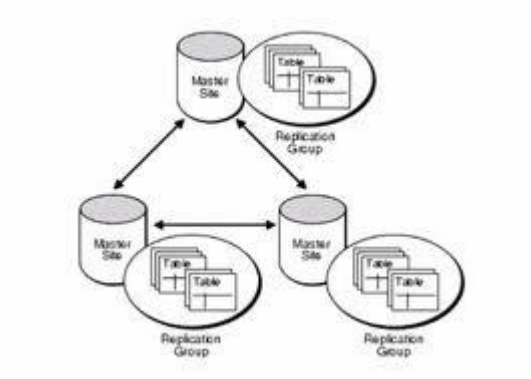
Unidad IV

Manejo de transacciones.
4.1 Transacciones.
A) Una transaccion en un sistema de gestion de bases de datos (SGBD), es un conjunto de ordenes que se ejecutan formando una unidad de trabajo, es decir, una forma indivisible o atomica.

B) Transaccion consiste en lograr hacer cualquier tipo de operacion en una base de datos, basandonos en consultas desde las mas simples hasta las de mayor grado de complejidad.

C) Transaccion se entiende en el ambito de las bases de datos en lograr hacer acciones sobre las bases de datos deseadas, logrando operaciones de ingreso, borrado, actualizacion y visualizar.



4.1.1 Estructura de transacciones.
La estructura de una transacción usualmente viene dada según el modelo de la transacción, estas pueden ser planas (simples) o anidadas.

· ***Transacciones planas:***

Consisten en una secuencia de operaciones primitivas encerradas entre las palabras clave **BEGIN** y **END**. Por ejemplo:

BEGIN _TRANSACTION Reservación
....
END.

· ***Transacciones Anidadas :***

Consiste en tener transacciones que dependen de otras, estas transacciones están incluidas dentro de otras de un nivel superior y se las conoce como subtransacciones. La transacción de nivel superior puede producir hijos (subtransacciones) que hagan más fácil la programación del sistema y mejoras del desempeño.
En las transacciones anidadas las operaciones de una transacción pueden ser así mismo otras transacciones. Por ejemplo:

```
BEGIN _TRANSACTION Reservación
                .....
                BEGIN _TRANSACTION Vuelo
                .....
              END.( Vuelo
)
            .....
                BEGIN _TRANSACTION Hotel
                .....
                END
                .....
 END.
```

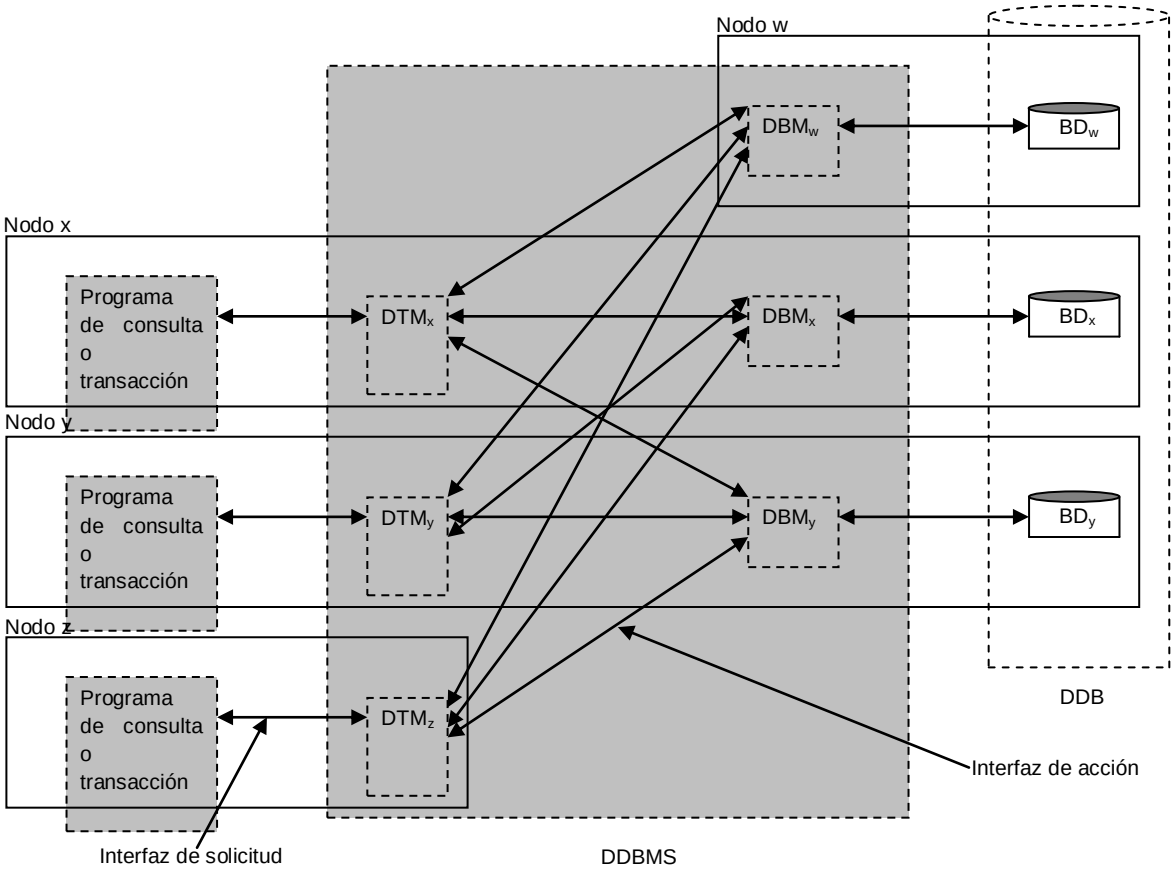
Una transacción anidada dentro de otra conserva las mismas propiedades que las de su padre, esto implica, que puede contener así mismo transacciones dentro de ella. Existen restricciones obvias en una transacción anidada: debe empezar después que su padre y debe terminar antes que el. El compromiso de una subtransaccion es condicional al compromiso de su padre, si el padre de una o varias subtransacciones aborta, las subtransacciones hijas también serán abortadas. Las transacciones anidadas brindan un nivel mas alto de concurrencia entre transacciones. Ya que una transacción consiste de varias transacciones es posible tener mayor concurrencia dentro de una sola transacción.
Así también, es posible recuperarse de de fallas de forma independiente de cada subtransaccion. Esto limita el daño a una parte mas pequeña de la transacción, haciendo que el costo de la recuperación sea el menor.

También se deben considerar el orden de las lecturas y escrituras. Si las acciones de lectura y escritura pueden ser mezcladas sin ninguna restricción, entonces, a este tipo de transacciones se les conoce como ***Generales*** .Por el contrario, si se restringe o impone que un dato debe ser leído antes de que pueda ser escrito entonces se tendrán transacciones ***Restringidas***. Si las transacciones son restringidas a que todas las acciones de lectura se realicen antes de las acciones de escritura entonces se les conoce como de ***Dos Pasos***. Finalmente existe un modelo de ***acción*** para transacciones restringidas en donde se aplica aun más la restricción de que cada par < read , write > tiene que ser ejecutado de manera atómica.

4.1.2 Ejecución de transacciones centralizada y distribuida.

El procesamiento de bases de datos distribuidas es el procesamiento de bases de datos en el cual la ejecución de transacciones y la recuperación y actualización de los datos acontece a través de dos o más computadoras independientes, por lo general separadas geográficamente. La figura 1 muestra un sistema de base de datos distribuida que involucra cuatro computadoras.

Figura 1
Arquitectura de base de datos distribuida



4.1.3 Estructura de transacciones.

- Transacciones Planas
  - Es una secuencia de operaciones primitivas entre las marcas BEGIN y END
- Transacciones Anidadas
  - Las operaciones de las transacciones pueden ser en si mismas una transacción

4.1.4 Ejecución de transacciones centralizada y distribuida.

El control de las transacciones también requiere de controlar la concurrencia del acceso y uso hacia el recurso que se esta manipulando, ese control de concurrencia tiene dos objetivos:

> ***< Como sincronizar la ejecución concurrente de transacciones***
> ***< Consistencia intra transacción (Aislamiento)***

Para llevar a cabo el control de concurrencia dentro de un proceso de transacciones se manejan dos modos:

> ***< Ejecución centralizada de transacciones*** (Figura B)
>
> ***< Ejecución distribuida de transacciones*** (Figura C)

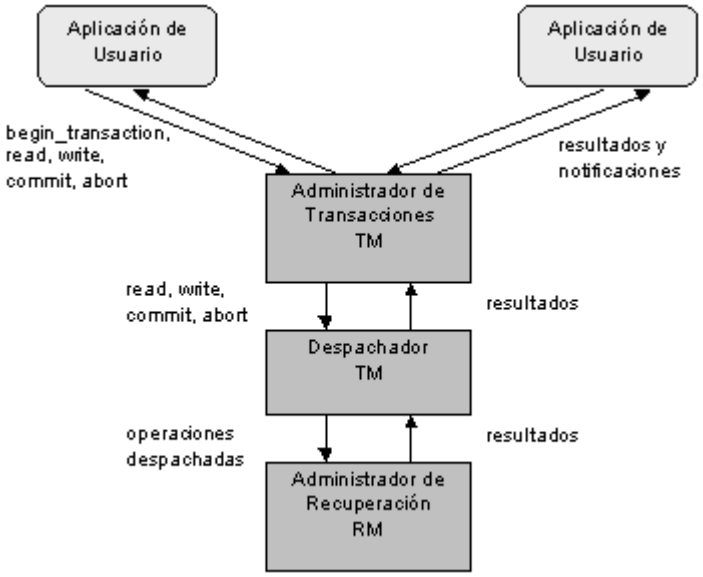


Figura B. Ejecución centralizada de transacciones.

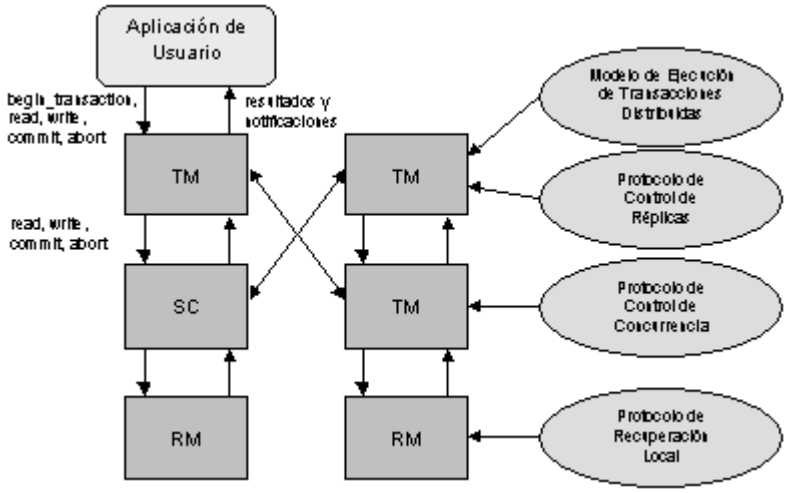


Figura C. Ejecución distribuida de transacciones.

4.2 Control de concurrencia.

El control de transacciones concurrentes en una base de datos brinda un eficiente desempeño del Sistema de Base de Datos, puesto que permite controlar la ejecución de transacciones que operan en paralelo, accesando a información compartida y, por lo tanto, interfiriendo potencialmente unas con otras.

El hecho de reservar un asiento en una avión mediante un sistema basado en aplicaciones web, cuando decenas de personas en el mundo pueden reservarlo también, nos da una idea de lo importante y crucial que es el control de concurrencia en un sistema de base de datos a mediana o gran escala.

Otro ejemplo en el que podemos observar la incidencia del control de concurrencia en el siguiente: en una Base de Datos bancaria podría ocurrir que se paguen dos cheques en forma simultánea sobre una cuenta que no tiene saldo suficiente para cubrirlos en su totalidad, esto es posible evitarlo si se tiene un control de concurrencia.

4.2.1 Serialización de transacciones.

Permite el proceso de transacciones asignándoles tiempos de procesamiento el cual permite incrementar el rendimiento del sistema ya que se ejecuta un máximo de procesos en forma concurrente y no a través de una serie. La ventaja es que a un mismo tiempo de reloj se pueden hacer dos operaciones, aunque el proceso de sincronización es mas complicado.

Un aspecto muy importante en el manejo de transacciones es el de mantener y aplicar algoritmos de control sobre los datos o recursos; para ese control también se utilizan protocolos que proporcionen confiabilidad como lo siguientes:

< Atomicidad
< Protocolos de recuperación total
< Protocolos de compromiso global

4.2.2 Algoritmos de control de concurrencia.

Deben sincronizar la ejecución de transacciones concurrentes bajo el criterio de correctitud. La consistencia entre transacciones se garantiza mediante el aislamiento de las mismas.

4.2.2.3 Pruebas de validación optimistas.

El protocolo mas reciente propuesta es el denominado optimista (OPT) el cualacentúa la pertenencia general del sistema reduciendo el bloqueo proveniente deaquellas transacciones que están preparadas para terminar pero que aun no lohicieron.Los mecanismos optimistas para el control de concurrencia fueron propuestosoriginalmente con el uso de estampas de tiempo.Sin embargo, en este tipo de mecanismos, no con los datos más aun así con lasestampas de tiempo no se asignan al inicio de una transacción sino justamente alinicio de su fase de validación. Esto se debe a que las estampas se requierenúnicamente durante la fase de validación.Los algoritmos optimistas, retrasan la fase de validación justo antes de la fase deescritura. De esta manera, una operación sometida a un despachador optimistanunca es retrasada. Las operaciones de lectura, cómputo y escrita de cadatransacción se procesan libremente sin actualizar la base de datos corriente. Cadatransacción inicialmente hace sus cambios en copias locales de los datos. La fase devalidación consiste en verificar si esas actualizaciones conservan la consistencia dela base de datos. Si la respuesta es positiva, los cambios se hacen globales. De otramanera, la transacción es abortada y tiene que reiniciar.Los mecanismos optimistas para control de concurrencia fueron propuestosoriginalmente con el uso de estampas de tiempo. Sin embargo, en este tipo demecanismos las estampas de tiempo se asocian únicamente con las transacciones,no con los datos.

Algoritmos optimistas

-

No realiza ninguna verificación durante la ejecución.

-

Los cambios se realizan sobre copias locales.

-

Al final de la ejecución existe una fase de validación que compruebaque cualquiera de las actualizaciones violaba la seriabilidad.

4.2.3 Disciplinas del Interbloqueo: prevención, detección, eliminación y recuperación.

Las estrategias de prevención de interbloqueo son muy conservadoras; resuelven el problema limitando el acceso a recursos e imponiendo restricciones sobre los procesos. En cambio, las estrategias de detección de interbloqueo, no limitan el acceso a recursos ni restringen las acciones del proceso. La detección del interbloqueo es el proceso de determinar si realmente existe un interbloqueo e identificar los procesos y recursos implicados en él. Una posibilidad detectar un interbloqueo es monitorear cada cierto tiempo el estado de los recursos. Cada vez que se solicita o se devuelve un recurso, se actualiza el estado de los recursos y se hace una verificación para observar si existe algún ciclo.
Este método está basado en suponer que un interbloqueo no ser presente y que los recursos del sistema que han sido asignados, se liberarán en el momento que otro proceso lo requiera.
Una comprobación para interbloqueo puede hacerse con igual o menor frecuencia que cada solicitud de recursos, dependiendo de que tan probable es que ocurra un interbloqueo. Comprobar cada solicitud de recursos tiene dos ventajas: Conduce a la detección temprana y el algoritmo es simple, de manera relativa porque se basa en cambios crecientes al estado del sistema. Además, las comprobaciones frecuentes consumen un tiempo considerable de procesador.
El empleo de algoritmos de detección de interbloqueo implica cierto gasto extra durante la ejecución. Así pues, se presenta de nuevo la cuestión de costeabilidad, tan habitual en los sistemas operativos. Los algoritmos de detección de interbloqueo determinan por lo general si existe una espera circular.

- **Un solo Tipo**

Usaremos una variante del grafo de asignación de recursos, llamado grafo de espera. Podemos obtener este grafo a partir del grafo de asignación de recursos, eliminando los nodos correspondientes al recurso y uniendo los arcos de forma que habrá un arco del proceso Pi al proceso Pj, si Pj tiene un recurso que Pi ha solicitado. Existirá un interbloqueo si y solo si hay un ciclo en el grafo resultante.
Para detectar un interbloqueo, el sistema necesita mantener el grafo de espera y periódicamente invocar un algoritmo que busque un ciclo en el grafo.

- **Varios tipos de recurso**

Este algoritmo de detección emplea estructuras de datos que varían con el tiempo, muy similares a las que se unas en el algoritmo del banquero:

Variable	Contenido
Disponible [m]	Número de recursos disponible de cada tipo
Asignado [n,m]	Cantidad de recursos asignados a los procesos
Petición [n,m]	Petición o solicitud actual de cada proceso

Estructuras de datos auxiliares:

Trabajo [m]: Acumula los recursos de los procesos que pueden evolucionar
Acabado[n]: Booleano que indica cuando un proceso ha acabado
Algoritmo:
Función Detección retorna Booleano
Trabajo:=Disponible
Para todo i
Sí Asignado [i] m<>0 Entonces
Acabado[i]:=False
Sino
Acabado[i]:= true
Fin Si
Fin Para
Mientras haya un i tal que Acabado [i]:=False y Petición [i]<=Trabajo
Trabajo:=Trabajo+Asignado[i]
Acabado[i]:= True
Fin Mientras
Si hay un i tal que Acabado [i]= False Entonces
Detección:=True
Sino
Detección:=False
Fin Si
El algoritmo de detección escrito se limita a investigar cada una de las posibles secuencias de asignación para los procesos que quedan por terminar. Aquellos procesos para los que Acabado[i] tengan un valor de falso, formarán parte de un interbloqueo.
Este algoritmo se puede invocar cada vez que ocurre una petición de recursos y no puede ser atendida de inmediato. Otra alternativa es invocarlo cada cierto intervalo de tiempo previamente establecido por el sistema.

Recuperación de Interbloqueo

Cuando se ha detectado que existe un interbloqueo, podemos actuar de varias formas. Una posibilidad es informar al operador que ha ocurrido un interbloqueo y dejar que el operador se ocupe de él manualmente. La otra posibilidad es dejar que el sistema se recupere automáticamente del interbloqueo. Dentro de esta recuperación automática tenemos dos opciones para romper el interbloqueo: Una consiste en abortar uno o más procesos hasta romper la espera circular, y la segunda es apropiar algunos recursos de uno o más de los procesos bloqueados.

La recuperación después de un interbloqueo se complica porque puede no estar claro que el sistema se haya bloqueado. Las mayorías de los Sistemas Operativos no tienen los medios suficientes para suspender un proceso, eliminarlo del sistema y reanudarlo más tarde.

Actualmente, la recuperación se suele realizar eliminando un proceso y quitándole sus recursos. El proceso eliminado se pierde, pero gracias a esto ahora es posible terminar. Algunas veces es necesario, eliminar varios procesos hasta que se hayan liberado los recursos necesarios para que terminen los procesos restantes.

Los procesos pueden eliminarse de acuerdo con algún orden de prioridad, aunque es posible que no existan prioridades entre los procesos bloqueados, de modo que el operador necesita tomar una decisión arbitraria para decidir que procesos se eliminarán.

- **Recuperación Manual**

Está forma de recuperación consiste en avisarle al administrador o al operador del sistema que se ha presentado un interbloqueo, y será el administrador el que solucione dicho problema de la manera más conveniente posible, de modo que su decisión no afecte demasiado a al usuario del proceso en conflicto, y sobre todo que no afecte a los demás usuarios del sistema.

- **Abortar los Procesos**

Para eliminar interbloqueos abortando un proceso, tenemos dos métodos; en ambos, el sistema recupera todos los recursos asignados a los procesos terminados.

1) Abortar todos los procesos interbloqueados. Esta es una de las soluciones más comunes, adoptada por Sistemas Operativos. Este método romperá definitivamente el ciclo de interbloqueo pero con un costo muy elevado, ya que estos procesos efectuaron cálculos durante mucho tiempo y habrá que descartar los resultados de estos cálculos parciales, para quizá tener que volver a calcularlos más tarde.

2) Abortar un proceso en cada ocasión hasta eliminar el ciclo de interbloqueo. El orden en que se seleccionan los procesos para abortarlos

debe basarse en algún criterio de costo mínimo. Después de cada aborto, debe solicitarse de nuevo el algoritmo de detección, para ver si todavía existe el interbloqueo. Este método cae enmucho tiempo de procesamiento adicional.
Quizá no sea fácil abortar un proceso. Si éste se encuentra actualizando un archivo, cortarlo a la mitad de la operación puede ocasionar que el archivo quede en un mal estado.
Si se utiliza el método de terminación parcial, entonces, dado un conjunto de procesos bloqueados, debemos determinar cuál proceso o procesos debe terminarse para intentar romper el interbloqueo. Se trata sobre todo de una cuestión económica, debemos abortar los procesos que nos representen el menor costo posible. Existen muchos factores que determinan el proceso que se seleccionará, siendo los principales los siguientes:
1) La prioridad del proceso. Se elimina el proceso de menor prioridad.
2) Tiempo de procesador usado. Se abortará aquel proceso que haya utilizado menos tiempo el procesador, ya que se pierde menos trabajo y será más fácil recuperarlo más tarde.
3) Tipos de recursos utilizados. Si los recursos son muy necesarios y escasos será preferible liberarlos cuanto antes.
4) Cuántos recursos más necesita el proceso. Es conveniente eliminar a aquellos procesos que necesitan un gran número de recursos.
5) Facilidad de suspención/reanudación.Se eliminarán aquellos procesos cuyo trabajo perdido sea más fácil de recuperar.

- **Apropiación de Recursos**

Para eliminar interbloqueos utilizando la apropiación de recursos, vamos quitando sucesivamente recursos de los procesos y los asignamos a otros hasta romper el ciclo de interbloqueo. Si se utiliza la apropiación de recursos para tratar los interbloqueos, hay que considerar tres aspectos:

- **Selección de la víctima**
- **Retroceso**
- **Bloqueo indefinido**

La detección y recuperación es la estrategia que a menudo se utiliza en grandes computadoras, especialmente sistemas por lote en los que la eliminación de un proceso y después su reiniciación suele aceptarse.

4.3 Confiabilidad.

La confiabilidad es otro requerimiento indiscutible – y probablemente el más importante. Una base de datos no confiable es simplemente inutilizable. Para la mayoría de las aplicaciones empotradas, en especial las empleadas en sistemas de tiempo real, la confiabilidad es una propiedad no negociable que deben tener todos los componentes.
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.

4.3.2 Protocolos REDO/UNDO.

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).

Considere el escenario mostrado en la Figura de abajo. 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.

4.3.3 Puntos de verificación (checkpoints).

Cuando ocurre una falla en el sistema es necesario consultar la bitácora para determinar cuáles son las transacciones que necesitan volver a hacerse y cuando no necesitan hacerse. Estos puntos de verificación nos ayudan para reducir el gasto de tiempo consultando la bitácora. El punto de verificación es un registro que se genera en la bitácora para concluir en todo lo que se encuentra antes de ese punto está correcto y verificado.

4.3.4 Protocolo 2PC de confiabilidad distribuida.

El protocolo 2PC básico un agente (un agente-DTM en el modelo) con un rol especial. Este es llamado el coordinador; todos los demás agentes que deben hacer commit a la vez son llamados participantes.
El coordinador es responsable de tomar la decisión de llevar a cabo un commit o abort finalmente. Cada participante corresponde a una subtransacción la cual ha realizado alguna acción de escritura en su base de datos local.
Se puede asumir que cada participante está en un sitio diferente. Aun si un participante y el coordinador se encuentran en el mismo sitio, se sigue el protocolo como si estuvieran en distintos sitios.
La idea básica del 2PC es determinar una decisión única para todos los participantes con respecto a hacer commit o abort en todas las subtransacciones locales.

El protocolo consiste en dos fases:

- La primera fase tiene como objetivo alcanzar una decisión común,
- La meta de la segunda fase es implementar esta decisión.

El protocolo procede como sigue:
Fase uno:
• El coordinador escribe “prepare” en la bitácora y envía un mensaje donde pregunta a todos los participantes si preparan el commit (PREPARE).
• Cada participante escribe “ready” (y registra las subtransacciones) en su propia bitácora si está listo o “abort” de lo contrario.
• Cada participante responde con un mensaje READY o ABORT al coordinador.
• El coordinador decide el commit o abort en la transacción como un resultado de las respuestas que ha recibido de los participantes. Si todos respondieron READY, decide hacer un commit. Si alguno ha respondido ABORT o no ha respondido en un intervalo de tiempo determinado se aborta la transacción.
Fase dos:
• El coordinador registra la decisión tomada en almacenamiento estable; es decir, escribe “global_commit” o “global_abort” en la bitácora.
• El coordinador envía mensaje de COMMIT o ABORT según sea el caso para su ejecución.
• Todos los participantes escriben un commit o abort en la bitácora basados en el mensaje recibido del coordinador (desde este momento el procedimiento de recuperación es capaz de asegurar que el efecto de la subtransacción no será perdido).
Finalmente:

- Todos los participantes envían un mensaje de acuse de recibo (ACK) al coordinador, y ejecutan las acciones requeridas para terminar (commit) o abortar (abort) la subtransacción.
- Cuando el coordinador ha recibido un mensaje ACK de todos los participantes, escribe un nuevo tipo de registro en la bitácora, llamado un registro “completo”.