Introducción a los Conceptos del Procesamiento de las Transacciones

22/Septiembre/2017

Transacción

Definición

Transacción

Definición

Una **transacción** es un conjunto de instrucciones que se ejecutan formando una unidad lógica de procesamiento. Una transacción puede incluir uno o más accesos a la BD a través del uso de diversas operaciones (inserción, eliminación, modificación, etc.).

Transacción

Definición

Una **transacción** es un conjunto de instrucciones que se ejecutan formando una unidad lógica de procesamiento. Una transacción puede incluir uno o más accesos a la BD a través del uso de diversas operaciones (inserción, eliminación, modificación, etc.).

Ejemplos de sistemas que utilizan transacciones

Transacción

Definición

Una **transacción** es un conjunto de instrucciones que se ejecutan formando una unidad lógica de procesamiento. Una transacción puede incluir uno o más accesos a la BD a través del uso de diversas operaciones (inserción, eliminación, modificación, etc.).

Ejemplos de sistemas que utilizan transacciones

- Reserva de vuelos
- Tranferencias bancarias
- Procesamiento de tarjetas de crédito
- Manejo de stock

Transacción

Definición

Una **transacción** es un conjunto de instrucciones que se ejecutan formando una unidad lógica de procesamiento. Una transacción puede incluir uno o más accesos a la BD a través del uso de diversas operaciones (inserción, eliminación, modificación, etc.).

Ejemplos de sistemas que utilizan transacciones

- Reserva de vuelos
- Tranferencias bancarias
- Procesamiento de tarjetas de crédito
- Manejo de stock

Son sistemas que requieren respuesta rápida y alta disponibilidad para muchos usuarios que acceden de manera concurrente.

Monousuarios vs. Multiusuarios

Clasificación DBMS según cantidad de usuarios concurrentes

- **Monousuario:** Sólo puede utilizar el sistema un usuario a la vez.

Monousuarios vs. Multiusuarios

Clasificación DBMS según cantidad de usuarios concurrentes

- **Monousuario:** Sólo puede utilizar el sistema un usuario a la vez.
- **Multiusuarios:** Varios usuarios pueden utilizar el sistema concurrentemente.

Monousuarios vs. Multiusuarios

Clasificación DBMS según cantidad de usuarios concurrentes

- **Monousuario:** Sólo puede utilizar el sistema un usuario a la vez.
- **Multiusuarios:** Varios usuarios pueden utilizar el sistema concurrentemente.

Ejemplos

Monousuarios vs. Multiusuarios

Clasificación DBMS según cantidad de usuarios concurrentes

- **Monousuario:** Sólo puede utilizar el sistema un usuario a la vez.
- **Multiusuarios:** Varios usuarios pueden utilizar el sistema concurrentemente.

Ejemplos

- Sistema de reservas de aerolíneas
- Cajeros electrónicos
- Supermercados

Monousuarios vs. Multiusuarios

Clasificación DBMS según cantidad de usuarios concurrentes

- **Monousuario:** Sólo puede utilizar el sistema un usuario a la vez.
- **Multiusuarios:** Varios usuarios pueden utilizar el sistema concurrentemente.

Ejemplos

- Sistema de reservas de aerolíneas
- Cajeros electrónicos
- Supermercados

Multiprogramación

Técnica por la que dos o más procesos son ejecutados concurrentemente por una CPU. Se trata de un paralelismo simulado, dado que la CPU sólo puede correr un proceso a la vez (proceso activo). Por eso se menciona que se ejecutan concurrentemente y no simultáneamente.

Multiprogramación

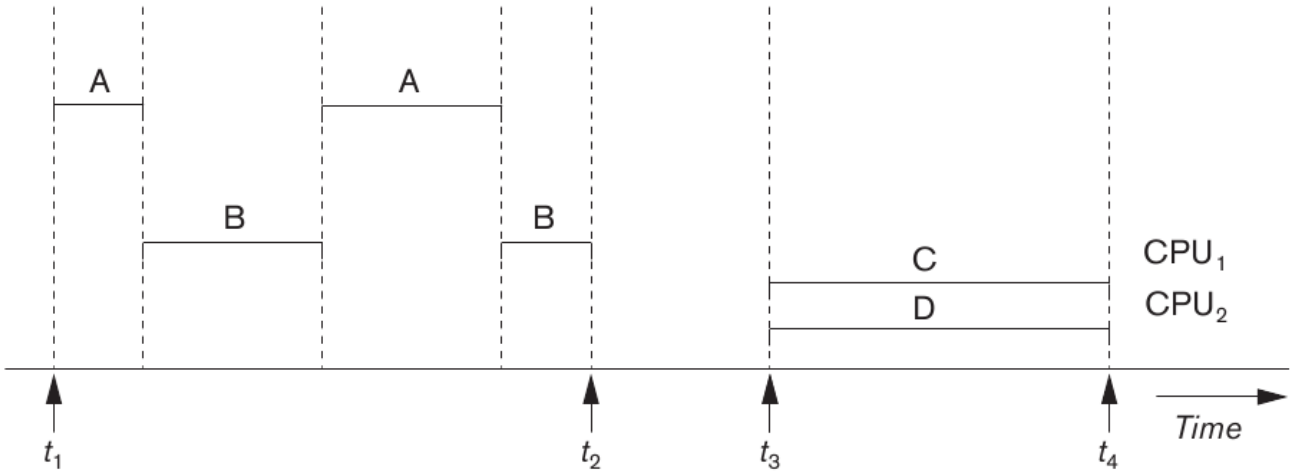

Figure 21.1
Interleaved processing versus parallel processing of concurrent transactions.

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Multiprogramación

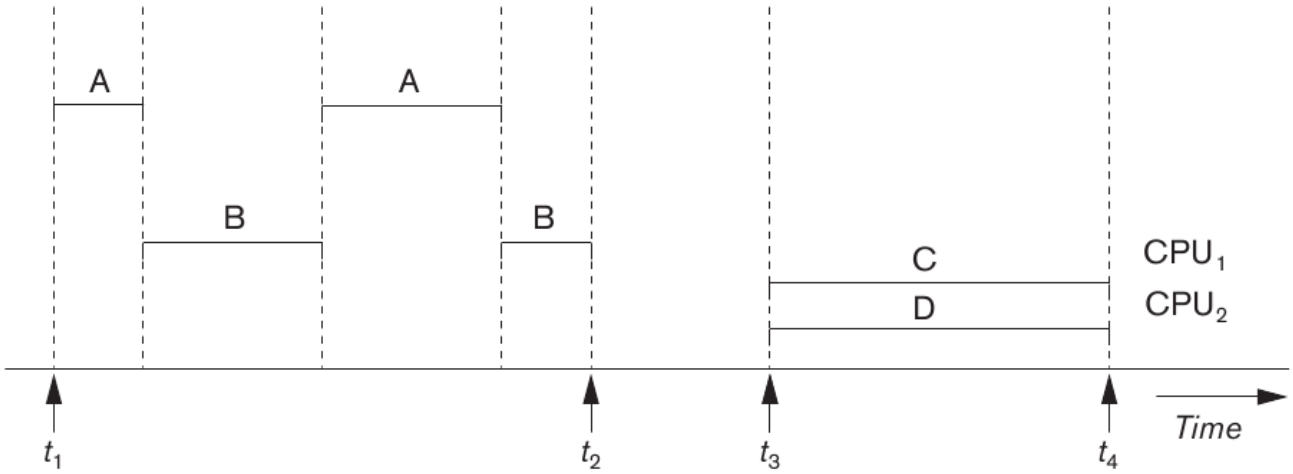

Figure 21.1
Interleaved processing versus parallel processing of concurrent transactions.

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Ejecución intercalada vs. Paralela

Asumimos SIEMPRE el modelo de ejecución intercalada (interleaved).

Transacciones

Una forma de especificar los límites, es hacerlo de forma explícita ...

```
begin transaction
      operación 1
      operación 2
      ...
      operación n
end transaction
```

Transacciones

Una forma de especificar los límites, es hacerlo de forma explícita ...

```
begin transaction
      operación 1
      operación 2
      ...
      operación n
end transaction
```

Tipo de transacciones

- **read-only transaction** (sólo lectura): Si ninguna de sus operaciones actualiza la Base de Datos
- **read-write transaction** (lectura/escritura): Si alguna de sus operaciones actualiza la Base de Datos

Modelo simplificado de Base de Datos

Base de datos

- BD es representada como una colección de **data items**.
- **granularidad**: Tamaño del **data item**
- Según la granularidad, un **data item** puede ser:
  - un registro de la base de datos
  - de mayor tamaño: un bloque de disco
  - de menor tamaño: el valor de un atributo
- **unique name**: Cada **data item** se identifica unívocamente a través de un identificador único (pocas veces utilizado por el programador). Ejemplo: Si la granularidad es el bloque de disco, entonces puede ser la dirección física del bloque.

Modelo simplificado de Base de Datos

Operaciones de acceso a la base de datos

- **read_item(X)**. Lee de la BD el **item** con id X y lo guarda (por simplicidad de notación) en la variable de programa llamada también X.
- **write_item(X)**. Escribe el valor de la variable de programa X en el **item** con id X de la BD.

Modelo simplificado de Base de Datos

Funcionamiento de Operaciones de acceso a la BD

- **read_item(X)**
  1. Encontrar la dirección del bloque de disco que contiene al **item** X.
  2. Copiar el contenido del bloque (disco) al buffer (memoria principal), sólo si no estaba previamente.
  3. Copiar el **item** X del buffer (memoria) a la variable del programa X

Modelo simplificado de Base de Datos

Funcionamiento de Operaciones de acceso a la BD

- **write_item(X)**
  1. Encontrar la dirección del bloque de disco que contiene al **item** X.
  2. Copiar el contenido del bloque (disco) al buffer (memoria principal), sólo si no lo estaba previamente.
  3. Copiar el contenido de la variable de programa X al lugar correcto del buffer (memoria)
  4. Almacenar el bloque actualizado del buffer nuevamente en el disco (inmediatamente o posteriormente en algún momento)

Modelo simplificado de Base de Datos

Funcionamiento de Operaciones de acceso a la BD

- **write_item(X)**
  1. Encontrar la dirección del bloque de disco que contiene al **item** X.
  2. Copiar el contenido del bloque (disco) al buffer (memoria principal), sólo si no lo estaba previamente.
  3. Copiar el contenido de la variable de programa X al lugar correcto del buffer (memoria)
  4. Almacenar el bloque actualizado del buffer nuevamente en el disco (inmediatamente o posteriormente en algún momento)

Copia de datos a disco

En el punto (4), la decisión de cuándo almacenar en disco es llevada a cabo por el **recovery manager** del DBMS en cooperación con el SO. Incluye políticas de caché vistas en SO.

Modelo simplificado de Base de Datos

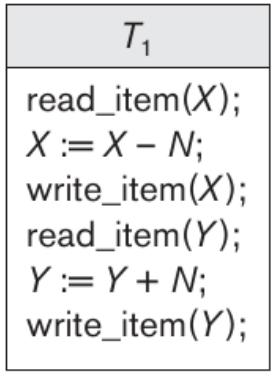

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Modelo simplificado de Base de Datos

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿T_1 es una read-only transaction o read-write transaction?

Modelo simplificado de Base de Datos

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿T_1 es una read-only transaction o read-write transaction? read-write transaction

Modelo simplificado de Base de Datos

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿T_1 es una read-only transaction o read-write transaction? read-write transaction

Conjunto de data items afectados por la transacción

- **read-set**: Conjunto de **items** leídos por la transacción.

Modelo simplificado de Base de Datos

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿T_1 es una read-only transaction o read-write transaction? read-write transaction

Conjunto de data items afectados por la transacción

- **read-set**: Conjunto de **items** leídos por la transacción.
  Ejemplo read-set(T_1) = $\{X, Y\}$

Modelo simplificado de Base de Datos

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿T_1 es una read-only transaction o read-write transaction? read-write transaction

Conjunto de data items afectados por la transacción

- **read-set**: Conjunto de **items** leídos por la transacción.
  Ejemplo read-set(T_1) = $\{X, Y\}$
- **write-set**: Conjunto de **items** escritos por la transacción.

Modelo simplificado de Base de Datos

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿T_1 es una read-only transaction o read-write transaction? read-write transaction

Conjunto de data items afectados por la transacción

- **read-set**: Conjunto de **items** leídos por la transacción.
  Ejemplo read-set(T_1) = $\{X, Y\}$
- **write-set**: Conjunto de **items** escritos por la transacción.
  Ejemplo write-set(T_1) = $\{X, Y\}$

¿Por qué es necesario el Control de Concurrencia?

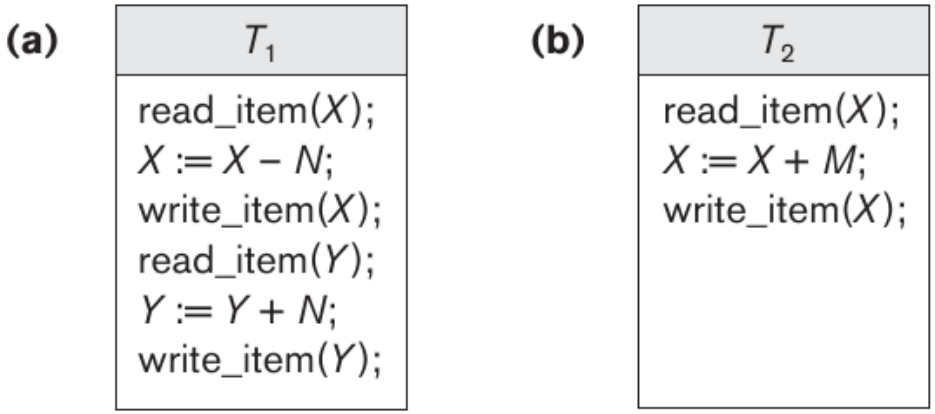

Figure 21.2
Two sample transactions. (a) Transaction T_1. (b) Transaction T_2.

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿Por qué es necesario el Control de Concurrencia?

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

Figure 21.2
Two sample transactions. (a) Transaction T_1. (b) Transaction T_2.

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Ejemplo: BD para reservas de una aerolínea

El **item** es identificado por el "id de vuelo" y contiene la información correspondiente a la "cantidad de asientos reservados"

¿Por qué es necesario el Control de Concurrencia?

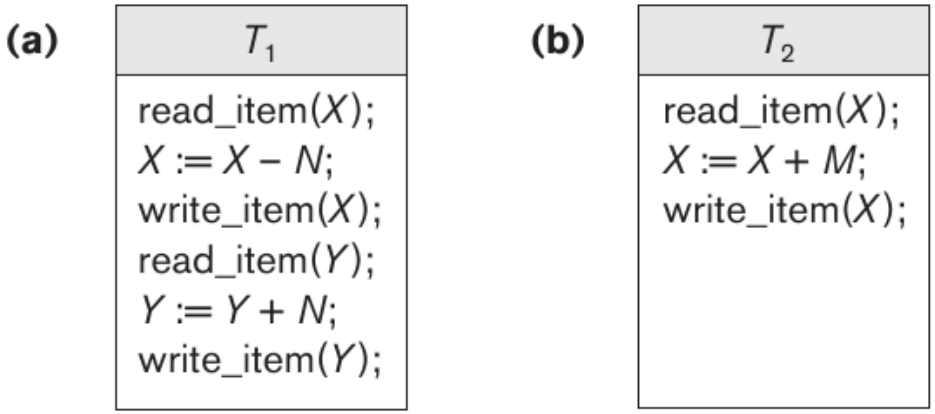

Figure 21.2
Two sample transactions. (a) Transaction T_1. (b) Transaction T_2.

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Ejemplo: BD para reservas de una aerolínea

El **item** es identificado por el "id de vuelo" y contiene la información correspondiente a la "cantidad de asientos reservados"

- T_1:

¿Por qué es necesario el Control de Concurrencia?

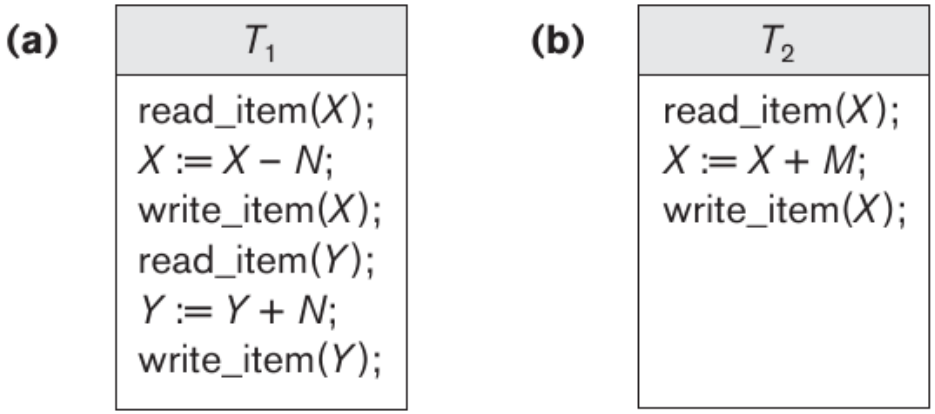

Figure 21.2
Two sample transactions. (a) Transaction T_1. (b) Transaction T_2.

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Ejemplo: BD para reservas de una aerolínea

El **item** es identificado por el "id de vuelo" y contiene la información correspondiente a la "cantidad de asientos reservados"

- T_1: Transfiere las reservas del vuelo X al Y

¿Por qué es necesario el Control de Concurrencia?

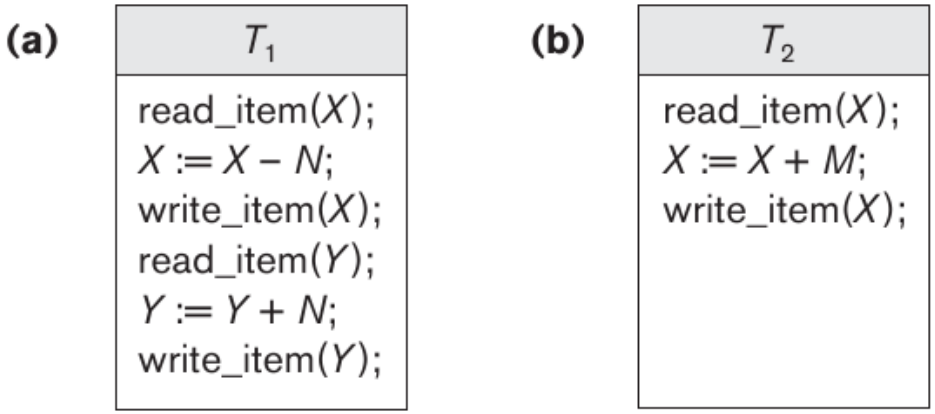

Figure 21.2
Two sample transactions. (a) Transaction T_1. (b) Transaction T_2.

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Ejemplo: BD para reservas de una aerolínea

El **item** es identificado por el "id de vuelo" y contiene la información correspondiente a la "cantidad de asientos reservados"

- T_1: Transfiere las reservas del vuelo X al Y
- T_2:

¿Por qué es necesario el Control de Concurrencia?

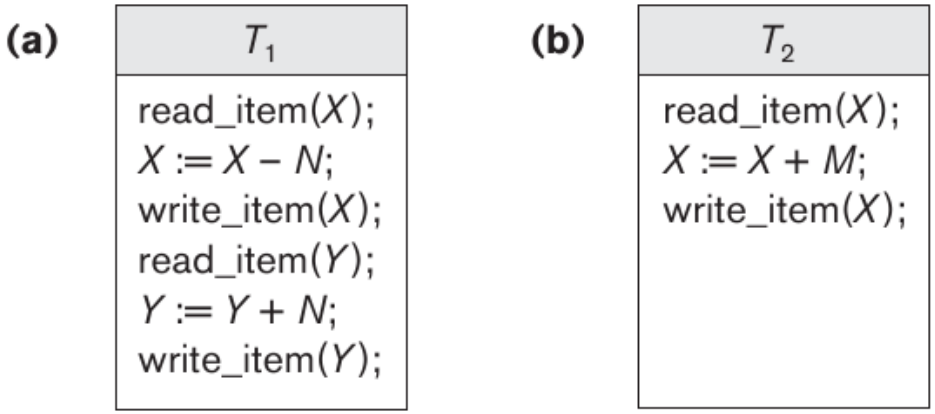

Figure 21.2
Two sample transactions. (a) Transaction T_1. (b) Transaction T_2.

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Ejemplo: BD para reservas de una aerolínea

El **item** es identificado por el "id de vuelo" y contiene la información correspondiente a la "cantidad de asientos reservados"

- T_1: Transfiere las reservas del vuelo X al Y
- T_2: Agrega reservas al vuelo X

¿Por qué es necesario el Control de Concurrencia?

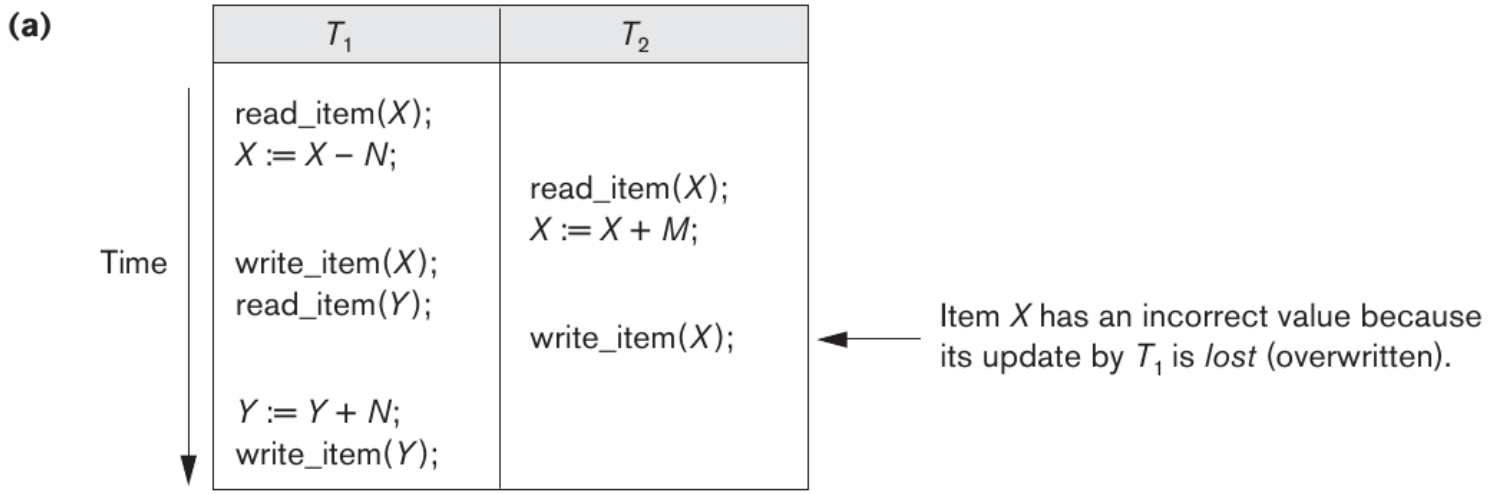

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿Por qué es necesario el Control de Concurrencia?

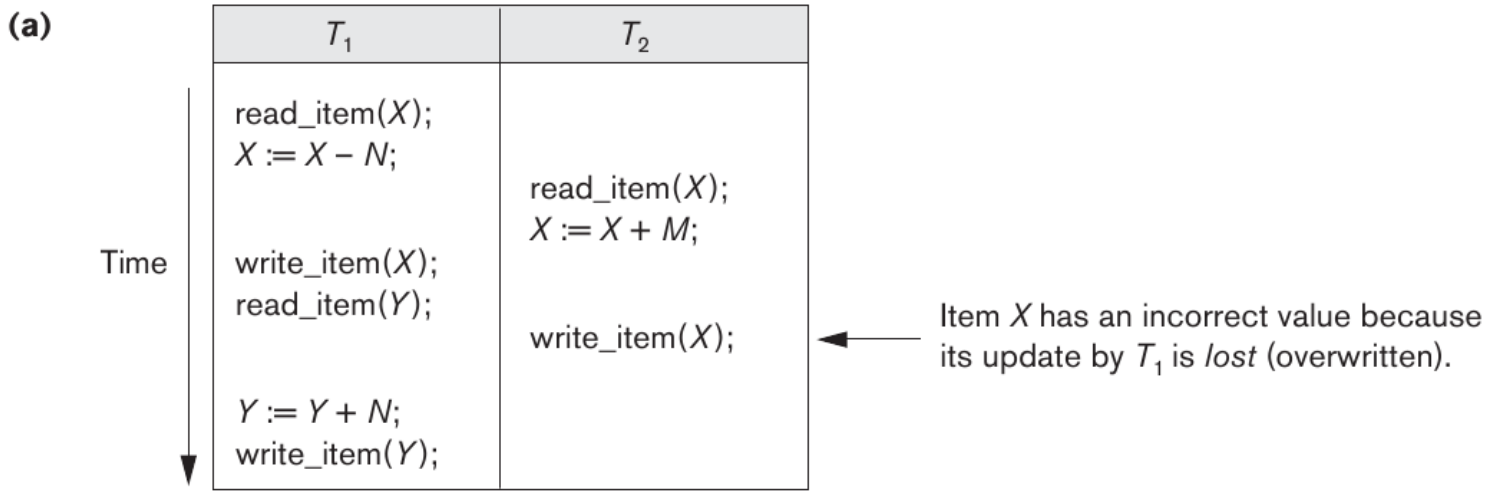

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Lost Update Problem: Problema por pérdida de actualización

T_2 lee el valor de X previo a que T_1 lo modifique en la BD.

¿Por qué es necesario el Control de Concurrencia?

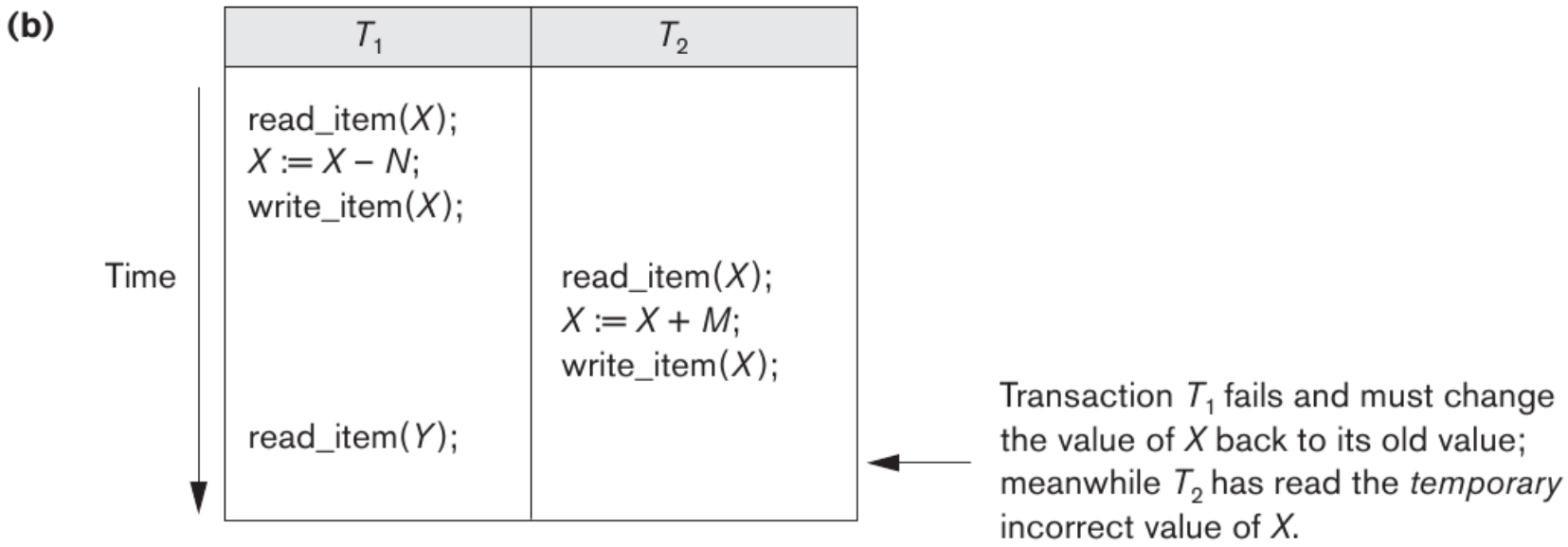

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿Por qué es necesario el Control de Concurrencia?

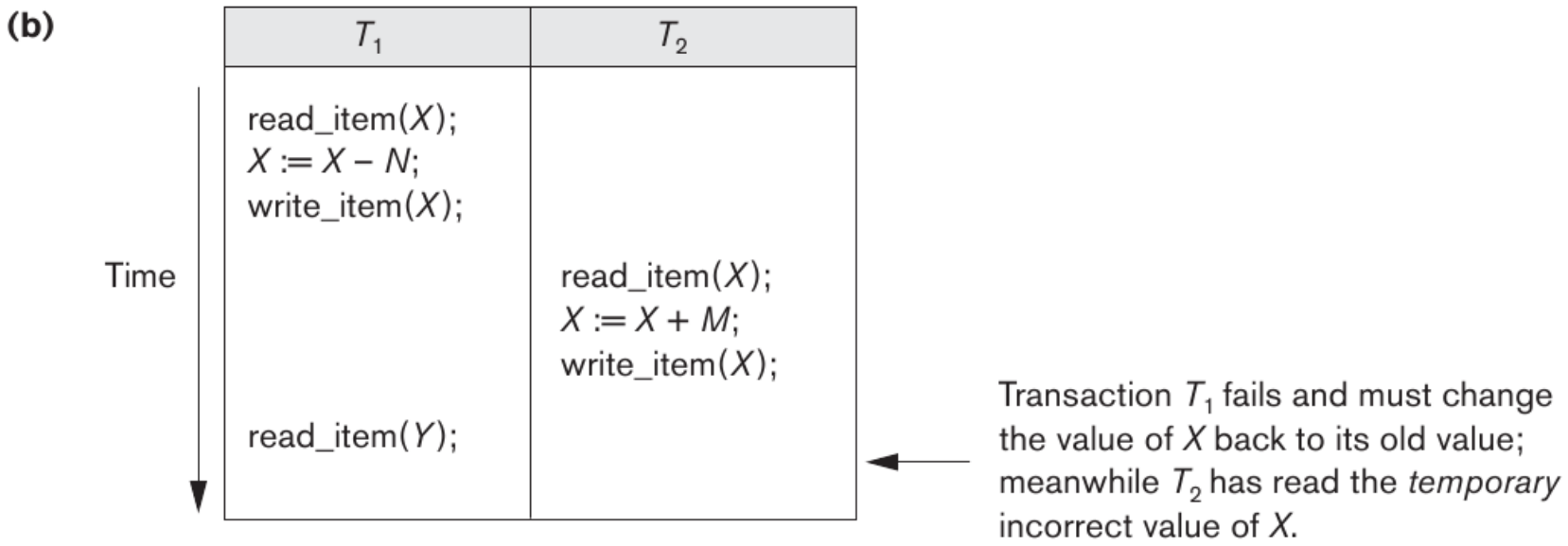

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Temporary Update (Dirty Read) Problem: Problema por actualización provisoria

T_2 lee el valor temporal de X, el cual no va a ser grabado en la BD debido a un fallo en medio de la transacción T_1.

¿Por qué es necesario el Control de Concurrencia?

(c)

T_1	T_3

	sum := 0;
	read_item(*A*);
	sum := *sum* + *A*;
	. . .

read_item(*X*);
X := *X* − *N*;
write_item(*X*);	

	read_item(*X*);
	sum := *sum* + *X*;
	read_item(*Y*);
	sum := *sum* + *Y*;

read_item(*Y*);
Y := *Y* + *N*;
write_item(*Y*);	

← T_3 reads *X* after *N* is subtracted and reads *Y* before *N* is added; a wrong summary is the result (off by *N*).

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

¿Por qué es necesario el Control de Concurrencia?

(c)

T_1	T_3

	sum := 0;
	read_item(*A*);
	sum := *sum* + *A*;
	⋮

read_item(*X*);
X := *X* − *N*;
write_item(*X*);	

	read_item(*X*);
	sum := *sum* + *X*;
	read_item(*Y*);
	sum := *sum* + *Y*;

read_item(*Y*);
Y := *Y* + *N*;
write_item(*Y*);	

← T_3 reads X after N is subtracted and reads Y before N is added; a wrong summary is the result (off by N).

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Incorrect Summary Problem: Problema de Suma Incorrecta

T_3 lee el valor de X después de que N asientos han sido sustraídos por T_1, pero lee el valor de Y antes de que esos N asientos hayan sido adicionados por T_1.

¿Por qué es necesario el Control de Concurrencia?

Unrepeatable Read Problem: Problema de Lecturas irrepetibles

T lee el mismo **item** dos veces, pero el **item** es modificado por otra transacción T' entre ambas lecturas. Así, T recibe diferentes valores para estas dos lecturas del mismo **item**.

¿Por qué el Recovery es necesario?

Siempre que una transacción es recibida por el DBMS para su ejecución el sistema es responsable de:

- que todas las operaciones de la transacción sean ejecutadas exitosamente y sus resultados sean almacenados en la BD (**committed**)
- ó, en caso contrario, que ninguna operación tenga efecto sobre la BD (**aborted**).

Transacciones que fallan

Ante un fallo en la transacción T previo a que se transforme en **committed**, el DBMS deberá deshacer los cambios realizados por las operaciones ejecutadas hasta el momento de T, dejando sin efecto lo hecho hasta ese momento.

Tipos de fallo

Tipos de fallo

¿Qué fallas pueden afectar la ejecución de una transacción?

Tipos de fallo

¿Qué fallas pueden afectar la ejecución de una transacción?

1. **Fallo de la Computadora (system crash):** Un error en el hardware, software o red ocurre mientras se ejecuta la transacción.
2. **Falla en la transacción o el sistema:** Una división por cero o un error en la lógica de la programación. Adicionalmente, el usuario puede interrumpir la transacción durante su ejecución.
3. **Errores locales o condiciones de excepción detectadas por la transacción:** Los datos necesarios para llevar a cabo la transacción no son encontrados. Otro ejemplo: una excepción por falta de fondos en la cuenta corriente.
4. **Ejecución del Control de Concurrencia:** El método de Control de Concurrencia puede decidir abortar una transacción por violar la serialización o por una situación de deadlock. Típicamente estas transacciones son reiniciadas automáticamente más tarde.
5. **Falla de Disco:** Puede ocurrir durante una lectura o escritura.
6. **Problemas físicos o catástrofes:** Incluyen suministro eléctrico, aire acondicionado, fuego, robo, sabotaje, etc.

Tipos de fallo

¿Qué fallas pueden afectar la ejecución de una transacción?

1. **Fallo de la Computadora (system crash):** Un error en el hardware, software o red ocurre mientras se ejecuta la transacción.
2. **Falla en la transacción o el sistema:** Una división por cero o un error en la lógica de la programación. Adicionalmente, el usuario puede interrumpir la transacción durante su ejecución.
3. **Errores locales o condiciones de excepción detectadas por la transacción:** Los datos necesarios para llevar a cabo la transacción no son encontrados. Otro ejemplo: una excepción por falta de fondos en la cuenta corriente.
4. **Ejecución del Control de Concurrencia:** El método de Control de Concurrencia puede decidir abortar una transacción por violar la serialización o por una situación de deadlock. Típicamente estas transacciones son reiniciadas automáticamente más tarde.
5. **Falla de Disco:** Puede ocurrir durante una lectura o escritura.
6. **Problemas físicos o catástrofes:** Incluyen suministro eléctrico, aire acondicionado, fuego, robo, sabotaje, etc.

Magnitud del esfuerzo

1-4 el sistema debería mantener suficiente información para restablecerse rápidamente.
5-6 el restablecimiento suele requerir de un esfuerzo importante.

Operaciones adicionales: **recovery manager**

El **recovery manager** del DBMS necesita llevar registro de las siguientes operaciones:

- **begin_transaction:** Marca el inicio de la ejecución de la transacción.
- **read/write:** Especifica las operaciones de lectura/escritura sobre los **ítems** de la BD que son ejecutados como parte de la transacción.
- **end_transaction:** Marca el fin de las lecturas y escrituras, y marca el final de la ejecución de la transacción. Sin embargo, en este punto es necesario chequear si los cambios introducidos por la transacción pueden ser aplicados permanentemente a la BD (**commited**) o si deben ser descartados porque violan la serialización o alguna otra razón (**abort**).
- **commit_transaction:** Esto marca un final exitoso de la transacción y la misma puede ser commiteada a la BD. A partir de acá dichos cambios no serán deshechos.
- **rollback (o abort):** Esto marca que la transacción no terminó de manera exitosa. Los cambios realizados en la BD por la transacción deben ser deshechos.

Operaciones adicionales: **recovery manager**

El **recovery manager** del DBMS necesita llevar registro de las siguientes operaciones:

- **begin_transaction:** Marca el inicio de la ejecución de la transacción.
- **read/write:** Especifica las operaciones de lectura/escritura sobre los **ítems** de la BD que son ejecutados como parte de la transacción.
- **end_transaction:** Marca el fin de las lecturas y escrituras, y marca el final de la ejecución de la transacción. Sin embargo, en este punto es necesario chequear si los cambios introducidos por la transacción pueden ser aplicados permanentemente a la BD (**commited**) o si deben ser descartados porque violan la serialización o alguna otra razón (**abort**).
- **commit_transaction:** Esto marca un final exitoso de la transacción y la misma puede ser commiteada a la BD. A partir de acá dichos cambios no serán deshechos.
- **rollback (o abort):** Esto marca que la transacción no terminó de manera exitosa. Los cambios realizados en la BD por la transacción deben ser deshechos.

Importante

Las transacciones que fallaron o han sido abortadas pueden ser reiniciadas (manual o automáticamente) posteriormente como una nueva transacción.

Diagrama de Estados de la Ejecución de una Transacción

Figure 21.4
State transition diagram illustrating the states for transaction execution.

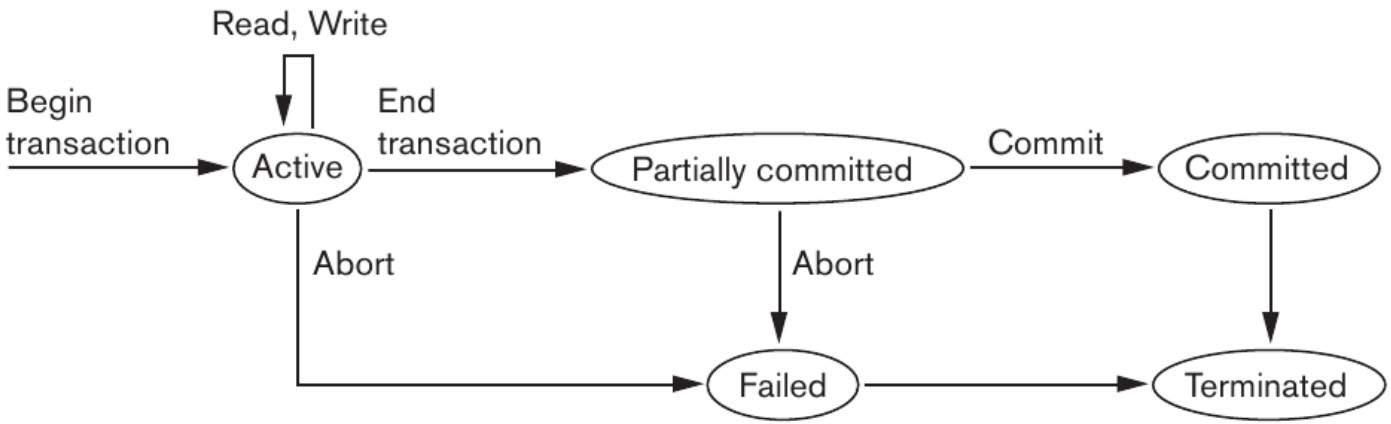

Figura de Elmasri/Navathe-*Fundamentals of DB Systems*, 6th Ed., Pearson, 2011

Log del sistema

Función del Log

Log (o DBMS journal) mantiene registro de todas las operaciones de las transacciones que afectan los valores de la BD. Permite restablecer el sistema ante fallos que afectan a las transacciones.

Log del sistema

Función del Log

Log (o DBMS journal) mantiene registro de todas las operaciones de las transacciones que afectan los valores de la BD. Permite restablecer el sistema ante fallos que afectan a las transacciones.

Características del Log

- Archivo incremental y secuencial
- Se almacena en disco
- Fallos que lo pueden afectar: Falla de Disco y Problemas físicos o catástrofes.

Log del sistema

Función del Log

Log (o DBMS journal) mantiene registro de todas las operaciones de las transacciones que afectan los valores de la BD. Permite restablecer el sistema ante fallos que afectan a las transacciones.

Características del Log

- Archivo incremental y secuencial
- Se almacena en disco
- Fallos que lo pueden afectar: Falla de Disco y Problemas físicos o catástrofes.

Esquema típico de almacenamiento del Log

- Última parte del log es mantenido en 1 o + buffers de memoria (por lo tanto, las entradas en el log se insertan primero en el buffer de la memoria principal)
- Cuando se llena el buffer que mantiene el log (o bajo alguna otra condición), el contenido se agrega al final del log en disco.
- El log en disco es periódicamente "backupeado" en otro dispositivo de almancenamiento para evitar fallos de tipo catástrofes

Punto de Commit

Condición de Commit

Una Transacción T alcanza su punto de commit cuando:

- Todas sus operaciones que acceden a la BD han sido ejecutadas exitosamente

Y

- Los efectos de TODAS sus operaciones sobre la BD han sido grabadas en el Log

Pasado este punto T se dice **commiteada**. La transacción escribe el registro de commit en el log.

Control de Concurrencia y Recovery

Anuncio

Control de Concurrencia y Recovery se verá más adelante en la materia, con mayor profundidad.

Propiedades deseables en las transacciones

Propiedades ACID para ejecutar métodos de control de concurrencia y recovery.

Propiedades

- **Atomicity (Atomicidad):** Transacción como unidad atómica. Las operaciones de una transacción se ejecutan en su totalidad o no se ejecuta ninguna.
- **Consistency preservation (Preservación de Consistencia):** Si la transacción se ejecuta completamente sin interferencia de otra transacción, entonces mueve a la BD de un estado consistente a otro también consistente.
- **Isolation (Aislamiento):** La ejecución de una transacción no debe interferir con la de otra transacción que se ejecute de manera concurrente. Una transacción debe aparentar ser ejecutada como si lo hiciera de forma aislada a las otras. Incluso si varias de ellas son ejecutadas a la vez.
- **Durability (Durabilidad):** Los cambios aplicados a la BD por una transacción commiteada debe persistir en la BD. Estos cambios no se pueden perder a causa de ningún fallo.

Propiedades deseables en las transacciones

Propiedades y subsistemas

- **Atomicidad:** Es responsabilidad del **subsistema de recovery** del DBMS asegurar esta propiedad. Las técnicas de restablecimiento deben poder deshacer cualquier efecto de la transacción en la BD. Las escrituras de una transacción commiteada deben ser escritas eventualmente a disco.
- **Consistencia:** Es responsabilidad de los **programadores** y del módulo que ejecuta las **restricciones de integridad** del DBMS . El **database state** es una colección de todos los **items** almacenados (valores) en la BD en un determinado momento. Un estado consistente satisface tanto las restricciones especificadas en el esquema como otras restricciones de la BD. El estado de la BD será consistente luego de la ejecución de una transacción asumiendo que no hubo interferencia con otra transacción.

Propiedades deseables en las transacciones

Propiedades y subsistemas

- **Atomicidad:** Es responsabilidad del **subsistema de recovery** del DBMS asegurar esta propiedad. Las técnicas de restablecimiento deben poder deshacer cualquier efecto de la transacción en la BD. Las escrituras de una transacción commiteada deben ser escritas eventualmente a disco.
- **Consistencia:** Es responsabilidad de los **programadores** y del módulo que ejecuta las **restricciones de integridad** del DBMS . El **database state** es una colección de todos los **items** almacenados (valores) en la BD en un determinado momento. Un estado consistente satisface tanto las restricciones especificadas en el esquema como otras restricciones de la BD. El estado de la BD será consistente luego de la ejecución de una transacción asumiendo que no hubo interferencia con otra transacción.

Consistencia

Si un programador programa por error que para realizar una transferencia de $100 desde una cuenta A a una B, se debe erróneamente restar $75 de A y sumar $125 a B, la BD será consistente con esa programación (y no con la intención que tenía el usuario). La BD no tiene forma de chequear este tipo de "inconsistencias", al menos que se lo especifique correctamente.

Propiedades deseables en las transacciones (Cont.)

Propiedades y subsistemas

- **Aislamiento:** Es impuesto por el **subsistema de control de concurrencia** del DBMS. Existen distintos niveles de aislamiento para una transacción T:
  - Nivel 0: T no sobrescribe los dirty reads de las transacciones de mayor nivel
  - Nivel 1: no hay lost updates
  - Nivel 2: no hay lost updates ni dirty reads
  - Nivel 3: También llamada aislamiento real, posee adicionalmente al Nivel 2, repeatable reads.

  Tanto en los lenguajes de programación como en SQL, existe una manera de establecer el nivel de aislamiento.
- **Durabilidad:** Es responsabilidad del **subsistema de recovery** del DBMS asegurar esta propiedad.

Normalización - Bibliografía

- Capítulo 21 (hasta 21.3 inclusive) Elmasri/Navathe - Fundamentals of Database Systems, 6th Ed., Pearson, 2011.