UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Práctica #9: Bloqueo y Concurrencia

PRESENTAN

Paredes Zamudio Luis Daniel
Rivera Morales David
Tapia Hernandez Carlos Alberto

PROFESOR

Erick Daniel Arroyo Martinez

ASIGNATURA

Fundamentos de Bases de Datos

19 de Noviembre de 2024

1. Identificación de Áreas de Mejora

Para mejorar el manejo de concurrencia en el procedimiento descrito, se pueden considerar las siguientes áreas de mejora:

1. **Transacciones Cortas**: Mantener las transacciones lo más cortas posibles para minimizar el tiempo durante el cual los recursos están bloqueados reduce la probabilidad de conflictos de concurrencia.

2. **Desacoplamiento de Lecturas y Escrituras**: Se pueden separar las operaciones de lectura y escritura en diferentes procedimientos o transacciones para reducir la contención. Las lecturas pueden realizarse sin bloqueos, mientras que las escrituras pueden manejarse de manera más controlada.

3. **Mejorar Niveles de Aislamiento** Ajustar los niveles de aislamiento de las transacciones según sea necesario. Por ejemplo, utilizar el nivel de aislamiento `READ COMMITTED` para permitir lecturas no bloqueantes mientras se asegura que las escrituras sean consistentes.

2. Argumentación y Elecciones de Implementación

1. **Bloqueos**:

   a) *Justificación*: Los bloqueos se utilizan para asegurar que solo una transacción pueda modificar un recurso específico a la vez, evitando así conflictos de concurrencia. En el código los usuarios pueden seleccionar productos sin interferir con el inventario, por lo que se están utilizando bloqueos de lectura (o no se están utilizando bloqueos) para permitir acceso concurrente a los datos de inventario sin que existan problemas.

   b) *Contribución*: Se permiten múltiples lecturas concurrentes sin bloqueos, lo cual mejora el rendimiento y la escalabilidad del sistema, ya que los usuarios pueden acceder a la información del producto sin esperar a que otros usuarios terminen sus operaciones.

2. **Transacciones**:

   a) *Justificación*: Las transacciones aseguran que un conjunto de operaciones se ejecute de manera atómica, consistente, aislada y duradera (ACID). De esto se busca que las operaciones de actualización del inventario y creación de pedidos se realicen dentro de transacciones para asegurar la consistencia de los datos.

   b) *Contribución*: Utilizar transacciones garantiza que las operaciones críticas se completen correctamente y que los datos permanezcan consistentes incluso en caso de fallos o errores.

3. **Procedimientos Almacenados**:

   a) *Justificación*: Los procedimientos almacenados encapsulan la lógica de negocio en la base de datos, permitiendo un acceso controlado y eficiente a los datos.

   b) *Contribución*: Facilitan la reutilización del código (como es el caso de `VerificarStock`), mejoran la seguridad al controlar el acceso a los datos y pueden optimizar el rendimiento al reducir la cantidad de datos transferidos entre la aplicación y la base de datos.

3. Ejemplo de Ejecución de la Solución

Escenario: Dos usuarios intentan procesar pagos y crear órdenes al mismo tiempo para el mismo producto.

1. *Selección de Productos*: Ambos usuarios seleccionan productos sin interferencia en el inventario.

```
                    -- Usuario 1 selecciona el producto
        CALL SeleccionarProducto('S10_1678', @nombreProducto1, @lineaProducto1,
 @escalaProducto1, @proveedorProducto1, @descripcionProducto1,
@cantidadEnStock1, @precioCompra1, @precioVentaSugerido1);

        -- Usuario 2 selecciona el mismo producto
        CALL SeleccionarProducto('S10_1678', @nombreProducto2, @lineaProducto2,
 @escalaProducto2, @proveedorProducto2, @descripcionProducto2,
@cantidadEnStock2, @precioCompra2, @precioVentaSugerido2);

```

2. **Procesamiento de Pagos y Creación de Órdenes**: Ambos usuarios intentan procesar pagos y crear órdenes simultáneamente.

```
                        -- Usuario 1 intenta procesar el pago y crear la orden
        CALL ProcesarPagoYOrden(103, 'CH12345', 500.00, 'S10_1678', 5,
100.00);

        -- Usuario 2 intenta procesar el pago y crear la orden al mismo
tiempo
        CALL ProcesarPagoYOrden(104, 'CH12346', 500.00, 'S10_1678', 5,
100.00);

```

3. **Detalle del Procedimiento** `ProcesarPagoYOrden`

   a) *Inicio de Transacción y Establecimiento del Nivel de Aislamiento:* Se inicia una transacción y se establece el nivel de aislamiento a `SERIALIZABLE` para evitar modificaciones concurrentes en el inventario.

   b) *Verificación de Stock:* El procedimiento `VerificarStock` bloquea la fila del producto para verificar si hay suficiente stock disponible.

   c) *Registro del Pago:* Si hay suficiente stock, se registra el pago en la tabla `payments`.

   d) *Creación de la Orden:* Se crea una nueva orden en la tabla `orders`.

   e) *Creación del Detalle de la Orden:* Se crea un nuevo detalle de orden en la tabla `orderdetails`

   f) *Actualización del Inventario:* Se actualiza el inventario, reduciendo la cantidad en stock del producto.

   g) *Commit de la Transacción:* Se realiza el commit de la transacción, liberando los bloqueos y permitiendo que otros procesos continúen.

4. **Ejemplo de Salida**:

   - *Usuario 1*: Si el stock es suficiente, el procedimiento se completa exitosamente y se crea la orden.
   - *Usuario 2*: Si el stock no es suficiente después de que el Usuario 1 ha procesado su orden, el procedimiento lanza una excepción indicando `"Stock insuficiente"`.