
Universidad
Tecmilenio®

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Procedimientos almacenados

Semana 4

Universidad Tecmilenio

Introducción

Cuando un proceso nuevo se desarrolla, este tiende a crecer y se necesitan más líneas de código. Lo conveniente es mantener un código estructurado y organizado, por tanto, tener instrucciones dentro de un procedimiento almacenado ayudará a implementar lógica de manera más eficiente y con la posibilidad de reducir errores.

Proyectar inventarios basados en ventas diarias, obtener horas trabajadas por línea de producción o calcular gastos por departamento son ejemplos de tareas específicas que las empresas requieren para su beneficio y crecimiento.

Explicación

Procedimientos almacenados

De acuerdo con Microsoft (2022a), los procedimientos almacenados son un grupo de una o más instrucciones Transact-SQL o una referencia a un método de *Common Runtime Language* (CLR) de Microsoft .NET Framework.

- La ejecución de un procedimiento almacenado sucede en el servidor.
- Por ello se deberá considerar mantener un código que optimice el rendimiento de los recursos.
- Si la lógica se mantiene en un objeto de este tipo, será sencillo agregar, eliminar o modificar código.

Universidad Tecmilenio

Explicación

Creación de un procedimiento almacenado

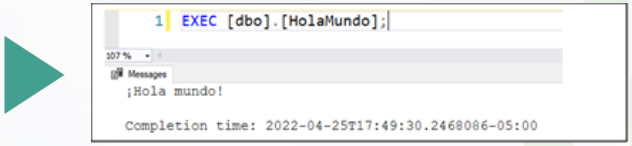
```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Ejemplo de un "Hola mundo" en un procedimiento almacenado

- Se iniciará con el conocido “¡Hola mundo!” para que el *script* anterior se pueda interpretar mejor. Por ahora, se definirá el procedimiento almacenado sin parámetros (W3Schools, s.f.).

```
CREATE PROCEDURE HolaMundo
AS
BEGIN
        PRINT '¡Hola mundo!'
END
GO
```

- Para la ejecución del procedimiento almacenado como nombre “HolaMundo”, será suficiente con hacer uso del comando **EXEC**, nombre del objeto y presionando el botón EXECUTE.



Explicación

Uso de parámetros de entrada en los procedimientos almacenados

Un procedimiento es encapsular instrucciones que ya conoces. Por otro lado, si se ejecuta cinco veces “HolaMundo”, el resultado siempre será el mismo.

El nuevo reto será imprimir “¡Hola mundo!” y el nombre de una persona, el cual cambiará debido a quien ejecuta el procedimiento almacenado.

```
ALTER PROCEDURE [dbo].[HolaMundo]
        @nombre AS VARCHAR(100)
AS
BEGIN
        PRINT '¡Hola mundo!'
        PRINT @nombre
END
GO
```

```
1  EXEC [dbo].[HolaMundo] @nombre = 'Axel Romero';
107 %
Messages
¡Hola mundo!
Axel Romero

Completion time: 2022-04-25T18:14:16.0383947-05:00
```

Universidad Tecmilenio

Explicación

Uso de parámetros de entrada en los procedimientos almacenados

Se definirán dos parámetros: edad y país de nacimiento, así se incrementará la habilidad de recepción de información en un procedimiento almacenado.

```
ALTER PROCEDURE [dbo].[HolaMundo]
	@nombre		AS	VARCHAR(100)
	,@edad		AS	INT
	,@pais		AS	VARCHAR(50) = NULL
AS
BEGIN
	PRINT '¡Hola mundo!'
	PRINT 'Tu nombre es: ' + @nombre
	PRINT 'Tienes: ' + CAST(@edad AS VARCHAR(3)) + ' de edad'
	PRINT 'Tu país de nacimiento es: ' + @pais
END
GO
```

"Cuando se ejecuta una función o un procedimiento almacenado, los parámetros de entrada pueden tener establecido su valor a una constante o usar el valor de una variable" (Microsoft, 2022b). Al asignar NULL, se está indicando que el parámetro puede ser opcional, no importando si se manda el parámetro.

```
1  EXEC [dbo].[HolaMundo] @nombre = 'Axel Romero', @edad = 32;
Messages
¡Hola mundo!
Tu nombre es: Axel Romero
Tienes: 32 de edad

Completion time: 2022-04-25T18:34:00.2602644-05:00
```

Universidad Tecmilenio

Explicación

Uso de parámetros de salida en los procedimientos almacenados

- El parámetro de salida “Estudiante” servirá para retornar el número de registros que contiene la tabla.
- La estructura del parámetro no cambia.
- Se agregará la instrucción OUTPUT para indicar que es un parámetro de salida.
- Se agregará un SELECT para conocer los datos de los estudiantes.

```
ALTER PROCEDURE [dbo].[InsertarEstudiante]
	@nombre			AS	VARCHAR(100)
	,@edad			AS	INT
	,@pais			AS	INT
	,@numeroRegistros	AS	INT OUTPUT
AS
BEGIN

	IF ISNULL(@nombre, '') = '' OR ISNULL(@edad, '') = '' OR ISNULL(@pais, '') = ''
	BEGIN
		PRINT 'Nombre, edad y país son datos requeridos para el proceso.'
	END
	ELSE
	BEGIN

		-- Insertar información en tabla [Estudiante]
		INSERT INTO [dbo].[Estudiante]
		([EstudianteNombre], [EstudianteEdad], [PaisID])
		VALUES
		(@nombre, @edad, @pais)

		--Obtener el número de registros en la tabla [Estudiante]
		SELECT @numeroRegistros = COUNT([EstudianteID])
		FROM [dbo].[Estudiante]

		--Obtener datos de los estudiantes.
		SELECT [EstudianteID], [EstudianteNombre], [EstudianteEdad], [PaisID]
		FROM [dbo].[Estudiante]
	END
END
GO
```

```
DECLARE @conteo AS INT = 0;

EXEC [dbo].[InsertarEstudiante]
    @nombre = 'Alondra Ramos'
    ,@edad = 30
    ,@pais = 2
    ,@numeroRegistros = @conteo OUTPUT

SELECT @conteo AS 'Registros'
```

	EstudianteID	EstudianteNombre	EstudianteEdad	PaisID
1	1	Axel Romero	32	1
2	2	Emilio Pacheco	30	1
3	3	Fernanda Ruiz	31	1
4	4	Alondra Ramos	30	2

	Registros
1	4

Actividad

Con base en lo descrito en el tema, reflexiona sobre las siguientes preguntas:

01 ¿De qué manera ayuda un procedimiento almacenado al procesamiento de información?

02 Menciona un proceso que sea candidato a convertirse en un procedimiento almacenado, teniendo como contexto una industria de venta de aparatos electrónicos.

Universidad Tecmilenio

Cierre

La creación de procedimientos almacenados te ayudará a contener procesos complejos de manera ordenada. Conforme se crean y consumen estos objetos, se observarán las ventajas.

Bibliografía

• Microsoft. (2022a). *Procedimientos almacenados (motor de base de datos)*. Recuperado de https://docs.microsoft.com/es-es/sql/relational-databases/stored-procedures/stored-procedures-database-engine

• Microsoft. (2022b). *Parámetros*. Recuperado de https://docs.microsoft.com/es-es/sql/relational-databases/stored-procedures/parameters?view=sql-server-ver15

• W3schools. (s.f.). *SQL Stored Procedures for SQL Server*. Recuperado de https://www.w3schools.com/sql/sql_stored_procedures.asp

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Funciones con valores de tabla

Semana 4

Introducción

El uso de funciones con valores de tabla te permitirá realizar operaciones, condiciones y hasta recibir parámetros de entrada, es decir, se podrá generar un flujo de código básico o complejo según se requiera para que el resultado que la función arroje sea exclusivamente de tipo tabla.

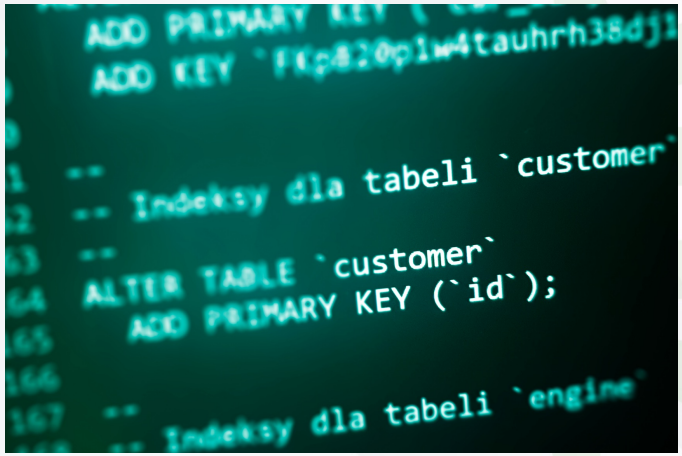
Las funciones con valores de tabla regresarán la información representada en columnas y filas, es decir, en una tabla, y por su naturaleza, podrán realizar INNER con otras tablas, pero tomando en cuenta que la función ya tiene una lógica encapsulada que evitará repetir código e implementando modificaciones a procesos en un tiempo corto.

Explicación

Funciones con valores de tabla

"Una función con valores de tabla es una función definida por el usuario que devuelve una tabla" (Microsoft, 2022).

Su ventaja es su fácil acceso para realizar modificaciones o implementar una nueva funcionalidad.



Creación de las funciones con valores de tabla

"La cláusula RETURN contiene una sola instrucción SELECT entre paréntesis. El resultado de SELECT es la tabla que se retorna. SELECT está sujeto a las mismas reglas que los SELECT de las vistas" (Moisset, 2022

```
CREATE FUNCTION ClientesActivosPorZona
(
	-- Add the parameters for the function here
	@zona AS INT
)
RETURNS TABLE
AS
RETURN
(
	-- Add the SELECT statement with parameter references here
	SELECT
		[c].[ClienteID]
		,[c].[ClienteNombre]
		,[z].[ZonaNombre]
	FROM
		[dbo].[Clientes] AS [c]
		INNER JOIN [dbo].[Zonas] AS [z]
			ON [c].[ZonaID] = [z].[ZonaID]
	WHERE
		[c].[ClienteStatus] = 1
		AND [z].[ZonaID] = @zona
)
GO
```

Uso de las funciones con valores de tabla

Un consumo básico de la función creada y el resultado que retorna será como se presenta en los siguientes ejemplos:

```
SELECT
    ClienteID
    ,ClienteNombre
    ,ZonaNombre
FROM
    [dbo].[ClientesActivosPorZona](1)
```

	ClienteID	ClienteNombre	ZonaNombre
1	10003	CEDIS CDMX	Centro
2	10005	Abarrotera del centro	Centro
3	10006	CEDIS Centro N1	Centro

```
SELECT
    [p].[PedidoID]
    ,[p].[ClienteID]
    ,[cpz].[ClienteNombre]
    ,[cpz].[ZonaNombre]
    ,[p].[PedidoFecha]
FROM
    [dbo].[Pedidos] AS [p]
    INNER JOIN [dbo].[ClientesActivosPorZona](1) AS [cpz]
        ON [cpz].[ClienteID] = [p].[ClienteID]
```

	PedidoID	ClienteID	ClienteNombre	ZonaNombre	PedidoFecha
1	4	10003	CEDIS CDMX	Centro	2022-04-25 ...
2	5	10005	Abarrotera del centro	Centro	2022-03-10 ...
3	7	10003	CEDIS CDMX	Centro	2022-04-28 ...
4	10	10006	CEDIS Centro N1	Centro	2022-02-12 ...
5	11	10006	CEDIS Centro N1	Centro	2022-03-22 ...

Uso de parámetros de entrada en funciones con valores de tabla

Ahora se creará nuevamente la función “ClientesActivosPorZona”, con la diferencia de que esta vez se define la tabla con las columnas y tipo de datos a retornar.

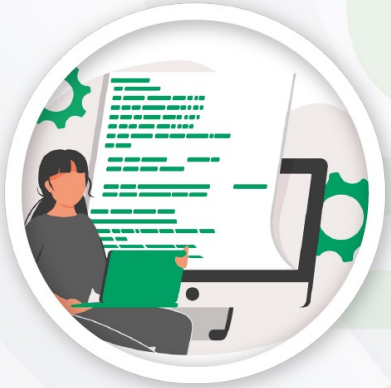
```
CREATE FUNCTION [dbo].[ClientesActivosPorZona]
(
	@zona AS INT = NULL
)
RETURNS @data TABLE
	(
		[Cliente]		INT		NULL
		,[Nombre]		VARCHAR(100)	NULL
		,[Zona]			VARCHAR(100)	NULL
	)
AS
BEGIN
	INSERT INTO @data
	SELECT
		[c].[ClienteID]
		,[c].[ClienteNombre]
		,[z].[ZonaNombre]
	FROM
		[dbo].[Clientes] AS [c]
		INNER JOIN [dbo].[Zonas] AS [z]
			ON [c].[ZonaID] = [z].[ZonaID]
	WHERE
		[c].[ClienteStatus] = 1
		AND [z].[ZonaID] = @zona

	RETURN
END
```

```
SELECT * FROM [dbo].[ClientesActivosPorZona](2)
```

Results | Messages

	Cliente	Nombre	Zona
1	10001	Abastecedora del sur	Sureste
2	10004	Abarrotes Campeche	Sureste
3	10007	CEDIS del sur F200	Sureste

Una **función con valores de tabla** te ayudará a concentrar las validaciones, la lógica y las relaciones entre tablas de un proceso para **regresar un resultado como tabla**, y se consumirá cuantas veces se requiera la información.



```
SELECT [Cliente], [Nombre], [Zona], [Transporte] FROM [dbo].[ClientesActivosPorZona](1)
UNION ALL
SELECT [Cliente], [Nombre], [Zona], [Transporte] FROM [dbo].[ClientesActivosPorZona](2)
UNION ALL
SELECT [Cliente], [Nombre], [Zona], [Transporte] FROM [dbo].[ClientesActivosPorZona](5)
```

	Cliente	Nombre	Zona	Transporte
1	10003	CEDIS CDMX	Centro	N1
2	10005	Abarrotera del centro	Centro	N1
3	10006	CEDIS Centro N1	Centro	N1
4	10001	Abastecedora del sur	Sureste	N1
5	10004	Abarrotes Campeche	Sureste	N1
6	10007	CEDIS del sur F200	Sureste	N1
7	10002	Comercializadora regiomontana	Norte	N2

Actividad

Con base en lo descrito en el tema, reflexiona sobre las siguientes preguntas:

01 ¿Cuál es una de las principales diferencias entre un procedimiento almacenado y una función con valores de tabla?

02 ¿Qué ventaja encuentras al usar funciones con valores de tabla y cómo se puede aplicar a un ejemplo real?

Cierre

Al explorar la aplicación de funciones con valores de tabla podemos concluir que es un objeto similar a una vista, pero con la capacidad de recibir parámetros, hacer uso de variables, condiciones e incluso crear un flujo de bloque de código que demanda dificultad.

Es significativo resaltar que, en algunas ocasiones, existen escenarios complejos, como una tarea automática que ejecuta un procedimiento almacenado que, a la vez, consume una función con valores de tabla que hace referencia a una vista y esta última a las tablas finales.

Bibliografía

• Microsoft. (2022). *Funciones con valores de tabla en CLR*. Recuperado de https://docs.microsoft.com/es-es/sql/relational-databases/clr-integration-database-objects-user-defined-functions/clr-table-valued-functions?view=sql-server-ver15

• Moisset, D. (2022). *137- Funciones con valores de tabla en línea*. Recuperado de https://www.tutorialesprogramacionya.com/sqlserverya/temarios/descripcion.php?cod=143&punto=137

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Control de errores

Semana 4

Universidad Tecmilenio

Introducción

T-SQL ofrece la capacidad de soportar la **gestión de errores y excepciones** durante la ejecución de bloques de código, que es una característica importante dentro del lenguaje, ya que ayudará a notificar que algo está fallando en el proceso y dependerá del programador obtener las alternativas para realizar los ajustes necesarios para que el segmento que estaba causando conflictos en el flujo del programa sea corregido.

Explicación

Introducción al control de errores

Un bloque de código puede estar funcionando a la perfección durante mucho tiempo, pero cuando se altera la funcionalidad de los procesos surgen nuevas circunstancias. Lo recomendable es que, si los procesos no cuentan con un **control de errores o excepciones**, este se implemente a la brevedad con el único fin de conservar la armonía en los procesos.

Universidad Tecmilenio

Explicación

Uso e implementación del control de excepciones en T-SQL

- Será de utilidad para prever problemas en el proceso durante la ejecución de un programa (Microsoft, 2022).

```
BEGIN TRY
     { sql_statement | statement_block }
END TRY
BEGIN CATCH
     [ { sql_statement | statement_block } ]
END CATCH
[ ; ]
```

- Las instrucciones a utilizar son BEGIN TRY... END TRY y BEGIN CATCH... END CATCH.

```
BEGIN TRY

      DECLARE @incremento  AS DECIMAL(18, 3) = .005

      UPDATE [dbo].[Productos]
      SET ProductoPrecio = ProductoPrecio + (ProductoPrecio * @incremento)

      PRINT 'Se ha incrementado el ' + CAST(@incremento AS VARCHAR(10)) + ' % a todos los productos.'

END TRY
BEGIN CATCH

      PRINT 'El proceso de actualización de precios ha fallado.'

END CATCH
```

Universidad Tecmilenio

Explicación

Se provocará un error intencionalmente con el fin de que T-SQL ejecute la sección BEGIN CATCH... END CATCH. Una de las operaciones matemáticas más comunes que se conoce que marcaría un error incluso en una calculadora sería dividir un número entre cero (porque es imposible dividir entre cero) (Artacho, 2019).

```
BEGIN TRY

	DECLARE @incremento		AS DECIMAL(18, 3) = .005
	DECLARE @division		AS DECIMAL(18, 2)

	SELECT @division = 100/0

	UPDATE [dbo].[Productos]
	SET ProductoPrecio = ProductoPrecio + (ProductoPrecio * @incremento)

	PRINT 'Se ha incrementado el ' + CAST(@incremento AS VARCHAR(10)) + ' % a todos los productos.'

END TRY
BEGIN CATCH

	PRINT 'El proceso de actualización de precios ha fallado.'

END CATCH
```

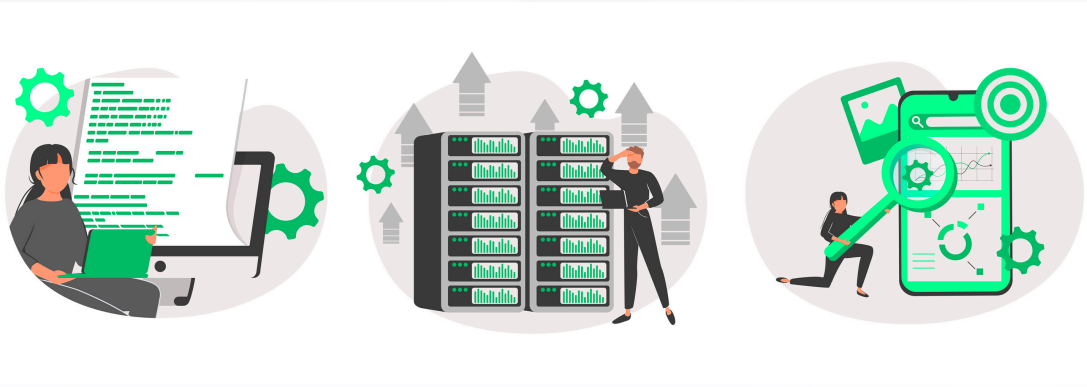
```
Messages
El proceso de actualización de precios ha fallado.

Completion time: 2022-05-09T12:10:16.8901948-05:00
```

Explicación

Uso e implementación del control de errores en T-SQL

Las siguientes funciones se deberán utilizar dentro de BEGIN CATCH... END CACTH:

- **ERROR_NUMBER()**: devuelve el número de error.
- **ERROR_MESSAGE()**: devuelve el texto completo del mensaje de error.
- **ERROR SEVERITY()**: devuelve la gravedad del error.
- **ERROR_STATE()**: devuelve el número de estado de error.
- **ERROR_LINE()**: devuelve el número de línea donde se produjo el error.
- **ERROR_PROCEDURE()**: devuelve el nombre del procedimiento donde se produjo el error.



Universidad Tecmilenio

Explicación

Ejemplo

Para que muestre un error en T-SQL y se pueda observar el **detalle de cada una de las funciones anteriores**, se realizará una suma entre un valor alfanumérico y numérico, lo cual no es posible, pues no se podría retornar un resultado al intentar sumar ABC + 5, por lo que marcará un error de conversión.

```
BEGIN TRY
    DECLARE @texto   AS VARCHAR(50)      = 'SQL Server'
           ,@numero AS INT               = 100

      SELECT @texto + @numero AS [Resultado]

      SELECT '... Aquí sigue el proceso.'
END TRY
BEGIN CATCH
    SELECT
    ERROR_NUMBER()          AS [Error_Numero],
    ERROR_SEVERITY()        AS [Error_Gravedad],
    ERROR_STATE()           AS [Error_Estado],
    ERROR_PROCEDURE()       AS [Error_Procedimiento],
    ERROR_LINE()            AS [Error_Linea],
    ERROR_MESSAGE()         AS [Error_Mensaje]
END CATCH
```

Resultado

	Error_Numero	Error_Gravedad	Error_Estado	Error_Procedimiento	Error_Linea	Error_Mensaje
1	245	16	1	NULL	5	Conversion failed when converting the varchar value 'SQL Server' to data type int.

Actividad

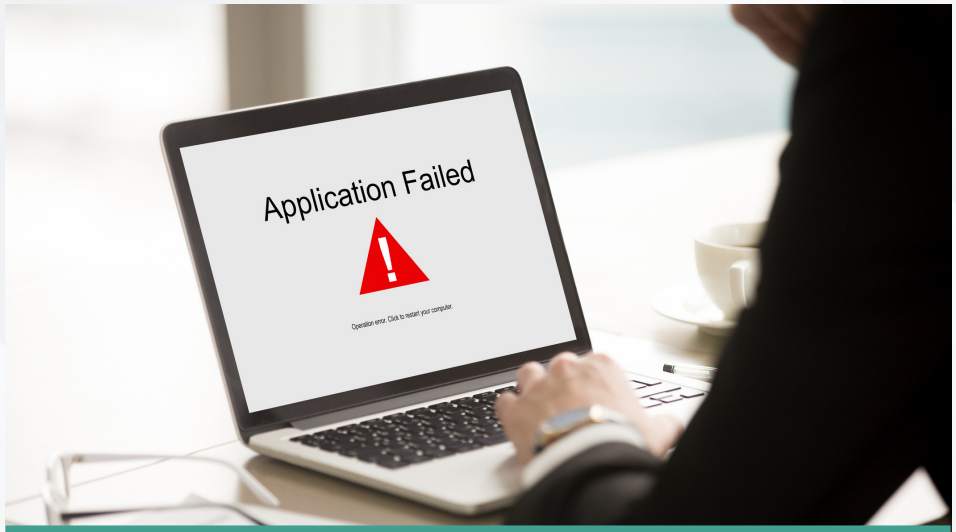
Con base en lo descrito en el tema, reflexiona sobre las siguientes preguntas:

01 ¿Debería ser obligatoria la implementación de errores y excepciones en una base de datos?

02 ¿De qué manera se pueden monitorear los procesos que han fallado?

Universidad Tecmilenio

Cierre



Todo proceso es susceptible a fallas, no importa si es pequeño, mediano o grande, lo recomendable será siempre tomar precauciones e incluso generar escenarios extremos para que el proceso quede lo más seguro posible.
Los beneficios que se van a obtener impactan positivamente en el tiempo que se invertirá en aplicar el manejo de errores y más cuando un proceso tiene un gran sentido de importancia para la empresa.

Bibliografía

• Artacho, A. (2019). *Dividir entre cero*... Recuperado de https://matematicascercanas.com/2019/06/09/dividir-entre-cero/

• Microsoft. (2022). *TRY...CATCH (Transact-SQL)*. Recuperado de https://docs.microsoft.com/en-us/sql/t-sql/language-elements/try-catch-transact-sql?view=sql-server-ver15

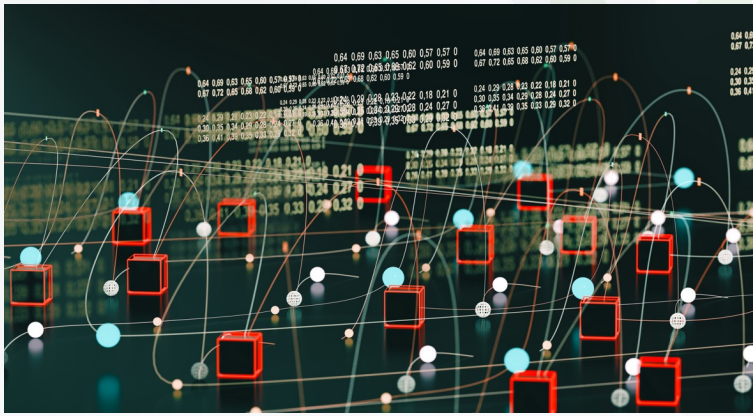
Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Transacciones

Semana 4

Se puede tener un proceso encapsulado en un objeto de tipo procedimiento almacenado o función, incluso todas las instrucciones dentro de una excepción, por si el proceso encuentra una falla en la ejecución, pero existen casos en donde falla únicamente una operación de 10, por lo tanto, el proceso quedará incompleto, situación que puede causar inconsistencia en los datos. Con el uso de transacciones en T-SQL se logrará mantener un bloque de código como uno solo, es decir, si falla una operación, no se aplica ningún cambio hasta que el error haya sido corregido.



Explicación

Descripción de las transacciones

Para Microsoft (2022), “una transacción es una sola unidad de trabajo. Si una transacción es exitosa, todas las modificaciones de datos realizadas durante la transacción se confirman y se convierten en parte permanente de la base de datos”.

Una transacción debe cumplir cuatro propiedades comúnmente conocidas como ACID, y las describe Lázaro (2018).

- **Atomicidad**
- **Consistencia**
- **Aislamiento**
- **Durabilidad**

Explicación

Creación y gestión de transacciones

La instrucción **BEGIN TRAN** indica el comienzo de una transacción. Si algo falla, revierte los cambios a sus valores iniciales con ayuda del comando **ROLLBACK TRAN**. En el escenario esperado, donde todo marcha bien y sin errores, se aplicará un **COMMIT TRAN** para completar la transacción con éxito y se apliquen los cambios en la base de datos (Erkec, 2021).

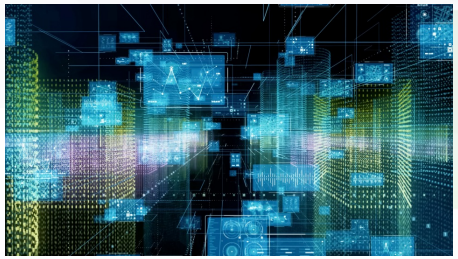
```
BEGIN TRANSACTION [ {transaction_name | @tran_name_variable }
    [WITH MARK ['description']]]
```

```
-- Iniciar la transacción.
BEGIN TRAN

-- Antes de actualizar precio.
SELECT [ProductoCodigo], [ProductoPrecio] FROM [dbo].[Productos] WHERE [ProductoCodigo] = 'CAM1'

-- Actualizar precio.
UPDATE [dbo].[Productos]
SET [ProductoPrecio] = 15000
WHERE [ProductoCodigo] = 'CAM1'

-- Después de actualizar precio.
SELECT [ProductoCodigo], [ProductoPrecio] FROM [dbo].[Productos] WHERE [ProductoCodigo] = 'CAM1'

-- Transacciones abiertas.
SELECT @@TRANCOUNT AS TransaccionesAbiertas
```

	ProductoCodigo	ProductoPrecio
1	CAM1	12800

	ProductoCodigo	ProductoPrecio
1	CAM1	15000

	TransaccionesAbiertas
1	1



Universidad Tecmilenio

Explicación

Operaciones con transacciones

BEGIN TRAN: iniciar la transacción.
COMMIT TRAN: aplicar cambios.
ROLLBACK TRAN: deshacer cambios.
SAVE POINT: revertir cambios de bloques determinados.

Actividad

Con base en lo descrito en el tema, reflexiona sobre las siguientes preguntas:

01 ¿Qué importancia tiene la implementación de transacciones en SQL Server?

02 Menciona un ejemplo de cómo aplicarías SAVEPOINT en un proceso crítico de una empresa.

Cierre

Utilizar las transacciones en T-SQL no representa mayor complejidad, incluso cuando se habla de una tarea relevante para la organización. Se puede concluir que una transacción es la colección de operaciones que se integran en una unidad lógica, la cual deberá tener un inicio y un fin. La ventaja más destacada será mantener la atomicidad, es decir, ejecutar todas las operaciones o descartarlas para mantener el equilibrio en la información y procesos.

Bibliografía

• Erkec, E. (2021). *Transactions in SQL Server for beginners*. Recuperado de https://www.sqlshack.com/transactions-in-sql-server-for-beginners/

• Lázaro, D. (2018). *Transacciones en SQL*. Recuperado de https://diego.com.es/transacciones-en-sql

• Microsoft. (2022). *Transactions (Transact-SQL).* Recuperado de https://docs.microsoft.com/en-us/sql/t-sql/language-elements/transactions-transact-sql?view=sql-server-ver15