**Universidad Nacional De Asunción**

**Facultad Politécnica**

**Ingeniería en Informática**

Sistemas Distribuidos

Tarea - Unidad 7

**Profesores:**

* Fernando Mancía.
* Moyses Molinas.

**Materia:** Sistemas Distribuidos.

**Alumnos:**

* Marcos Raúl Flores Duarte.
* Fabrizio Amadeo Rejala Galeano.

**Sección:** TQ.

2024

**Ejercicio 1**

Una aplicación de gestión escolar registra la asistencia diaria de los estudiantes. Cada vez que se marca una asistencia, se actualizan varias tablas en la base de datos, incluyendo las tablas de asistencia, historial del estudiante y estadísticas diarias de asistencia.

**1**. Describe cómo estructurarías una transacción única que incluya la actualización de estas tres tablas. Explica cómo se asegura la consistencia de la base de datos si ocurre un fallo durante el registro de asistencia.

Para estructurar una transacción que actualice las tres tablas (asistencia, historial del estudiante y estadísticas diarias de asistencia), se debe realizar una transacción atómica, donde todas las operaciones dentro de la transacción se completan correctamente o ninguna de ellas se aplica.

Dentro de esta transacción única, podríamos seguir estos pasos:

* **Insertar o actualizar** en la tabla de asistencia con la información del día.
* **Actualizar** el historial del estudiante, agregando un registro que refleje la asistencia o inasistencia del día.
* **Actualizar** las estadísticas diarias de asistencia, incrementando los contadores de estudiantes presentes o ausentes.

Esta estructura asegura que todas las actualizaciones relacionadas con el registro de asistencia se realicen de manera coordinada y bajo una misma transacción, garantizando que no existan inconsistencias parciales.

**Consistencia ante fallos:** Si ocurre un fallo en algún momento durante el proceso de actualización, el sistema revertirá todas las operaciones dentro de la transacción, volviendo la base de datos a su estado inicial, como si ninguna operación hubiese ocurrido. Esto asegura que no se dejarán registros incompletos o inconsistentes en ninguna de las tablas.

**2.** Analiza qué sucedería si la transacción falla en mitad del proceso. ¿Cómo el rollback ayuda a mantener la integridad de los datos?

Si la transacción falla a la mitad del proceso, por ejemplo, después de actualizar la tabla de asistencia pero antes de modificar las estadísticas diarias, el sistema invocará un **rollback**. El rollback deshará todas las operaciones realizadas hasta el punto del fallo, restaurando el estado anterior de la base de datos.

**Importancia del rollback para la integridad:** El rollback es esencial para mantener la integridad de los datos, ya que evita que se guarden registros parciales o inconsistentes. En este caso, ninguna de las tres tablas se actualizaría si la transacción no completa todos sus pasos. Esto asegura que la base de datos siempre se mantenga en un estado coherente, sin entradas incompletas o conflictos entre registros relacionados.

**Ejercicio 2**

Un sistema de paquetes turísticos permite a los usuarios reservar varios servicios, como vuelos, hotel y alquiler de auto, en una sola operación. Cada servicio es proporcionado por un proveedor diferente, y si cualquiera de estos servicios no se confirma, la reserva entera debe cancelarse.

**1.** Proponer una estrategia de transacciones anidadas para modelar este sistema de reservas, donde cada servicio representa una transacción anidada dentro de la transacción principal. ¿Cómo debería diseñarse la transacción principal para que cada servicio (transacción anidada) dependa de la confirmación de los demás?

La transacción principal se diseñará para contener tres transacciones anidadas, una para cada servicio (vuelo, hotel y alquiler de auto). Cada una de estas transacciones anidadas representa la confirmación de un servicio individual dentro del paquete de reserva. En este caso, la transacción principal actúa como un "padre" que sólo se confirma si todas sus transacciones hijas (las reservaciones individuales) se confirman.

Cada transacción anidada (hija) debe completarse de forma exitosa e independiente; sin embargo, al estar dentro de la transacción principal, depende de esta para que sus cambios sean permanentes. La transacción principal, diseñada como una unidad indivisible (propiedad de atomicidad), requiere que cada subtransacción llegue a su fin de forma satisfactoria. Si alguna de las transacciones anidadas falla (por ejemplo, si el vuelo o el hotel no están disponibles), la transacción principal debe abortarse, provocando que todas las subtransacciones se deshagan y ninguna reservación quede en estado parcial.

**2.** Explicar cómo se gestionaría el sistema en caso de que uno de los proveedores de servicios no pueda confirmar la reserva, y cómo se implementaría el rollback de la transacción completa.

Si uno de los proveedores no puede confirmar la reserva, el sistema debe ejecutar un rollback de toda la transacción principal para mantener la consistencia de la operación. En este escenario, el rollback desharía cualquier cambio que las transacciones anidadas ya hubieran realizado antes de que ocurriera el fallo. Este comportamiento asegura que el sistema no contenga datos inconsistentes, como una reserva de hotel sin el vuelo correspondiente.

La implementación de este rollback se basaría en el concepto de "todo o nada" en transacciones. Para cada operación que se realiza en las subtransacciones, el sistema mantiene un registro (log) de los cambios en un almacenamiento temporal. Si una subtransacción falla, el log permite revertir cada cambio realizado en el proceso, devolviendo el sistema al estado anterior al inicio de la transacción principal, como si nada hubiera ocurrido.

**3.** ¿Qué tipos de bloqueos y técnicas de control de concurrencia serían necesarios para evitar que otros usuarios reserven el mismo vuelo o habitación de hotel en el mismo instante?  
Para evitar que otros usuarios reserven el mismo vuelo o habitación al mismo tiempo, se deben implementar bloqueos de tipo exclusivo (bloqueo X o bloqueo de escritura) sobre los recursos (vuelos, habitaciones, autos) mientras se procesa la transacción.

Además, se puede emplear control de concurrencia basado en bloqueos, donde cada recurso queda bloqueado hasta que se confirme la transacción completa. Una técnica como el protocolo de bloqueo en dos fases puede garantizar que todos los bloqueos se retengan hasta que la transacción principal confirme o aborte.

Otra técnica a considerar sería el uso de un mecanismo de timestamp o marcas de tiempo, que permita controlar el acceso concurrente y manejar los conflictos de