Deadlock apunte

En las BD, es común que múltiples procesos o solicitudes concurrentes intenten acceder simultáneamente a recursos compartidos (como archivos, registros en una base de datos o servirios). Esto puede llevar a **deadlocks** si no se gestionan correctamente los mecanismos de sincronización.

Deadlock

Un *deadlock* ocurre cuando **dos o más hilos o procesos están esperando indefinidamente a que otros liberen un recurso**, pero ninguno puede avanzar porque todos están bloqueados entre sí. Esta situación puede aparecer en:

- **Backends web con acceso concurrente a bases de datos**
- **Sistemas de caché compartida**
- **Módulos de bloqueo en servidores concurrentes (como Node.js con worker threads)**

Starvation

La **muerte por inanición** (*starvation*) se da cuando un proceso o hilo **espera indefinidamente para acceder a un recurso** porque otros procesos siempre tienen prioridad o se adelantan en la cola.

En sistemas de bases de datos, puede suceder si:

- Una consulta tiene menor prioridad que otras (por ejemplo, `SELECT` vs `UPDATE`)
- Hay una política de espera injusta en la aplicación
- Una transacción nunca logra obtener los bloqueos necesarios

Condiciones para prevenir deadlocks

Una solución al problema de la sección crítica (ya sea en memoria compartida o en recursos como archivos o registros) debe cumplir con:

1. **Exclusión mutua:**
   Si un proceso está ejecutando su sección crítica, **ningún otro** puede hacerlo al mismo tiempo.
2. **Progreso:**
   Si ningún proceso está en su sección crítica y hay procesos que desean

entrar, **la decisión sobre quién entra debe tomarse sin demoras innecesarias**.

3. **Espera limitada (Bounded Waiting):**
   Debe existir un límite en el número de veces que otros procesos pueden entrar en su sección crítica antes de que se permita el ingreso de uno en espera.

Aplicación en desarrollo web

En sistemas web reales, esto puede significar:

- **Bloqueos de filas o tablas en SQL:**
  Uso de `LOCK`, `SELECT ... FOR UPDATE`, o el motor de almacenamiento (como InnoDB en MySQL) puede llevar a interbloqueos.
- **Colas de tareas en backends asincrónicos:**
  Si no se maneja la prioridad y concurrencia correctamente, algunas tareas podrían quedar esperando indefinidamente.

Cómo evitarlo

- Uso de **transacciones** con buen diseño de orden de bloqueo
- Aplicación de **timeouts** en operaciones concurrentes
- Diseño de lógica de negocio para **evitar ciclos de dependencia**
- En bd: configurar **niveles de aislamiento** correctamente