

Dpto. Ciencias e Ingeniería de la Computación
 Universidad Nacional del Sur

**ELEMENTOS DE BASES
 DE DATOS**

Segundo Cuatrimestre 2014

**Clase 18:
 Gestión de Transacciones –
 Sistema de Recuperación**



Mg. María Mercedes Vitturini
 [mvitturi@uns.edu.ar]

Sistema de Recuperación 

“ A **computer system**, like any other device, is subject to failure from a variety of causes: **disk crash, power outage, software error, a fire in the machine room, even sabotage**. In any failure, **information may be lost**. Therefore, the database system must take actions in advance to ensure that the **atomicity and durability** properties of transactions, introduced in Chapter 14, are preserved. An integral part of a database system is a **recovery scheme that can restore the database to the consistent state that existed before the failure**. The recovery scheme must also provide **high availability**; that is, it must minimize the time for which the database is not usable after a failure.”

DATABASE System Concepts – A. Silberschatz

EBD2014_18 - Mg. Mercedes Vitturini

DBMS y Fallos

Cualquier sistema está sujeto a fallos: un fallo eventualmente puede ocasionar pérdidas o inconsistencia de información.

“La ocurrencia de un fallo se puede presentar en cualquier etapa de la vida de las distintas transacciones que gestiona el SMDB.”

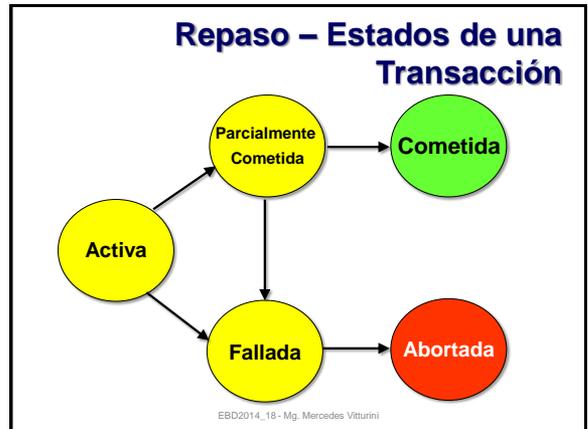
- Ante el fallo, el DBMS se debe **‘recuperar’** y dejar sus bases de datos en el estado consistente anterior al fallo.

Ejemplo:

Sea T una transacción que realiza transferencias de fondo de la cuenta A a la cuenta B. El fallo podría ocurrir después de descontar 50\$ de A. Los valores iniciales: A=1000, B=2000. El fallo ocurre después del débito, pero antes de acreditarlos en B. (A=950, B=2000)

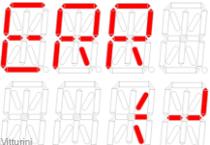
- El DBMS debe recuperarse al estado consistente anterior, antes de iniciar la transferencia (A=1000, B=2000).

EBD2014_18 - Mg. Mercedes Vitturini



Tipos de Fallos

- Para un **SMBD centralizado concurrente** se identifican los siguientes tipos de fallos:
 - Fallo de Transacción.
 - Caída de Sistema.
 - Fallo con pérdida de disco



EBD2014_18 - Mg. Mercedes Vitturini

Fallo de Transacción

FALLO DE TRANSACCIÓN – cualquier error que cause que una transacción deba retroceder:

- **Error lógico**: la transacción no puede continuar su ejecución a causa de alguna condición interna.
- **Error bajo el entorno del sistema**: el sistema debe terminar con la ejecución de una transacción activa debido a alguna condición de error (deadlock, problemas de estapillas, etc.)
 - La transacción vuelve a ejecutarse más tarde.

Alcance del Fallo de Transacción – la transacción activa que falla y **potencialmente otras transacciones activas dependientes por el efecto ‘fallo en cascada o cadena’**. Otras transacciones activas no afectadas prosiguen su ejecución normal.

EBD2014_18 - Mg. Mercedes Vitturini

Caída del Sistema

Caída del sistema (crash) – fallos de hardware o software que causan la falla general y caída del sistema.

- Características:
 - Causan **pérdida de información en memoria volátil** (memoria principal y cache).
 - El contenido de la información en la memoria no volátil (disco) permanece intacto.
 - El DBMS posee numerosos chequeos de consistencia para prevenir la corrupción de los datos.

Alcance del fallo Caída de Sistema – afecta a *todas las transacciones activas* al momento del error y potencialmente *también* a transacciones recientemente cometidas.

EBD2014_18 - Mg. Mercedes Vitturini

Fallo de Disco

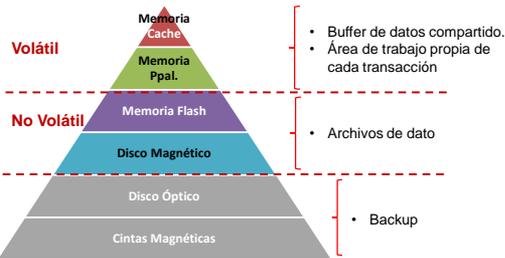
Fallo de disco – i.e. daño físico en el medio de almacenamiento masivo.

- Se asume que el error es detectable (checksum).
- Se **requiere de copias** de datos de backup en otros discos u otros medios de almacenamiento para recuperar el fallo.
- La recuperación combina los pasos de recuperar un backup con recuperación por fallo del tipo caída del sistema.

backup + recuperación por caída de sistema

EBD2014_18 - Mg. Mercedes Vitturini

Repaso Estructura de Almacenamiento



EBD2014_18 - Mg. Mercedes Vitturini

Estructura de Almacenamiento

Almacenamiento volátil

- Incluye memoria principal (MP) y memoria cache (MC).
- No **sobrevive a las caídas de sistemas**.

Almacenamiento no volátil

- Ejemplo: discos.
- No representa por sí solo almacenamiento estable.

Almacenamiento estable

- Organización del almacenamiento que asegura *'nunca'* perder información.
- Sobrevive a todos los fallos.
- Se basa en **replicas en distintos medios de almacenamiento con modos de fallo independiente**, geográficamente separados y actualización de la información controlada.

EBD2014_18 - Mg. Mercedes Vitturini

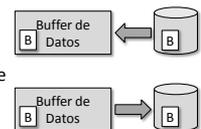
Memoria Estable

- La información que reside en memoria estable **nunca** se pierde.
- Mecanismo para alcanzarlo: **repetir la información (de manera controlada) en varios medios de almacenamiento**.
 - En el caso de **discos espejados**, ambos bloques están físicamente en la misma locación.
 - En el caso de **copias remotas**, uno de los bloques es local y el otro está en un sitio remoto.

EBD2014_18 - Mg. Mercedes Vitturini

Repaso - Acceso a los datos

- La **base de datos** reside en archivos en memoria estable y se organiza en unidades denominadas **bloques físicos**.
- Los datos se trasladan **desde/hacia** los **bloques físicos** a **bloques de registro intermedio o buffer de MP**.
- Las transferencias disco/MP se inician a través de las operaciones:
 - **INPUT(B)**: transfiere el bloque físico B a memoria principal.
 - **OUTPUT(B)**: transfiere el bloque de registro intermedio B (buffer) y sustituye el bloque físico apropiado.



EBD2014_18 - Mg. Mercedes Vitturini

Acceso a los datos

- Las operaciones de entrada y salida desde el disco a la MP se hacen en *unidad de bloque*.
- Cada transacción posee su *área de trabajo (work-area)* en MP donde mantiene las variables locales con los valores de los ítems de datos accedidos y actualizados por ella.
- Las **transacciones** transfieren ítems de datos entre el **buffer de datos compartido** y su propia **área de trabajo** usando las operaciones: **READ (X)** y **WRITE(X)**.

EBD2014_18 - Mg. Mercedes Vitturini

Transferencias de Datos

READ(X, x_i) asigna el valor del elemento de dato X a la variable local x_i.

Esta operación se ejecuta como sigue:

- Si el bloque B_x en el que reside X **no está en memoria principal**, entonces se ejecuta **INPUT(B_x)**.
- Se asigna a x_i el valor de X del bloque en el registro intermedio.

EBD2014_18 - Mg. Mercedes Vitturini

Transferencias de Datos

WRITE(X, x_i): asigna el valor de la variable local x_i al elemento de información X.

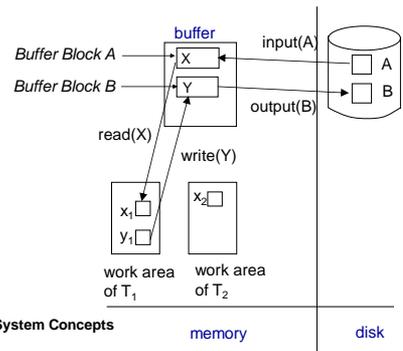
Esta operación se ejecuta como sigue:

- Si el bloque B_x en el que reside X **no está en memoria principal**, entonces se ejecuta **INPUT(B_x)**.
- Se asigna el valor de x_i a X en el bloque que contiene a X del **buffer compartido**.

- En algún momento se realiza el **OUTPUT(B_x)**.



EBD2014_18 - Mg. Mercedes Vitturini



Fuente: Database System Concepts
 A. Silberschatz

EBD2014_18 - Mg. Mercedes Vitturini

Write (X, x) – Output(X)

- El **Output(B)** de bloques modificados a almacenamiento estable tiene lugar en algún momento posterior actualizar el bloque en el buffer de datos.
- El orden en el que se realiza el output puede ser diferente del orden en el que se modificaron.
- La operación **Output(X)** **no se siempre ejecuta inmediatamente después de write (X, x_i)**.
 - Por ejemplo, el bloque en el que reside X contiene otros elementos de información a los que se accede todavía.
- Ante un **fallo de sistema después de ejecutar write(X, x_i)** y **antes de ejecutar Output(X)**, existe riesgo de perder el nuevo valor para X.

EBD2014_18 - Mg. Mercedes Vitturini

Sistema de Recuperación



Los sistemas están sujetos a fallos que pueden ocasionar estados inconsistentes

Algoritmo de Recuperación (AR)

Los **algoritmos de recuperación (AR)** son procesos usados para asegurar **atomicidad y durabilidad** de las transacciones **a pesar de los fallos**, así como la **consistencia** de la BD.

- En un AR se identifican:
 - Acciones preventivas** a tomar durante el procesamiento normal de las transacciones para asegurar que exista suficiente información para permitir la recuperación de fallos (**acciones preventivas**).
 - Acciones de recuperación** como consecuencia de un fallo para asegurar la consistencia, atomicidad y durabilidad de las transacciones (**acciones preventivas**).

EBD2014_18 - Mg. Mercedes Vitturini

Acciones Preventivas

- Modificar directamente la base de datos antes de saber si la transacción cometerá puede conducir a estados inconsistentes. **Se necesita guardar en memoria estable información sobre las modificaciones realizadas.**

La **bitácora** (o registros *log*) se mantiene en **memoria estable** y mantiene una secuencia de **registros con información sobre las actualizaciones realizadas por cada transacción.**



Framework: la bitácora y el buffer de datos **son compartidos por todas las transacciones del sistema.**

Framework inicial: cada uno de los registros de la bitácora se graban en **memoria estable** y **las transacciones se ejecutan en serie.**

EBD2014_18 - Mg. Mercedes Vitturini

Registros Bitácora

- Una estructura muy utilizada para registrar las modificaciones a la base de datos es a través del uso de **registros log** o **registros bitácora**.
- Los **registros log dejan pista de todas las actividades de actualización.**
- Típicamente** un registro log mantiene la siguiente información:
 - Identificador de la transacción.**
 - Identificador del dato modificado.**
 - Valor viejo del dato**, anterior a la modificación.
 - Valor nuevo del dato**, posterior a la modificación.



EBD2014_18 - Mg. Mercedes Vitturini

Registros Bitácora

- Cuando la **transacción T_i se inicia**, se agrega en la bitácora el registro:

$\langle T_i, \text{start} \rangle$

- Cada vez que la transacción T_i ejecuta **write(X)**, se agrega en bitácora un registro:

$\langle T_i, X, v_1, v_2 \rangle$

- donde v_1 representa el valor viejo de X antes de la escritura,
- v_2 el nuevo valor después de la misma,
- T_i y X identifican la transacción y el dato modificado.

EBD2014_18 - Mg. Mercedes Vitturini

Registros de la Bitácora

- Cuando la transacción T_i **finalizó** con su última instrucción agrega en bitácora el registro:

$\langle T_i, \text{commit} \rangle$

- Si la transacción T_i **aborta** agrega en bitácora el registro:

$\langle T_i, \text{abort} \rangle$

EBD2014_18 - Mg. Mercedes Vitturini

Transacción cometida

- Una transacción se dice que está **cometida** si el registro **$\langle T, \text{commit} \rangle$** de bitácora está en **memoria estable.**

- Todos los registros log anteriores de la transacción también deben haber sido almacenados en memoria estable previamente.**

- Dada una transacción en estado cometida, eventualmente, las **escrituras realizadas por ella podrían haber sido actualizada en el buffer de datos** y el output estar pendiente.



EBD2014_18 - Mg. Mercedes Vitturini

Operaciones Redo y Undo

- **Undo de un registro:** $\langle T_i, X, V_1, V_2 \rangle$ escribe a X con el valor **anterior** V_1 .
- **Redo de un registro:** $\langle T_i, X, V_1, V_2 \rangle$ escribe a X con el valor **nuevo** V_2 .

Undo y Redo de Transacciones

- **undo(T_i)** restaura el valor todos los ítems de datos modificados por T_i a sus valores anteriores, recorriendo la bitácora en sentido hacia atrás desde el último registro para T_i
- **redo(T_i)** escribe el valor de todos los ítems de datos actualizados por T_i con el nuevo valor, recorriendo la bitácora hacia adelante desde el primer registro de T_i

EBD2014_18 - Mg. Mercedes Vitturini

Recuperación con Bitácora

IMPORTANTE:

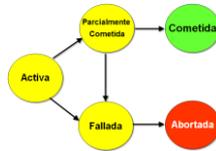
- Los **registros de la bitácora deben siempre ser escritos antes** que la modificación en la base de datos.
 - Asumimos además que la operación output de la bitácora es inmediata.
 - El output de los registros de los datos puede ocurrir en cualquier momento posterior.

EBD2014_18 - Mg. Mercedes Vitturini

Esquemas de Modificación

Existen dos esquemas de modificación

- Modificación Inmediata
- Modificación Diferida



EBD2014_18 - Mg. Mercedes Vitturini

Esquema modificación inmediata

- Bajo el esquema de **modificación inmediata** las modificaciones a la base de datos se realizan **en cualquier momento aún mientras la transacción está activa** (modificaciones no cometidas).
- En **caso de que ocurra un fallo**, es necesario **deshacer los cambios** hechos por una transacción no cometida.

EBD2014_18 - Mg. Mercedes Vitturini

Modificación inmediata: acciones preventivas

- Cuando la transacción T se inicia se agrega en bitácora el registro $\langle T, \text{start} \rangle$
- Cada operación **Write(X)** de T, agrega el registro de bitácora con los **valores anterior y actual** para X,

$$\langle T, X, V_1, V_2 \rangle$$
- Luego se actualiza la base de datos.
- Cuando T está cometida se agrega en bitácora el registro $\langle T, \text{commit} \rangle$

Nunca se actualiza la base de datos **antes de que se escriba el registro de bitácora en memoria estable.**

EBD2014_18 - Mg. Mercedes Vitturini

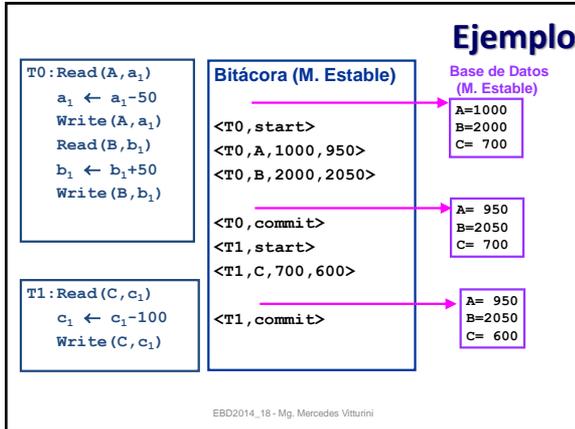
Actualización efectiva de los datos

Bitácora		Base de Datos	
M. Principal	M. Estable	M. Principal	M. Estable
...	...	A=100	A=100
$\langle T, A, 100, 200 \rangle$	$\langle T, A, 100, 200 \rangle$		
		A=200	
			A=200

La actualización en memoria principal del valor de A=200 **debe ser posterior** al grabado en memoria estable del registro de bitácora $\langle T, A, 100, 200 \rangle$.

↓ Tiempo

EBD2014_18 - Mg. Mercedes Vitturini



Modificación inmediata: recuperación ante un fallo

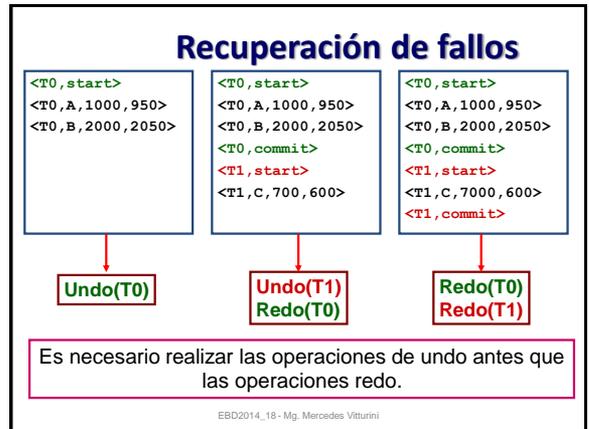
- El esquema de recuperación con modificación inmediata necesita de las operaciones **UNDO** y **REDO** usando información de la bitácora.
 - UNDO(T)**: restaura todos los ítems de dato que T actualiza a los valores que tenía anteriormente.
 - REDO(T)**: asigna los nuevos valores a todos los ítems de dato que actualiza la transacción T.
- Las operaciones undo y redo son **idempotentes**:
UNDO(T)=UNDO(UNDO(T)) Y **REDO(T)=REDO(REDO(T))**

EBD2014_18 - Mg. Mercedes Vitturini

Recuperación ante fallos

- Consideremos que se produjo un **fallo del sistema**. El esquema de recuperación inmediata consulta la bitácora para determinar cuáles transacciones deben deshacerse o rehacerse.
- ¿Cuándo aplicar **undo(T)**?
 - Cuando la bitácora contiene el registro <T,start> pero no contiene el registro <T,commit>.
- ¿Cuándo aplicar **redo(T)**?
 - Cuando la bitácora contiene tanto el registro <T,start> como el registro <T,commit>.

EBD2014_18 - Mg. Mercedes Vitturini



Esquema Modificación Diferida

- En el esquema de **modificación diferida** las modificaciones que realice una transacción (**write**)
 - Durante la ejecución de la transacción sólo se graban en los registros de la bitácora,
 - Las escrituras en la base de datos se demoran hasta que la transacción este cometida.
- La información en la bitácora se utiliza cuando se ejecutan las escrituras diferidas.
- Si ocurre un **fallo de sistema** o **fallo de transacción** antes de que la transacción este cometida se ignora la información de la bitácora.

EBD2014_18 - Mg. Mercedes Vitturini

Modificación diferida: acciones preventivas

- Antes de que comience la ejecución de la transacción T_p, se agrega en bitácora el registro <T_p, start>.
- Si T_p realiza una operación **write(X,v)**, se agrega en bitácora el registro <T_p, X, v>.
 - Modificación diferida requiere solamente del nuevo valor del dato, omitiendo el valor antiguo.
- Cuando T_p está cometida, se escribe el registro de bitácora <T_p, commit>.
- Los registros *log* se usan para las escrituras diferidas.

EBD2014_18 - Mg. Mercedes Vitturini

Modificación Diferida

BD

A=1000
B=2000

MP

A=1000
B=2000

T1

```

Read (A, a1)
a1 = a1 - 50
Write (A, a1)
Read (B, b1)
b1 = b1 + 50
Write (B, b1)
                    
```

Bitacora

<start T1>

<T1, A, 950>

<T1, B, 2050>

<commit T1>

MP

A=950
B=2050

BD

EBD2014_18 - Mg. Mercedes Vitturini

Recuperación y modificación diferida

- Supongamos una secuencia de transacciones T_1, T_2, \dots, T_n ejecutándose en serie.
- En cualquier momento puede ocurrir **una caída del sistema**.
- Para recuperar el estado consistente, *necesitarán rehacerse (REDO T_i)* de todas aquellas transacciones que tienen en bitácora **<T_i, start>** y **<T_i, commit>**.
- ¿Y si ocurriese una caída del sistema mientras se realizan tareas de recuperación?

EBD2014_18 - Mg. Mercedes Vitturini

Ejemplo

T0 transfiere 50 pesos de la cuenta A a la cuenta B.

T1 quita 100 pesos de la cuenta C.

T0:Read (A, a₁)

```

a1 ← a1-50
Write (A, a1)
Read (B, b1)
b1 ← b1+50
Write (B, b1)
                    
```

T1:Read (C, c₁)

```

c1 ← c1-100
Write (C, c1)
                    
```

Inicialmente se asume única transacción activa por vez (ejecuciones en serie)

EBD2014_18 - Mg. Mercedes Vitturini

Ejemplo

T0:Read (A, a₁)

```

a1 ← a1-50
Write (A, a1)
Read (B, b1)
b1 ← b1+50
Write (B, b1)
                    
```

Bitácora

<T0, start>

<T0, A, 950>

<T0, B, 2050>

<T0, commit>

<T1, start>

<T1, C, 600>

<T1, commit>

Buffer de Datos

A=1000
B=2000
C= 700

A= 950
B=2050
C= 700

A= 950
B=2050
C= 600

T1:Read (C, c₁)

```

c1 ← c1-100
Write (C, c1)
                    
```

EBD2014_18 - Mg. Mercedes Vitturini

Recuperación ante Fallos

- El esquema de recuperación ante fallos con modificación diferida usa la operación **REDO** y la información de la bitácora.
 - **REDO(T)**: asigna los nuevos valores a todos los datos que actualiza la transacción T.
- La operación redo es *idempotente*, esto es:

REDO(T) = REDO(REDO(T))
- ¿Cuándo aplicar redo(T) después de un fallo?
 - Siempre que en bitácora existan los registros <T, start> y <T, commit>.

EBD2014_18 - Mg. Mercedes Vitturini

Recuperación de fallos

<T0, start>

<T0, A, 950>

<T0, B, 2050>

<T0, start>

<T0, A, 950>

<T0, B, 2050>

<T0, commit>

<T1, start>

<T1, C, 600>

<T0, start>

<T0, A, 950>

<T0, B, 2050>

<T0, commit>

<T1, start>

<T1, C, 600>

<T1, commit>

No es necesario hacer ninguna acción de recuperación.

Redo(T0)

Redo(T0)
Redo(T1)

EBD2014_18 - Mg. Mercedes Vitturini

Temas de la clase de hoy

- Sistema de Recuperación.
 - Tipos de Fallos.
 - Estructuras de Almacenamiento.
- Mecanismos de recuperación usando bitácora para ambientes no concurrentes.
 - Modificación Diferida.
 - Modificación Inmediata.

Bibliografía

- “Database System Concepts” – A. Silberschatz.
Capítulos 16 (ed. 2010).

EBD2014_18 - Mg. Mercedes Vitturini