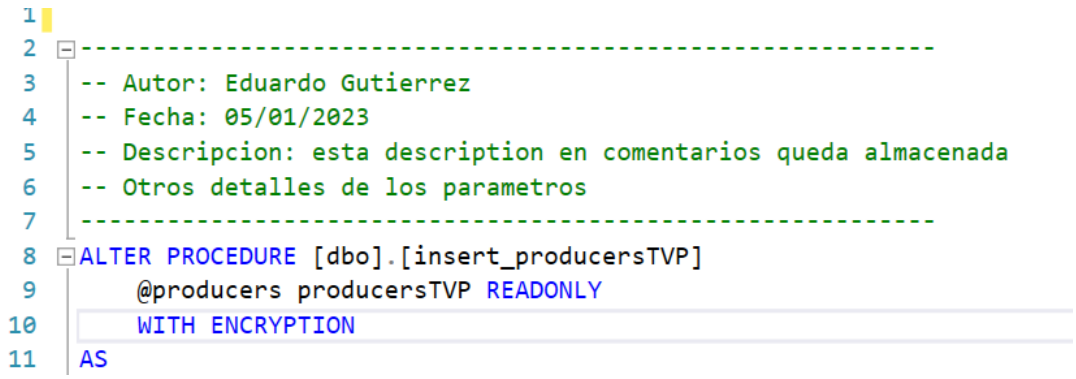
Instituto Tecnológico de Costa Rica
Prof. Rodrigo Núñez Núñez
Emanuel Rodríguez Oviedo c.2022108678
José Eduardo Gutierrez Conejo 2019073558

Caso 4 - Bases de datos I

Tabla de contenidos

Encriptación de SP

Para poder encriptar un script en SQLSEVER, es tan sencillo como ejecutar un query de ALTER PROCEDURE donde se agregue la línea WITH ENCRYPTION, lo cual se ve demostrado en la siguiente imágen:



Como se puede observar, el alter procedure repite todo el código, excepto por la línea 10. Para comprobar que efectivamente este método nos permite encriptar y proteger a nuestro script de los ataques de agentes externos podemos usar los siguientes dos métodos:

1. Ejecutar el siguiente query, en un archivo sin encriptar este mostraría el código que se ejecutaría al llamar al query, sin embargo, como está encriptado ahora mostrará “NULL”

```
63
64
65  SELECT O.name, M.definition, O.type_desc, O.type
66  FROM sys.sql_modules M
67  INNER JOIN sys.objects O ON M.object_id=O.object_id
68  WHERE O.type IN ('P');
69  GO
```

125 %

Results | Messages

	name	definition	type_desc	type
1	product_sale_SP	---------- -- Autor:...	SQL_STORED_PROCEDURE	P
2	sp_upgraddiagrams	CREATE PROCEDURE dbo.sp_upgraddiagrams AS ...	SQL_STORED_PROCEDURE	P
3	sp_helpdiagrams	CREATE PROCEDURE dbo.sp_helpdiagrams (@di...	SQL_STORED_PROCEDURE	P
4	sp_helpdiagramdefinition	CREATE PROCEDURE dbo.sp_helpdiagramdefinition ...	SQL_STORED_PROCEDURE	P
5	insert_producersTVP	NULL	SQL_STORED_PROCEDURE	P
6	sp_creatediagram	CREATE PROCEDURE dbo.sp_creatediagram (@d...	SQL_STORED_PROCEDURE	P
7	sp_renamediagram	CREATE PROCEDURE dbo.sp_renamediagram (...	SQL_STORED_PROCEDURE	P
8	sp_alterdiagram	CREATE PROCEDURE dbo.sp_alterdiagram (@dia...	SQL_STORED_PROCEDURE	P
9	sp_dropdiagram	CREATE PROCEDURE dbo.sp_dropdiagram (@dia...	SQL_STORED_PROCEDURE	P

Insert_producersTVP encriptado

	name	definition	type_desc	type
1	product_sale_SP	NULL	SQL_STORED_PROCEDURE	P
2	sp_upgraddiagrams	CREATE PROCEDURE dbo.sp_upgraddiagrams AS ...	SQL_STORED_PROCEDURE	P
3	sp_helpdiagrams	CREATE PROCEDURE dbo.sp_helpdiagrams (@di...	SQL_STORED_PROCEDURE	P
4	sp_helpdiagramdefinition	CREATE PROCEDURE dbo.sp_helpdiagramdefinition ...	SQL_STORED_PROCEDURE	P
5	insert_producersTVP	NULL	SQL_STORED_PROCEDURE	P
6	sp_creatediagram	CREATE PROCEDURE dbo.sp_creatediagram (@d...	SQL_STORED_PROCEDURE	P
7	sp_renamediagram	CREATE PROCEDURE dbo.sp_renamediagram (...	SQL_STORED_PROCEDURE	P
8	sp_alterdiagram	CREATE PROCEDURE dbo.sp_alterdiagram (@dia...	SQL_STORED_PROCEDURE	P
9	sp_dropdiagram	CREATE PROCEDURE dbo.sp_dropdiagram (@dia...	SQL_STORED_PROCEDURE	P

produtc_sale_SP encriptado

2. Podemos llamar al built-in stored procedure que nos ofrece SQL Server: rsp_HelpText usp_FetchSalesByYear, el cual cumple la misma función que el query anterior pero cambiando la manera en que nos muestra el código;

```
71
72  sp_HelpText [insert_producersTVP];
73
```

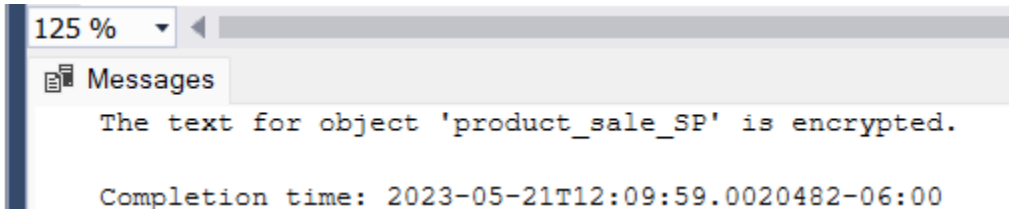
125 %

Messages

```
The text for object 'insert_producersTVP' is encrypted.

Completion time: 2023-05-21T12:00:22.5843733-06:00
```

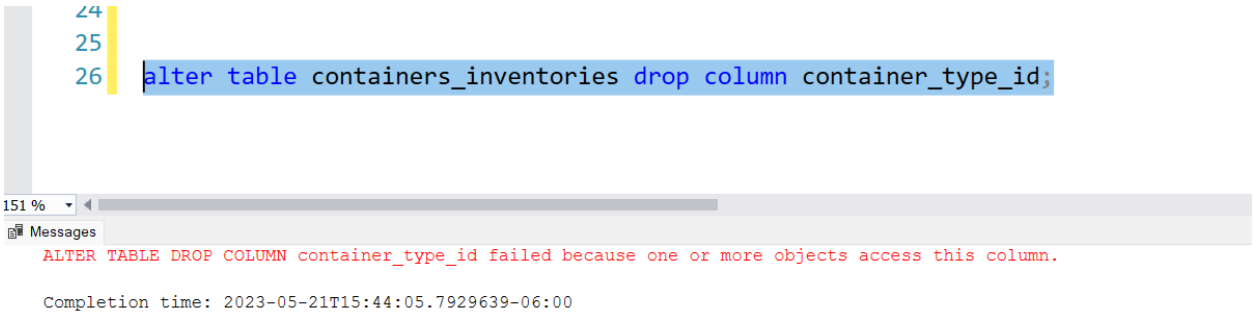
rsp_HelpText Insert_producersTVP retornando mensaje de encriptado



rsp_HelpText prodcut_sales_SP retornando mensaje de encriptado

Schemabinding

Para poder asegurar la integridad de la lógica de negocios de nuestra base de datos, podemos aplicar la “clause” WITH SCHEMABINDING en un ALTER VIEW y en cualquier función que hayamos definido sobre nuestra base de datos. Así, al tratar de alterar cualquier tabla que esté siendo usada en una de estas, de modo que dicha alteración pueda corromper la lógica que aplican, se dará un mensaje de advertencia y se abortará la ejecución de tal query.



Jobs

Job a)

1. Crea un nuevo job en SQL Server Agent. Para ello, abrimos SQL Server Management Studio, expandimos la carpeta "SQL Server Agent" en el Explorador de objetos y hacemos clic derecho en "Jobs". Luego, selecciona "New Job".

2. En la ventana de configuración del job, proporcionamos un nombre y una descripción para el job en la pestaña "General".

3. En la pestaña "Steps", creamos un nuevo paso para el job haciendo clic en "New...". Le damos un nombre descriptivo al paso.

4. En el campo "Command", escribimos el siguiente código T-SQL:

```
DECLARE @sql NVARCHAR(MAX);
SET @sql = '';
SELECT @sql = @sql + 'EXEC sp_recompile ''' +
QUOTENAME(SCHEMA_NAME(schema_id)) + '.' + QUOTENAME(name) + ''';' +
CHAR(13)
FROM sys.procedures;
EXEC sp_executesql @sql;
```

5. Se hace clic en "OK" para guardar el paso.

6. En la pestaña "Schedules", configuramos la frecuencia con la que ejecutar el job y hacemos clic en "OK" para guardar el job.

Job b)

1. Instalar sqlserver nuevamente desde el instalador en la ventana de “Custom”
2. Crear una nueva instancia de un servidor
3. En el linked server, nos vamos en el object explorer a server objects, linked servers, click derecho y new linked server.
4. En linked server, ponemos el nombre del server al que queremos conectarnos
5. Seleccionamos en el server type SQL Server
6. luego nos vamos en la ventana de la izquierda a Security, en la selección de abajo colocamos Be made using this security context, donde colocamos un usuario con suficientes permisos.
7. Le damos abajo al boton ok
8. Creamos un job como en el punto anterior, con dos pasos, dentro del linked server
9. En los pasos colocamos respectivamente el siguiente código:

```
-- Insert into event_types table
INSERT INTO [Esencial Verde Linked Copy].[dbo].[event_types] (event_type_name)
SELECT event_type_name
FROM [thisisnotapc].[Esencial Verde].[dbo].[event_types]
WHERE event_type_id NOT IN (SELECT event_type_id FROM [Esencial Verde Linked
Copy].[dbo].[event_types]);

-- Insert into levels table
INSERT INTO [Esencial Verde Linked Copy].[dbo].[levels] (level_name)
SELECT level_name
FROM [thisisnotapc].[Esencial Verde].[dbo].[levels]
WHERE level_id NOT IN (SELECT level_id FROM [Esencial Verde Linked Copy].[dbo].[levels]);

-- Insert into sources table
INSERT INTO [Esencial Verde Linked Copy].[dbo].[sources] (source_name)
SELECT source_name
FROM [thisisnotapc].[Esencial Verde].[dbo].[sources]
WHERE source_id NOT IN (SELECT source_id FROM [Esencial Verde Linked
Copy].[dbo].[sources]);

-- Insert into object_types table
INSERT INTO [Esencial Verde Linked Copy].[dbo].[object_types] (object_type_name)
SELECT object_type_name
FROM [thisisnotapc].[Esencial Verde].[dbo].[object_types]
WHERE object_type_id NOT IN (SELECT object_type_id FROM [Esencial Verde Linked
Copy].[dbo].[object_types]);

-- Insert into event_logs table
INSERT INTO [Esencial Verde Linked Copy].[dbo].[event_logs] (description, reference_id1,
reference_id2, value1, value2, event_type_id, object_type_id, level_id, source_id, created_at,
updated_at, computer, username, checksum)
SELECT description, reference_id1, reference_id2, value1, value2, event_type_id,
object_type_id, level_id, source_id, created_at, updated_at, computer, username, checksum
FROM [thisisnotapc].[Esencial Verde].[dbo].[event_logs]
WHERE event_log_id NOT IN (SELECT event_log_id FROM [Esencial Verde Linked
Copy].[dbo].[event_logs]);



DELETE FROM [THISISNOTAPC].[Esencial Verde].[dbo].[event_logs]
WHERE event_log_id NOT IN (
    SELECT MAX(event_log_id)
    FROM [THISISNOTAPC].[Esencial Verde].[dbo].[event_logs]
```

);

10. Creamos el schedule adecuado y guardamos el job con OK

Dirty Read Error

Este error ocurre cuando una transacción lee datos que aún no se han enviado (commit). Por ejemplo, suponga que hay una transacción 1 que actualiza una fila. Una transacción 2 lee la fila actualizada antes de que la transacción 1 confirme y envíe los datos de la actualización.
Ejecución:

1. Ejecutar los scripts de creación de los Stored procedures con el sufijo Original.
2. Abrir los archivos con el sufijo EXEC.
3. Se procede a ejecutar el get_producer_debt_SP para verificar la información original de la consulta.
4. Se procede a ejecutar el update_producer_SP cambiando la información del productor.
5. Como se puede observar la transacción va a durar alrededor de unos 10 segundos, esto porque tiene un waitfor delay para poder realizar la prueba ya que este problema de Dirty Read solo se da en ambientes de alta concurrencia y haciendo que este SP dure unos segundos extras se puede demostrar que el get_producer_SP lee los datos sin haberse realizado el commit.
6. Durante la ejecución del update_producer_SP se ejecuta el get_producer_debt_SP, y se obtienen los datos del update sin haberse realizado el commit.
7. Una vez que la ejecución del update falla se puede volver a ejecutar el get para verificar que esta vez se ve la consulta con la información sin cambios.

Solución:
El nivel de aislamiento de la transacción original es de READ UNCOMMITTED, lo que indica que se van a leer datos de transacciones que no han realizado un commit. Si se cambia el nivel de aislamiento a READ COMMITTED se puede evitar este problema ya que no se estaría leyendo esta información hasta que se realice el commit.

- Para demostrar esto se puede abrir el archivo con el sufijo READ COMMITTED el cual es un archivo que modifica el SP Original y le agrega un nivel de aislamiento de READ COMMITTED, el cual como se mencionó soluciona el error de Dirty Read.

Deadlock

Un deadlock es un tipo de error en bases de datos que puede ocurrir cuando dos o más transacciones se bloquean entre sí, es decir los recursos bloqueados de una transacción 1 son solicitados por una transacción 2 y a su vez la transacción 1 solicita recursos bloqueados por la transacción 2, por lo que ninguno de los dos logra terminar su ejecución y se decide por abortar alguna de las 2.

Ejecución:

1. Ejecutar los queries con sufijo Original ya que estos son los Stored Procedures que presentan el error
2. Como se puede observar los SPs presentan en los selects un WITH TABLOCKX lo que representan un lock exclusivo para la tabla, de esta forma ninguna otra transacción puede ver los datos de la tabla que se está seleccionando. Y como se puede observar esto se presenta en ambos SPs.
3. Ejecuta los SP en cualquier orden ya que el waitfor se encarga que simular la concurrencia, por medio de los archivos con sufijo EXEC.
4. Se va a poder observar que el SP que se ejecuta de segundo manda error de deadlock

Solución:
Al utilizar el TABLOCK se está bloqueando todas las lecturas y escrituras de una forma exclusiva en el SP, esto causa que cualquier otra SP no puede utilizarlo, una forma de solucionarlo es definir un orden de de prioridades para las solicitudes de la tabla, pero esto requiere mucho análisis sobre todos los SP, por lo que lo que se recomienda es trabajar con otro tipos de LOCKS no exclusivos para evitar este tipo de bloqueos. Por ejemplo si lo que se desea es evitar updates durante la ejecución del select se podría utilizar un UPDLOCK.

- Ejecuta los SPs con sufijo WITHOUT TABLOCKX para mostrar que con los cambios no ocurre el deadlock.

Lost Update

El error de Lost Update se produce cuando dos o más transacciones intentan actualizar los mismos datos simultáneamente y como resultado, la actualización realizada por una transacción se pierde o se sobrescribe con la actualización realizada por otra transacción.

Ejecución:

1. Ejecutar la creación del SP con el sufijo Original.
2. Seguidamente abra los archivos con el Sufijo EXEC para ejecutar las consultas.
3. Como se va a poder observar al realizar las consultas, las transacciones leen un mismo dato y luego realizan cambios sobre el datos y sobre escriben el resultado, a esto pasar de una forma recurrente los modificaciones de uno caen sobre el otro lo que causa una pérdida de información.

Solución:
La solución planteada es evitar sacar información antes de realizar el cambio ya que esto puede causar inconsistencia si se modifica la información, por lo tanto lo que se propone es que en vez de realizar cambios en una variable y luego asignarla es realizar los cálculos y la asignación dentro del update.

- Ejecute el SP con sufijo SELECT IN TRANSACT donde se realiza esta solución y corra los EXEC nuevamente, se va a poder observar que no se pierden datos.

Phantom

El error de Phantom se produce cuando una transacción ejecuta una consulta basada en un conjunto de filas y, posteriormente, otra transacción realiza una operación de inserción o eliminación que afecta a las filas del conjunto original de la primera transacción. Como resultado, la primera transacción puede observar un conjunto de filas "fantasma" que no existían cuando se ejecutó la consulta inicial.

Ejecución:

1. Ejecuta los SPs con el sufijo Original y el tipo para el TVP.
2. Ejecute el archivo con sufijo EXEC para que tenga información para la consulta a realizar.
3. Abra los archivos con sufijo EXEC FOR TRIAL para realizar la demostración.
4. El SP en primera instancia consulta la cantidad de productores a los cuales repartir las deducciones de puntos, después de esto realiza un update sobre los productores. En el intervalo de tiempo entre la consulta y el update se inserta un nuevo campo que cumple con las condiciones y entonces a la hora de realizar el update se realiza sobre un campo extra.

Solución:
La solución planteada para evitar esto es utilizar un nivel de aislamiento de repeatable read ya que esta evita que otras transacciones realicen cambios sobre las tablas que se están consultando.

- Ejecute el SP con sufijo REPEATABLE READ para verificar la solución,
- Para las verificaciones se recomienda utilizar el archivo de EXTRA QUERIES para poder visualizar los cambios que realiza el SP.

Glosario

Schemabinding:

En SQL, SCHEMABINDING (enlace de esquema) es una propiedad que se puede aplicar a objetos de base de datos, como vistas y funciones, para vincularlos al esquema subyacente de

las tablas a las que hacen referencia. Cuando se aplica el enlace de esquema, los objetos dependientes se vuelven "vinculados" al esquema y cualquier cambio en las tablas subyacentes requerirá que se actualicen o se eliminen y se vuelvan a crear los objetos vinculados.

Esta propiedad de SCHEMABINDING proporciona varios beneficios. En primer lugar, garantiza la integridad de los objetos vinculados al asegurar que las estructuras de las tablas subyacentes no se alteren inadvertidamente. Esto evita que se realicen modificaciones en las tablas que puedan afectar negativamente a las vistas o funciones dependientes.

En segundo lugar, SCHEMABINDING mejora el rendimiento al permitir que el optimizador de consultas del motor de base de datos realice mejoras en la ejecución de las consultas. Al estar vinculados al esquema subyacente, los objetos pueden beneficiarse de la eliminación de redundancias y la optimización de consultas, lo que resulta en consultas más rápidas y eficientes.

Job:

En SQL, un "job" es una tarea programada o un proceso automatizado que se ejecuta en un momento específico o a intervalos regulares. Los jobs son utilizados para realizar actividades recurrentes, como realizar copias de seguridad de bases de datos, generar informes, realizar mantenimiento programado, entre otros.

Un job consta de una serie de instrucciones o comandos SQL que se ejecutan de forma secuencial. Estos comandos pueden incluir consultas, actualizaciones, inserciones, eliminaciones u otras acciones que se deseen realizar en la base de datos.

Además de las instrucciones SQL, un job también puede incluir instrucciones adicionales para controlar su programación y ejecución. Estas instrucciones pueden especificar la frecuencia de ejecución (por ejemplo, diaria, semanal, mensual), la hora de inicio, la duración máxima permitida, entre otros parámetros.

Linked Server

Un "Linked Server" (servidor vinculado) en SQL Server es una configuración que permite a un servidor de base de datos acceder y trabajar con datos de otro servidor de base de datos, incluso si los servidores se encuentran en entornos diferentes o utilizan sistemas de gestión de bases de datos distintos.

Un Linked Server se establece mediante la configuración de una conexión y una definición de nombre lógico en el servidor principal para el servidor remoto al que se desea acceder. Esta

configuración permite realizar consultas, ejecutar procedimientos almacenados y realizar operaciones de escritura y lectura en el servidor remoto como si estuviera siendo accedido localmente.