UNIVERSIDAD CATÓLICA de Colombia

Transacciones y Control de Concurrencia en los Sistemas Distribuidos

Diego Alberto Rincón Yáñez MSc

darincon@ucatolica.edu.co

Sistemas Distribuidos

Departamento de Ingeniería de Sistemas y Computación

Introducción

- Primer acercamiento: Algoritmos "**Semi-Centralizados**"
- Objetivo de las **transacciones:**
  - Asegurar que todos los objetos gestionados por **un servidor** permanecen en un estado (c) cuando dichos objetos son **accedidos por múltiples transacciones** y en **presencia de fallos**

Sincronización Sencilla

- Las operaciones realizadas en nombre de **diferentes clientes** pueden **interferir** **a veces unas con otras**.
- La **interferencia** puede producir valores **incorrectos** **en los objetos**.
- Frecuentemente se usan **hilos** para permitir concurrentemente las operaciones de varios clientes y aun accediendo, posiblemente, a los **mismos objetos**.

Sincronización Sencilla

- Si los métodos no están diseñados para su utilización en un programa multi-hilo, es posible que las acciones de dos o más ejecuciones concurrentes del método puedan combinarse arbitrariamente y tener efectos extraños.
- Ejemplo en java:

```
public syncronized void deposita(int cantidad) throws RemoteException{

}
```

Sincronización Sencilla

- En varias ocasiones es necesario hacer que los hilos se comuniquen.
- Ejemplo en java:
  - Metodos: wait y notify.
  - Wait: Un hilo llama a wait en un objeto para suspenderse él mismo y permitir a otro hilo ejecutar un método en ese objeto.
  - Notify: Un hilo llama a notify en un objeto para informar a cualquier hilo que esta esperando en el objeto que ha cambiado alguno de sus datos.

Modelos de fallos para transacciones

- En este modelo (Lampson) se intenta que los algoritmos trabajen correctamente en presencia de fallos predecibles, pero no se hacen consideraciones sobre su comportamiento cuando ocurre un desastre.
- El modelo establece:
  - Las escrituras/lecturas pueden fallar.

Modelos de fallos para transacciones

- Los servidores pueden fallar ocasionalmente.
- Puede existir un retardo arbitrario antes que llegue un mensaje.

Transacciones

UNIVERSIDAD CATÓLICA de Colombia

- ¿Qué es?
  - ACID
- Atomicidad

  Todo o nada: Una transacción finaliza correctamente, y los efectos de todas sus operaciones son registrados en los objetos, o si fallan no tienen ningún efecto.

Transacciones

UNIVERSIDAD CATÓLICA de Colombia

- Isolation
  - Cada transacción debe realizarse sin interferencia de otras transacciones.
- Consistency
  - La transacción en el caso que se haga debe hacer pasar el sistema de un estado consistente a otro.
- Durability
  - Los cambios registrados en los recursos debido a transacciones que hagan COMMIT deben soportar fallos siguiendo el modelo de lampson.

Transacciones

UNIVERSIDAD CATÓLICA de Colombia

Con éxito	*Abortado por el cliente*	*Abortado por el servidor*	
AbreTransacción	*AbreTransacción*		*AbreTransacción*
Operación	*Operación*		*Operación*
Operación	*Operación*		*Operación*
⋮	⋮	El servidor aborta la transacción →	⋮
Operación	*Operación*		*ERROR en la operación informado al cliente*
CierraTransacción	*AbortaTransacción*		

Transacciones

UNIVERSIDAD CATÓLICA de Colombia

- Acciones de servicio relacionadas con la ruptura del proceso
  - Si un proceso servidor falla, se reemplaza en algún momento.
    - El nuevo proceso aborta todas las transacciones no finalizadas
    - Usa un proceso de recuperación para restablecer los valores de los objetos producidos por la transacción finalizada de forma correcta.

Transacciones

- Acciones de un cliente relativas a la ruptura del proceso servidor
  - Si un servidor falla mientras una transacción esta en progreso, el cliente será consciente de ello cuando una de las operaciones devuelve una excepción.