1

BASES DE DATOS DISTRIBUIDAS

TEMA 5

PROFESOR: M.C. ALEJANDRO GUTIÉRREZ DÍAZ

2

5. CONFIABILIDAD

5.1 Conceptos básicos de confiabilidad

5.2 Protocolos Redo - Undo

5.3 Puntos de verificación - checkpoints

5.4 Protocolo 2PC de confiabilidad distribuida

3

Un sistema de manejo de bases de datos confiable es aquel que puede continua procesando las solicitudes de usuario aún cuando el sistema sobre el que opera no es confiable.

En otras palabras, aun cuando los componentes de un sistema distribuido fallen, un DDMBS confiable debe seguir ejecutando las solicitudes de usuario sin violar la consistencia de la base de datos.

En este capítulo se discutirán las características de un DDBMS confiable.

En este capítulo se considerará el caso de que un sistema distribuido no sea confiable y, particularmente desde el punto de vista de los DDMBS, se presentarán los protocolos para recuperación de información.

4

Definiciones

A lo largo de estas notas nos hemos referido a la confiabilidad y disponibilidad de la base de datos sin definir esos términos de manera precisa. En esta sección daremos sus definiciones generales para posteriormente elaborarlas de manera más formal. La confiabilidad se puede interpretar de varias formas.

La confiabilidad se puede ver como una medida con la cual un sistema conforma su comportamiento a alguna especificación. También se puede interpretar como la probabilidad de que un sistema no haya

experimentado ninguna falla dentro de un periodo de tiempo dado. La confiabilidad se utiliza típicamente como un criterio para describir sistemas que no pueden ser reparados o donde la operación continua del sistema es crítica.
Disponibilidad, por otro lado, es la fracción del tiempo que un sistema satisface su especificación. En otras palabras, la probabilidad de que el sistema sea operacional en un instante dado de tiempo.

5

Sistema, estado y falla

Un ***sistema*** se refiere a un mecanismo que consiste de una colección de componentes y sus interacciones con el medio ambiente que responden a estímulos que provienen del mismo con un patrón de comportamiento reconocible (ver Figura 5.1).
Cada componente de un sistema puede ser así mismo un sistema, llamado comúnmente *subsistema*. Un ***estado externo*** de un sistema se

puede definir como la respuesta que un sistema proporciona a un estímulo externo.
Por lo tanto, es posible hablar de un sistema que se mueve dentro de estados externos de acuerdo a un estímulo proveniente del medio ambiente. Un *estado interno* es, por lo tanto, la respuesta del sistema a un estímulo interno.
Desde el punto de vista de confiabilidad, es conveniente definir a un estado interno como la unión de todos los estado externos de las componentes que constituyen el sistema. Así, el cambio de estado interno se da como respuesta a los estímulos del medio ambiente.

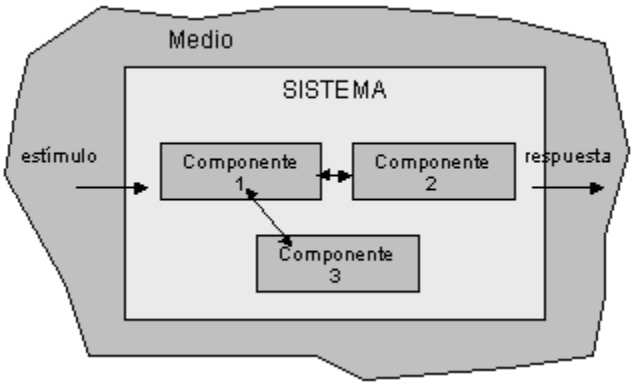

Figura 5.1. Conceptos básicos de un sistema.
El comportamiento del sistema al responder a cualquier estímulo del medio ambiente necesita establecerse por medio de una *especificación*, la cual indica el comportamiento válido de cada estado del sistema. Su especificación es no sólo necesaria para un buen diseño sino también es esencial para definir los siguientes conceptos de confiabilidad.

Cualquier desviación de un sistema del comportamiento descrito en su especificación se considera como una ***falla.*** Cada falla necesita ser rastreada hasta su causa. En un sistema confiable los cambios van de estados válidos a estados válidos

6.

en un sistema no confiable, es posible que el sistema caiga en un estado interno el cual no obedece a su especificación; a este tipo de estados se les conoce como *estados erróneos*. Transiciones a partir de este estado pueden causar una falla.
La parte del estado interno que es incorrecta se le conoce como *error* del sistema. Cualquier error en los estados internos de las componentes del sistema se le conoce como una *falta* en el sistema. Así, una falta causa un error lo que puede inducir una falla del sistema (ver Figura 5.2).
Las faltas del sistema se pueden clasificar como severas (hard) y no severas (soft). Las faltas severas casi siempre son de tipo permanente y conducen a fallas del sistema severas. Las faltas no severas por lo general son transitorias o intermitentes. Ellas inducir fallas del sistema no severas las cuales representan, por lo general, el 90 % de todas las fallas.

Figura 5.2. De faltas a fallas

7

Se presentará ahora la definición formal de confiabilidad y disponibilidad.
La confiabilidad de un sistema, $R(t)$, se define como la siguiente probabilidad condicional:
$R(t)$ = Pr{ 0 fallas en el tiempo [0,t] | no hubo fallas en t = 0 }
Si la ocurrencia de fallas sigue una distribución de Poisson, entonces,
$R(t)$ = Pr{ 0 fallas en el tiempo [0,t] }

8

El cálculo de la confiabilidad y disponibilidad puede ser tedioso. Por lo tanto, es acostumbrado usar dos métricas de un sólo parámetro para modelar el comportamiento del sistema. Las dos métricas son el *tiempo medio entre fallas* (MTBF por sus siglas en inglés) y el *tiempo medio para reparaciones* (MTTR). El MTBF puede ser calculado ó a partir de datos empíricos ó de la función de confiabilidad como:
El MTTR está relacionado al rango de reparación de la misma forma que

el MTBF está relacionado al rango de fallas. Usando estas dos métricas, la disponibilidad de un estado estable de un sistema con rangos de falla y reparación exponencial se puede especificar como

9

Tipos de fallas en SMBDD

Diseñar un sistema confiable que se pueda recuperar de fallas requiere identificar los tipos de fallas con las cuales el sistema tiene que tratar. Así, los tipos de fallas que pueden ocurrir en un SMBD distribuido son:

1. Fallas de transacciones. Las fallas en transacciones se pueden deber a un error debido a datos de entrada incorrectos así como a la detección de un interbloqueo. La forma usual de enfrentar las fallas en transacciones es abortarlas. Experimentalmente, se ha determinado que el 3% de las transacciones abortan de manera anormal.

2. Fallas del sistema. En un sistema distribuido se pueden presentar fallas en el procesador, la memoria principal o la fuente de energía

de un nodo. En este tipo de fallas se asume que el contenido de la memoria principal se pierde, pero el contenido del almacenamiento secundario es seguro. Típicamente, se hace diferencia entre las fallas parciales y fallas totales del nodo. Una falla total se presenta en todos los nodos del sistema distribuido. Una falla parcial se presenta solo en algunos nodos del sistema.

3. Fallas del medio de almacenamiento. Se refieren a las fallas que se pueden presentar en los dispositivos de almacenamiento secundario que almacenan las bases de datos. Esas fallas se pueden presentar por errores del sistema operativo, por errores del controlador del disco, o del disco mismo.

4. **Fallas de comunicación**. Las fallas de comunicación en un sistema distribuido son frecuentes. Estas se pueden manifestar como pérdida de mensajes lo que lleva en un caso extremo a dividir la red en varias subredes separadas.

10

La arquitectura correspondiente a la recuperación de errores consiste de un sistema de almacenamiento constituido por dos partes. La primera, llamada memoria principal, es un medio de almacenamiento volátil. La segunda parte, llamada almacenamiento secundario, es un medio de almacenamiento permanente el cual, en principio, no es infalible a fallas. Sin embargo, por medio de una combinación de técnicas de hardware y de software es posible garantizar un medio de almacenamiento estable, capaz de recuperarse de fallas.

A todos los elementos utilizados para obtener un almacenamiento estable se les agrega el atributo "*estable*" con el propósito de reconocer que ellos han sido modificados o creados para este fin. Así tendremos una base de datos estable y operaciones de lectura y escritura estables.
En la Figura se presenta la interfaz entre el administrador de recuperación local y el administrador de buffers de la base de datos. El administrador de buffers de la base de datos mantiene en memoria principal los datos más recientemente accesados, esto se hace con el propósito de mejorar el rendimiento.
Típicamente, el buffer se divide en páginas que son del mismo tamaño que las páginas de la base de datos estable. La parte de la base de datos que se encuentra en el buffer se le conoce como *base de datos*

volátil. Es importante notar que el LRM ejecuta las operaciones solicitadas por una transacción sólo en la base de datos volátil. En un tiempo posterior, la base de datos volátil es escrita a la base de datos estable.

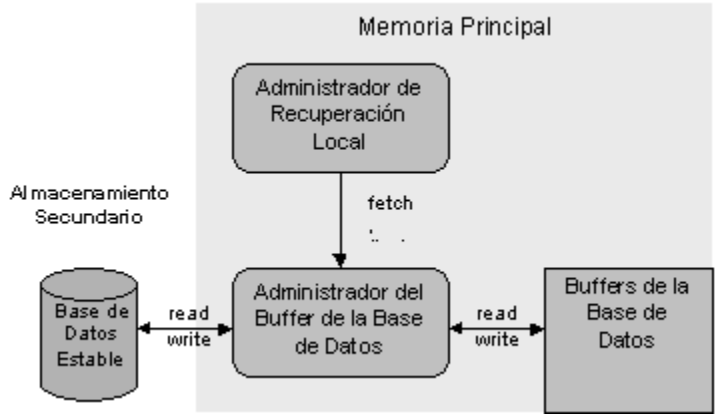

Figura 5.3. Interfaz entre el administrador local de recuperación y el administrador de buffers de la base de datos.
Cuando el LRM solicita una página de datos, envía una instrucción conocida como ***fetch***. El administrador del buffer verifica si la página ya existe en el buffer, y si es así la hace disponible a la transacción que la solicita. En caso contrario, lee la página de la base de datos estable y la coloca en un buffer vacío.
Si no existe un buffer vacío, selecciona uno, la página que contiene es enviada a la base de datos estable y la página solicitada es colocada en el buffer liberado. Para forzar que se descarguen las páginas almacenadas en los buffers, existe el comando ***flush*.**

11

Información de recuperación
Cuando una falla del sistema ocurre, el contenido de la base de datos

volátil se pierde.
EL DBMS tiene que mantener cierta información acerca de su estado en el momento de la falla con el fin de ser capaz de llevar a la base de datos al estado en el que se encontraba antes de la falla. A esta información se le denomina *información de recuperación*.

La información de recuperación que el sistema mantiene depende del método con el que se realizan las actualizaciones. Existen dos estrategias para efectuarlas: en el lugar (*in place*) y fuera de lugar (*out-ofplace*). En el primer caso, cada actualización se hace directamente en los valores almacenados en las páginas de los buffers de la base de datos. En la segunda alternativa, al crear un nuevo valor para un dato, éste se almacena en forma separada del valor anterior. De esta manera, se mantiene los valores nuevo y anterior.

12

BREAK

EN LA SIGUIENTE SESION SE VERAN EN MAS DETALLE

13

Recuperación in-place
Dado que las actualización *in-place* hacen que los valores anteriores se pierdan, es necesario mantener suficiente información de los cambios de estado en la base de datos. Esta información se mantiene, por lo general, en el *registro de la base de datos* (database log). Así cada actualización, no solo cambia la base de datos, sino es también guardada en el registro de la base de datos VER (Figura

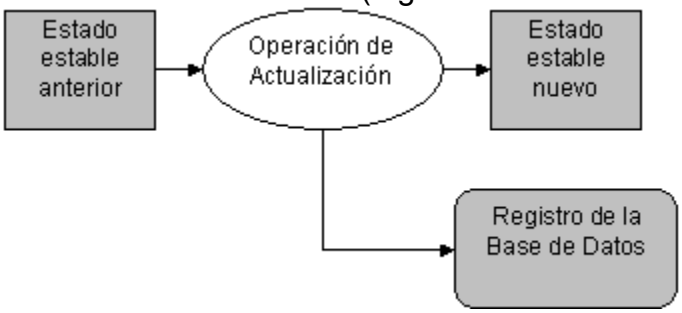

Figura 5.4. Ejecución de una operación de actualización.
El registro de la base de datos contiene información que es utilizada por el proceso de recuperación para restablecer la base de datos a un estado consistente. Esta información puede incluir entre otras cosas:

el identificador de la transacción,

el tipo de operación realizada,
los datos accesados por la transacción para realizar la acción,
el valor anterior del dato (imagen anterior), y
el valor nuevo del dato (imagen nueva).

14

Considere el escenario mostrado en la Figura IZQ 5.5 El DBMS inicia la ejecución en el tiempo 0 y en el tiempo t se presenta una falla del sistema. Durante el periodo [0,t] ocurren dos transacciones, T_1 y T_2. T_1 ha sido concluida (ha realizado su commit) pero T_2 no pudo ser concluida. La propiedad de durabilidad requiere que los efectos de T_1 sean reflejados en la base de datos estable. De forma similar, la propiedad de atomicidad requiere que la base de datos estable no contenga alguno de los efectos de T_2.

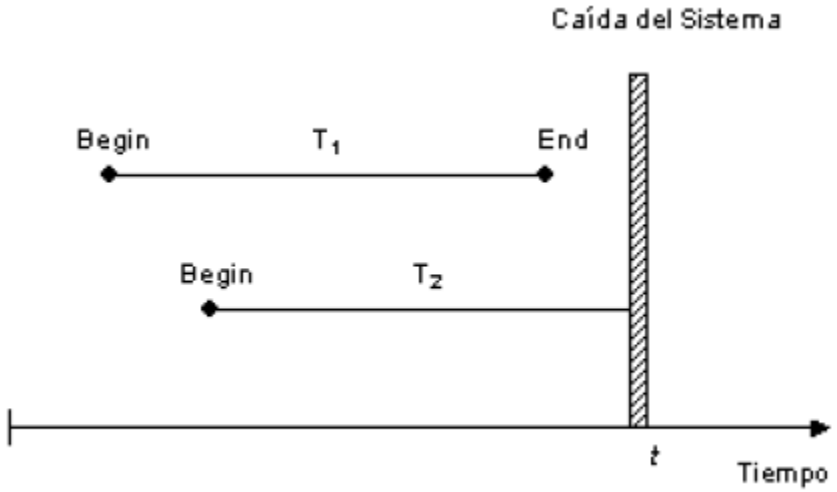

Figura 5.5. Ejemplo de una falla del sistema.
A pesar que T_1 haya sido terminada, puede suceder que el buffer correspondiente a la página de la base de datos modificada no haya sido escrito a la base de datos estable. Así, para este caso la recuperación tiene que volver a realizar los cambios hechos por T_1. A esta operación se le conoce como REDO y se presenta en la Figura DERECHA
La operación de REDO utiliza la información del registro de la base de datos y realiza de nuevo las acciones que pudieron haber sido realizadas antes de la falla. La operación REDO genera una nueva imagen.

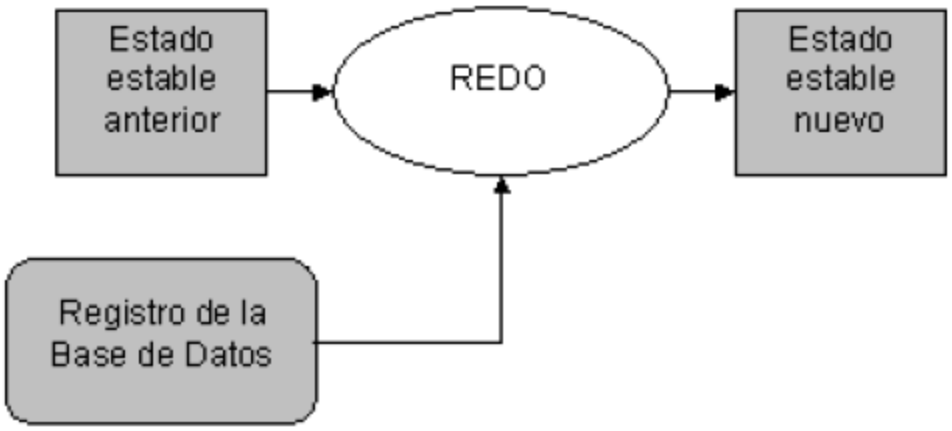

Figura 5.6. Operación REDO.

15

Por otra parte, es posible que el administrador del buffer haya realizado la escritura en la base de datos estable de algunas de las páginas de la base de datos volátil correspondientes a la transacción T_2. Así, la información de recuperación debe incluir datos suficientes para permitir deshacer ciertas actualizaciones en el nuevo estado de la base de datos y regresarla al estado anterior.
A esta operación se le conoce como UNDO y se muestra en la Figura 5.7. La operación UNDO restablece un dato a su imagen anterior utilizando la información del registro de la base de datos.

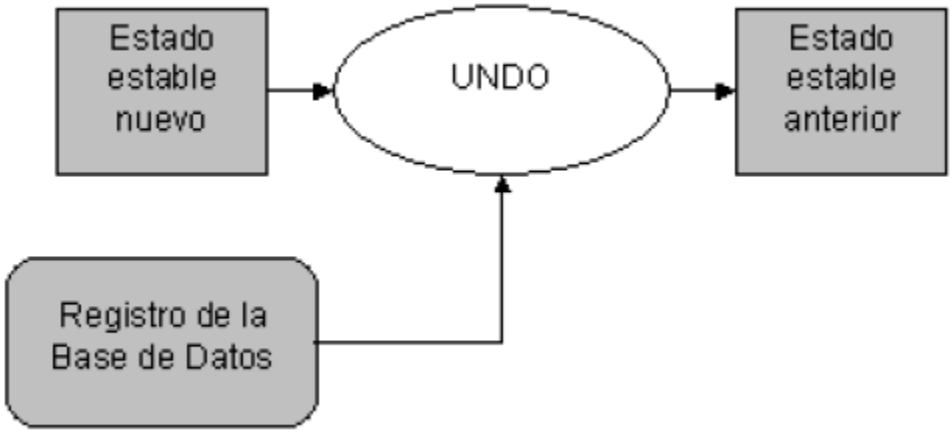

Figura 5.7. Operación UNDO.

16

De forma similar a la base de datos volátil, el registro de la base de datos se mantiene en memoria principal (llamada los *buffers de registro*) y se

escribe al almacenamiento estable (llamado *registro estable*). Las páginas de registro se pueden escribir en el registro estable de dos formas: *síncrona* o *asíncrona*.

En forma síncrona, también llamada un registro forzado, la adición de cada dato en el registro requiere que la página del registro correspondiente se mueva al almacenamiento estable. De manera asíncrona, las páginas del registro se mueven en forma periódica o cuando los buffers se llenan.

Independientemente de que la escritura del registro sea síncrona o asíncrona, se debe observar un protocolo importante para el mantenimiento del registro de la base de datos. Considere el caso donde las actualizaciones a la base de datos son escritas en el almacenamiento estable antes de que el registro sea modificado en el registro estable para reflejar la actualización.

Si una falla ocurre antes de que el registro sea escrito, la base de datos permanecerá en forma actualizada, pero el registro no indicará la actualización lo que hará imposible recuperar la base de datos a un estado consistente antes de la actualización.

Por lo tanto, el registro estable siempre debe ser actualizado antes de la actualización de la base de datos. A este protocolo se le conoce como registro antes de la escritura (write-ahead logging) y se puede especificar de la manera siguiente:

1. Antes de que la base de datos estable sea actualizada, las imágenes anteriores deben ser almacenadas en el registro estable. Esto facilita la realización de un UNDO.
2. Cuando la transacción realiza un commit, las imágenes nuevas tienen que ser almacenadas en el registro estable antes de la actualización de la base de datos estable. Esto facilita la realización de una operación REDO.

La interfaz completa del registro de la base de datos volátil y estable se presenta en la Figura 5.8.

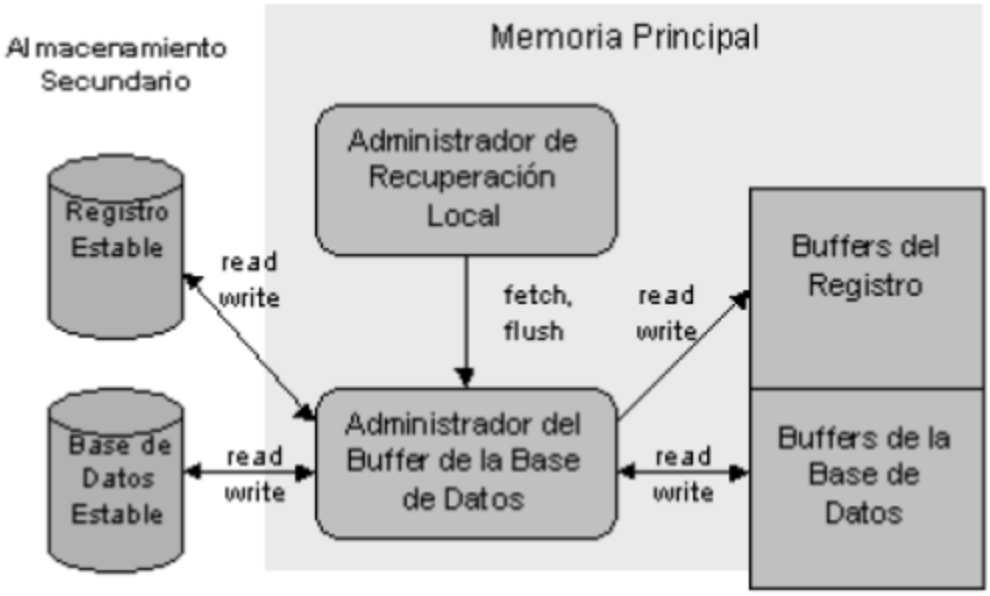

Figura 5.8. Interfaz entre el registro de la base de datos volátil y estable.

17

Recuperación out-of-place

Las técnicas de recuperación más comunes son de tipo in-place. Por lo tanto, aquí se presenta solo una breve descripción de las técnicas out-ofplace.

1. Shadowing. Cuando ocurre una actualización, no se cambia la página anterior, sino se crea una página sombra con el nuevo valor y se escribe en la base de datos estable. Se actualizan los caminos de acceso de manera que los accesos posteriores se hacen a la nueva página sombra. La página anterior se retiene para propósitos de recuperación, para realizar una operación UNDO.

2. **Archivos diferenciales**. Para cada archivo F se mantiene una parte de solo lectura (FR), un archivo diferencial que consiste de la parte de inserciones DF+ y la parte de supresiones DF-. Así, el archivo completo consistirá de la unión de la parte de lectura más la parte de inserciones y a todo esto se le eliminan las supresiones realizadas.

$F = (FR \cup DF+)$ O DFTodas las actualizaciones se tratan como la supresión de un valor anterior y la inserción de un nuevo valor. Periódicamente, el archivo diferencial tiene que ser mezclado con el archivo base de solo lectura.

18

Verificación

La operación de recuperación requiere recorrer todo el registro de la base de datos. Así, el buscar todas las transacciones a las cuales es necesario aplicarles un UNDO o REDO puede tomar una cantidad de trabajo considerable.

Para reducir este trabajo se pueden poner puntos de verificación (checkpoints) en el registro de la base de datos para indicar que en esos puntos la base de datos está actualizada y consistente. En este caso, un REDO tiene que iniciar desde un punto de verificación y un UNDO tiene que regresar al punto de verificación más inmediato anterior. La colocación de puntos de verificación se realiza con las siguientes acciones:

1. Se escribe un "begin_checkpoint" en el registro de la base de datos.
2. Se recolectan todos los datos verificados en la base de datos estable.
3. Se escribe un "fin_de_checkpoint" en el registro de la base de datos.

19

Compromisos de dos fases

El compromiso de dos fases (two-phase commit) o 2PC es un protocolo muy simple y elegante que asegura la atomicidad de las transacciones distribuidas.

Extiende los efectos de una operación local de commit a transacciones distribuidas poniendo de acuerdo a todos los nodos involucrados en la ejecución de una transacción antes de que los cambios hechas por la transacción sean permanentes.

Las fases del protocolo son las siguientes:

Fase 1: el coordinador hace que todos los participantes estén listos para escribir los resultados en la base de datos.

Fase 2: Todos escriben los resultados en la base de datos.

La terminación de una transacción se hace mediante la *regla del compromiso global*:

1. El coordinador aborta una transacción si y solamente si al menos un participante vota por que se aborte.
2. El coordinador hace un commit de la transacción si y solo si todos los participantes votan por que se haga el commit.

La operación del compromiso de dos fases entre un coordinador y un participante en ausencia de fallas se presenta en la Figura 5.9, en donde

los círculos indican los estados y las líneas entrecortadas indican mensajes entre el coordinador y los participantes. Las etiquetas en las líneas entrecortadas especifican la naturaleza del mensaje.

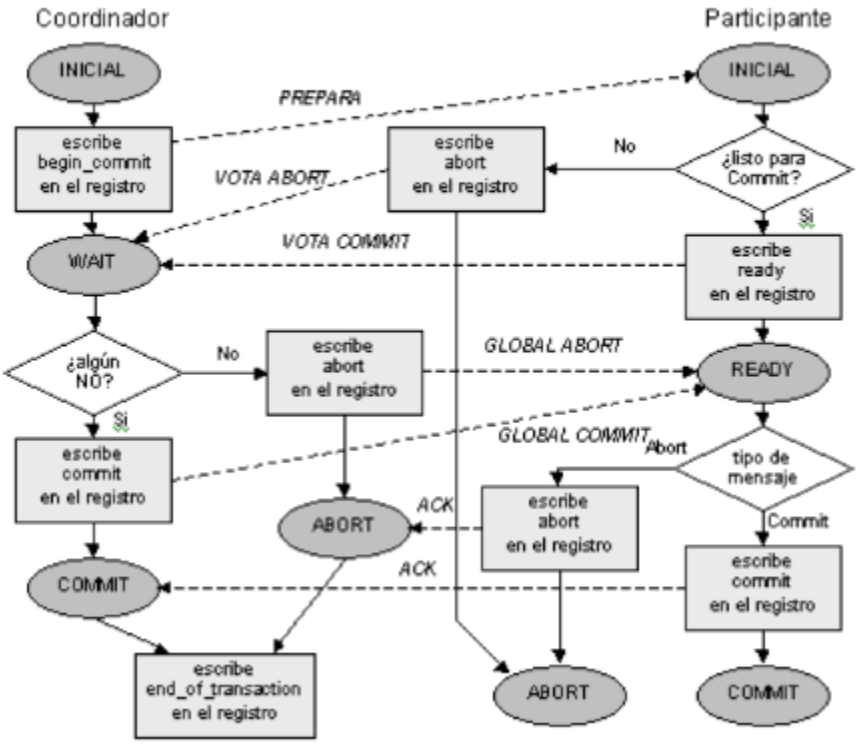

Figura 5.9. Acciones del compromiso de dos fases.

20

Otra alternativa es una comunicación lineal, en donde los participantes se pueden comunicar unos con otros. Existe un ordenamiento entre los nodos del sistema. La estructura se presenta en la Figura 5.11. Un participante, P, recibe un mensaje vote-abort o vote-commit de otro participante, Q. Si P no está listo para hacer un commit, envía un voteabort al siguiente participante, R, y la transacción se aborta en este punto. Si P el participante está de acuerdo en hacer un commit, envía un vote-commit a R y entra al estado de LISTO. Este proceso continúa hasta el último participante poniendo fin a la primera fase.

En la segunda fase, si el último participante está de acuerdo en hacer un commit envía un global-commit al participante anterior. En caso contrario, envía un global-abort. Así en la segunda fase, P recibe un mensaje de R informándole si se hace un global-commit o un global-abort. Este mensaje se propaga a Q y así hasta alcanzar al coordinador.

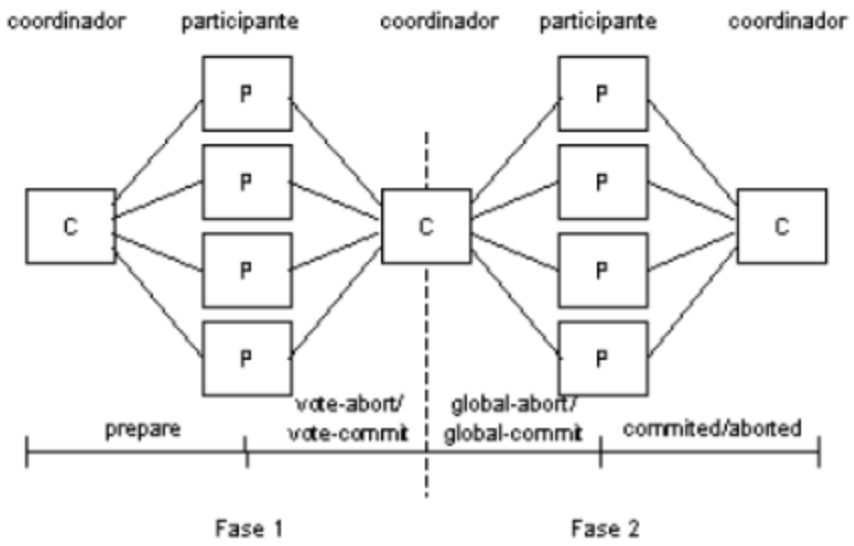

Figura 5.10. Estructura centralizada de comunicaciones para el compromiso de dos fases.

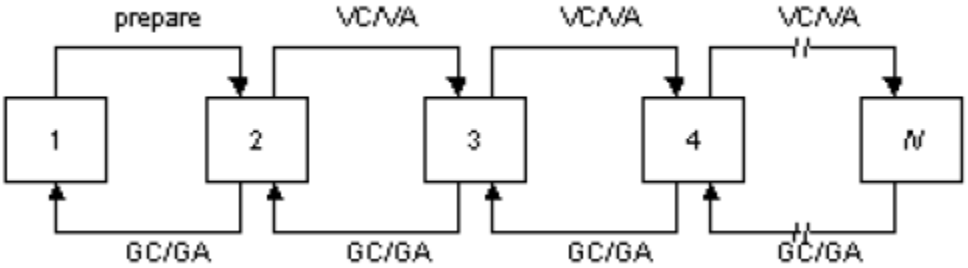

Figura 5.11. Estructura lineal de comunicaciones para el compromiso de dos fases.

21

Otro estructura de comunicación usual para implementar los compromisos de dos fases involucra la comunicación entre todos los participantes durante la primera fase del protocolo de manera que todos ellos alcanzan su punto de terminación en forma independiente.
Esta versión, llamada la *estructura distribuida*, no requiere la segunda fase. En la Figura 5.12 se presenta la estructura distribuida en la cual el coordinador envía el mensaje de preparación a todos los participantes. Cada participante, entonces, envía su decisión a todos los otros participantes y al coordinador indicándola como un vote-commit o voteabort. Cada participante espera los mensajes de todos los otros participantes y toma su decisión de acuerdo a la regla de compromiso global.

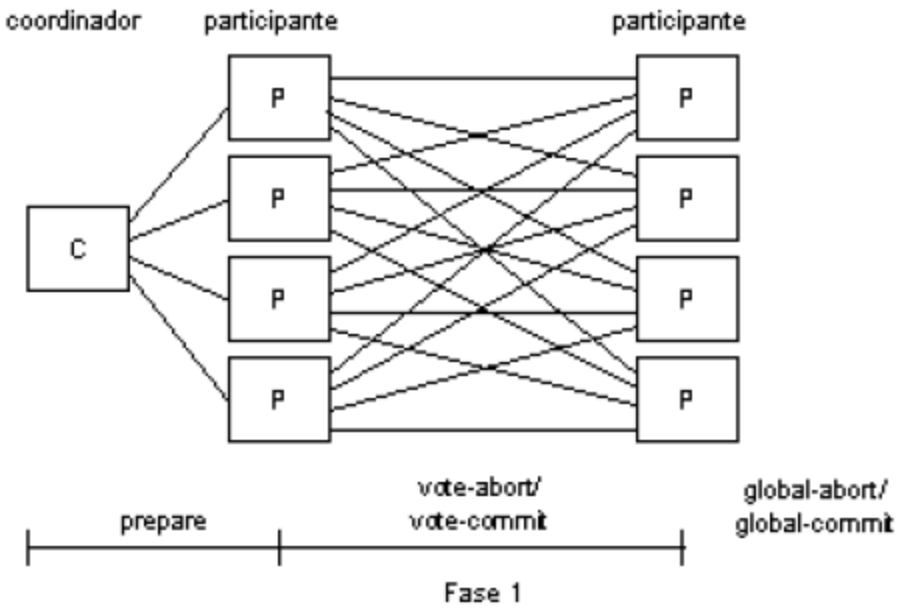

Figura 5.12. Estructura distribuida de comunicaciones para el compromiso de dos fases.

22

Independientemente de la forma en que se implemente el compromiso de dos fases, éste puede ser modelado por medio de un diagrama de transición de estados. En la Figura 5.13 se presentan los diagramas de transición para el coordinador y los participantes.

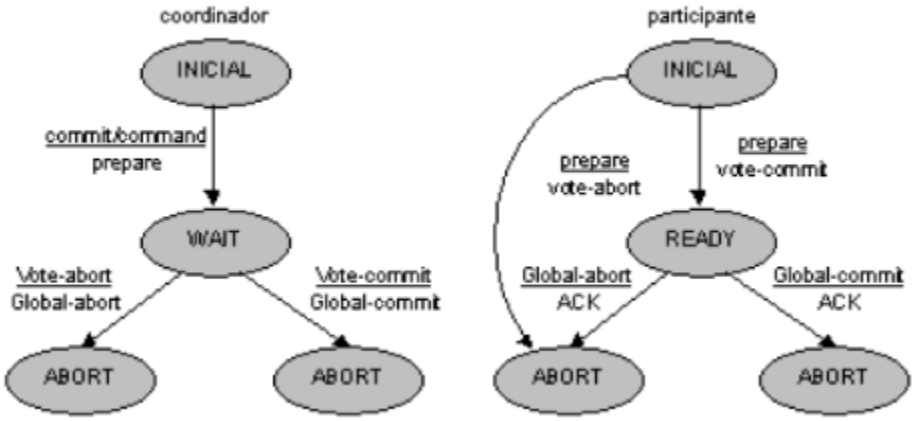

Figura 5.13. Diagrama de transición de estados para el compromiso de dos fases del coordinador y de un participante.