TRANSACCIÓN: Transacción: Es una unidad lógica de procesamiento de la Base de Datos que incluye una o más operaciones de acceso a la Base de datos, que pueden ser de inserción, eliminación, modificación o recuperación.

El problema de controlar la concurrencia surge cuando varias transacciones enviadas por varios usuarios interfieren entre sí de un modo que produce unos resultados incorrectos

Los **sistemas de procesamiento de transacciones** son sistemas con grandes B. D. y cientos de usuarios concurrentes ejecutando transacciones de bases de datos (reservas en aerolíneas, banca, procesamiento de tarjetas de crédito, etc.)
□ Requieren una alta disponibilidad y una respuesta rápida para cientos de usuarios simultáneos.
□ Un criterio que sirve para clasificar un sistema de B. D.es el número de usuarios que lo pueden utilizar al mismo tiempo.

□ Un DBMS es **monousuario** si sólo lo puede utilizar un usuario a la vez (principalmente restringidos a los sistemas de computación personal).
□ Un DBMS es **multiusuario** si varios usuarios pueden utilizar el sistema (y, por tanto, acceder a la base de datos) simultáneamente.
□ A la B. D. Pueden acceder simultáneamente múltiples usuarios debido a la **multiprogramación**, que permite al computador ejecutar varios procesos al mismo tiempo. Si sólo hay una CPU, realmente sólo se ejecuta un proceso a la vez.

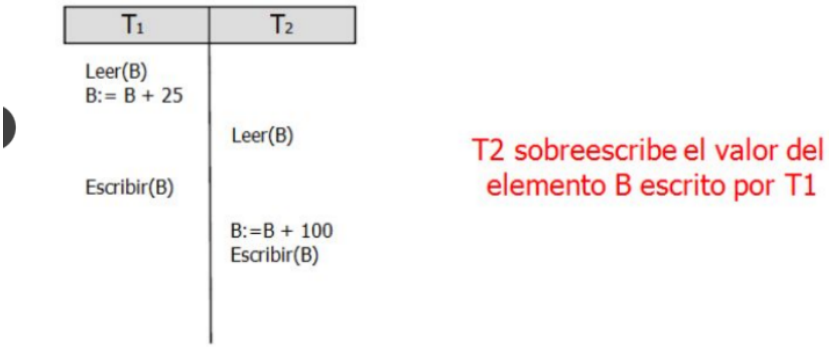
Sin embargo los S. O. multiprogramación con una CPU ejecutan algunos comandos de un proceso, después suspenden ese proceso y ejecutan algunos comandos del siguiente proceso, y así sucesivamente. El proceso se reanuda en el mismo punto en que se suspendió, siempre que consiga turno para utilizar de nuevo la CPU. Por lo tanto, la ejecución de procesos realmente es **interpolada.**
□ Si el computador tiene varias CPUs, es posible el procesamiento paralelo de varios procesos.
□ **U**na transacción se inicia por la ejecución de un programa de usuario escrito en un lenguaje de manipulación de datos de alto nivel o en un lenguaje de programación (SQL, Java) y está delimitado por instrucciones begin transaction y end transaction.
□ Un programa de aplicación puede contener más de una transacción si contiene varios límites de transacción.
□ Si las operaciones de B. D. de una transacción no la actualizan, sino que únicamente recuperan datos, se dice que la transacción es de sólo lectura.

Base de Datos: está representada básicamente como una colección de elementos de datos con nombre.

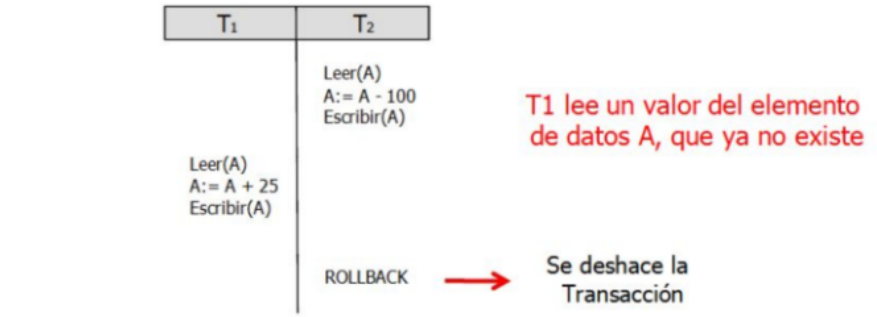
OPERACIONES BÁSICAS DE UNA TRANSACCIÓN:
□ **Leer/Read (X):** Transfiere el dato X de la base de datos a una memoria intermedia local perteneciente a la transacción que ejecuta la operación leer.
□ **Escribir/Write (X)**: Transfiere el dato X desde la memoria intermedia local de la transacción que ejecuta la operación escribir a la base de datos

Si no se controla la concurrencia pueden ocurrir 3 cosas:

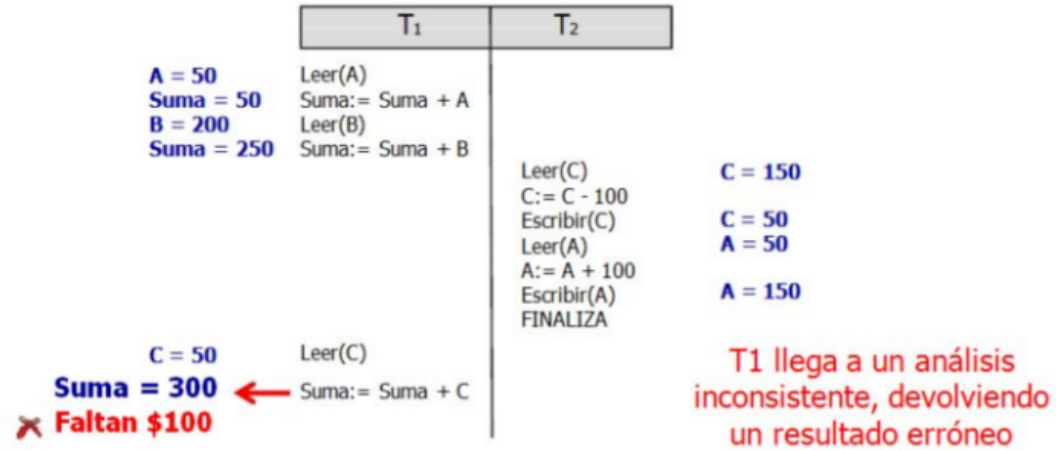
- **Actualización perdida:** cuando dos transacciones que intentan modificar un elemento de datos,ambas leen el valor antiguo del elemento, una de ellas (T1) actualiza el dato, pero esa actualización se pierde dado que la otra transacción (T2) sobreescribe ese valor sin siquiera leerlo



- **Lectura sucia o dependencia no confirmada:** cuando una transacción T1 lee o actuliza un elemento de datos que ha sido actualizado por otra transacción T2 que aun no ha sido confirmada. Por lo tanto, existe la posibilidad de que se deshaga T2 y T1 haya visto un valor que ya no existe. T1 opera sobre una suposición falsa



- **Análisis inconsistente:** se produce cuando un transacción T1, producto de haber leído un dato actualizado por otra transacción T2 ya finalizada, incurre en un análisis inconsistente



¿Por qué es necesario la recuperación?

- siempre que se envía una transacción a un DBMS para su ejecución, el sistema es responsable de garantizar que todas las operaciones de la transacción se completen

satisfactoriamente y que su efecto se graba permanentemente en la BD o de que la transacción no afecte a la BD o a cualquier otra transacción
- El DBMS no debe permitir q algunas operaciones de una transacción T se apliquen a la BD, mientras que otras no. Esto puede ocurrir si una transacción falla después de ejecutar algunas de sus operaciones, pero antes de ejecutar todas ellas

Una transacción puede fallar por:
- Caída del sistema (error de hardware, software o red)
- Un error de la transacción (división por 0)
- Errores locales o condiciones de excepción detectadas por la transacción (no se encuentran los datos)
- Control de concurrencia (estado de bloqueo)
- Fallo del disco
- Problemas físicos y catástrofes (cortes de luz, incendio)

Estado de las transacciones:
- **Activa:** inmediatamente después de iniciarse su ejecución, puede emitir operaciones LEER y ESCRIBIR.
- **Parcialmente confirmada:** usa protocolos para comprobar que los cambios de la transacción sean efectivos y una vez comprobados se dice que alcanzó su punto de confirmación y entra en el estado de confirmado
- **Falla:** si falla alguna de las comprobaciones o si la transacción es cancelada durante su estado activo
- **Terminado:** es cuando la transacción abandona el sistema

Propiedades de las transacciones: ACID
- **ATOMICIDAD (A) :** una T es una unidad atómica de procesamiento, o se ejecuta en su totalidad o no se ejecuta
- **CONSISTENCIA © :** una T está conservando la consistencia si su ejecución completa lleva a la BD de un estado consistente a otro
- **AISLAMIENTO (I) :** una T debe aparecer como si estuviera ejecutándose de forma aislada a las demás. Es decir, la ejecución de una transacción no debe interferir con la ejecución de ninguna otra T simultánea
- **DURABILIDAD (D) :** los cambios aplicados a la BD por una T confirmada deben persistir en la BD. Estos cambios no deben perderse por culpa de un fallo

PLANIFICACIÓN: es un ordenamiento global de las operaciones de esas transacciones que respete el orden interno de las operaciones dentro de cada transacción. Las operaciones de cada T pueden interpolar. Si no se permite la interpolación las operaciones de cada T se deben ejecutar consecutivamente

PLANIFICACION SERIE: por cada T que participa en la planificación , todas las operaciones de T se ejecutan consecutivamente en la planificación. En caso contrario se dice que es **no serie.** En una planificación serie solo hay una transacción activa al mismo tiempo

Una planificación S de n transacciones es serializable si es equivalente a alguna planificación en serie de las mismas n transacciones

A partir de n transacciones es posible generar n! planificaciones ens erie y muchas mas planificaciones no serie
Se pueden formar dos grupos distintos con las planificaciones no serie:

- Aquellos que son equivalentes a una (o más) planificaciones en serie y, por tanto, serializables
- Aquellas que no son equivalentes a ninguna planificación en serie y.por tanto, no son serializables

□ Decir que una planificación no serie S es serializable es equivalente a decir que es correcta, porque es equivalente a una planificación en serie, que se considera correcta.
□ ¿Cuándo se considera que dos planificaciones son equivalentes? Analizaremos la equivalencia por conflicto.
□ Dos planificaciones son **equivalentes por conflicto** si el orden de cualquier par de operaciones en conflicto es el mismo en las dos planificaciones.

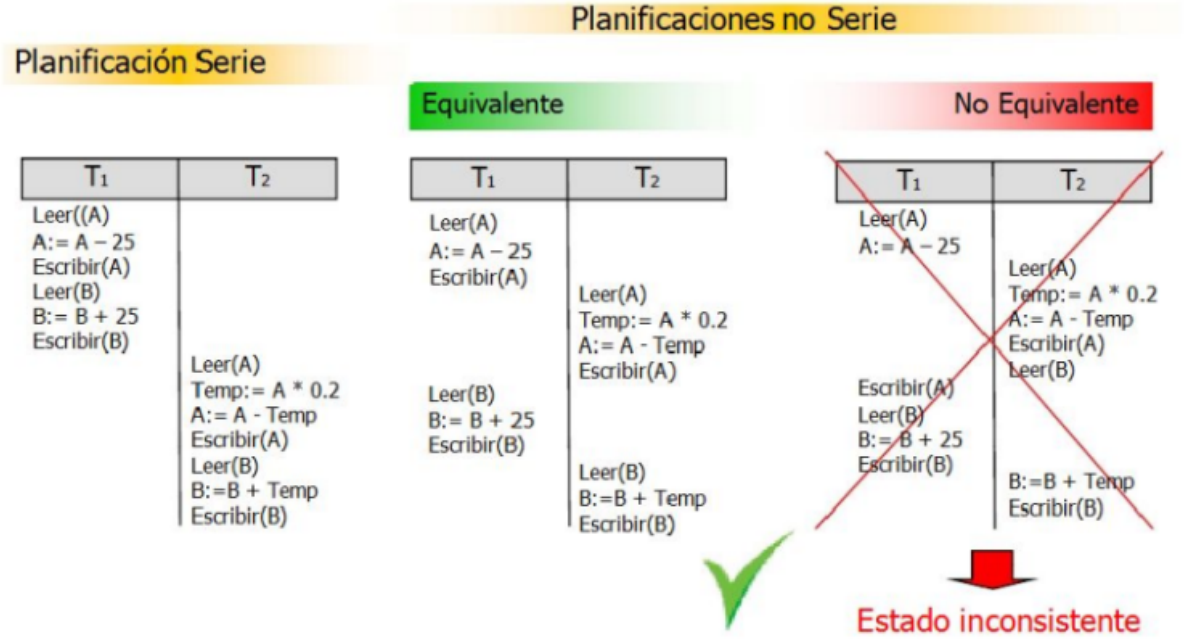
□ Si las instrucciones operan sobre datos diferentes, no generan conflicto. Supongamos I1 e I2 instrucciones de T1 y T2 respectivamente que operan sobre el dato D.

- Si I1=READ(D) e I2=READ(D), no hay conflicto.
- Si I1=READ(D) e I2=WRITE(D), hay conflicto.
- Si I1=WRITE(D) e I2=READ(D), hay conflicto.
- Si I1=WRITE(D) e I2=WRITE(D), hay conflicto.

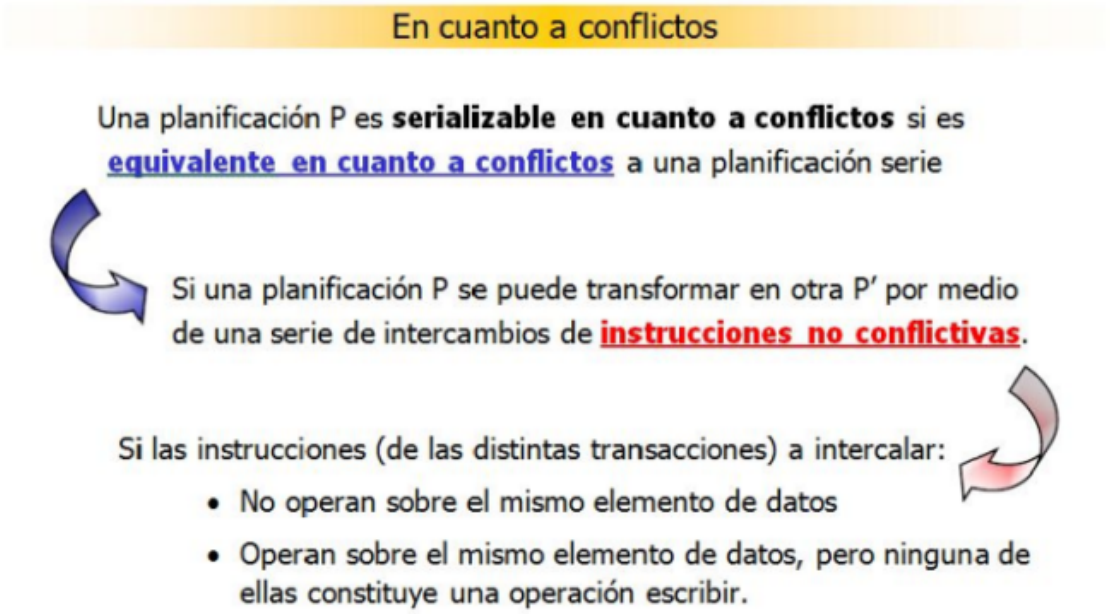
□ En conclusión, dos instrucciones son conflictivas si operan sobre el mismo dato, y al menos una de ellas es una operación de escritura.

Grafo de precedencia: es la forma de graficar las transiciones entre cada transacción.
Si en el grafo de precedencia aparece un ciclo (es cuando el nodo inicial de cada arco es el mismo en el nodo final del arco anterior y el nodo inicial del primer arco es el mismo que el nodo final del último no) la planificacion S no es serializable (por conflicto); si no hay ciclo S es serializable

Como aspecto positivo tenemos: mayor productividad, mejor utilización de los recursos, tiempo de espera reducido
Como aspecto negativo tenemos mayor probabilidad de inconsistencias



PLANIFICACIONES SERIALIZABLES EN CUANTO A CONFLICTOS:



CONTROL DE CONCURRENCIA: es cuando se ejecutan varias transacciones concurrentemente en la BD, puede que deje de conservarse la propiedad de aislamiento Es necesario que el sistema controle la interacción entre las transacciones concurrentes; dicho control se lleva a cabo a través de uno de los mecanismos existentes llamado **esquema de control** de concurrencia. Los **esquemas de control** que analizaremos aseguran que las planificaciones son serializables.

Hay dos formas de implementar el aislamiento:

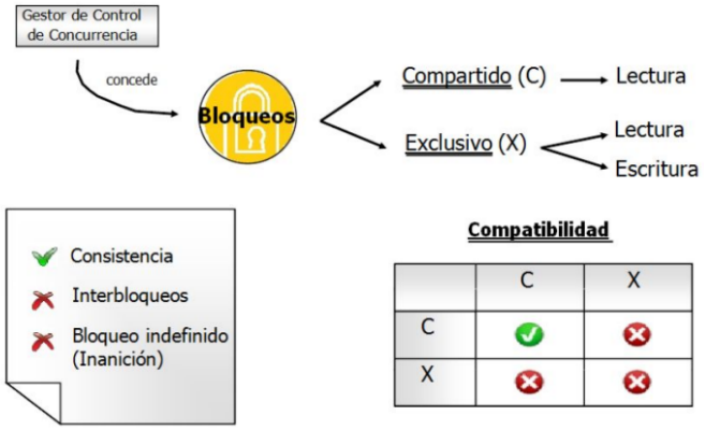
- **Protocolo de bloqueo**

- **Protocolo basado en marcas temporales**

Analizaremos dos tipos de bloqueo a realizar sobre un elemento de datos:

- **Bloqueo compartido:** es realizado por una transacción para leer un dato que no modificara. Mientras dure este bloqueo, se impide que otra transacción pueda escribir el mismo dato
- **Bloqueo exclusivo:** se genera cuando una transacción necesita escribir un dato. Este bloqueo garantiza que otra transacción no pueda utilizar ese dato (ni lectura ni escritura)

- ☐ Un bloqueo compartido puede coexistir con otro bloqueo compartido sobre el mismo dato. Por ej, si dos transacciones requieren leer el dato D, ambas transacciones deben impedir que otra transacción pueda escribir ese dato; como ambas solo quieren leerlo, no se genera conflicto entre ellas
- ☐ Un bloqueo exclusivo es único. Una transacción que modifica un dato requiere que ninguna otra esté operando con ese dato en ese momento
- ☐ Cada transacción debe solicitar al gestor de control de concurrencia el tipo de bloqueo que necesite. Si obtiene el dato, debe utilizarlo y luego liberarlo
- ☐ Una transacción que solicita un bloqueo y no lo obtiene puede quedar en situación de espera. Si se posee certeza de que el dato bloqueado será rápidamente liberado, es posible esperar. En caso contrario, resulta conveniente que la transacción falle. En general el porcentaje de transacciones que fallan por problemas de bloqueo es menor

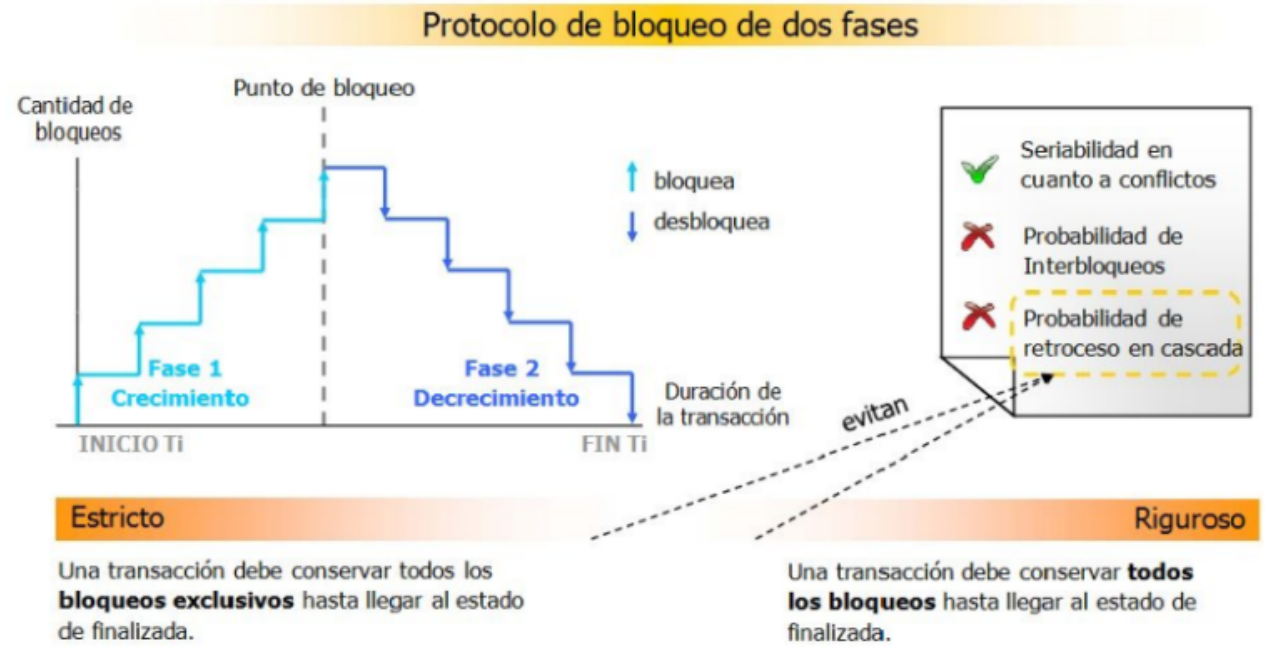


Protocolos de bloqueo: me da las pautas que debe tener una planificación P para que sea legal bajo un protocolo de bloqueo dado si P es una planificación posible para un conjunto de transacciones que sigan las reglas del protocolo. Un protocolo asegura la secuencialidad en cuanto a conflictos si y sólo si todas las planificaciones legales son serializables en cuanto a conflictos.

- **Protocolos de bloqueo de dos fases:** exige que cada transacción realice las peticiones de bloqueo. No asegura la ausencia de interbloqueos. También puede ocurrir el retroceso en cascada.
    1. **Fase crecimiento:** una transacción puede obtener bloqueos pero no puede liberarlos

2. **Fase de decrecimiento:** una transacción puede liberar bloqueos pero no puede obtener ninguno nuevo

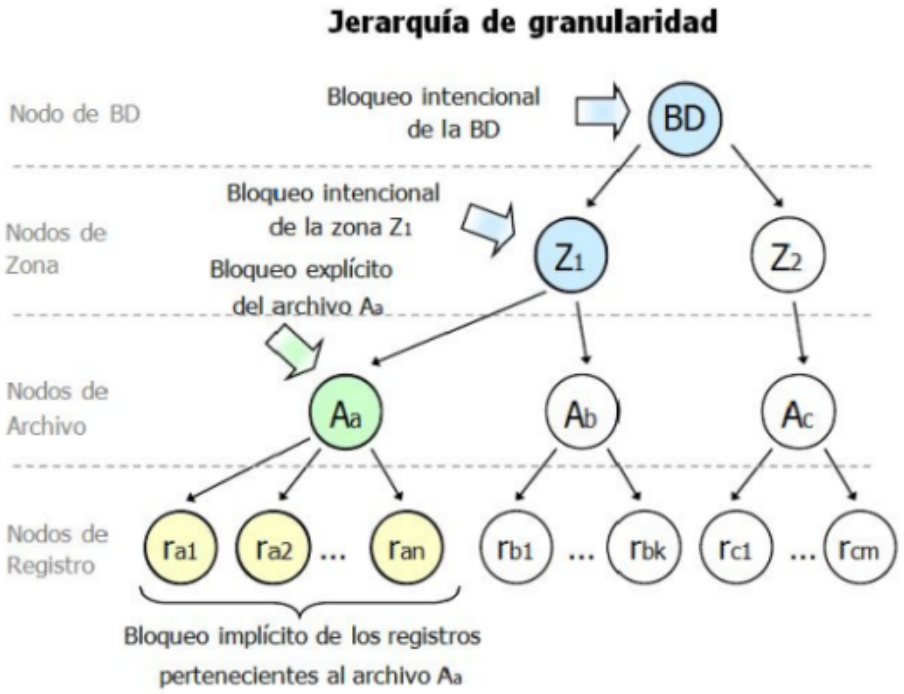
El retroceso en cascada se puede evitar utilizando el **protocolo de bloqueo estricto de dos fases.** Este protocolo exige que una transacción deba poseer todos los bloqueos en modo exclusivo que tome hasta que dicha transacción se finalice
Una variante es el **protocolo de bloqueo riguroso de dos fases**, el cual exige que se posean todos los bloqueos hasta que finalice la transacción



Protocolo basado en marcas temporales: se determina el orden entre 2 transacciones conflictivas en tiempo de ejecución a través del primer bloqueo conflictivo que soliciten ambas. Otra forma de determinar el orden de seriabilidad es seleccionar previamente un orden entre las transacciones. El método más común para hacer esto es utilizar un esquema de ordenación por marcas temporales.
Existen dos métodos simples para implementar este esquema:

- Usar el valor del **reloj del sistema** como marca temporal
- Usar un **contador lógico** que se incrementa cada vez que se asigna una nueva marca temporal

GRANULARIDAD: es el tamaño de los elementos de datos. Se necesita un mecanismo que permita definir múltiples niveles de granularidad. Se puede construir uno permitiendo que los elementos de datos sean de varios tamaños y definiendo una jerarquía de granularidades de los datos, en la cual las granularidades pequeñas están anidadas en otras más grandes



Tiene como característica:

- Se puede bloquear individualmente cada nodo del árbol
- Cuando una transacción bloquea un nodo también bloquea todos los descendientes con el mismo modo de bloqueo
- Si un nodo se bloquea en modo intencional se está haciendo un bloqueo explícito en un nivel inferior del árbol
- Los bloqueos intencionales se colocan en todos los ascendientes de un nodo antes de bloquearlo explícitamente, así no es necesario que una transacción busquen en todo el árbol
- Si una transacción quiere bloquear un nodo N debe recorrer el camino en el árbol desde la raíz hasta N y durante el recorrido debe ir bloqueando los distintos nodos en modo intencional
- Granularidad fina indica elementos de tamaño pequeño, en tanto que granularidad gruesa designa elementos grandes. A la hora de elegir el tamaño de los elementos de datos debo analizar ventajas y desventajas. A mayor tamaño menor será el grado de concurrencia permitido. A menor tamaño habrá más elementos en la B, esto ocasionará más operaciones de bloquear y desbloquear, dando lugar a una mayor sobrecarga además de requerir más espacio de almacenamiento para la tabla de bloqueos. El mejor tamaño para los elementos va a depender del tipo de transacciones

INTERBLOQUEOS: Existe cuando un conjunto de transacciones está esperando un elemento de datos bloqueado por otra transacción del conjunto

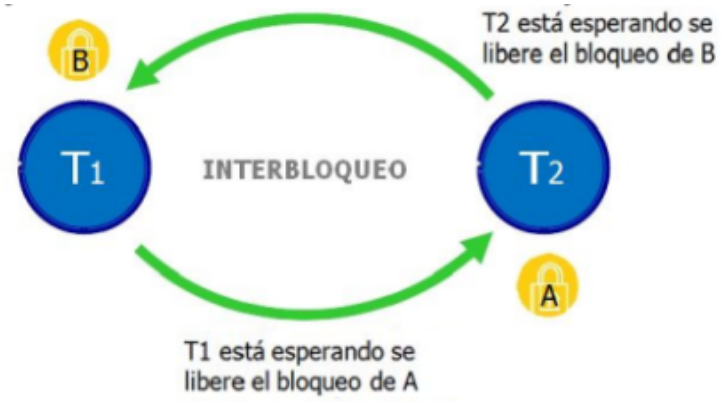
- Son un mal necesario si se quieren evitar estados inconsistentes
- Es preferible tener interbloqueos que estados inconsistentes porque el interbloqueo se puede tratar haciendo retroceder las transacciones

PREVENCIÓN DE INTERBLOQUEOS:

- **Protocolo de prevención de interbloqueo:** se utiliza en el bloqueo de dos fases. Requiere que cada T bloquee con antelación todos los elementos que necesita, si alguno de esos elementos no puede obtenerse ninguno de los elementos se bloquea. Limita la concurrencia
- **Marco de tiempo de transacción TS(T):**
- **Esperar morir:** una T más antigua tiene que esperar a una T mas nueva, mientras que una T mas nueva que solicita un elemento que posee una T más antigua es abortado y reiniciado
- **Herir-esperar:** un T mas nueva tiene q esperar a una T mas vieja, mientras que una T mas vieja que solicita un elemento que posee una T mas nueva se apropia de la T nueva abortando

Ambos terminan con la cancelación de la T más nueva. Están libres de Interbloqueos. Se utiliza en T largas y cada una utiliza muchos elementos

Una forma de detectar un estado de interbloqueo es mediante un grafo de espera:



Cuando libera los bloqueos el arco desaparece del grafo de espera
Tenemos un estado de interbloqueo si el grafo de espera tiene un ciclo
Un problema de este método es determinar cuándo un sistema debe comprobar si hay interbloqueo
Si el sistema se encuentra en estado de interbloqueo algunas T q lo provocan deben abortarse. Para seleccionar cual abortar se utiliza una técnica llamada **selección de la víctima** (evita la selección de T que han estado ejecutando durante mucho tiempo y q han realizado muchas actualizaciones)

INTERBLOQUEOS. TIEMPOS LIMITADOS: si una T lleva en esperó un tiempo superior a un tiempo definido por el sistema, este asume que la T puede estar bloqueada y la elimina, no importa si existe o no un interbloqueo

TÉCNICAS DE RECUPERACIÓN DE BASES DE DATOS: técnicas de recuperación basadas en la actualización diferida e inmediata, paginación a la sombra
Cuando falla una T hay que llevar a cabo las siguientes acciones para asegurar la integridad de la BD:

- Una acción es volver a ejecutar la T fallada. El problema surge cuando la nueva T comienza de un estado de inconsistencia. Por lo tanto, volver a ejecutar una transacción que anteriormente fallo no retrotrae la BD a un estado de consistencia

- otra acción es no hacer nada, pero la BD permanecerá inconsistente

CONCLUSIÓN: reejecutar o no una transacción fallida no asegura la consistencia de la BD. Es necesario realizar acciones que permitan retrotraer el estado de la BD al instante anterior al comienzo de ejecución de la transacción. Estas acciones tienen que ver con dejar constancia de las actividades llevadas a cabo por la transacción que permitiran deshacer los cambios producidos

Para recuperarse del fallo de una T normalmente significa que la B. D. se restaura al estado coherente más reciente, inmediatamente anterior al momento del fallo.Para esto, el sistema debe guardar información sobre los cambios que las distintas transacciones aplicaron a los elementos de datos. Esta información normalmente se guarda en el registro del **sistema (también llamado registro del sistema o bitácora).** Este registro se guarda en el disco y periódicamente se realiza una copia de seguridad del registro para protegerse contra fallos catastróficos.
Las entradas del registro son:
[start_transaction, T] Indica que se ha iniciado la ejecución de la transacción T.
[write_item, T, X, valorViejo, valorNuevo] Indica que la transacción T ha cambiado el valor del elemento X de valorViejo a valorNuevo.
[commit, T] Indica que la transacción T se ha completado satisfactoriamente, y afirma que su efecto puede grabarse permanentemente en la B. D.
[abort, T] Indica que la transacción T se abortó.
Otro tipo de entrada en el registro es el **punto de control (checkpoint).** En el registro del sistema se escribe periódicamente un registro [checkpoint] en el punto en que el sistema escribe en la base de datos en disco todos los búferes del DBMS que se han modificado.El gestor de recuperaciones de un DBMS debe decidir a qué intervalos tomar un punto de control. El intervalo puede medirse en tiempo o como un número t de transacciones

Toma de un punto de control:

1. Suspender temporalmente la ejecución de las transacciones.
2. Forzar la escritura en disco de todos los búferes de memoria que se hayan modificado.
3. Escribir un registro [checkpoint] en el registro del sistema, y forzar la escritura del registro en disco.
4. Reanudar la ejecución de transacciones.

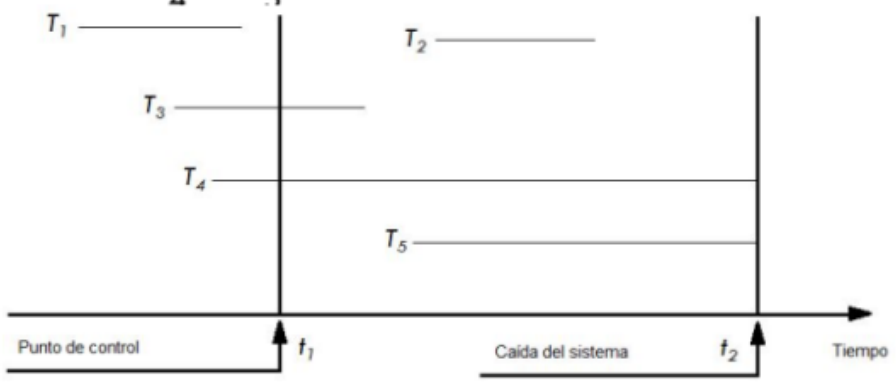
La idea es posponer las actualizaciones de la BD hasta q la T complete su ejecución satisfactoriamente
Durante la ejecución de la T las actualizaciones se graban en la bitácora y en los búferes
Una vez q la T alcanza su punto de confirmación y se fuerza a la escritura del registro en disco, las actualizaciones se graban en la BD

Un protocolo de actualización diferida típico es:

1. Una transacción no puede modificar la B. D. en disco hasta haber alcanzado su punto de confirmación.
2. Una transacción no alcanza su punto de confirmación hasta que todas sus operaciones de actualización se han grabado en el registro del sistema y éste se ha escrito en el disco.

El algoritmo de recuperación utiliza el procedimiento REHACER:
Utiliza dos listas de transacciones: una con las transacciones confirmadas T desde el último punto de control, y otra con las transacciones activas T´.
Aplica la operación REHACER a todas las operaciones write_item de las transacciones confirmadas del registro del sistema en el orden en que se escribieron en el registro. Las transacciones que están activas y que no se confirmaron son efectivamente canceladas y deben reenviarse.

Según el algoritmo de recuperación, no hay necesidad de rehacer las operación write_elemento de T_1. Sin embargo, las operaciones write_elemento de T_2 y T_3 deben rehacerse, ya que alcanzaron sus puntos de confirmación después del último checkpoint. Las transacciones T_4 y T_5 son ignoradas.



Cuando una transacción emite un comando de actualización, la B. D. puede actualizarse inmediatamente, sin que haya que esperar a que la transacción alcance su punto de confirmación.
□ No obstante, una operación de actualización tiene que grabarse en el registro del sistema (en disco) antes de que se aplique a la B. D. de modo que podamos recuperarnos de un fallo.
□ Hay que prepararse para deshacer el efecto de las operaciones de actualización aplicadas a la B. D. por transacciones fallidas.

□ El **algoritmo de recuperación:**

1. Utiliza dos listas de transacciones: una con las transacciones confirmadas desde el último punto de control,y otra con las transacciones activas.
2. Deshace todas las operaciones write_elemento de las transacciones activas, utilizando el procedimiento **DESHACER**. Las operaciones deben deshacerse en el orden inverso al orden en que se escribieron en el registro del sistema.
3. Rehace todas las operaciones write_item de las transacciones confirmadas a partir del registro del sistema, en el orden en que se escribieron en dicho registro, utilizando el procedimiento **REHACER.**

PAGINACIÓN A LA SOMBRA:

□ El directorio se guarda en la memoria principal si no es demasiado grande, y todas las referencias (lecturas y escrituras) a las páginas de la B. D. en disco pasan por él.

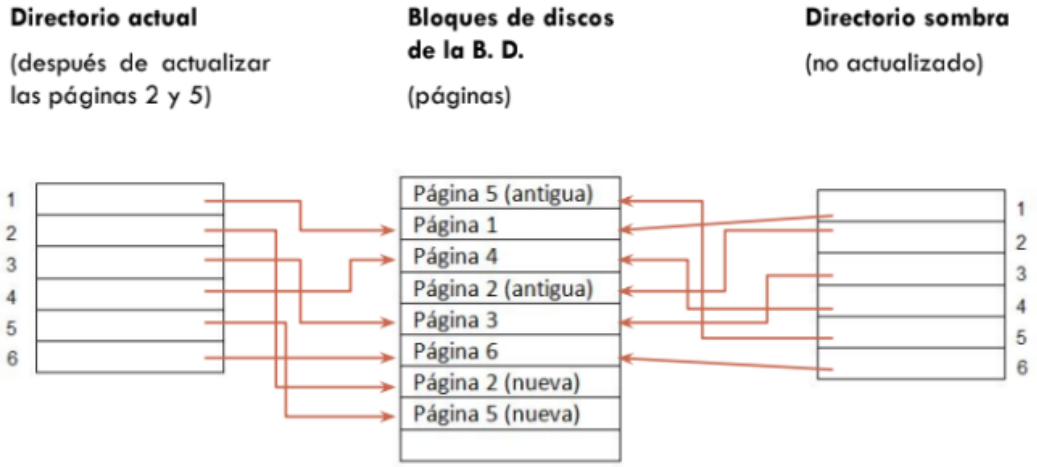
□ Cuando empieza la ejecución de una transacción, el directorio actual (cuyas entradas apuntan a las páginas más recientes o últimas de la B. D. en disco) se copia en un directorio sombra.

□ El directorio sombra se guarda en el disco, mientras que la transacción utiliza el directorio actual.

□ Durante la ejecución de la transacción, el directorio sombra nunca se modifica.

□ Cuando se efectúa una operación write_item, se crea una nueva copia de la página de B. D. modificada, pero no se sobrescribe la copia antigua. En vez de ello, la nueva página se escribe en algún bloque de disco que no se ha utilizado anteriormente.

□ La entrada del directorio actual se modifica de modo que apunte al nuevo bloque de disco, mientras que el directorio sombra no se modifica y sigue apuntando al antiguo bloque de disco, no modificado.



- Se guardan dos versiones de las páginas actualizadas. La versión antigua es referenciado por el directorio sombra y la versión nueva por el directorio actual.

□ Para recuperarse ante un fallo durante la ejecución de una transacción, es suficiente con liberar las páginas modificadas de la B. D. y descartar el directorio actual.

□ El estado de la B. D. anterior a la ejecución de la transacción está disponible a través del directorio sombra, y ese estado se recupera restableciendo el directorio sombra.

□ La confirmación de una transacción corresponde a descartar el directorio sombra anterior.