**Conceptos del Manejo de transacciones y transacciones anidadas**

¿Una pregunta que debemos hacernos es cómo se gestionan los datos en las bases de datos? Existen diferentes puntos que desembocan en una buena gestión de las mismas y las transacciones en base de datos son una parte fundamental de este proceso.

Una transacción en base de datos es una secuencia de operaciones que se realizan de manera indivisible, operaciones agrupadas como una unidad. Es como un paquete que contiene múltiples acciones que deben completarse en su totalidad o deshacerse por completo. Las operaciones que contiene una transacción se van almacenando temporalmente, no a nivel de disco. Es hasta que termina la transacción que se tienen efecto de manera permanente o no. Esto asegura que los datos se mantengan consistentes y evita problemas en caso de errores o fallos en el sistema.

**Condiciones de terminación**

Una transacción aplica a datos recuperables, puede estar formada por operaciones simples o compuestas y su intención es que sea atómica.

Diagrama

Descripción generada automáticamente con confianza media

Una transacción siempre termina, aun en la presencia de fallas. Si una transacción termina de manera exitosa se dice que la transacción hace un commit (consumación).

Si la transacción se detiene sin terminar su tarea, se dice que la transacción aborta.

Cuando la transacción es abortada, su ejecución se detiene y todas las acciones ejecutadas hasta el momento se deshacen (undone) regresando a la base de datos al estado antes de su ejecución. A esta operación también se le conoce como rollback

**Características de una transacción**

Una transacción en base de datos debe cumplir con las siguientes características, conocidas como las propiedades ACID:

● **Atomicidad:** Una transacción se considera atómica, lo que significa que se ejecuta como una unidad completa o no se ejecuta en absoluto. Si alguna de las operaciones falla, se deshacen todas las operaciones anteriores, asegurando que los datos no queden en un estado inconsistente.

● **Consistencia:** Una transacción debe llevar la base de datos desde un estado válido a otro estado válido. Esto garantiza que se respeten las restricciones y reglas definidas para los datos.

**● Aislamiento:** Cada transacción se ejecuta de manera aislada y no se ve afectada por otras transacciones concurrentes. Esto evita problemas de concurrencia y garantiza la integridad de los datos.

**● Durabilidad:** Una vez que una transacción se ha completado correctamente, los cambios realizados en la base de datos se mantienen permanentemente, incluso en caso de fallos del sistema. Los datos actualizados son duraderos y no se perderán.

Un ejemplo que podemos analizar es, imaginar que estás realizando una compra en línea. Cuando seleccionas los productos y procedes al pago, se lleva a cabo una transacción en la base de datos. En este caso, la transacción incluye operaciones como reducir el inventario de los productos comprados, registrar la transacción en la cuenta del cliente y generar un recibo de compra. Si algo falla durante este proceso, todas las operaciones se deshacen para evitar inconsistencias.

**Importancia de las transacciones en bases de datos**

Las transacciones son vitales en entornos donde múltiples usuarios pueden acceder y modificar la misma información al mismo tiempo. Garantizan la integridad y consistencia de los datos, evitando conflictos y asegurando que los cambios se realicen de manera segura.

**Casos de uso de las transacciones**

Las transacciones son utilizadas en una amplia gama de aplicaciones y sistemas, como:

● **Sistemas bancarios:** Cuando realizas una transferencia de dinero en línea, se utiliza una transacción para asegurar que el dinero se deduzca de una cuenta y se acredite en otra de manera precisa.

● **Sistemas de reserva**: Al reservar un vuelo o una habitación de hotel, se utiliza una transacción para garantizar que la disponibilidad se actualice correctamente y que no se realicen reservas duplicadas.

● **Sistemas de gestión de inventarios:** Cuando se registra una venta o se actualiza el inventario de productos, una transacción asegura que los datos se actualicen de manera coherente.

**Primitivas para el manejo de transacciones**

● **BEGIN TRAN:** comienza una transacción y aumenta en 1 @@TRANCOUNT.

● **COMMIT TRAN:** reduce en 1 @@TRANCOUNT y si @@TRANCOUNT llega a 0 guarda la transacción.

● **ROLLBACK TRAN**: deshace la transacción actual, o si estamos en transacciones anidadas deshace la más externa y todas las internas. Además pone @@TRANCOUNT a 0.

● **SAVE TRAN:** guarda un punto (con nombre) al que podemos volver con un ROLLBACK TRAN si es que estamos en transacciones anidadas y no queremos deshacer todo hasta la más externa.

● **ABORT\_TRANSACTION**: deshacer operación.

**Tipos de Transacciones Clasificación de acuerdo a su estructura**

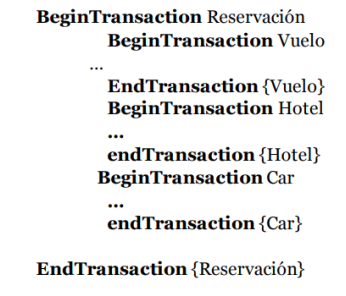
**Transacciones planas:** Estas transacciones tienen un punto de partida simple

(Begin\_transaction) y un punto simple de terminación (End\_transaction).

Texto, Carta

Descripción generada automáticamente

**Transacciones anidadas:** las operaciones de una transacción anidada pueden incluir otras transacciones.



Una transacción anidada dentro de otra transacción conserva las mismas propiedades que la de sus padres, esto implica, que puede contener así mismo transacciones dentro de ella.

Diagrama

Descripción generada automáticamente

**Existen restricciones para una transacción anidada:**

● Debe empezar después que su padre y debe terminar antes que él.

● El commit de una transacción padre está condicionada al commit de sus transacciones hijas.

● Si alguna transacción hija aborta (rollback), la transacción padre también será abortada (rollback).

**Estados de una transacción**

● Transacción Activa: se encuentra en este estado justo después de iniciar su ejecución.

● Transacción Parcialmente Confirmada: en este punto, se han realizado las operaciones de la transacción pero no han sido almacenados de manera permanente.

● Transacción Confirmada: Ha concluido su ejecución con éxito y se almacenan de manera permanente.

● Transacción Fallida: En este caso, es posible que la transacción deba ser cancelada.

● Transacción Terminada: indica que la transacción ha abandonado el sistema.

Diagrama

Descripción generada automáticamente

En conclusión, al abordar el tema de transacciones en bases de datos, se destaca la importancia crítica de estas operaciones para garantizar la integridad y consistencia de los datos. Como estudiante de licenciatura en sistemas, este conocimiento proporciona una base sólida para comprender cómo las transacciones, al adherirse a las propiedades ACID, aseguran que las operaciones en la base de datos se ejecuten de manera segura y confiable.

La comprensión de las primitivas de manejo de transacciones, como BEGIN TRAN, COMMIT TRAN, y ROLLBACK TRAN, junto con la identificación de casos de uso prácticos, como sistemas bancarios o reservas en línea, demuestra cómo estos conceptos se aplican en entornos del mundo real.

Es crucial reconocer la necesidad de cautela al trabajar con transacciones anidadas, entendiendo las restricciones asociadas, y tener presente la clasificación según su estructura, ya sea transacciones planas o anidadas.

**TRIGGERS**

Los triggers consisten en una serie de reglas predefinidas que se asocian a una tabla. Estas reglas se aplican a la base de datos cuando se realizan determinadas operaciones en la tabla, por ejemplo, al añadir, actualizar o eliminar registros. Dicho de otra manera, el trigger desencadena determinadas acciones de forma automática en las tablas de la base de datos cuando se insertan, modifican y se añaden nuevos datos. Generalmente es muy utilizado para auditorias en empresas

Por otra parte, entre sus principales ventajas es que todas estas funciones se pueden realizar desde la propia base de datos, es decir, no es necesario recurrir a lenguajes externos de programación.

En el manejo de bases de datos es muy frecuente realizar operaciones como

insertar, modificar o borrar registros de tablas en nuestra base de datos y de

forma involuntaria cometer errores. Por ello, se precisa el uso de

procedimientos de control del manejo de estos datos a través de triggers o

disparadores.

Los triggers mejoran la administración de la Base de datos, sin necesidad de contar con que

el usuario ejecute la sentencia de SQL, es decir de forma automatizada,

ayudando a, maximizar el tiempo de ejecución de las acciones, aumentar la

seguridad e integridad de la información.

II. OBJETIVOS ESPECÍFICOS

Utilizaremos en este proyecto triggers (disparadores) de auditorías por cada

insert, update y delete sobre la tabla conserje, que a su vez de forma

automática registrara esos movimientos en una tabla auxiliar llamada

auditoria\_conserje, incorporando la fecha y hora en que se registro la acción,

usuario de la base de datos que realizó la operación y la acción que realizo.