

# Desarrollo de Aplicaciones Web

Tema VIII: Concurrencia

# Temas de la Unidad

- Bloqueo Optimista Fuera de Línea
- Bloqueo Pesimista Fuera de Línea
- Bloqueo de Granularidad Gruesa
- Bloqueo Implícito

# Bloque Optimista Fuera de Línea

## Planteamiento del Problema

- Cuando hay un conflicto entre transacciones
  - La idea es detener el conflicto para evitarlo
  - En el proceso se hacer retroceder la transacción
- Dependiendo del ámbito de trabajo
  - Se usan sistemas transaccionales
  - Se usan sistemas no transaccionales
- En el Segundo no se puede depender de la BD
  - Para asegurar que cada transacción sea coherente
  - Y que los datos del registro son confiables

# Bloque Optimista Fuera de Línea

## Planteamiento del Problema(Cont)

- La integridad de datos se pone en riesgo
  - Cuando 2 sesiones trabajan con los mismos datos
  - Se puede perder procesos de actualización
  - Generando inconsistencias de datos
- Si hay 2 sesiones (1 que lee, 1 que escribe)
  - Se puede provocar una lectura inconsistente

# Bloque Optimista Fuera de Línea

## Solución

- El bloqueo optimista fuera de línea
  - Resuelve el problema mediante validación
    - Para que los cambios de 1 actualización
    - No entre en conflicto con la otra sesión
- La validación exitosa significa que
  - Se puede realizar el bloqueo sin problema
  - Proceder con los cambios de datos en el registro
  - Todo de forma consistente
- Si la validación y actualización
  - Están en una única transacción
  - Se puede asegurar su consistencia

# Bloque Optimista Fuera de Línea Solución (Cont)

- La técnica pesimista asume que
  - La probabilidad de conflicto es alta
- La técnica optimista asume que
  - La probabilidad de conflicto es baja
- Lo anterior permite la concurrencia del sistema
  - Esto no limita el trabajo de múltiples sesiones
  - Permitiendo que varios usuarios trabajen
    - Utilizando los mismos datos al mismo tiempo
    - Dando más flexibilidad al sistema

# Bloque Optimista Fuera de Línea

## Implementación

- Se debe validar el tiempo transcurrido
  - Donde la sesión fue abierta y cargó el registro
  - Que otra sesión no lo haya alterado
- Se permite leer en cualquier momento
  - Esta transacción no es “Peligrosa”
  - Es válida solo en el momento que se lee
- Para operar y evitar datos corruptos
  - Se adquiere un bloqueo optimista
  - Para cada miembro del conjunto de cambio
  - Para aplicar los cambios en la base de datos

# Bloque Optimista Fuera de Línea

## Implementación (Cont)

- Normalmente se asocia un número de versión
  - A cada registro en el sistema
  - Cuando se carga el registro ese número se mantiene
  - Para conseguir el bloqueo optimista se compara
    - Los números de versión actual con los datos de registro
    - Si la verificación es exitosa se puede aplicar cambios
    - Incluyendo un incremento en el # de versión
    - Este último evita los datos de registro inconsistentes
    - Una sesión con versión antigua no se puede aplicar

# Bloque Optimista Fuera de Línea

## Implementación (Cont)

- Con BD relacional RDBMS la verificación
  - Es cuestión de agregar # Sesión al WHERE
  - Para SQLs de actualización o eliminación de info
  - Este SQL puede obtener el bloqueo
    - Con esto puede proceder con la actualización
    - Todo realizado de forma exclusiva
  - Se debe realizar una inspección final
    - Si se devuelve 1 fila, se realizó todo con éxito
    - Si se devuelve 0 filas, no se realizó el cambio
      - Esto debe hacer un Rollback para evitar cambios
      - La transacción debe abortarse y tratar de resolver el conflicto

# Bloque Optimista Fuera de Línea

## Implementación (Cont)

- Se puede manejar información adicional
  - Para dar seguimiento al último registro cambiado
  - Útil en la administración de conflicto
    - Cuando existen conflictos simultáneos
  - Se puede informar al usuario del error
    - Con trazabilidad de cuándo y quién realizó el cambio
  - Aquí es más recomendable el # de versión
    - Los relojes del sistema son poco fiables
    - Máxime si la coordinación es entre varios servidores

# Bloque Optimista Fuera de Línea

## Implementación (Cont)

- Otra opción es comparar todos los campos
  - Se verifican los datos antes de modificarlos
  - Esto elimina la necesidad del # de versión
    - Útil si no se puede hacer cambios a las tablas en la BD
  - Pero esta versión puede ser complicada
    - Necesitando cláusulas WHERE complejas y largas
    - Haciendo el UPDATE difícil
      - De desarrollar
      - De mantener
    - Esto además provoca un impacto en el rendimiento

## Ventajas y Desventajas

- Cuando la posibilidad de conflicto es baja
  - El bloqueo optimista es ideal
- En casos donde el conflicto es alto
  - El ambiente es poco amigable para el usuario
  - Se le informa al final de toda la transacción
    - Ha ocurrido un error, todo se debe ejecutar nuevamente
  - 1 usuario puede hacer la actualización
    - Los demás tendrán un problema
    - Tendrán que ejecutar todo de nuevo
  - Si la labor de usuario es extensa por transacción
    - El sistema será impopular, poco amigable

## Ventajas y Desventajas (Cont)

- Es fácil de implementar
- Tiene pocos problemas en su desarrollo
- Ambas razones lo hacen ideal para iniciar
- Sirve como pivote para arrancar el proceso
  - Maneja los casos donde es ideal
  - Se complementa con el pesimista
  - En caso de ser necesario

# Quiz

Saque 1 hoja

Coloque su nombre y ID

Tiene 10 minutos para entregarlo

# Indique en sus propias palabras

- Arquitectura en capas
  - Cuál es la arquitectura en capas más usada?
  - De una breve descripción de cada capa
  - Cual es una ventaja de esta arquitectura?
- Explique con 3 ejemplos
  - Cómo se apega su desarrollo del proyecto 1 a la teoría que se ha visto?

# Bloqueo Pesimista Fuera de Línea

# Bloque Pesimista Fuera de Línea

- Se permite el acceso exclusivo a los datos
  - De forma que solo 1 transacción tiene acceso
  - Al mismo tiempo
- En un ambiente ideal se podría abrir 1 sesión
  - A lo largo de toda la transacción
  - Esto no es normalmente fácil o útil
  - Máxime cuando las transacciones son muy largas
- Se puede trabajar con múltiples transacciones
  - Para manejar el acceso simultáneo a los datos
  - Y para manipular los recursos propios

# Bloque Pesimista Fuera de Línea

- Se previene el conflicto por completo
- Se obliga a la transacción a adquirir el bloqueo
- Esto se realiza antes de empezar a usar la info
- Esto asegura que se puede completar la tarea
- Sin encontrar un rechazo por concurrencia

# Bloque Pesimista Fuera de Línea

## Implementación

- Los bloqueos pesimistas pueden ser
  - Bloqueo de Escritura Exclusivo
  - Bloqueo de Lectura Exclusivo
  - Bloqueo de Lectura/Escritura Exclusivo
    - Es una combinación de los dos anteriores

# Bloqueo de Escritura Exclusivo

- Se requiere cuando una transacción
  - Adquiere un bloqueo para realizar un modificación
  - Los datos que se modifican son los de la sesión
- Esto evita 2 transacciones
  - Conectadas al mismo registro
  - Ejecutando tareas al mismo tiempo
- Este esquema ignora la lectura de datos
- Óptimo cuando la lectura no es crítica
  - Para que tenga los datos más recientes

# Bloqueo de Lectura Exclusivo

- Necesario al ser crítico que la info sea correcta
  - Sin importar la intención de edición
- La transacción adquiere un bloqueo exclusivo
  - En el proceso de carga de los datos
- Este método restringe severamente el sistema
  - Evitando a toda costa la concurrencia
- En ambientes empresariales
  - No es muy recomendable su uso

# Bloqueo de Lectura/Escritura Exclusivo

- Este provee lo mejor de ambos mundos
  - Da la seguridad del bloqueo de lectura
  - Permite más concurrencia del bloqueo de escritura
- Es más complicado por su configuración
  - Se debe manejar bien la relación entre ambos
  - Para sacar provecho de ambos escenarios
- Los bloqueos de lectura/escritura
  - Son mutuamente excluyentes

# Bloqueo de Lectura/Escritura Exclusivo

- No se puede bloquear para escribir
  - Si existe un bloqueo previo sobre el registro
  - Aún cuando el registro anterior sea de lectura
  - Lo mismo ocurre en sentido contrario
- Si existe un bloqueo de lectura
  - Se puede permitir accesos concurrentes de lectura
  - Esto no provocará ningún cambio
  - Se asegura que todos tienen bien la información

# Bloqueo de Lectura/Escritura Exclusivo

- Es complicado de implementar
- Para su diseño se debe
  - Maximizar la concurrencia
  - Satisfacer las necesidades
  - Minimizar la complejidad lo más posible
- Antes de tomar decisiones
  - Se debe asegurar que todos entienden bloqueos
  - Esto tanto para el diseño como para el desarrollo

# Bloqueo de Lectura/Escritura Exclusivo

- Un bloqueo incorrecto
  - Puede bloquear todos los registros
- Un bloqueo de los tipos incorrectos de registro
  - Creará un método ineficaz
- Esto puede generar
  - No realizar bloqueos adecuados
  - Degradar la concurrencia de un sistema multi-usuario
  - Tal estrategia no se puede corregir con una técnica correcta

# Bloque Pesimista Fuera de Línea

## Implementación

- Una vez decidido el tipo de bloqueo
  - Se debe definir el administrador de bloqueos
- El administrador de bloqueos tiene que
  - Aceptar o denegar la solicitud de bloqueo
  - Con esto se adquiere o quita un bloqueo
  - Necesita saber que está siendo bloqueado
  - Así como el dueño del bloqueo actual
  - Para que el administrador pueda tomar decisiones

# Bloque Pesimista Fuera de Línea

## Implementación

- El proceso de identificar una transacción
  - Puede ser un proceso complejo
  - Esto hace difícil la tarea del administrador
  - Para ello se considera el concepto de sesión
  - Sesión y transacción pueden ser intercambiables
    - Mientras la transacción sea parte de una sesión
    - La sesión está bien como propietaria de un bloqueo
- El gestor de bloqueos puede manejar
  - Una tabla hash en memoria
  - Una tabla en base de datos

# Bloque Pesimista Fuera de Línea

## Implementación

- Debe existir eso si solamente 1 tabla de bloqueos
  - Esta debe ser de instancia única (singleton)
- En los casos que hay más de 1 servidor
  - La solución para el manejo del cluster
  - Es en este caso la tabla de base de datos
  - Y el gestor accediendo a los datos almacenados

# Bloque Pesimista Fuera de Línea

## Implementación

- Es importante manejar el protocolo
  - Para interactuar entre
    - La transacción y el administrador de bloqueos
  - El protocolo debe incluir
    - Qué se va a bloquear
    - Cuándo se va a bloquear
    - Cuándo liberar el bloqueo
    - Cómo responder cuando el bloqueo no se puede adquirir

# Bloque Pesimista Fuera de Línea

## Implementación

- La transacción debe obtener el bloqueo
  - Antes de poder cargar los datos
  - Esto asegurar cargar la última versión asegurada
  - Así se puede bloquear en firme
  - Trabajando con los datos correctos
- Según el tipo de bloqueo
  - Puede ser que el orden no sea importante
  - En caso contrario se puede
    - Realizar una comprobación optimista
    - Después de adquirir el bloqueo pesimista

# Bloque Pesimista Fuera de Línea

## Implementación

- Se debe estar seguro de tener la última versión
  - Esto después de haberlo bloqueado
  - Esto se traduce en bloquear antes de cargar
- La regla más simple para el desbloqueo
  - Es liberar cuando la transacción se completa
  - En algunos casos se puede hacer antes de finalizar
    - Esto en función del tipo de bloqueo
    - La intención de la utilización del objeto en la transacción
    - Esto debe suceder solo en casos especiales justificados

# Bloque Pesimista Fuera de Línea

## Implementación

- Si no puedo adquirir el bloqueo
  - Lo ideal es abortar el proceso
  - Esto es aceptable aún para los usuarios
    - Debido a que sucede de forma temprana
- Es importante que el diseño y desarrollo
  - Tome esto en cuenta para evitar fallos
    - Para transacciones polémicas
    - Que es mejor informar de su imposibilidad para ejecutar
    - Por eso es importante adquirir los bloqueos a priori

# Bloque Pesimista Fuera de Línea

## Implementación

- Es importante administrar los tiempos de espera
  - Para las sesiones que fallaron en obtener la cerradura
- Si una transacción en progreso falla
  - No podrá liberar los bloqueos que tiene asignados
- Otro escenario similar es cuando
  - El usuario abandona el proceso a mitad del camino
  - Es necesario administrar los tiempos de espera
    - Ya sea por el servidor de aplicaciones o
    - Administrado por el contexto de la aplicación

# Bloque Pesimista Fuera de Línea

## Implementación

- Es importante administrar los tiempos de espera
  - Para las sesiones que fallaron en obtener la cerradura
- Si una transacción en progreso falla
  - No podrá liberar los bloqueos que tiene asignados
- Otro escenario similar es cuando
  - El usuario abandona el proceso a mitad del camino
  - Es necesario administrar los tiempos de espera
    - Ya sea por el servidor de aplicaciones o
    - Administrado por el contexto de la aplicación

# Bloque Pesimista Fuera de Línea

## Implementación

- La liberación se puede manejar con 1 objeto
  - Un objeto utilitario que libera los bloqueos
  - Para las sesiones que no son válidas
- Otra opción es un time-stamp de transacción
  - Asignado al bloqueo
  - Considerando un bloqueo inválido
  - Si se supera cierta edad

## Ventajas y Desventajas

- El bloqueo pesimista es apropiado
  - Si hay posibles conflictos entre sesiones
  - Si el coste del conflicto es muy alto
  - Así el usuario no tiene que desechar su trabajo
- Este tipo de bloqueo es seguro que
  - Creará altos niveles de contención de datos
  - Es importante que sea un complemento del optimista
  - Se debe utilizar solo donde es necesario

# Bloqueo de Granularidad Gruesa

# Bloqueo de Granularidad Gruesa

- Se bloquea un conjunto de objetos relacionados
- La edición se puede agrupar y hacer un bloqueo
- Puede suceder en ambientes donde
  - 1 Cliente tiene un set de direcciones asignadas
- Esto tiene sentido en ambientes que lo permite
- El bloqueo individual se vuelve muy complejo
  - Habría que tener un código de búsqueda especial
  - Que los encuentre y bloquee individualmente
  - La complejidad crece exponencialmente
  - Dependiente de los grupos a ser bloqueados

# Bloqueo de Granularidad Gruesa

- Se cubre muchos objetos al mismo tiempo
- Permite simplificar la agrupación
  - Para bloquear los objetos relacionados
  - Para liberar los objetos relacionados
  - Para el proceso de carga de los miembros
  - Todo se trabaja por grupo como una unidad

# Bloqueo de Granularidad Gruesa

## Implementación

- Primer paso
  - Se debe crear un único punto de contención
  - Con esto se hace un bloqueo de los objetos del grupo
  - Esto permite 1 bloqueo por todo el conjunto
- Segundo paso
  - Seleccionar la ruta más corta para el bloqueo único
  - Esto minimiza el número de miembros a identificar
  - Para facilitar su carga en memoria y obtener el bloqueo

# Bloqueo de Granularidad Gruesa

## Implementación

- Si se usa bloqueo optimista compartido
  - Cada elemento compartirá un objeto tipo versión
  - Este es el punto de contención
  - Al incrementar la versión se bloqueará el grupo
  - Generando el bloqueo compartido
  - Cada miembro del grupo referencia la “Versión”
  - Esto minimiza la ruta de acceso a la contención

# Bloqueo de Granularidad Gruesa

## Implementación

- Si se usa bloqueo pesimista compartido
  - Todos los miembros del grupo comparten 1 objeto
  - Dicho objeto puede ser bloqueado
  - Un objeto similar al bloqueo optimista
    - Puede ser un buen candidato para dicha tarea

# Bloqueo de Granularidad Gruesa

## Implementación

- Otra alternativa: Definir un agregado
  - Como un conjunto de objetos asociados
  - Se tratan como una única unidad
  - Sobre la cual se hacen cambios de datos
  - Cada agregado tiene una raíz
  - Desde aquí se hace el bloque de raíz
  - La raíz es el único punto de acceso a los miembros
  - La raíz define que está incluido en el conjunto
  - El trabajo de 1 miembro exige el bloque de todos

# Bloqueo de Granularidad Gruesa

## Ventajas y Desventajas

- Se utiliza por requerimientos de negocio
- En ambientes de factura
  - Se tiene una serie de artículos relacionados
  - Que se contienen dentro de la factura
  - Aquí editar la factura y los artículos por aparte
    - Pierde todo sentido
- Un efecto positivo es que
  - La adquisición y liberación del bloqueo es barata

# Bloqueo Implícito

# Bloque Implícito

- Permite al ambiente realizar los bloqueos
- Esto se hace de forma implícita
- Algunos ambientes ya tienen este ambiente
  - Viene pre-construido
  - Se puede sacar provecho de su código
  - Y su esquema prestablecido

# Bloque Implícito

- Es importante no dejar "Lagunas" en su uso
  - Si se olvida escribir una línea de código
  - Y con esta se adquiere el bloqueo
  - Se puede generar un esquema inútil
- Estos ambientes son difíciles de probar
  - Aún algunos errores pueden pasar las pruebas
  - Materializándose únicamente en producción

# Bloque Implícito

- La tarea de bloqueo no se puede pasar por alto
  - Es mejor que no sea manipulada por el programador
  - Sino manejado implícitamente por la aplicación
- Las aplicaciones empresariales combinan
  - Frameworks, librerías, clases
  - Generación de código
  - Todo esto ofrece ventajas y oportunidades
  - Para facilitar el bloqueo implícito

# Bloque Implícito

## Implementación

- Se debe restructurar (Refactoring)
  - Modificando el código para incluir el bloqueo
  - Incluyendo todo mecanismo necesario de código
  - Para que el bloqueo sea manejado por el framework
- Las herramientas de generación de código
  - Sirven para cumplir con el bloqueo
  - Siguiendo un esquema adecuado

# Bloque Implícito

## Implementación

- Es importante tener un checklist
  - De tareas obligatorias para 1 transacción
  - Para incluirla en 1 estrategia de bloqueo
- En bloqueo optimistas
  - El checklist elementos tales como
    - Almacenar números de versión por registro
    - Incluir la versión en los criterios de actualización SQL
    - Almacenar la versión incrementada al cambiar el registro

# Bloque Implícito

## Implementación

- En bloqueo pesimista
  - El checklist incluye elementos como
    - Adquirir bloqueos necesarios para cargar información
    - Normalmente un bloqueo exclusivo de lectura
    - O la parte de lectura en un esquema Lectura/Escritura
    - Liberación de los bloqueos al finalizar la sesión

# Bloque Implícito

## Implementación

- Ambos tipos de bloqueo limitan la concurrencia
  - Se puede impactar en gran medida los tiempos de respuesta
  - Es importante adquirir los bloqueos para escritura
  - Previo a cualquier tipo de transacción
  - Esto se puede manejar con verificación del framework
  - Para asegurar que el proceso sigue ese orden
  - Si no se adquirió el bloqueo es un error de programación
  - El código debería generar
    - Un error en el proceso
    - Una excepción para informar del problema en cuestión

# Bloque Implícito

## Ventajas y Desventajas

- Facilita su creación y administración
  - Gracias a las ventajas de los frameworks de desarrollo
  - Pese a esto no le permite al desarrollador ignorar
    - Que debe crear los mecanismos necesarios para su uso
  - Es hasta cierto punto difícil de probar
  - Puede ser propenso a generar deadlocks
  - Las aplicaciones pueden fallar de forma inesperada
  - Deja en manos del framework su manejo