Procedimientos-Funciones Almacenadas

¿Que son los procedimientos y funciones almacenadas?

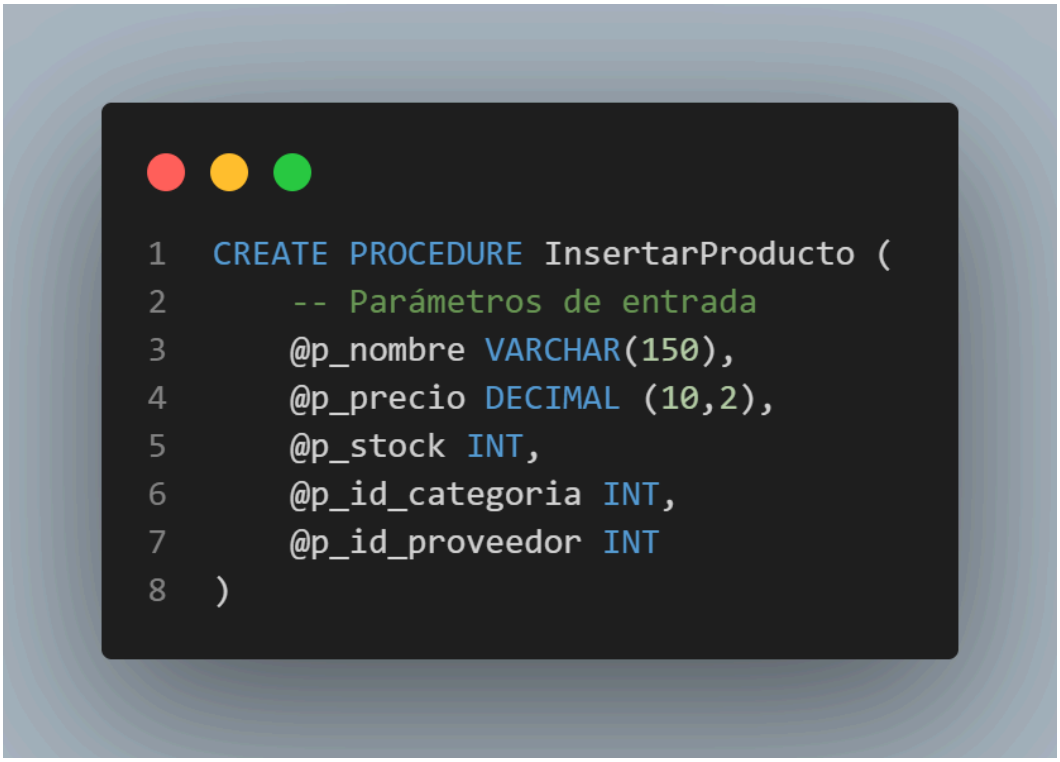
Son bloques de códigos escritos una sola vez y guardados en la base de datos, permitiendo reutilizarlos cuando se requiera realizar una tarea especifica donde el bloque de código puede devolver un valor o no.

Facilitando la reutilización, mantenibilidad, legibilidad y rendimiento en las consultas de la base de dato en la que se trabaja.

Procedimientos Almacenados

Un procedimiento almacenado de SQL Server es un grupo de una o varias instrucciones Transact-SQL (funciones que permiten manipular datos). Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:

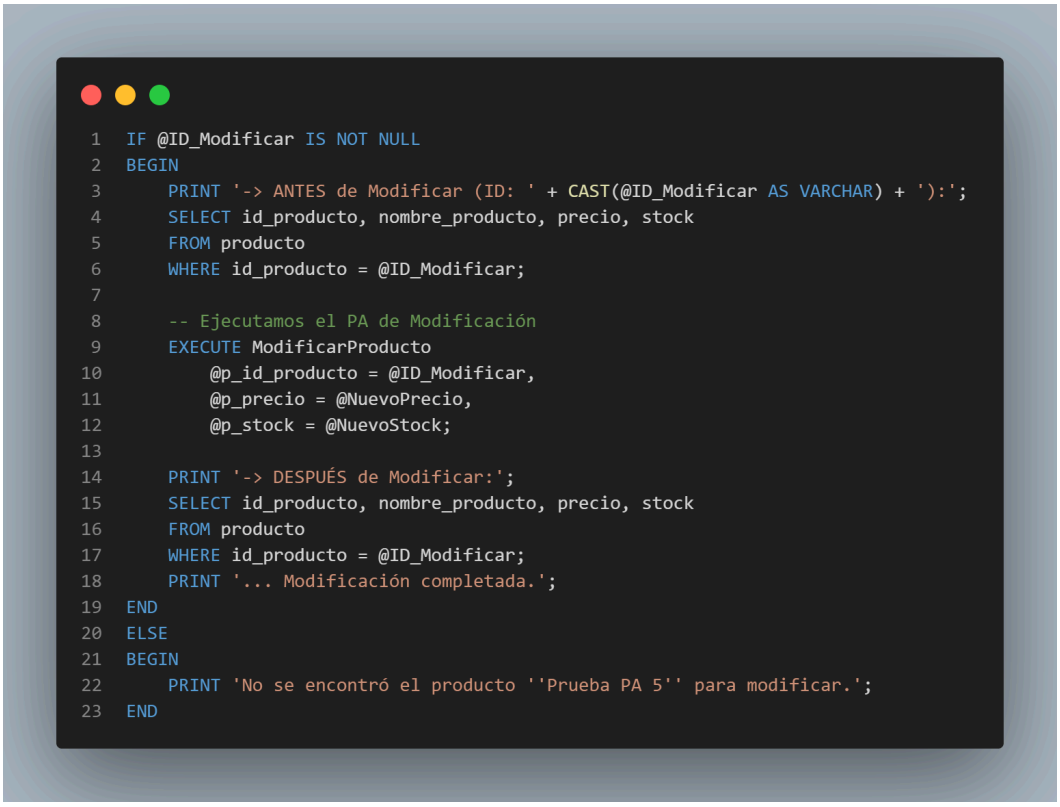
- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.



```
CREATE PROCEDURE InsertarProducto (
    -- Parámetros de entrada
    @p_nombre VARCHAR(150),
    @p_precio DECIMAL (10,2),
    @p_stock INT,
    @p_id_categoria INT,
    @p_id_proveedor INT
)
```

- Contener instrucciones de programación que realicen operaciones en la base de datos. Entre otras, pueden contener llamadas a otros procedimientos.
- Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

Devuelve conjuntos de resultados, parámetros de salida o ambos, pero como tal un valor como una función.



```
IF @ID_Modificar IS NOT NULL
BEGIN
    PRINT '-> ANTES de Modificar (ID: ' + CAST(@ID_Modificar AS VARCHAR) + '):';
    SELECT id_producto, nombre_producto, precio, stock
    FROM producto
    WHERE id_producto = @ID_Modificar;

    -- Ejecutamos el PA de Modificación
    EXECUTE ModificarProducto
        @p_id_producto = @ID_Modificar,
        @p_precio = @NuevoPrecio,
        @p_stock = @NuevoStock;

    PRINT '-> DESPUÉS de Modificar:';
    SELECT id_producto, nombre_producto, precio, stock
    FROM producto
    WHERE id_producto = @ID_Modificar;
    PRINT '... Modificación completada.';
END
ELSE
BEGIN
    PRINT 'No se encontró el producto ''Prueba PA 5'' para modificar.';
END
```

¿Como se utiliza?

Cuando queremos mejorar el rendimiento al reducir el tráfico de red (reducir cantidad de datos que se envían), aumentar la seguridad al controlar el acceso a los datos y facilitar el mantenimiento al permitir reutilizar código o modularizar la lógica, entra en acción ese mecanismo.

Creación de Procedimientos Almacenados (Forma 1)

> Aclaración: En este artículo se describe cómo crear un procedimiento almacenado de SQL Server mediante SQL Server Management

1. En el **Explorador de objetos**, conéctese a una instancia de SQL Server
2. Expanda la instancia y, a continuación, expanda **Bases de datos**.
3. Expanda la base de datos que desee y, a continuación, expanda **Programabilidad**.
4. Haga clic con el botón derecho en **Procedimientos almacenados** y, a continuación, seleccione **Nuevo**>**Procedimiento almacenado**.

   Se abrirá una nueva ventana Consulta con una plantilla para el procedimiento almacenado.
5. En el menú **Consulta**, seleccione **Especificar valores para parámetros de plantilla**.
6. En el cuadro de diálogo **Especificar valores para los parámetros de plantilla**, proporcione la siguiente información de los campos **Valor**:

   - **Author**: reemplace `Name` por su nombre.

- **Create Date**: introduzca la fecha de hoy.
- **Description**: describa brevemente lo que hace el procedimiento.
- **Procedure_Name**: reemplace `ProcedureName` por el nuevo nombre del procedimiento almacenado.
- **@Param1**: reemplace `@p1` por el nombre del primer parámetro, como *@ColumnName1*.
- **@Datatype_For_Param1**: según corresponda, reemplace `int` por el tipo de datos del primer parámetro, como *nvarchar(50)*.
- **Default_Value_For_Param1**: según corresponda, reemplace `0` por el valor predeterminado del primer parámetro o *NULL*.
- **@Param2**: reemplace `@p2` por el nombre del segundo parámetro, como *@ColumnName2*.
- **@Datatype_For_Param2**: según corresponda, reemplace `int` por el tipo de datos del segundo parámetro, como *nvarchar(50)*.
- **Default_Value_For_Param2**: según corresponda, reemplace `0` por el valor predeterminado del segundo parámetro o *NULL*.

7. Select **OK**.
8. En el **Editor de Power Query**, reemplace la instrucción SELECT por la consulta del procedimiento.

Creación de Procedimientos Almacenados (Forma 2 - Utilizada en el proyecto)

1. En SSMS, conéctese a una instancia de SQL Server.
2. Seleccione **Nueva consulta** en la barra de herramientas.
3. Escriba el código siguiente en la ventana Consulta, reemplazando `<ProcedureName>`, los nombres y los tipos de datos de cualquier parámetro y la instrucción SELECT por sus propios valores.

```
CREATE PROCEDURE <ProcedureName>
  @<ParameterName1> <data type>,
  @<ParameterName2> <data type>
AS

  SET NOCOUNT ON;
  SELECT <your SELECT statement>;
GO
```

4. Seleccione **Ejecutar** en la barra de herramientas para ejecutar la consulta. Se creará el procedimiento almacenado.

Ejecución del Procedimiento Almacenado

> Aclaración: En este artículo se describe cómo crear un procedimiento almacenado de SQL Server mediante SQL Server Management

1. En SSMS, conéctese a una instancia de SQL Server o Azure SQL Database.
2. En la barra de herramientas, seleccione **Nueva consulta**.
3. Escriba una instrucción EXECUTE con la siguiente sintaxis en la ventana Consulta y proporcione valores para todos los parámetros esperados:

   ```
   EXECUTE <ProcedureName> N'<Parameter 1 value>, N'<Parameter x value>;
   GO
   ```

   Por ejemplo, la siguiente instrucción de Transact-SQL ejecuta el procedimiento almacenado `uspGetCustomerCompany` y con `Cannon` como valor del parámetro `@LastName` y `Chris` como valor del parámetro `@FirstName`:

   ```
   EXEC SalesLT.uspGetCustomerCompany N'Cannon', N'Chris';
   GO
   ```

4. En la barra de herramientas, seleccione **Ejecutar**. Se ejecutará el procedimiento almacenado.

Opciones de parámetro

Si bien hay varias maneras de introducir los parámetros en la instrucción EXECUTE del procedimiento almacenado creado. Las formas más comunes son las siguientes:

- Si proporciona los valores de parámetro en el mismo orden que se definen en el procedimiento almacenado, no es necesario indicar los nombres de parámetro. Por ejemplo:

  ```
  EXEC SalesLT.uspGetCustomerCompany N'Cannon', N'Chris';
  ```

- Si proporciona nombres de parámetro en el patrón de `@parameter_name=value`, no es necesario especificar los nombres de parámetro y los valores en el mismo orden que se definen. Por ejemplo, las dos instrucciones siguientes son válidas:

  ```
  EXEC SalesLT.uspGetCustomerCompany @FirstName = N'Chris', @LastName = N'Cann
  on';
  ```

  ```
  EXEC SalesLT.uspGetCustomerCompany @LastName = N'Cannon', @FirstName = N'Ch
  ris';
  ```

- Si usa el formulario `@parameter_name=value` para cualquier parámetro, debe usarlo para todos los parámetros posteriores de esa instrucción. Por ejemplo, no puede usar `EXEC SalesLT.uspGetCustomerCompany1 @FirstName = N'Chris', N'Cannon';`.

Sintaxis básica

```
-- Crear
CREATE PROCEDURE NombreDelProcedimiento
AS
```

```
BEGIN
   -- Sentencias SQL (INSERT, UPDATE, SELECT, etc.)
END;
GO

-- Ejecutar
EXECUTE NombreDelProcedimiento;
GO

-- Modificar
ALTER PROCEDURE NombreDelProcedimiento
AS
BEGIN
   -- Nuevas sentencias SQL
END;
GO

-- Eliminar
DROP PROCEDURE NombreDelProcedimiento;
GO
```

Funciones Almacenadas

Al igual que los procedimientos almacenados son objetos de bases de datos que contienen un conjunto de instrucciones SQL.

Ofreciendo reutilización, ingreso de parametros, devuelve un valor escalar o tabla, Realizar operaciones sobre datos y devuelven el resultado.

¿Como se usa?

Cuando requerimos de obtener un valor. Las funciones están diseñadas para usarse en instrucciones SQL donde se pueda usar una expresión. A menudo se utilizan para cálculos de valores (sumar, restar, calcular impuestos, calcular edades etc.) o transformaciones de datos (convertir formatos, limpiar datos, derivar nuevos valores, aplicar reglas de negocios).

Creación de funciones almacenadas

> Aclaración: En este artículo se describe cómo crear un procedimiento almacenado de SQL Server mediante SQL Server Management

1. Expanda la instancia y la base de datos en el **Explorador de objetos**.
2. Expanda `Programabilidad`.
3. Haga clic derecho en `Funciones` y seleccione `Nueva función` > `Función con valores de tabla en línea` (u otro tipo de función).

4. Se abrirá una nueva ventana de consulta con una plantilla para empezar a escribir su código.

Desglose de la sintaxis

- `CREATE FUNCTION`: Indica que está creando una función.
- `dbo.GetProductName`: El esquema y el nombre de la función.
- `(@ProductID INT)`: Los parámetros de entrada. Se define el nombre del parámetro (con `@`) y su tipo de dato (`INT`).
- `RETURNS NVARCHAR(100)`: Especifica el tipo de dato que la función devolverá.
- `AS`: Introduce el cuerpo de la función.
- `BEGIN` ... `END`: Delimita las instrucciones de código que forman el cuerpo de la función.
- `GO`: Es un comando de lote en SQL Server que separa los comandos y se ejecuta por separado.

Ejecución del Funciones Almacenado

Hay dos formas basicas de ejecutar una función almacenada.

- **Usando SQL Server Management Studio**
  1. Abre SQL Server Management Studio y conéctate a tu instancia de SQL Server.
  2. En el **Explorador de objetos**, expande la base de datos deseada, luego "Programabilidad" y busca el objeto en "Procedimientos almacenados" o "Funciones".
  3. Haz clic derecho sobre el procedimiento o función que deseas ejecutar.
  4. Selecciona la opción "Ejecutar procedimiento almacenado".
  5. Si el objeto tiene parámetros, aparecerá un cuadro de diálogo donde puedes ingresar los valores para cada uno antes de hacer clic en "Aceptar".
- **Usando Transact SQL (T-SQL)**
  1. Abre una nueva ventana de consulta en **SSMS**.
  2. Usa la palabra clave `EXEC` o `EXECUTE` seguida del nombre de la función o procedimiento que quieres ejecutar.
  3. Si el objeto requiere parámetros de entrada, proporciónalos entre paréntesis después del nombre.

```
-- Para ejecutar un procedimiento almacenado
EXEC dbo.uspGetCustomerCompany @LastName = 'Cannon', @FirstName = 'Chris';

-- Para ejecutar una función (el resultado se devuelve directamente)
SELECT dbo.MiFuncion(Parametro1, @Parametro2) FROM MiTabla;
```

  - `EXEC` es un acortamiento de `EXECUTE` y se usa para llamar procedimientos o ejecutar sentencias SQL dinámicas.
  - Ten en cuenta que las **funciones** se usan principalmente en la cláusula `SELECT` y retornan un único valor, mientras que los **procedimientos almacenados** ejecutan un conjunto de

instrucciones y pueden tener parámetros de entrada y salida.

Sintaxis Básica

```
CREATE FUNCTION [nombre_esquema.] nombre_función (
   @parametro1 tipo_dato1,
   @parametro2 tipo_dato2
)
RETURNS tipo_dato_retorno
AS
BEGIN
   -- Lógica de la función SQL
   RETURN (sentencia_sql);
END
GO
```

Diferencias entre Funciones y procedimientos

Característica	Funciones almacenadas	Procedimientos almacenados
Valor de retorno	Obligatoriamente un valor (escalar o tabla)	Puede o no devolver un valor o un conjunto de resultados
Uso	Se pueden llamar dentro de sentencias `SELECT`, `WHERE`, etc.	Se ejecutan como comandos separados
Modificación de datos	No pueden modificar datos directamente en la base de datos (no realizan operaciones de `INSERT`, `UPDATE`, `DELETE`)	Pueden modificar datos, manejar transacciones y lógica compleja
Parámetros	Solo aceptan parámetros de entrada	Pueden aceptar parámetros de entrada y salida

¿Como Aplicar los prodecimientos y funciones en implementaciones de operaciones CRUD?

Tanto para los procedimientos y funciones tiene su forma de hacerlo.

Aplicación en procedimientos - Funciones Almacenados

Primero de debe crear procedimientos almacenados separados.

- Procedimiento para `INSERT`

```
CREATE PROCEDURE sp_insert_producto
   @nombre VARCHAR(50),
   @precio DECIMAL(10,2)
AS
BEGIN
   INSERT INTO Productos (Nombre, Precio)
```

```
    VALUES (@nombre, @precio);
END
GO
```

- Procedimiento para `SELECT`

```
CREATE PROCEDURE sp_select_producto_por_id
    @id INT
AS
BEGIN
    SELECT *
    FROM Productos
    WHERE ProductoID = @id;
END
GO
```

- Procedimiento para `UPDATE`

```
CREATE PROCEDURE sp_update_producto
    @id INT,
    @nombre VARCHAR(50),
    @precio DECIMAL(10,2)
AS
BEGIN
    UPDATE Productos
    SET Nombre = @nombre, Precio = @precio
    WHERE ProductoID = @id;
END
GO
```

- Procedimiento para `DELECT`

```
CREATE PROCEDURE sp_delete_producto
    @id INT
AS
BEGIN
    DELETE FROM Productos
    WHERE ProductoID = @id;
END
GO
```

Una vez creados los procedimientos para el CRUD, se realiza la ejecución de los procedimientos creados.

En una nueva consulta se puede realizar con el comando `EXE`

- Para insertar un registro:

```
EXEC sp_insert_producto @nombre = 'Laptop', @precio = 1200.50;
```

- Para seleccionar un producto por ID:

```
EXEC sp_select_producto_por_id @id = 1;
```

- Para actualizar un producto:

```
EXEC sp_select_producto_por_id @id = 1;
```

- Para eliminar un producto:

```
EXEC sp_delete_producto @id = 1;
```

Analisis de rendimiento - eficiencia

Para cumplir con el objetivo de la materia de "Comparar la eficiencia de las operaciones directas versus el uso de procedimientos y funciones", se implementará una metodología de prueba cuantitativa.

Esta metodología no solo busca validar las afirmaciones teóricas sobre la reducción del tráfico de red y la mejora del rendimiento, sino también medir el impacto real en nuestro SGBD (SQL Server).

Herramientas de Medición

Para obtener métricas objetivas de rendimiento, se utilizarán los siguientes comandos de Transact-SQL en SQL Server Management Studio (SSMS) antes de ejecutar cada prueba:

- `SET STATISTICS TIME ON`: Este comando instruye al servidor para que devuelva el tiempo de CPU y el tiempo transcurrido (en milisegundos) necesarios para analizar, compilar y ejecutar cada sentencia.
- `SET STATISTICS IO ON`: Este comando proporciona estadísticas sobre la cantidad de actividad de E/S (Entrada/Salida) generada por las sentencias. Se registrarán las "lecturas lógicas" (lecturas desde la caché de datos) y las "lecturas físicas" (lecturas desde el disco), que son indicadores clave del costo de una consulta.

Caso de Prueba: Inserción de Lote de Datos (Tarea 2)

Se realizará una comparación directa entre la inserción de un lote de 100 registros en la tabla `producto` (o `usuario`) mediante dos métodos:

1. **Prueba A (SQL Directo):** Se ejecutará un *script* T-SQL que contenga 100 sentencias `INSERT` individuales en un solo lote. Se registrarán las métricas de `STATISTICS TIME` y `STATISTICS IO`.
2. **Prueba B (Procedimiento Almacenado):** Se ejecutará un *script* T-SQL que llame 100 veces, en un bucle, al procedimiento almacenado `sp_insertar_producto` (previamente creado). Se registrarán las métricas de `STATISTICS TIME` y `STATISTICS IO`.

Criterios de Comparación

Los resultados de "tiempo transcurrido" y, especialmente, las "lecturas lógicas" de ambas pruebas (Prueba A y Prueba B) serán registrados y comparados. Se espera que el método de Procedimiento Almacenado demuestre una eficiencia comparable o superior, principalmente validando la teoría de reutilización del plan de ejecución.

Los resultados numéricos, capturas de pantalla de las estadísticas y las conclusiones finales de esta prueba se presentarán en el **Capítulo IV: Desarrollo del Tema/Resultados** del documento final.

Tarea

- Realizar al menos 3 procedimientos almacenados que permitan: Insertar, modificar y borrar registros de alguna de las tablas del proyecto

  np_PA(CRUD).sql

- Insertar un lote de datos en las tablas mencionadas (guardar script) con sentencias `INSERT` y otro lote invocando a los procedimientos creados.

  np_PA(LoteDatos).sql

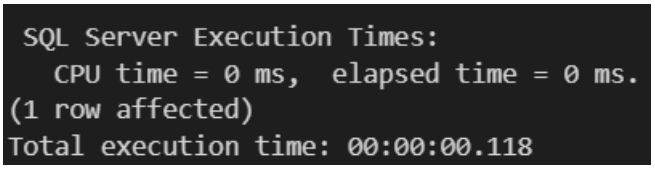
- Realizar update y delete sobre alguno de los registros insertados en esas tablas invocandos a los procedimientos.

  np_PA_(Update_Delete).sql

- Desarrollar al menos 3 funciones almacenadas.

  np_FA.sql

- Comparar la eficiencia de las operaciones directas versus el uso de procedimientos y funciones.
  - Eficiencia de Inserción directa

  - Eficiencia de Procedimiento Almacenado

```
 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
>>> Tarea 2 completada. Dos lotes de 100 productos insertados para el análisis de eficiencia.
Total execution time: 00:00:00.179
```

Análisis

El tiempo que registraste (118 ms para SQL Directo vs 179 ms para el PA) es el **Tiempo Transcurrido**, que es el tiempo total desde que envías el comando hasta que recibes la respuesta.
El SQL directo fue más rápido por la forma en que se ejecutó:

1. SQL Directo (Bloque A) fue más rápido por el Lote Único
El Bloque A envió las 100 sentencias `INSERT` como un **lote único y optimizado** al servidor.
• **Ventaja:** El servidor procesó las 100 sentencias en una única transacción de red entre SSMS y la base de datos.
• **Ahorro:** Esto reduce el tiempo de espera por la red (*latency*) y la sobrecarga de envío/recepción de 100 comandos separados.

2. PA (Bloque B) fue más lento por el Bucle Secuencial
El Bloque B ejecutó el PA **100 veces consecutivas dentro de un bucle** `WHILE`.
• **Desventaja:** Aunque el PA reutiliza el plan de ejecución (que es la ventaja teórica), el bucle añade una **sobrecarga de programación y control de flujo** (el proceso de incrementar la variable `@i`, verificar la condición y volver a llamar al PA, 100 veces) que se suma al tiempo total transcurrido.

Conclusión para tu Tarea 5: Reutilización vs. Bucle
El resultado de la prueba demuestra lo siguiente:
• **PA (Bloque B) ganó en eficiencia de cómputo:** Si revisas la estadística **Tiempo de CPU** (Time) y **Lecturas Lógicas** (IO), es muy probable que la prueba del PA tenga valores comparativamente más bajos por *cada inserción individual*, porque el plan de ejecución **solo se compiló una vez**.
• **SQL Directo (Bloque A) ganó en tiempo transcurrido:** Ganó porque fue un **lote masivo**, minimizando la sobrecarga secuencial.

Fuentes

Documentación técnica de SQL Server - SQL Server
Elija el área de SQL Server que le interese.

https://learn.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver17