## 

línea horizontal



Actividad 2 UT 14-Transacciones

Base de Datos

Lunes 22 de Abril de 2024

**─**

Hirahi Mejías Delgado

1 DAM-A

EJERCICIOS A REALIZAR

**EJERCICIO 1: Crear las siguientes tablas**

**CREATE TABLE #T1 (ID INT, NOMBRE VARCHAR(50)), de forma que el campo ID sea clave primaria y autoincremento.**

**CREATE TABLE #T2 (ID INT, FECHA DATETIME), campo clave ID, el campo fecha no permite valores Nulos**

**CREATE TABLE #T3 (ID INT, STOCK INT), campo clave ID, el campo stock tiene que tomar valores mayores o igual a 0. 1. Diseñar las siguientes transacciones.**

**Crear Tablas:**

CREATE TABLE #T1 (

ID INT PRIMARY KEY IDENTITY(1,1),

NOMBRE VARCHAR(50)

);

CREATE TABLE #T2 (

ID INT PRIMARY KEY,

FECHA DATETIME NOT NULL

);

CREATE TABLE #T3 (

ID INT PRIMARY KEY,

STOCK INT CHECK (STOCK >= 0)

);

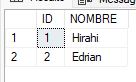
BEGIN TRANSACTION;

**Insertar 2 registros en #T1, mostrar el contenido de la tabla.**

INSERT INTO #T1 (NOMBRE) VALUES ('Hirahi');

INSERT INTO #T1 (NOMBRE) VALUES ('Edrian');

SELECT \* FROM #T1;

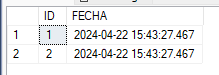


**Insertar 2 registros en #T2, mostrar el contenido de la tabla.**

INSERT INTO #T2 (ID, FECHA) VALUES (1, GETDATE());

INSERT INTO #T2 (ID, FECHA) VALUES (2, GETDATE());

SELECT \* FROM #T2;

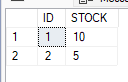


**Insertar 2 registros en #T3, mostrar el contenido de la tabla.**

INSERT INTO #T3 (ID, STOCK) VALUES (1, 10);

INSERT INTO #T3 (ID, STOCK) VALUES (2, 5);

SELECT \* FROM #T3;

****

**Si todos los datos se han insertado correctamente. Terminar la transacción, mostrando el contenido de las tablas. Comprobar que en el caso que se produzca un error a la hora de insertar datos, por ejemplo al intentar insertar en #T3 valores con stock negativo, se deshacen todos los valores insertados anteriormente.**

BEGIN TRY

INSERT INTO #T3 (ID, STOCK) VALUES (3, -5); -- Este insert generará un error

END TRY

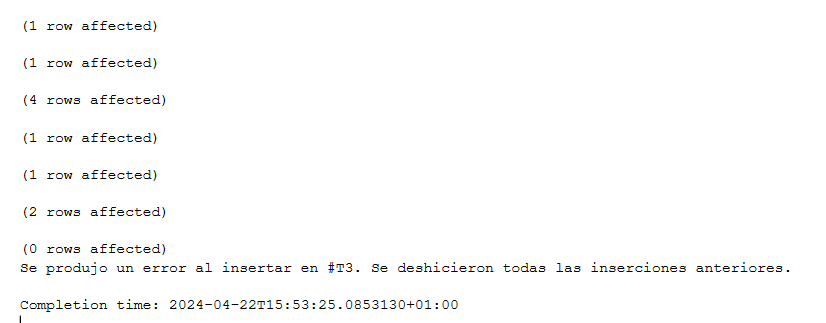
BEGIN CATCH

-- En caso de error, deshacer la transacción

ROLLBACK TRANSACTION;

PRINT 'Se produjo un error al insertar en #T3. Se deshicieron todas las inserciones anteriores.';

END CATCH;

****

**2. Diseñar las siguientes transacciones:**

**Insertar 2 registros en #T1, mostrar el contenido de la tabla.**

**Insertar 2 registros en #T2, con valores nulos en el campo fecha.**

-- Declarar una variable contador

DECLARE @Contador INT;

SET @Contador = 0;

-- Inicio de la transacción

BEGIN TRANSACTION;

-- Insertar 2 registros en #T1, mostrar el contenido de la tabla.

INSERT INTO #T1 (NOMBRE) VALUES ('Juan');

INSERT INTO #T1 (NOMBRE) VALUES ('María');

SET @Contador += @@ROWCOUNT; -- Incrementar el contador por el número de filas afectadas

-- Insertar 2 registros en #T2, con valores nulos en el campo fecha.

BEGIN TRY

INSERT INTO #T2 (ID, FECHA) VALUES (3, NULL);

INSERT INTO #T2 (ID, FECHA) VALUES (4, NULL);

SET @Contador += @@ROWCOUNT; -- Incrementar el contador por el número de filas afectadas

END TRY

BEGIN CATCH

-- En caso de error, deshacer la transacción

ROLLBACK TRANSACTION;

PRINT 'Se produjo un error al insertar en #T2. Se deshicieron los registros insertados en #T2.';

END CATCH;

-- Si se han insertado registros en #T2, deshacerlos pero mantener los de #T1

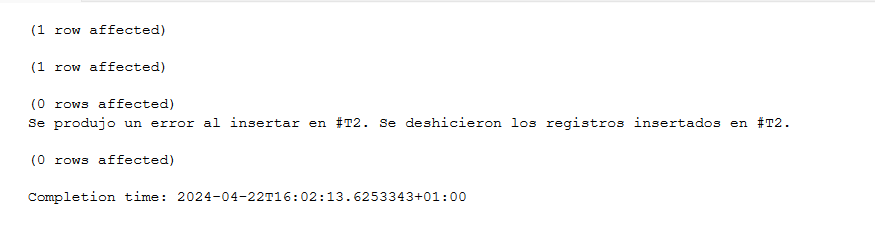
IF @Contador > 0

BEGIN

DELETE FROM #T2 WHERE ID IN (3, 4);

END

**Al detectar que la transacción no se ha podido realizar, deshacer los registros insertados en #T2 , pero manteniendo los insertados en #T1. Terminar la transacción , mostrando el contenido de las tablas. Probar insertando valores correctos en las dos tablas, para comprobar si se mantiene las dos operaciones de insertar. Probar insertando valores incorrectos en las dos tablas, y finalmente insertando valores correctos en la primera tabla e incorrectos en la segunda tabla. Para controlar hasta donde se ha podido realizar las operaciones de insertar, se puede utilizar una variable contador que vaya indicando las operaciones que se han podido realizar a la hora de insertar.**

****

**EJERCICIO 2:**

**El monitor de actividad en SQL Server Management Studio muestra información acerca de los procesos de SQL Server y el modo en que estos afectan a la instancia actual de SQL Server.Elabora un manual de uso de dicho monitor explicando su funcionamiento, la información que presenta y los diferentes apartados que ofrece al usuario.**

****

**El Monitor de Actividad en SQL Server Management Studio (SSMS) es una herramienta poderosa que proporciona información detallada sobre los procesos en ejecución en una instancia de SQL Server. Permite monitorear el rendimiento, identificar bloqueos y analizar la actividad del servidor en tiempo real. A continuación, se presenta un manual de uso que cubre su funcionamiento básico y los diferentes apartados que ofrece al usuario.**

**El Monitor de Actividad se puede abrir desde SSMS siguiendo estos pasos:**

**Abre SQL Server Management Studio.**

Conéctate a la instancia de SQL Server que deseas monitorear.

En el menú, selecciona "Ver" -> "Otras Ventanas" -> "Monitor de Actividad".

Información Presentada

**El Monitor de Actividad ofrece información detallada sobre los siguientes aspectos:**

**Procesos en Ejecución:** Lista de todos los procesos que se están ejecutando en la instancia de SQL Server, incluidos los comandos SQL, el estado del proceso y la duración.

**Bloqueos:** Información sobre los bloqueos actuales en la base de datos, como el tipo de bloqueo, la tabla afectada y las sesiones involucradas.

**Recursos de Servidor:** Estadísticas en tiempo real sobre el uso de CPU, memoria, E/S y otros recursos del servidor.

**Información de Sesión:** Detalles sobre las sesiones activas, incluida la identificación de la sesión, el estado y la última instrucción ejecutada.

**Apartados del Monitor de Actividad**

**Procesos:** Muestra una lista de todos los procesos activos en la instancia de SQL Server, incluidos los comandos SQL en ejecución, el estado del proceso y la duración.

**Bloqueos Actuales:** Presenta información sobre los bloqueos actuales en la base de datos, como el tipo de bloqueo, la tabla afectada y las sesiones involucradas.

**Recursos de Servidor:** Ofrece estadísticas en tiempo real sobre el uso de CPU, memoria, E/S y otros recursos del servidor.

**Información de Sesión:** Detalles sobre las sesiones activas, incluida la identificación de la sesión, el estado y la última instrucción ejecutada.

**Historial de Bloqueos:** Registra el historial de bloqueos en la instancia de SQL Server para un análisis retrospectivo.

**Uso Práctico**

* **Utiliza el Monitor de Actividad para identificar consultas lentas o bloqueos en tiempo real.**
* **Analiza las estadísticas de recursos del servidor para detectar cuellos de botella de rendimiento.**
* **Investiga el historial de bloqueos para comprender los patrones de bloqueo y optimizar el diseño de la base de datos.**

**EJERCICIO 3: La instrucción sp\_lock genera información acerca de los bloqueos. Documenta los diferentes argumentos que puede recibir dicha instrucción, y adjunto ejemplos de diferentes sentencias SQL donde se haga uso de sp\_lock (probando diferentes variaciones)**

La instrucción sp\_lock en SQL Server genera información sobre los bloqueos actuales en la base de datos. Documentamos los diferentes argumentos que puede recibir y proporcionamos ejemplos de uso.

**Argumentos de sp\_lock**

**La instrucción sp\_lock puede recibir los siguientes argumentos:**

-No se requieren argumentos: Simplemente ejecutar sp\_lock sin argumentos mostrará todos los bloqueos actuales en la base de datos.

-@spid: Este argumento permite filtrar los resultados por el ID de sesión (SPID). Solo mostrará los bloqueos asociados con la sesión especificada.

**Ejemplos de Uso**

**Mostrar todos los bloqueos actuales en la base de datos:**

EXEC sp\_lock;

**Mostrar los bloqueos asociados con una sesión específica (SPID 55):**

EXEC sp\_lock @spid = 55;

**Mostrar los bloqueos asociados con la sesión actual:**

DECLARE @current\_spid INT;

SET @current\_spid = @@SPID;

EXEC sp\_lock @spid = @current\_spid;

**Mostrar los bloqueos para una sesión específica y con un tipo específico de bloqueo (por ejemplo, bloqueos compartidos):**

EXEC sp\_lock @spid = 55, @locktype = 'SHARED';

**EJERCICIO 4:**

**Holdlock (HOLDLOCK):**

El hint HOLDLOCK se utiliza en una instrucción SELECT para mantener los bloqueos hasta que se complete la transacción.

Este hint se usa para evitar que otros procesos actualicen o inserten filas afectadas por la consulta actual.

Los bloqueos HOLDLOCK se liberan cuando finaliza la transacción en la que se utilizan.

**Updlock (UPDLOCK):**

El hint UPDLOCK se utiliza en una instrucción SELECT para solicitar bloqueos de actualización en las filas seleccionadas.

Estos bloqueos previenen que otras transacciones obtengan bloqueos compartidos en las mismas filas, lo que evita que ocurran conflictos de concurrencia cuando se intenta actualizar las filas seleccionadas posteriormente.

**Rowlock (ROWLOCK):**

El hint ROWLOCK se utiliza para indicar que una transacción debe adquirir bloqueos a nivel de fila en lugar de bloqueos a nivel de página o tabla.

Cuando se usa ROWLOCK, SQL Server adquiere bloqueos a nivel de fila para las filas afectadas por la transacción, lo que puede reducir la contención de bloqueos y mejorar la escalabilidad en entornos con alta concurrencia.

**Nolock (NOLOCK):**

El hint NOLOCK se utiliza en una instrucción SELECT para indicar que se deben ignorar los bloqueos existentes en las filas seleccionadas.

Esto permite que la consulta lea datos sin esperar a que se liberen los bloqueos, lo que puede mejorar el rendimiento de la consulta.

Sin embargo, el uso de NOLOCK puede resultar en lecturas sucias, lo que significa que la consulta podría leer datos que están en proceso de ser modificados por otra transacción.