallos definidos por el diseñador.

estados de terminación aceptable comprometido: Se alcanzan los objetivos del flujo de trabajo

estados de terminación aceptable abortado: Es un estado de terminación válido pero en el que no se alcanzan los objetivos → hay que deshacer todos los efectos no deseables de la ejecución parcial del flujo de trabajo de acuerdo con los requisitos de atomicidad ante fallos de dicho flujo

→ **estados de terminación no aceptable**: Son el resto. Cabe la posibilidad de que se violen los requisitos de atomicidad ante fallos

Los flujos de trabajo deben alcanzar un estado de terminación aceptable **incluso en caso de fallos del sistema** → Si el flujo de trabajo se encontraba en un estado de terminación no aceptable en el momento de producirse el fallo, se le debe conducir a un estado de terminación aceptable (sea abortado o comprometido) durante la recuperación del sistema

20.7.3 Ejecución de los flujos de trabajo

Se puede controlar la ejecución de las tareas mediante un coordinador humano o mediante un sistema de software denominado **sistema de gestión de flujos de trabajo** consistente en:

Un planificador: Es un programa que procesa los flujos remitiendo varias tareas para su ejecución, controlando varios sucesos y evaluando las condiciones relacionadas con las dependencias entre las tareas

Agentes para las tareas: Controlan la ejecución de las tareas mediante entidades de proceso

Mecanismo para consultar estado Hay 3 enfoque arquitectónicos:

Centralizado: Un unico planificador. Se utiliza en los sistemas de flujo de trabajo en que los datos se guardan en una base de datos central

Parcialmente distribuido: Un planificador por cada flujo de trabajo

Totalmente distribuido: No tiene planificadores, pero los agentes de tareas coordinan su ejecución comunicándose entre sí. Los más sencillos siguen este enfoque y se basan en el correo electrónico. Es mas complicado que el centralizado

20.7.4 Recuperación de flujos de trabajo

El **objetivo es hacer cumplir la atomicidad ante fallos** de los flujos de trabajo. Los procedimientos de recuperación deben asegurarse de que si se produce un fallo en cualquiera de los componentes de procesamiento del flujo de trabajo (incluyendo el planificador), el flujo llegue a alcanzar un **estado de terminación aceptable** (sea abortado o comprometido)

Se da por supuesto que las entidades de procesamiento implicadas en el flujo de trabajo poseen sus propios sistemas locales de recuperación y tratan sus propios fallos. Para recuperar el contexto del entorno de ejecución, las rutinas de recuperación de fallos tienen que restaurar la información del estado del planificador en el momento del fallo, incluyendo la información sobre el estado de

ejecución de cada tarea. Por tanto, la información de situación adecuada debe registrarse en un almacenamiento estable.