



Departamento de
Ciencias de la Computación y Tecnologías de Información
Universidad del Bío-Bío
Sede Chillán

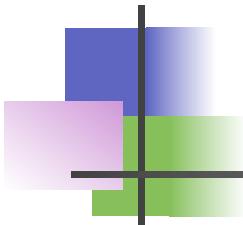
Bases de Datos:

Administración de Transacciones

M^a Angélica Caro Gutiérrez

<http://www.face.ubiobio.cl/~mcaro/>

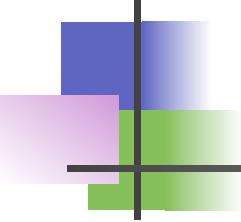
mcaro@ubiobio.cl



Administración de Transacciones



- Introducción
- Transacciones
- Control de Conurrencia
- Recuperación de BD

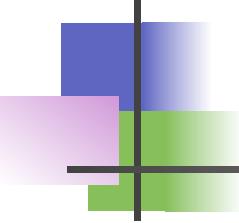


Introducción

- Los sistemas informáticos fallan...



- Falla suministro energía, falla disco, errores del software, etc.

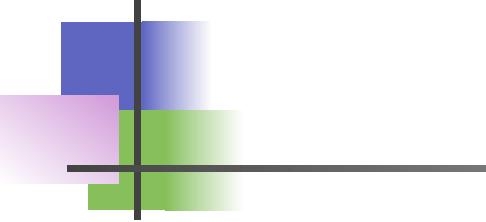


Introducción

- ¿Consecuencia de una falla?



Pérdida de Datos!!!



Introducción

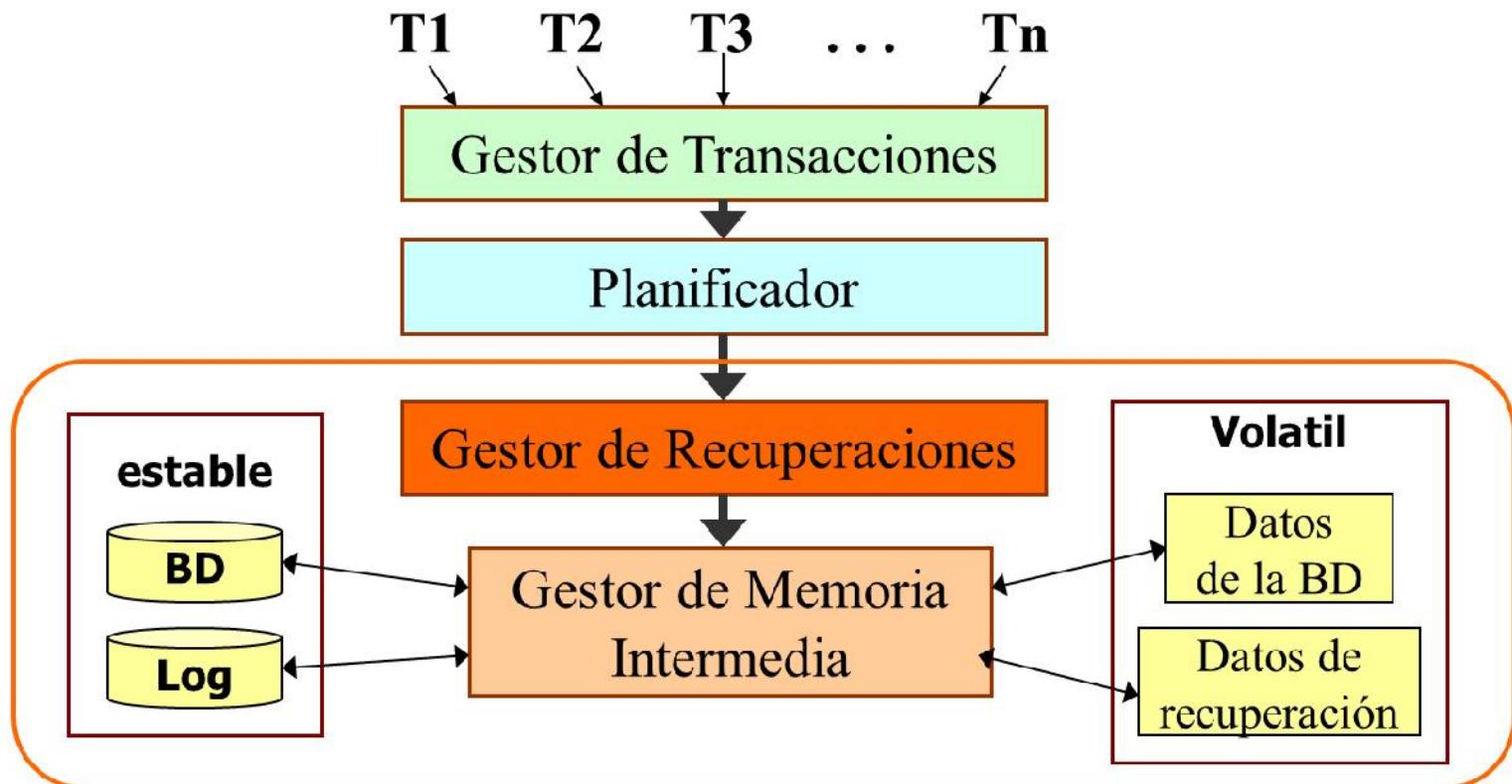


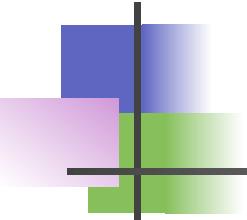
■ SGBD:

- Posee un esquema de recuperación que elimina las fallas y restaura la BD.
- Debe asegurarse de que BD vuelva a quedar en el estado consistente antes de la falla.
- En este contexto, un concepto clave es el de transacción, que representa una unidad atómica de trabajo simple en la BD.
- Una transacción es una colección de operaciones que realiza una única función lógica en una BD.
- Cada transacción es atómica, lo que es vital ante fallas.

Introducción

- SGBD:



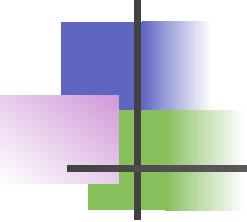


Introducción

■ Almacenamiento:

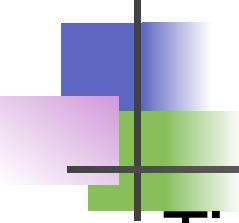
- Volátil (RAM)
- No Volátil (Disco)
- Estable (Varios dispositivos no volátiles)





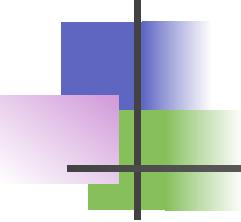
Introducción

- Tipos de Fallas en un sistema de BD:
 - **Errores locales previstos por la aplicación (rollback explícito o programado)** Durante la ejecución de una transacción pueden presentarse condiciones que requieran la cancelación de la misma (por ejemplo, un saldo insuficiente en una cuenta bancaria).
 - **Errores locales no previstos (overflow, división por cero...)** Errores lógicos de programación (bugs), o interrupciones provocadas (ctrl-C). Una falla local sólo afecta a la transacción que se está ejecutando donde ha ocurrido la falla. Normalmente supone la pérdida de los datos en memoria principal y en los búferes de E/S.



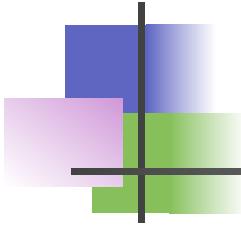
Introducción

- Tipos de Fallas en un sistema de BD:
 - **Fallas por imposición del control de concurrencia** El Subsistema de Control de Concurrencia puede decidir abortar una transacción T (para reiniciarla posteriormente) porque viola la seriabilidad o porque varias transacciones están en un estado de bloqueo mortal y T es la víctima.
 - **Fallas del sistema (caídas suaves)** Consisten en un mal funcionamiento del Hw, errores Sw (del SGBD o del SO) o de red, que afectan a todas las transacciones en progreso. No dañan físicamente el disco (el contenido de la BD permanece intacto y no se corrompe), pero se pierden los datos en la memoria principal y en los búferes de E/S.



Introducción

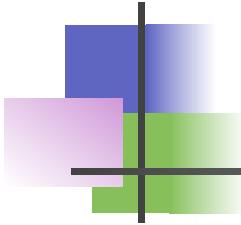
- Tipos de Fallas en un sistema de BD:
 - **Fallas de disco (caídas duras)** Son fallas en los dispositivos de almacenamiento (por ejemplo, si ocurre una rotura o un aterrizaje de alguna cabeza lectora-escritora en el disco). Afectan a todas las transacciones en curso y suponen la pérdida de datos en disco (es decir, algunos bloques del disco pueden perder sus datos).
 - **Fallas catastróficas o físicas** Algunos ejemplos son el corte del suministro eléctrico, robo del disco, incendio, sabotaje, sobreescritura en discos o cintas por error, etc. **Estas fallas suceden con menos frecuencia que el resto, pero su recuperación es más complicada.**



Administración de Transacciones



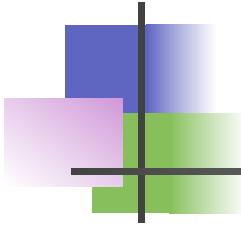
- Introducción
- Transacciones
- Control de Conurrencia
- Recuperación de BD



Administración de Transacciones

■ Concepto de Transacción:

- Una **transacción** es una unidad **atómica de ejecución**, es decir, una secuencia de operaciones que, o bien se realizan todas, o bien no se realiza ninguna.
- Por definición, una BD se encuentra en un estado consistente antes y después de la ejecución de una transacción.
- Las transacciones son el fundamento de la **ejecución concurrente** y la **recuperación de una falla del sistema** en un SGBD.



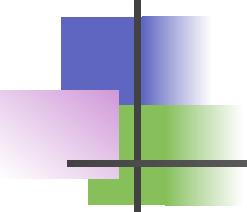
Administración de Transacciones

- ¿Qué es ejecución concurrente?
- Algunos ejemplos de sistemas que pueden funcionar con ejecución concurrente:

- Sistema Bancario

¿Cómo se da la concurrencia?...





Administración de Transacciones

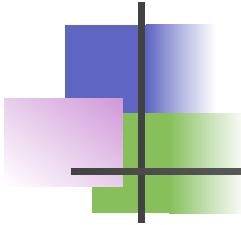
- Comente los problemas de concurrencia que se pueden presentar los siguientes escenarios :



Gestión de inventarios



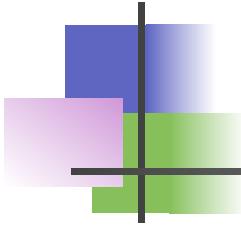
Cajeros automáticos



Administración de Transacciones

■ Concepto de Transacción:

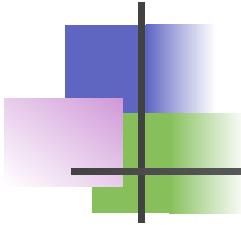
- Una transacción es una unidad de programa que accede y posiblemente actualiza varios elementos de datos.
- La transacción lee una sola vez cada elemento de datos y en caso que lo vaya a actualizar escribe como mucho una vez cada dato.
- Durante la ejecución de una transacción puede ser necesario permitir inconsistencia temporal.
- Lo anterior es complejo ante alguna falla.



Administración de Transacciones

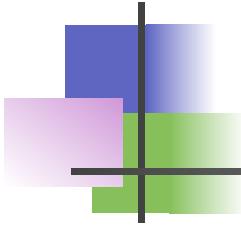
■ Propiedad de Transacciones:

- Desde el punto de vista de un SGBD, una transacción es una serie de operaciones de escritura y lectura a la BD
- Un SGBD debe garantizar cuatro propiedades en una transacción (ACID):
 - Atomicidad
 - Consistencia
 - Aislamiento
 - Durabilidad



Administración de Transacciones

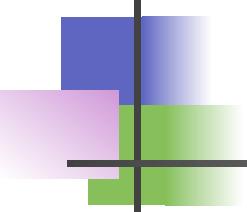
- Propiedad de Transacciones (ACID):
 - La ejecución de cada transacción es **atómica**: o se realizan todas las acciones o no se realiza ninguna
 - **Consistencia**: las transacciones deben preservar la consistencia de la BD
 - **Aislamiento**: las transacciones están aisladas, o protegidas, de los efectos de otras transacciones.
 - **Durabilidad**: si la transacción finaliza exitosamente, sus efectos deben persistir incluso si se produce una caída del sistema



Administración de Transacciones

■ Acciones de Transacciones:

- Las acciones que puede ejecutar una transacción son:
lectura y **escritura** de objetos de la BD
 - $R_t(O)$ denota lectura del objeto O por la transacción t
 - $W_t(O)$ denota escritura de O por t
 - Cada transacción debe indicar como última acción:
 - **commit** (comprometer): indica que la transacción termina satisfactoriamente, o
 - **abort** (abortar): indica que la transacción termina pero se deshacen los cambios realizados a la BD

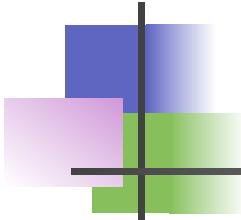


Administración de Transacciones

- Ejemplo de Transacciones:
 - Consideremos las transacciones T_1 y T_2 :
 - T_1 transfiere 100 desde la cuenta A a la B
 - T_2 incrementa A y B por el 10%

T_1 : **R(A);**
 $A = A - 100;$
W(A);
R(B);
 $B = B + 100;$
W(B);
commit;

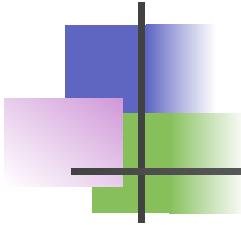
T_2 : **R(A);**
 $A = A * 1.1;$
W(A);
R(B);
 $B = B * 1.1;$
W(B);
commit;



Administración de Transacciones



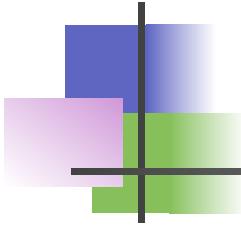
- Introducción
- Transacciones
- Control de Conurrencia
- Recuperación de BD



Administración de Transacciones

■ Conurrencia y Planificación:

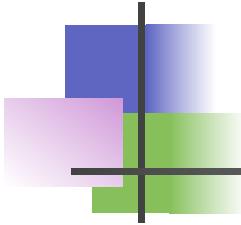
- Se denomina **planificación** a una secuencia de ejecución.
- Una **planificación** en serie es una planificación en donde las instrucciones pertenecientes a una transacción aparecen todas juntas.
- En **entornos multiprogramados**, es posible ejecutar varias transacciones de manera **concurrente**.
- Cuando se ejecutan varias transacciones de manera concurrente, se puede perder la consistencia de la base de datos aún cuando cada una de las transacciones individuales sea correcta



Administración de Transacciones

■ Conurrencia y Serialización

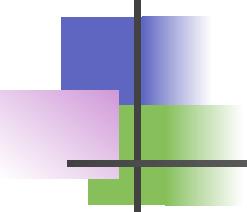
- Cuando varias transacciones se ejecutan concurrentemente, **no puede asegurarse** que la planificación sea en serie.
- Una **planificación es serializable** si el resultado de su ejecución equivale a alguna planificación en serie.
- Garantizar planificaciones serializables es el modo de garantizar la propiedad de **aislamiento** de las transacciones.



Administración de Transacciones

■ Planes:

- Un **plan de ejecución** (planificación) es una lista de acciones (lectura, escritura, comprometer o abortar) de un conjunto de transacciones.
- El orden en el cual dos acciones de una transacción T aparecen en un plan debe ser el mismo orden que el orden en el que aparecen en T.



Administración de Transacciones

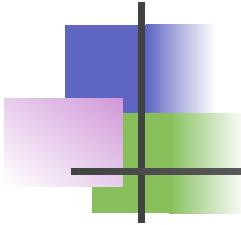
■ Ejemplo Plan:

- Consideremos los siguientes planes:

| Plan no serial | |
|------------------|------------------|
| T_1 | T_2 |
| $R(A)$ $W(A)$ | |
| | $R(B)$ $W(B)$ |

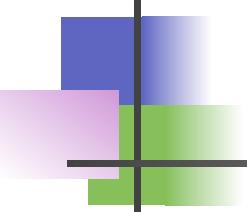
| Serial: $T_1 - T_2$ | |
|---------------------|------------------|
| T_1 | T_2 |
| $R(A)$ $W(A)$ | |
| $R(C)$ $W(C)$ | |
| $commit_{T_1}$ | |
| | $R(B)$ $W(B)$ |
| | $commit_{T_2}$ |

- Si las acciones de diferentes transacciones no son intercaladas, i.e. las transacciones son ejecutadas de inicio a fin, una por una, el plan se denomina plan serial
- Para el ejemplo existen dos posibles planes seriales: $T_1 - T_2$ y $T_2 - T_1$



Administración de Transacciones

- Ejecución concurrente de Transacciones:
 - Los SGBD entrelazan las acciones de distintas transacciones para mejorar el rendimiento
 - Si una transacción necesita leer desde disco, el SGBD podría empezar a procesar otra transacción
 - Surge entonces el concepto de **planes serializables**
 - Un **plan serializable** sobre un conjunto S de transacciones que comprometen los cambios, es un plan cuyo efecto sobre cualquier instancia consistente de la BD se garantiza idéntico al de alguna planificación secuencial sobre S



Administración de Transacciones

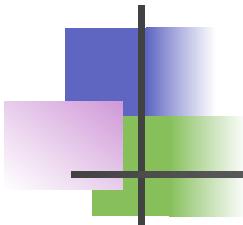
- Ejemplo Planes serializables:

- Los siguientes planes son serializables, el primero produce el mismo efecto que ejecutar T_1-T_2 , el segundo es equivalente a T_2-T_1

| T_1 | T_2 |
|------------------|------------------|
| $R(A)$ $W(A)$ | |
| | $R(A)$ $W(A)$ |
| $R(B)$ $W(B)$ | |
| | $R(B)$ $W(B)$ |
| | $commit_{T_2}$ |
| $commit_{T_1}$ | |

| T_1 | T_2 |
|----------------|----------------------------|
| | $R(A)$ $W(A)$ |
| | $R(B)$ $W(B)$ |
| | $W(A)$ $R(B)$ $W(B)$ |
| | $commit_{T_2}$ |
| $commit_{T_1}$ | |

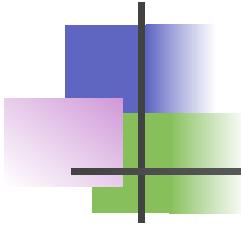
- Las ejecuciones secuenciales pueden producir distintos resultados, pero todos se suponen aceptables; el SGBD no garantiza cuál de ellos será el resultado de una ejecución entrelazada



Administración de Transacciones

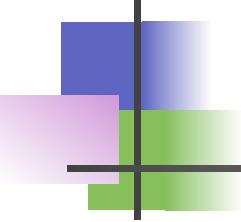
■ Problemas al intercalar Transacciones:

- Dos acciones sobre el mismo objeto de la BD presentan conflicto si por lo menos una de las acciones es una **escritura**
- Existen tres tipos de conflictos entre dos transacciones T_1 y T_2 :
 - Conflicto de **escritura-lectura** (WR) o lectura de datos no comprometidos
 - Conflicto de **lectura-escritura** (RW) o lecturas no repetidas
 - Conflicto de **escritura-escritura** (WW) o sobre-escritura de datos no comprometidos



Administración de Transacciones

- Conflicto de escritura-lectura (WR)
 - El principal problema de WR es la lectura de datos no comprometidos, i.e. T_2 lee un objeto que fue modificado por T_1 , pero que aún no ha sido comprometido
 - En tal caso, la lectura es conocida como **lectura sucia**



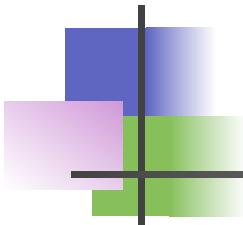
Administración de Transacciones

Ejemplo Lectura Sucia:

- Consideremos las transacciones T_1 y T_2 :
 - T_1 transfiere 100 desde la cuenta A a la B
 - T_2 incrementa A y B por el 10%
- Supongamos los valores iniciales $A = 1000$ y $B = 500$

| T_1 | T_2 |
|-------------------|------------------|
| $R(A)$, $A=1000$ | |
| $W(A)$, $A=900$ | |
| | $R(A)$, $A=900$ |
| | $W(A)$, $A=990$ |
| | $R(B)$, $B=500$ |
| | $W(B)$, $B=550$ |
| | $commit_{T_2}$ |
| $R(B)=550$ | |
| $W(B)=650$ | |
| $commit_{T_1}$ | |

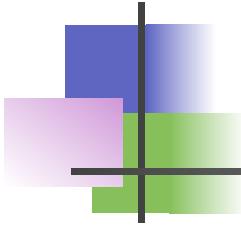
- Este plan produce los valores $A = 990$ y $B = 650$
 - **¿Qué valores produce $T_1 - T_2$?**
 - **¿Qué valores produce $T_2 - T_1$?**
- $T_1 - T_2 \rightarrow A = 990$ y $B = 660$
- $T_2 - T_1 \rightarrow A = 1000$ y $B = 650$
- El problema es que T_2 lee el valor de A modificado por T_1 pero que no ha sido comprometido



Administración de Transacciones

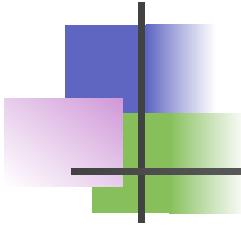
■ Conflicto de escritura-lectura (WR)

- Más aún, T_1 podría escribir algún valor para A que hace la BD inconsistente
- Es importante que T_1 vuelva a escribir un valor correcto para A antes que T_2 lea A
- Ningún daño se produce si las transacciones son ejecutadas de manera serial, ya que T_2 no verá tal inconsistencia
- Una transacción puede dejar la base de datos inconsistente por un período de tiempo, pero una vez comprometida la transacción, la BD debe quedar en un estado consistente



Administración de Transacciones

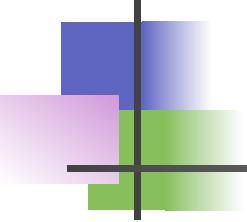
- Conflicto de lectura-escritura (RW)
 - El principal problema de lectura-escritura es la lectura no repetida de datos, i.e. T_2 modifica el valor de un objeto A que es leído por una transacción T_1 , mientras T_1 está aún en progreso
 - Si T_1 vuelve a leer A, el valor será diferente
 - Ejemplo: Consideremos las transacciones T_1 y T_2 , y sea A el número de copias disponibles de un libro
 - T_1 lee A = 1
 - T_2 lee A = 1, resta 1 y compromete (A=0)
 - T_1 trata de restar 1 a A y recibe un error, A ya no tiene el valor 1!



Administración de Transacciones

■ Conflicto de escritura-escritura(WW)

- T_2 sobre-escribe el valor de un objeto A que ya ha sido modificado por una transacción T_1 , mientras T_1 todavía está en progreso
- Este problema se conoce como sobre-escritura de datos no comprometidos
- Supongamos que los salarios de Pedro y Juan deben ser iguales
 - T_1 asigna salarios iguales a 2000
 - T_2 asigna salarios iguales a 1000
- El plan $T_1 - T_2$ asigna salarios iguales a 1000, $T_2 - T_1$ asigna salarios iguales a 2000



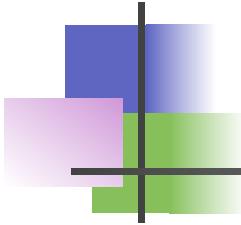
Administración de Transacciones

Conflicto de escritura-escritura(WW)

- Supongamos la siguiente ejecución intercalada:

| T_1 | T_2 |
|---|--|
| $W(Juan)$, Salario=2000 | $W(Pedro)$, Salario=1000 |
| $W(Pedro)$, Salario=2000 $commit_{T_1}$ | $W(Juan)$, Salario=1000 $commit_{T_2}$ |

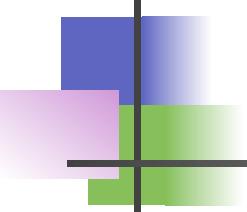
- El plan no produce salarios iguales para Juan y Pedro
- Por lo tanto este plan no es serializable



Administración de Transacciones

■ Planes con Abort

- Cuando una transacción aborta, todas las acciones son deshechas y la BD vuelve al estado inicial
- Un plan serializable sobre un conjunto de transacciones S es un plan cuyo efecto sobre cualquier BD consistente es idéntico a alguna ejecución serial sobre el conjunto de transacciones comprometidas en S
- Esta definición de plan serializable asume que las transacciones que abortan deben ser completamente deshechas, lo cual puede ser imposible en algunas situaciones



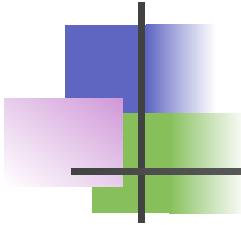
Administración de Transacciones

Ejemplo Planes con Abort:

- Consideremos las transacciones T_1 y T_2 :
 - T_1 transfiere 100 desde la cuenta A a la B
 - T_2 incrementa A y B por el 10%
- Supongamos los valores iniciales
 $A = 1000$ y $B = 500$

| T_1 | T_2 |
|--------------------|---------------------|
| $R(A)$, A=1000 | |
| $W(A)$, A=900 | |
| | $R(A)$, A=900 |
| | $W(A)$, A=990 |
| | $R(B)$, B=500 |
| | $W(B)$, B=550 |
| | <i>commit</i> T_2 |
| <i>abort</i> T_1 | |

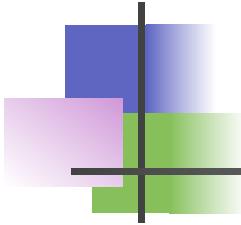
- T_2 ha leído un valor de A que no debería haber estado nunca ahí
- Si T_2 no comprometiera antes que T_1 aborta, podríamos tratar la situación propagando el aborto de T_1 en cascada y abortar T_2
- Pero T_2 ya comprometió y sus acciones no pueden ser deshechas! Este plan **no es recuperable**



Administración de Transacciones

■ Planes Recuperables

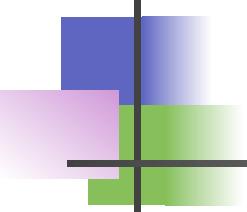
- En un plan recuperable las transacciones se comprometen solo después de que se comprometen todas las transacciones de las cuales se han leído cambios
- Si las transacciones leen solamente los cambios de transacciones comprometidas, el plan no solo es recuperable, sino que puede abortar una transacción sin abortar otras transacciones en cascada (planes que evitan abortos en cascada)



Administración de Transacciones

Planes conflicto equivalentes:

- Dos planes son conflicto equivalentes si involucran el mismo conjunto de acciones de las mismas transacciones y ordenan cada par de acciones conflictivas de dos transacciones que comprometen de la misma manera
- Dos acciones están en conflicto si operan sobre el mismo objeto y al menos una de ellas es escritura
- Podemos cambiar el orden de acciones que no son conflictivas sin alterar el efecto del plan sobre la BD
- Si dos planes son conflicto equivalentes, tienen el mismo efecto sobre la BD
- Un plan es conflicto serializable si es conflicto equivalente con algún plan serial
- Todo plan conflicto equivalente es serializable



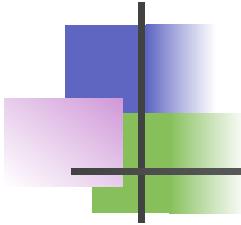
Administración de Transacciones

Ejemplo Planes Conflicto Equivalentes:

- El siguiente plan produce el mismo efecto que ejecutar $T_1 - T_2 - T_3$ pero no es conflicto equivalente a $T_1 - T_2 - T_3$

| T_1 | T_2 | T_3 |
|--|--|--|
| $R(A)$ | | |
| $W(A)$ <i>commit_{T_1}</i> | $W(A)$ <i>commit_{T_2}</i> | $W(A)$ <i>commit_{T_3}</i> |

- Las operaciones de escritura de T_1 y T_2 aparecen en diferente orden y no pueden ser reordenadas



Administración de Transacciones

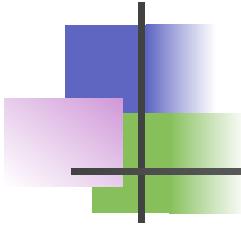
Ejemplo Planes Conflicto Equivalentes:

- Consideremos el siguiente plan:

R₂(y);R₁(x);R₃(y);R₂(x);W₂(y);W₁(x);R₃(x)

- Podemos reordenar las operaciones de la siguiente manera:

- La lectura de T1 puede ser localizada junto con la escritura de T₁
 $R_2(y);R_3(y);R_2(x);W_2(y);R_1(x);W_1(x);R_3(x)$
- Luego, las operaciones de T2 se reordenan de la siguiente forma:
 $R_3(y);R_2(x);R_2(y);W_2(y);R_1(x);W_1(x);R_3(x)$
- Sin embargo, no podemos mover R₃(y) ni R₃(x), por lo tanto este plan no es conflicto equivalente a ninguna ejecución serial de las transacciones



Administración de Transacciones

Ejemplo Planes Conflicto Equivalentes:

- Consideremos el siguiente plan:

R₁(x);R₂(y);W₃(x);R₂(x);R₁(y)

- Las operaciones pueden ser re-ordenadas de la siguiente manera:

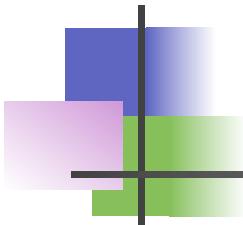
 - Las lecturas de T1 pueden ser ejecutadas en el siguiente orden:

R₁(x);R₁(y);R₂(y);W₃(x);R₂(x)

 - La lectura de T2 se puede reordenar de la siguiente forma:

R₁(x);R₁(y);W₃(x);R₂(y);R₂(x)

 - El plan es conflicto equivalente a la ejecución serial T₁-T₃-T₂, y como consecuencia el plan es serializable



Administración de Transacciones

Planes Conflicto Equivalentes: Ejercicio

- Consideremos el siguiente plan (R=Lectura, W=Escritura, C=Commit, A=Abort)

$$S = R_3(c), W_2(a), W_2(b), R_1(a), R_3(a), R_2(c), R_3(b), C_3, W_1(a), C_2, C_1$$

- ¿El plan es conflicto equivalente?
- ¿A qué plan es equivalente?

| T1 | T2 | T3 |
|------|--------|--------|
| | | R(c) |
| | W(a) | |
| | W(b) | |
| R(a) | | |
| | | R(a) |
| | R(c) | |
| | | R(b) |
| | | commit |
| W(a) | | |
| | commit | |
| | commit | |