LOGGING

ÍNDICE

INTRODUCCIÓN

- El objetivo de la recuperación es asegurar que una base de datos puede procesar transacciones en un modo a prueba de fallas
- Existen básicamente 3 tipos de fallas
  - De transacciones
  - De sistema
  - De almacenamiento
- Estas fallas tienen que ver , básicamente con la perdida de los datos que están en memora

EJEMPLO

- Analicemos las siguientes instrucciones SQL
  1. Begin Transaction
  2. Select saldo from cuenta where numero = 23
  3. Update cuenta set saldo = saldo + 1000
  4. Insert into movimientos (tipo, fecha, cuenta, monto) values("Deposito", 27/10/2017, 23, 1000)
  5. Commit
- ¿Qué sucede internamente en cada una de estas operaciones?
- ¿En qué momento los cambios se reflejan en la base de datos?

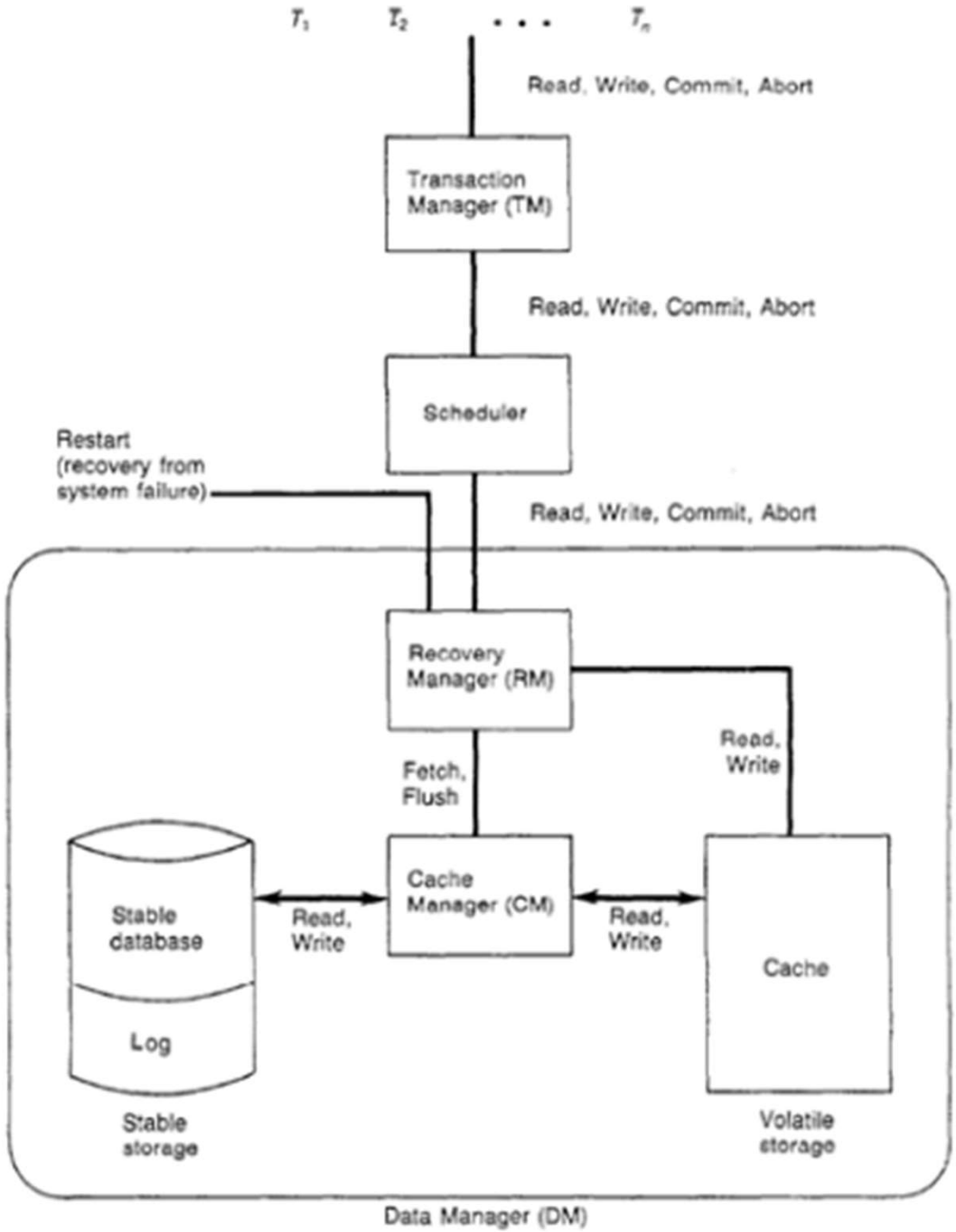


FIGURE 6–1
Model of a Centralized Database System

ÍNDICE

- Introducción
- **Cache Manager**
- Recovery Manager
- Checkpoint

CACHE MANGER

- Es el responsable de interactuar con el almacenamiento no volátil. Sus principales funciones son
  - Fetch (leer de disco)
  - Flush (escribir en disco)
  - Pin
  - Unpin

CACHE MANAGER - FETCH

- Fetch recibe el data item X como parámetro . Hace que el CM efectúe las siguientes acciones
- 1. Selecciona un slot vacío (digamos c) . Si todos los slots están ocupados , selecciona un slot c, le efectúa un flush y lo usa como si estuviera vacío
- 2. Copia el valor que x tiene en el disco a c
- 3. Inicializa el “dirty bit” de c en 0.
- 4. Actualiza el directorio del cache para indicar que el slot c es ocupado por x.

ÍNDICE

- Introducción
- Cache Manager
- **Recovery Manager**
- Checkpoint

RECOVERY MANAGER

- La interface del RM interface está definida por medio de 5 procedimientos:
  - RM-Read(Ti, x): lee el valor de x para la transacción Ti;
  - RM-Write(Ti, x, v): escribe v en x en nombre de la transacción Ti
  - RM-Commit(Ti): commit Ti;
  - RM-Abort(T_i): abort T_i; and
  - Restart: lleva a la base de datos al último estado "comiteado" antes de que haya fallado el sistema.

ALGUNAS CONSIDERACIONES SOBRE EL RECOVERY MANAGER

- Decimos que un recovery manager requiere
  - undo si permite que transacciones no comiteadas escriban en disco
  - Redo si solo escribe transacciones que haya comiteado
- Existen básicamente 4 tipos de recovery manager
  - Redo
  - Undo
  - Undo / Redo
  - No Undo / No redo

REGLAS QUE DEBE OBSERVAR EL RECOVERY MANAGER

- Undo Rule si el disco contiene el último valor comiteado de X antes de reemplazarlo con un valor que no tiene commit es necesario preservar el valor original
- Redo Rule: Antes de que una transacción haga commit los valores que escribió para cada data ítem deben ser almacenados en disco.
- Garbage Collection Rule: Una entrada [Ti, x, V] puede ser eliminada del log sii
- (1) Ti aborto
- o (2) Ti hizo commit , pero hay otra transacción que hizo commit que escribió x después de T , lo que quiere decir que V no es el último valor comiteado de x.

UNDO / REDO 1

- Es el más complicado de todos los algoritmos
- RM-Write(Ti, x, v)
  - 1. Si Ti no está en la lista de las transacciones activas, la agrega
  - 2. Si x no está en el cache , le hace un fetch
  - 3. Agrega [Ti, X, V] al log.
  - 4. Escribe v en el slot del cache ocupado por x
  - 5. Informa que termino el procesamiento al scheduler.
- RM-Read(Ti, x)
  - 1. Si x no está en el cache , le hace un fetch
  - 2. Devuelve el valor de x al scheduler

UNDO / REDO 2

- RM-Commit(Ti)
  - 1. Agrega Ti a la lista de las transacciones commiteadas
  - 2. Informa al scheduler que proceso el commit de Ti
  - 3. Borra Ti de la lista de transacciones activas.
- RM-Abort(Ti)
  - 1. Por cada data ítem x que fue actualizado por Ti
    - Si x no está en el cache le asigna un slot
    - Copia la imagen que tenía X antes de Ti en ese slot
  - 2. Agrega Ti a la lista de transacciones abortadas
  - 3. Informa al scheduler que aborto la transacción Ti
  - 4. Borra Ti de la lista de transacciones activas.

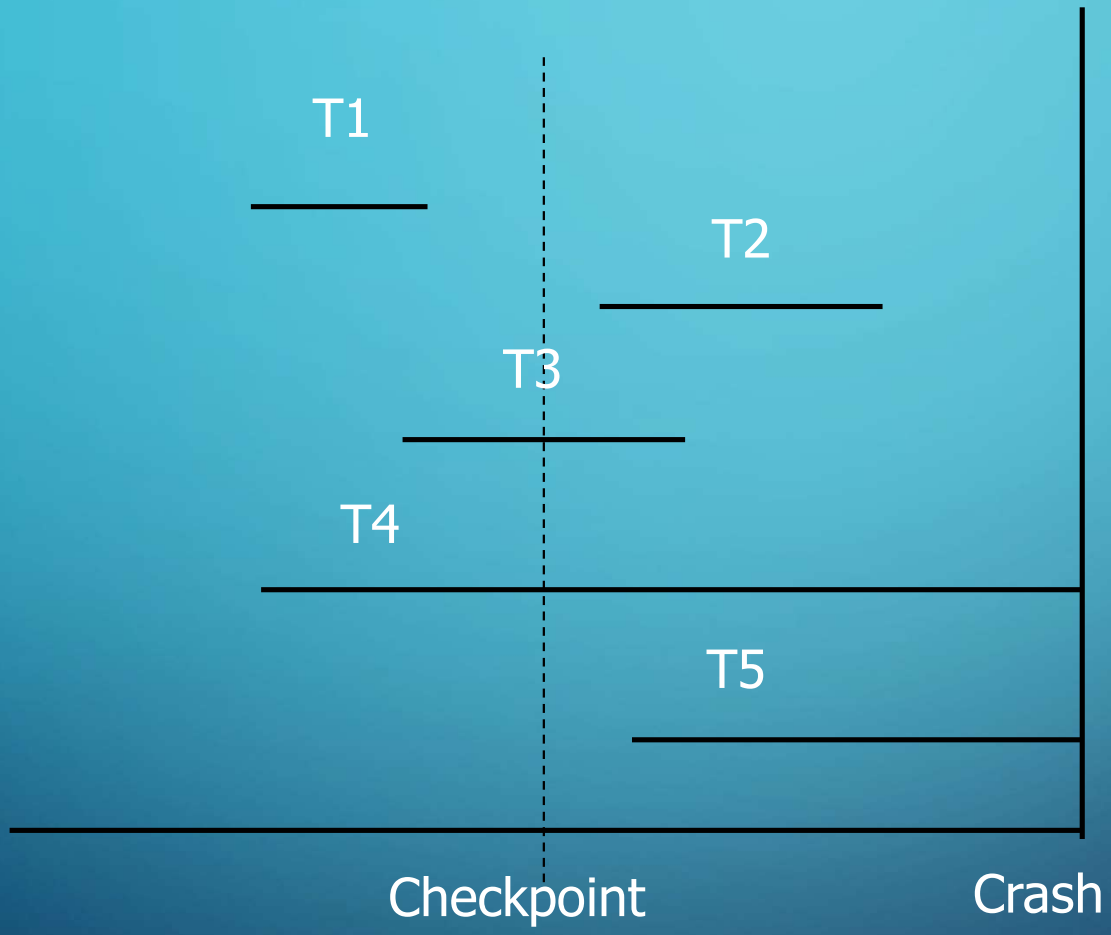
UNDO / REDO : RESTART

1. Borrar todos los slots del cache
2. redone : = { } y undone : = { }
3. Empezar desde el final del log y recorrerlo hasta el comienzo. Repeitr los siguientes pasos hasta que (redone U undone) = todas los data items de la base o hasta que se haya terminado de procesar el log
   - Para cada entrada[Ti, x, v],
     - Si x no pertence a redone U undone, entonces
       - Si x no está en el cache asignarle un slot ;
     - Si Ti está en la lista de transacciones comiteadas,
       - Copiar v en el slot alocado para x
       - redone : = redone U {x} ;
     - Sino (i.e., T, is in the abort list or in the active but not in the commit list),
       - Copiar la imagen anterior de x en el slot que le corresponde a x
       - undone : = undone U {x}
4. Para cada Ti en la lista de transacciones comiteadas
   - Si Ti está en la lista de transacciones activas removerlo
5. Informar al Schedule la finalización del Restart.

ÍNDICE

- Introducción
- Cache Manager
- Recovery Manager
- **Checkpoint**

CHECKPOINT, INTRODUCCIÓN



CHECKPOINT

- El log no puede ser mantenido para siempre y algunos datos son bajados a disco.
- Esto se resuelve mediante el uso de técnicas de checkpoint
- El checkpoint lleva a cabo su acción mediante la combinación de dos tipos de actualización al disco
  - (1) actualizar el log, la lista de transacciones comiteadas y abortadas para indicar cuales modificaciones ya están escritas o deshechas en,
  - y (2) escribir las imágenes "posteriores" de las modificaciones efectuadas por transacciones commiteadas o las imágenes previas de las modificaciones efectuadas por transacciones abortadas en disco
  - La técnica 1 le dice al Restart que updates no tienen que deshacerse o rehacerse. La técnica 2 reduce la cantidad de trabajo que tiene que hacer el Restart, pasando parte de este trabajo al checkpoint
- La técnica (1) es **esencial** a cualquier mecanismo de checkpoint, mientras que la técnica (2) es opcional

TIPOS DE CHECKPOINT

- **Quiescente**, deja de aceptar nuevas transacciones mientras procesa
- **Non quiescente**, sigue aceptando transacciones mientras procesa, el flush real se produce cuando todas las transacciones que estaban activas al comienzo finalizan con COMMIT o ABORT

ENTRADAS AL LOG

- <START Ti>
- <COMMIT Ti>
- <ABORT Ti>
- <Ti, x, v> : para redo logging indica que Ti escribió el valor v en x
- <Ti,x,w> : para undo logging indica que “w” es el valor que tenia x cuando Ti la escribió
- <Ti,x,v> : indica que v es lo que escribió Ti en x
- Checkpoint quiescente
  - <CKPT>
- Checkpoint non quiescente
  - <START CKPT (lista de transacciones activas)>
  - <END CKPT>

LOGGING

- Esta presentación fue armada utilizando, además de material propio, material de
  - "Concurrency Control and Recovery in Database Systems" de Bernstein, Hadzilacos y Goodman

LOGGING

MUCHAS GRACIAS!