
Universidad
Tecmilenio®

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Diseño e implementación de funciones definidas por el usuario

Semana 7

Universidad Tecmilenio

Introducción

GETDATE(), SUM() y MIN() son funciones muy utilizadas en T-SQL, debido a que proporcionan un resultado de manera simple y eficaz. Durante esta experiencia de aprendizaje se creará algo similar, con la diferencia de que esa nueva función personalizada va a contener lógica y validaciones definidas por el programador T-SQL para retornar un resultado.

Explicación

Descripción general de las funciones

¿Qué es una función?

Es una operación denotada por un nombre, seguido por uno o más operandos colocados entre paréntesis, relacionando un conjunto de valores de entrada y un conjunto de valores de resultado (IBM, 2021).

Una manera de saber cuándo diseñar y crear una función, será cuestionar ¿existe otro proceso que realiza la misma operación?

Puntos importantes para considerar cuando se decida diseñar e implementar funciones en SQL Server (Microsoft, 2022a):

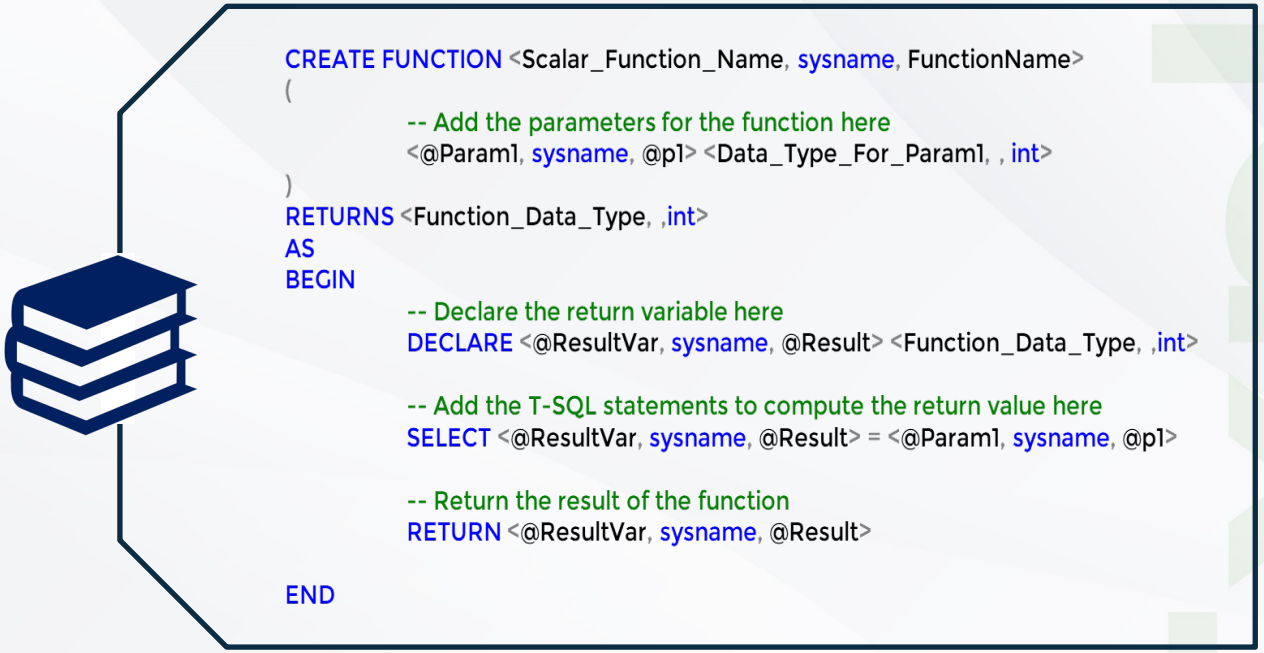
- Permiten una programación modular.
- Permiten una ejecución más rápida.
- Pueden reducir el tráfico de la red.

Universidad Tecmilenio

Explicación

Diseño e implementación de funciones escalares

Las **funciones escalares** en T-SQL son rutinas que aceptan parámetros, realizan una acción, como un cálculo complejo, y devuelven el resultado de esa acción como un valor, el cual puede ser escalar o un conjunto.

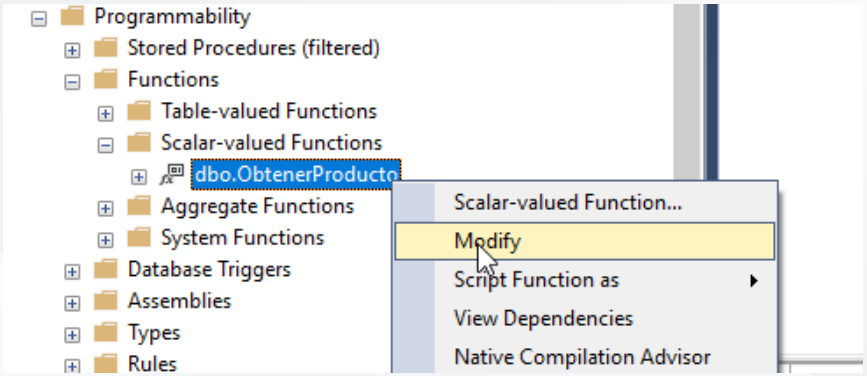


```
CREATE FUNCTION <Scalar_Function_Name, sysname, FunctionName>
(
        -- Add the parameters for the function here
        <@Param1, sysname, @p1> <Data_Type_For_Param1, , int>
)
RETURNS <Function_Data_Type, ,int>
AS
BEGIN
        -- Declare the return variable here
        DECLARE <@ResultVar, sysname, @Result> <Function_Data_Type, ,int>

        -- Add the T-SQL statements to compute the return value here
        SELECT <@ResultVar, sysname, @Result> = <@Param1, sysname, @p1>

        -- Return the result of the function
        RETURN <@ResultVar, sysname, @Result>

END
```

Estructura de una función escalar en T-SQL.

Fuente: BI TALKS BI. (2018). *Difference Between Stored Procedure & Functions*. Recuperado de https://bitalksbi.com/stored-procedure-function-sql-example.

Explicación



Modificación de una función escalar en modo gráfico en T-SQL.

Explicación

Diseño e implementación de funciones con valores de tabla

De acuerdo con Microsoft (2022b), aquella función que devuelve una tabla es denominada como **función con valores de tabla**, las cuales materializan, en Transact-SQL, los resultados obtenidos al llamar a la función en una tabla intermedia.

```
CREATE FUNCTION <Inline_Function_Name, sysname, FunctionName>
(
        -- Add the parameters for the function here
        <@param1, sysname, @p1> <Data_Type_For_Param1, , int>,
        <@param2, sysname, @p2> <Data_Type_For_Param2, , char>
)
RETURNS TABLE
AS
RETURN
(
        -- Add the SELECT statement with parameter references here
        SELECT 0
)
GO
```

Estructura de una función con valores de tabla en T-SQL.

Fuente: BI TALKS BI. (2018). *Difference Between Stored Procedure & Functions*. Recuperado de https://bitalksbi.com/stored-procedure-function-sql-example.

Universidad Tecmilenio

Explicación

```
-- Modificar una función
ALTER FUNCTION [dbo].[ObtenerProducto]
(
		-- Modificaciones
		...
)
		-- Modificaciones
RETURNS ...
AS
BEGIN
		-- Modificaciones
		...
END

-- Eliminar una función
DROP FUNCTION  [dbo].[ObtenerProductoCompleto]

-- Renombrar una función
sp_rename 'ObtenerProductoCompleto2' , 'ObtenerProductoCompletoTbl'
```

Modificar, eliminar y renombrar funciones en T-SQL.

Actividad

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

01 Explica un escenario ideal para aplicar una función con valores de tabla.

02 ¿Cómo saber cuándo aplicar una función escalar y una función con valores de tabla?

Cierre

La **sintaxis** entre una función escalar y una función con valores de tabla puede ser similar, pero el resultado que retorna es diferente, es decir, cuando se diseña una función, se debe analizar el propósito por el cual es creada. No se trata de ver cuál es mejor, sino de crear la función que facilite la implementación en uno o más procesos.

Las funciones, básicamente, existen en todos los lenguajes de programación, y si aún queda duda del concepto, se puede interpretar como una caja negra en la cual entran datos, se procesan y regresa un resultado.

Bibliografía

• BI TALKS BI. (2018). *Difference Between Stored Procedure & Functions*. Recuperado de https://bitalksbi.com/stored-procedure-function-sql-example

• IBM. (2021). *Funciones*. Recuperado de https://www.ibm.com/docs/es/db2/11.1?topic=elements-functions

• Microsoft. (2022a). *Funciones definidas por el usuario*. Recuperado de https://docs.microsoft.com/es-es/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver16

• Microsoft. (2022b). *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

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Manipulación de datos a través de disparadores

Semana 7

Introducción

Se explorará una funcionalidad en SQL Server llamada **disparadores** que, basados en reglas y parametrizaciones, se ejecutará de manera automática para realizar procesos determinados con resultados en el tiempo y forma que se requieren.

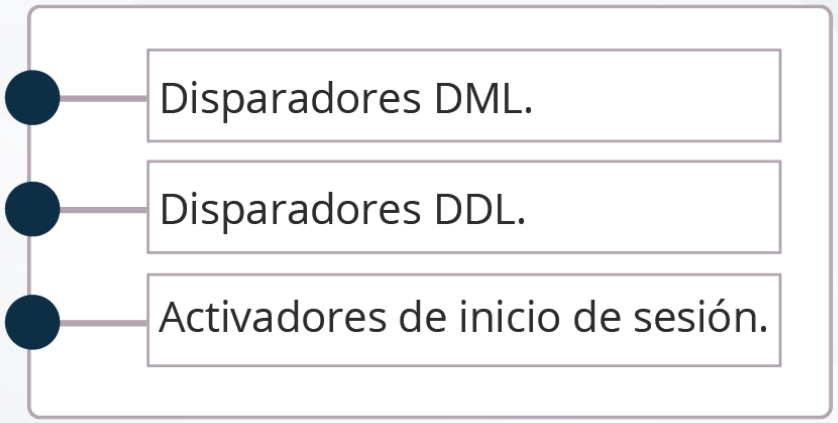
Explicación

Diseño de disparadores (*triggers*)

¿Qué es un disparador en SQL Server?

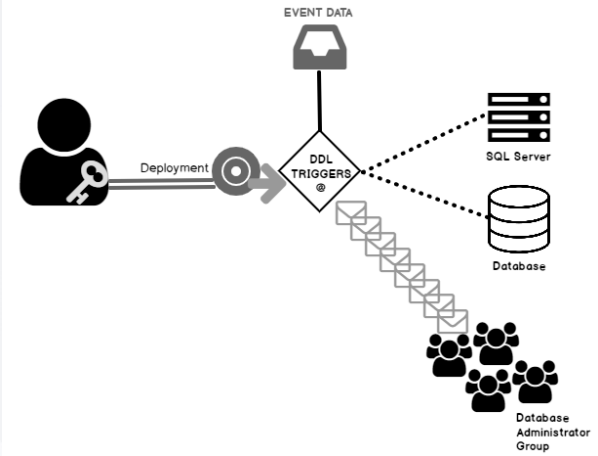
De acuerdo con Microsoft (2022), un **disparador** es un procedimiento especial de almacenado que cuenta con una ejecución automática al momento de cumplirse un evento en el servidor de la base de datos.

Emil Drkusic (2020), explica que tenemos tres grupos de disparadores utilizados en SQL:



Universidad Tecmilenio

Explicación



Representación gráfica de un disparador con DDL.

Fuente: Raiyani, J. (2019). *Database Level DDL Triggers for Views, Procedures and Functions.* Recuperado de https://www.sqlshack.com/database-level-ddl-triggers-for-views-procedures-and-functions/

```
CREATE TRIGGER <Schema_ Name, sysname,
Schema_ Name>.<Trigger_ Name, sysname, Trigger_ Name>
  ON  <Schema_ Name, sysname,
Schema_ Name>.<Table_ Name, sysname, Table_ Name>
  AFTER <Data_ Modification_ Statements, ,
INSERT,DELETE,UPDATE>
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets
from
```

Estructura de disparador en T-SQL.

Universidad Tecmilenio

Explicación

Implementación de disparadores (triggers)

Con el comando **CREATE TRIGGER**, seguido del nombre **"TR_InsertarVentaWS"**, y después del comando ON, se debe indicar el nombre de la tabla que se estará observando si ocurre un evento. Con **AFTER INSERT** se indica que se ejecutarán las siguientes sentencias en cuanto se termine de insertar la información en la tabla "Ventas". Dentro del **BEGIN… END** se deberá colocar toda la lógica que se desee.

```
CREATE TRIGGER TR_ InsertarVentasWS
  ON [dbo].[Ventas]
  AFTER INSERT
AS
BEGIN
      DECLARE @ventaID   AS INT
            ,@clienteID      AS INT
            ,@ventaFecha     AS DATETIME
            ,@descuentoID  AS INT

      SELECT
            @ventaID = [i].[VentaID]
            ,@clienteID      = [i].[ClienteID]
            ,@ventaFecha = [i].[VentaFecha]
            ,@descuentoID = [i].[DescuentoID]
      FROM
            INSERTED AS [i]


      INSERT INTO [dbo].[VentasWS]
      ([VentaID], [ClienteID], [VentaFecha], [DescuentoID])
      VALUES (@ventaID, @clienteID, @ventaFecha,
@descuentoID)

END
GO
```

Creación del disparador "TR_InsertarVentaWS" en T-SQL.

Universidad Tecmilenio

Explicación

Conceptos avanzados

Para modificar un trigger, es necesario usar el comando **ALTER TRIGGER** o, en su defecto, en el Object Explorer para posicionar el puntero en el objeto, hacer clic derecho, seleccionar y hacer clic en modificar. Una vez realizada esta operación, se habilitará el trigger para aplicar las adecuaciones.

Características de un **disparador** (trigger), en SQL Server:

1. Es un objeto que no soporta parámetros de entrada ni de salida.
2. Se puede accionar sobre un **INSERT, UPDATE** o **DELETE.**
3. Se dispara automáticamente asociado a un evento.
4. Es apto para mantener la integridad de los datos.
5. Se puede tener más de un disparador asociado a una tabla.
6. Se puede combinar **INSERT, UPDATE** o **DELETE** en un mismo disparador y se recomienda usar el comando **MERGE** para dicha acción.
7. Dentro de la lógica del trigger se pueden invocar funciones, tablas, vistas e incluso procedimientos almacenados.

Actividad

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

01 ¿Sería una buena opción utilizar un disparador para monitorear la cantidad de productos que hay en una bodega?

02 Menciona cómo diseñarías un disparador para alertar la eliminación de información relacionada con los clientes.

Universidad Tecmilenio

Cierre

El uso de **disparadores** en SQL Server se puede extender tanto como se desee. Ahora que se conocen las bases, será tarea de cada programador T-SQL explorar todos los beneficios que proporciona crear, diseñar y poner en marcha este tipo de objetos.

En cualquier industria de producción de bienes siempre será importante manejar un inventario sano, lo cual significa tener los insumos necesarios para no detener la producción y contar con el mínimo de stock necesario para no tener inconvenientes con la entrega del producto final al cliente.

Bibliografía

• Drkusic, E. (2020). *Learn SQL: SQL Triggers*. Recuperado de https://www.sqlshack.com/learn-sql-sql-triggers/

• Microsoft. (2022). *CREATE TRIGGER (Transact-SQL)*. Recuperado de https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver16

• Raiyani, J. (2019). *Database Level DDL Triggers for Views, Procedures and Functions*. Recuperado de https://www.sqlshack.com/database-level-ddl-triggers-for-views-procedures-and-functions/

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Almacenamiento y consulta de datos XML

Semana 7

Universidad Tecmilenio

Introducción

Es común encontrar el tipo de dato **XML** en SQL Server cuando se trabaja con un sistema que intercambia información con otro sistema, ya que este formato se ha vuelto un estándar para la comunicación entre diferentes fuentes de datos. En esta experiencia de aprendizaje se utilizarán sentencias T-SQL para aprovechar al máximo todo lo que se puede lograr con este tipo de dato.

¿Qué es un XML (*extensible markup language*)?

Es un lenguaje de marcado extensible de propósito general y una especificación de W3C, ya que no está predefinido, por lo que debes definir tus propias etiquetas. Su propósito principal es compartir datos a través de diferentes sistemas (MDN web docs, 2022).

Universidad Tecmilenio

Explicación

Almacenamiento de datos y esquemas XML

- La empresa Comercializadora Siglo XXI.

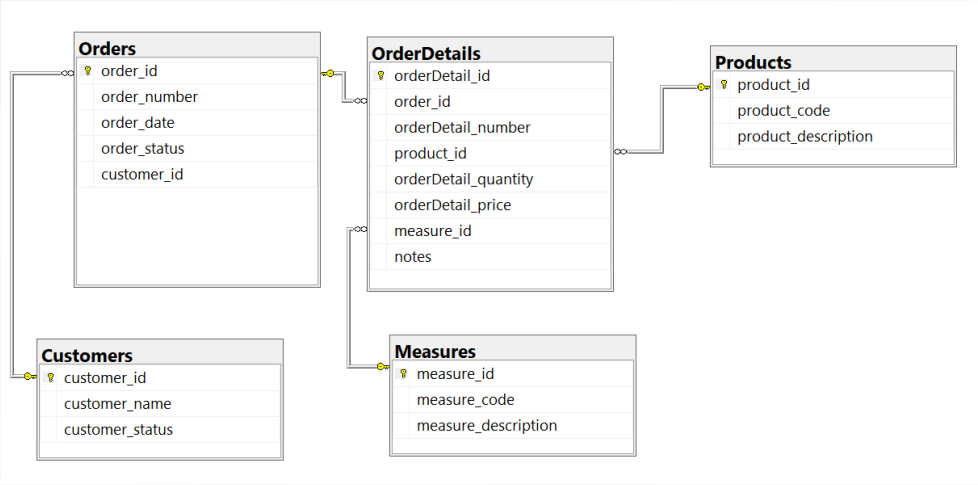


Diagrama entidad-relación de “DBSiglo” en T-SQL.

```
CREATE TABLE [dbo].[OrdersSharing](
	[ID]	[int]	PRIMARY KEY IDENTITY(1,1) NOT NULL,
	[DateInserted]	[datetime] NOT NULL,
	[XMLData]	[xml]	NOT NULL,
	[Processed]	[bit]	NOT NULL,
)
```

Crear tabla “OrdersSharing” en T-SQL.

Explicación

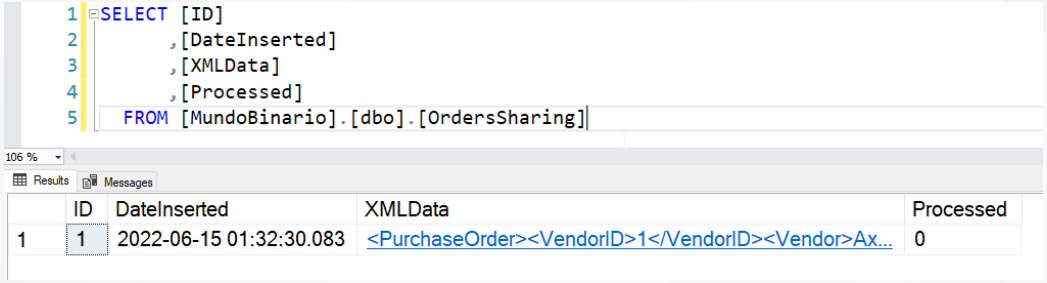
```
DECLARE @xml NVARCHAR(MAX)

SET @xml = N'
<PurchaseOrder>
  <VendorID>1</ VendorID>
  <Vendor>Axel Romero</ Vendor>
  <PONumber>001001</ PONumber>
  <Date>2021-02-10T14:20:18.133</ Date>
  <LineItem>
    <Product>Libreta profesional con rayas</ Product>
    <Qty>10.00</ Qty>
    <Price>5.00</ Price>
    <Amount>50.0000</ Amount>
  </ LineItem>
  <LineItem>
    <Product>Goma bicolor azul-rojo</ Product>
    <Qty>10.00</ Qty>
    <Price>5.00</ Price>
    <Amount>50.0000</ Amount>
  </ LineItem>
  <LineItem>
    <Product>Lápiz de puntillas finas</ Product>
    <Qty>10.00</ Qty>
    <Price>5.00</ Price>
    <Amount>50.0000</ Amount>
  </ LineItem>
  <LineItem>
    <Product>Hoja blanca tamaño carta</ Product>
    <Qty>10.00</ Qty>
    <Price>5.00</ Price>
    <Amount>50.0000</ Amount>
  </ LineItem>
</ PurchaseOrder>'

INSERT INTO [dbo].[OrdersSharing]
([DateInserted], [XMLData], [Processed])
VALUES
(GETDATE(), @xml, 0)
```

Insertar información a la “OrdersSharing” en T-SQL.

Explicación

```
SELECT [ID]
      ,[DateInserted]
      ,[XMLData]
      ,[Processed]
FROM [MundoBinario].[dbo].[ OrdersSharing]
```

SELECT a la tabla “OrdersSharing” en T-SQL.



SELECT a la tabla “OrdersSharing” en T-SQL.

Universidad Tecmilenio

Explicación

```xml
<PurchaseOrder>
  <VendorID>1</VendorID>
  <Vendor>Axel Romero</Vendor>
  <PONumber>001001</PONumber>
  <Date>2021-02-10T14:20:18.133</Date>
  <LineItem>
    <Product>Libreta profesional con rayas</Product>
    <Qty>10.00</Qty>
    <Price>5.00</Price>
    <Amount>50.0000</Amount>
  </LineItem>
  <LineItem>
    <Product>Goma bicolor azul-rojo</Product>
    <Qty>10.00</Qty>
    <Price>5.00</Price>
    <Amount>50.0000</Amount>
  </LineItem>
  <LineItem>
    <Product>Lápiz de puntillas finas</Product>
    <Qty>10.00</Qty>
    <Price>5.00</Price>
    <Amount>50.0000</Amount>
  </LineItem>
  <LineItem>
    <Product>Hoja blanca tamaño carta</Product>
    <Qty>10.00</Qty>
    <Price>5.00</Price>
    <Amount>50.0000</Amount>
  </LineItem>
</PurchaseOrder>
```

Visualización de un XML en T-SQL.

Universidad Tecmilenio

Explicación

Uso de la instrucción FOR XML

- Se puede usar la instrucción **FOR XML** para formar contenido de manera automática en una tabla en SQL Server.

```
SELECT
        [o].[customer_ id]                          AS [VendorID]
        ,[c].[customer_ name]                           AS [Vendor]
        ,FORMAT([o].[order_ number],'000000')       AS [PONumber]
        ,[o].[order_ date]                          AS [Date]
        ,[p].[product_ description]                 AS [Product]
        ,[od].[orderDetail_ quantity]               AS [Qty]
        ,[od].[orderDetail_ price]                  AS [Price]
        ,[od].[orderDetail_ quantity] *
        [od].[orderDetail_ price]                   AS [Amount]
FROM
        [dbo].[Orders] AS [o]
        INNER JOIN [dbo].[Customers] AS [c]
                ON [o].[customer_ id] = [c].[customer_ id]
        INNER JOIN [dbo].[OrderDetails] AS [od]
                ON [o].[order_ id] = [od].[order_ id]
        INNER JOIN [dbo].[Products] AS [p]
                ON [od].[product_ id] = [p].[product_ id]
WHERE
        [o].[order_ id] = 1
FOR XML RAW
```

Uso de FOR XML RAW en T-SQL.

Universidad Tecmilenio

Explicación

```
<row VendorID="1" Vendor="Axel Romero" PONumber="001001" Date="2021-02-10T14:20:18.133" Product="Libreta profesional con rayas" Qty=
<row VendorID="1" Vendor="Axel Romero" PONumber="001001" Date="2021-02-10T14:20:18.133" Product="Goma bicolor azul-rojo" Qty="10.00"
<row VendorID="1" Vendor="Axel Romero" PONumber="001001" Date="2021-02-10T14:20:18.133" Product="Lápiz de puntillas finas" Qty="10.0
<row VendorID="1" Vendor="Axel Romero" PONumber="001001" Date="2021-02-10T14:20:18.133" Product="Hoja blanca tamaño carta" Qty="10.0
```

Visualización de un XML con FOR XML RAW en T-SQL.

```
SELECT
	[o].[customer_id]                    AS [VendorID]
	,[c].[customer_name]                 AS [Vendor]
	,FORMAT([o].[order_number],'000000') AS [PONumber]
	,[o].[order_date]                    AS [Date]
	,[p].[product_description]           AS [Product]
	,[od].[orderDetail_quantity]         AS [Qty]
	,[od].[orderDetail_price]            AS [Price]
	,[od].[orderDetail_quantity] *
	[od].[orderDetail_price]             AS [Amount]
FROM
	[dbo].[Orders] AS [o]
	INNER JOIN [dbo].[Customers] AS [c]
		ON [o].[customer_id] = [c].[customer_id]
	INNER JOIN [dbo].[OrderDetails] AS [od]
		ON [o].[order_id] = [od].[order_id]
	INNER JOIN [dbo].[Products] AS [p]
		ON [od].[product_id] = [p].[product_id]
WHERE
	[o].[order_id] = 1
FOR XML AUTO, ELEMENTS
```

Uso de FOR XML AUTO, ELEMENTS en T-SQL.

Universidad Tecmilenio

Explicación

```xml
<o>
  <VendorID>1</VendorID>
  <Date>2021-02-10T14:20:18.133</Date>
  <c>
    <Vendor>Axel Romero</Vendor>
    <PONumber>001001</PONumber>
    <p>
      <Product>Libreta profesional con rayas</Product>
      <od>
        <Qty>10.00</Qty>
        <Price>5.00</Price>
        <Amount>50.0000</Amount>
      </od>
    </p>
    <p>
      <Product>Goma bicolor azul-rojo</Product>
      <od>
        <Qty>10.00</Qty>
        <Price>5.00</Price>
        <Amount>50.0000</Amount>
      </od>
    </p>
    <p>
      <Product>Lápiz de puntillas finas</Product>
      <od>
        <Qty>10.00</Qty>
        <Price>5.00</Price>
        <Amount>50.0000</Amount>
      </od>
    </p>
    <p>
      <Product>Hoja blanca tamaño carta</Product>
      <od>
        <Qty>10.00</Qty>
        <Price>5.00</Price>
        <Amount>50.0000</Amount>
      </od>
    </p>
  </c>
</o>
```

Visualización de un XML con FOR XML AUTO, ELEMENTS en T-SQL.

Universidad Tecmilenio

Explicación

```
SELECT
	[order].[customer_ id]			AS [VendorID]
	,[customer].[customer_ name]		AS [Vendor]
	,FORMAT([order].[order_ number],'000000')	AS [PONumber]
	,[order].[order_ date]			AS [Date]
	,[product].[product_ description]		AS [Product]
	,[orderDetail].[orderDetail_ quantity]	AS [Qty]
	,[orderDetail].[orderDetail_ price]		AS [Price]
	,[orderDetail].[orderDetail_ quantity] *
	[orderDetail].[orderDetail_ price]		AS [Amount]
FROM
	[dbo].[Orders] AS [order]
	INNER JOIN [dbo].[Customers] AS [customer]
		ON [order].[customer_ id] = [customer].[customer_ id]
	INNER JOIN [dbo].[OrderDetails] AS [orderDetail]
		ON [order].[order_ id] = [orderDetail].[order_ id]
	INNER JOIN [dbo].[Products] AS [product]
		ON [orderDetail].[product_ id] = [product].[product_ id]
WHERE
	[order].[order_ id] = 1
FOR XML AUTO, ELEMENTS, ROOT('PurchaseOrder')
```

Uso de FOR XML AUTO, ELEMENTS, ROOT(“) en T-SQL.

Universidad Tecmilenio

Explicación

```xml
<PurchaseOrder>
  <order>
    <VendorID>1</VendorID>
    <Date>2021-02-10T14:20:18.133</Date>
    <customer>
      <Vendor>Axel Romero</Vendor>
      <PONumber>001001</PONumber>
      <product>
        <Product>Libreta profesional con rayas</Product>
        <orderDetail>
          <Qty>10.00</Qty>
          <Price>5.00</Price>
          <Amount>50.0000</Amount>
        </orderDetail>
      </product>
      <product>
        <Product>Goma bicolor azul-rojo</Product>
        <orderDetail>
          <Qty>10.00</Qty>
          <Price>5.00</Price>
          <Amount>50.0000</Amount>
        </orderDetail>
      </product>
      <product>
        <Product>Lápiz de puntillas finas</Product>
        <orderDetail>
          <Qty>10.00</Qty>
          <Price>5.00</Price>
          <Amount>50.0000</Amount>
        </orderDetail>
      </product>
      <product>
        <Product>Hoja blanca tamaño carta</Product>
        <orderDetail>
          <Qty>10.00</Qty>
          <Price>5.00</Price>
          <Amount>50.0000</Amount>
        </orderDetail>
      </product>
    </customer>
  </order>
</PurchaseOrder>
```

Visualización de un XML con FOR XML AUTO, ELEMENTS, ROOT(“) en T-SQL.

Universidad Tecmilenio

Explicación

```
SELECT
	[o].[customer_ id]			AS [VendorID]
	,[c].[customer_ name]			AS [Vendor]
	,FORMAT([o].[order_ number],'000000')	AS [PONumber]
	,[o].[order_ date]			AS [Date]
	,(
		SELECT
		[p].[product_ description]	AS [Product]
		,[od].[orderDetail_ quantity]	AS [Qty]
		,[od].[orderDetail_ price]	AS [Price]
		,[od].[orderDetail_ quantity] *
		[od].[orderDetail_ price]	AS [Amount]
		FROM
			[dbo].[OrderDetails] AS [od]
			INNER JOIN [dbo].[Products] AS [p]
				ON [od].[product_ id] = [p].[product_ id]
		WHERE
			[o].[order_ id] = [od].[order_ id]
		FOR XML PATH('LineItem'), TYPE
	)
FROM
	[dbo].[Orders] AS [o]
	INNER JOIN [dbo].[Customers] AS [c]
		ON [o].[customer_ id] = [c].[customer_ id]
WHERE
	[o].[order_ id] = 1
FOR XML PATH ('PurchaseOrder'), TYPE
```

Uso de FOR XML PATH en T-SQL.

Universidad Tecmilenio

Explicación

```xml
<PurchaseOrder>
  <VendorID>1</VendorID>
  <Vendor>Axel Romero</Vendor>
  <PONumber>001001</PONumber>
  <Date>2021-02-10T14:20:18.133</Date>
  <LineItem>
    <Product>Libreta profesional con rayas</Product>
    <Qty>10.00</Qty>
    <Price>5.00</Price>
    <Amount>50.0000</Amount>
  </LineItem>
  <LineItem>
    <Product>Goma bicolor azul-rojo</Product>
    <Qty>10.00</Qty>
    <Price>5.00</Price>
    <Amount>50.0000</Amount>
  </LineItem>
  <LineItem>
    <Product>Lápiz de puntillas finas</Product>
    <Qty>10.00</Qty>
    <Price>5.00</Price>
    <Amount>50.0000</Amount>
  </LineItem>
  <LineItem>
    <Product>Hoja blanca tamaño carta</Product>
    <Qty>10.00</Qty>
    <Price>5.00</Price>
    <Amount>50.0000</Amount>
  </LineItem>
</PurchaseOrder>
```

Visualización de un XML con FOR XML PATH en T-SQL.

Universidad Tecmilenio

Introducción a la consulta de XML

Explicación

```
-- Variables a utilizar en la extracción de datos.
DECLARE @i_XmlDoc               NVARCHAR(MAX)
       ,@xmlDocID               INTEGER

-- Obtener XML de OrdersSharing y guardarlo en @i_XmlDoc
SELECT @i_XmlDoc = CONVERT(NVARCHAR(MAX), [XMLDATA])
FROM [dbo].[OrdersSharing]
WHERE ID = 1

-- Crear tabla temporal para almacenar información que se extraiga del
XML.
CREATE TABLE #purchaseOrders
(
       [PONumber]     VARCHAR(10)
      ,[Date]         DATETIME
)

-- Se usa este comando para indicar que se accede a un XML.
EXEC sp_xml_preparedocument @xmlDocID OUTPUT, @i_XmlDoc

-- Se inserta información en la tabla temporal con un SELECT y OPENXML.
INSERT     #purchaseOrders
       (
              [PONumber], [Date]
       )

SELECT     [xml].[PONumber], [xml].[Date]
FROM       OPENXML(@xmlDocID, 'PurchaseOrder', 2) WITH
       (
              [PONumber]     VARCHAR(10)
              ,[Date]        DATETIME
       ) AS [xml]

-- Se remueve el documento XML que se ha utilizado.
EXEC sp_xml_removedocument @xmlDocID

-- Verificar si se obtuvieron datos del XML.
SELECT [PONumber], [Date] FROM #purchaseOrders

-- Eliminar tabla temporal
DROP TABLE #purchaseOrders
```

OPENXML en T-SQL.

Se utilizará el comando **OPENXML** y se leerá el XML que se insertó en la tabla “OrdersSharing”.

El *script* mostrado contiene comandos nuevos que permitirán generar un entendimiento completo.

	PONumber	Date
1	001001	2021-02-10 14:20:18.133

Ejecución del script de la tabla 11 OrderDetails en T-SQL.

Actividad

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

01 Cuando viaja información de un sistema a otro, ¿por qué es más eficiente que se envié/reciba en formato XML?

02 ¿En qué ocasiones es necesario transformar la información a XML?

Antes de comenzar a escribir comandos T-SQL para formar un **XML**, se recomienda un trabajo exhaustivo de análisis y diseño de la estructura que va a contener, con esto se evitarán trabajos extra a la hora de la creación del XML.

Es fundamental precisar los elementos, *tags*, atributos y valores, pues estos ambientes deberán estar en el mismo contexto para ahorrar tiempo en las pruebas al compartir información. Una vez definida la estructura, será tarea de los programadores T-SQL manipular este tipo de datos, tanto la creación (XML PATH), como la recepción (OPENXML).

Bibliografía

• MDN web docs. (2022). *Introducción a XML*. Recuperado de https://developer.mozilla.org/es/docs/Web/XML/XML_introduction

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Creación de objetos temporales

Semana 7

Introducción

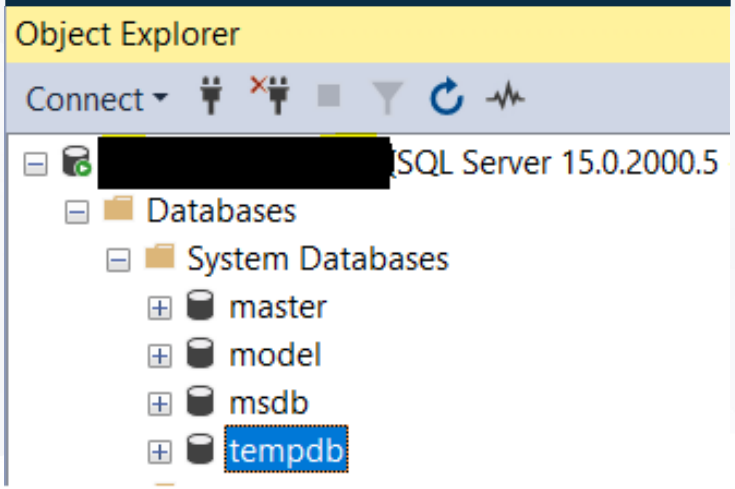
Dentro de los componentes tecnológicos, las bases de datos son el fundamento para las operaciones y a veces para la definición de las estrategias del negocio. Por lo que resulta necesario que las bases de datos estén en su óptimo funcionamiento para el negocio.

La creación y el manejo de objetos temporales dentro de SQL Server se vuelve una práctica importante de entender, ya que define, en gran parte, si las consultas en las bases de datos son eficientes o no.

Explicación

Base de datos tempdb

La base de datos **tempdb** es un recurso global que funciona para todos los usuarios y que almacena una diversidad de objetos temporales que se van creando, como índices, tablas, variables, procedimientos almacenados y resultados de consultas (Microsoft, 2022a).



Base de datos tempdb.

Universidad Tecmilenio

Explicación

Uso de tablas temporales

- Las tablas son los objetos más importantes y pertinentes de las bases de datos dentro de SQL Server.
- Las tablas dependen de otros elementos para su buen funcionamiento, como las vistas y los índices. En el caso de las tablas temporales, resulta ser lo mismo.
- Una **tabla temporal** es una tabla versionada que ayuda al usuario a conservar un histórico de los cambios y modificaciones de los datos para facilitar su análisis (Microsoft, 2022b).

Universidad Tecmilenio

Explicación

Una manera para consultar los datos temporales es a través de la cláusula **FOR SYSTEM_TIME**.

```
SELECT * FROM Trabajos
  FOR SYSTEM_TIME
    FROM  '2016-01-01  00:00:00.0000000'  TO  '2022-01-01
00:00:00.0000000'
      WHERE TrabajosID = 1000 Groupby TipoTrabajo;
```

Cláusula FOR SYSTEM_TIME.

Las **tablas temporales** locales son accesibles dentro de la misma sesión establecida en la base de datos, pero cuando se desconecta la sesión, se borran automáticamente los datos.

Explicación

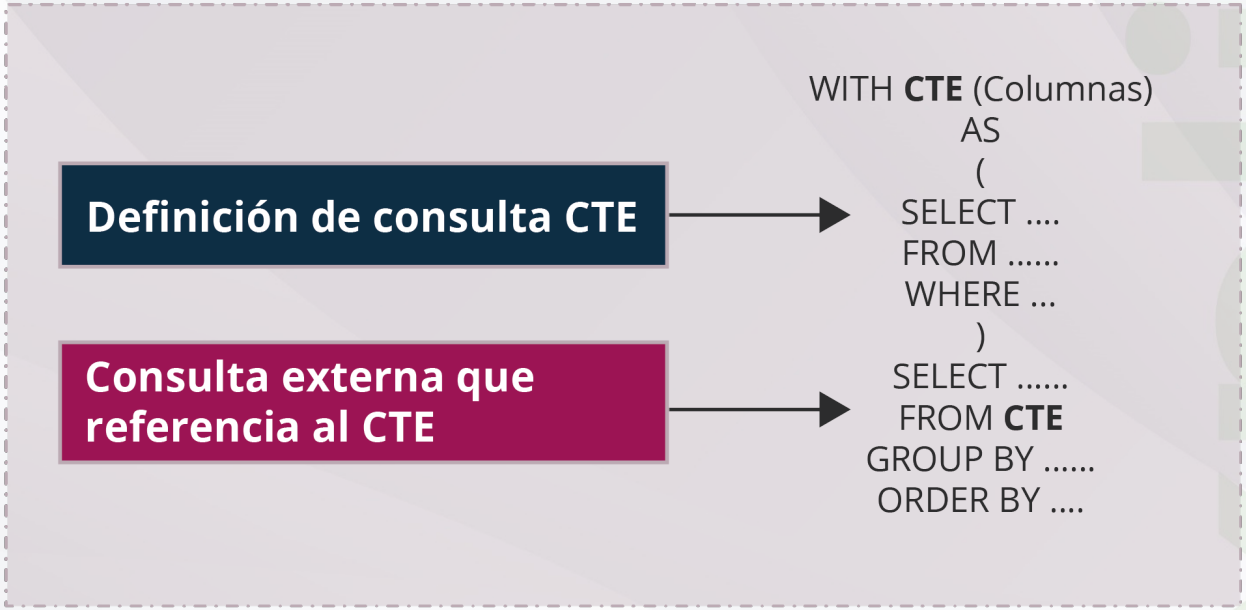
Uso de expresiones de tablas comunes

De acuerdo con Microsoft (2022c), **las expresiones de tablas comunes**, también conocidas con el acrónimo CTE (***common table expressions***), son un conjunto de resultados que tienen un nombre temporal. Estos resultados se obtienen a través de una subconsulta determinada.

Las expresiones de tablas comunes permiten lo siguiente:

- Que se realicen agrupaciones sobre datos derivados de operaciones escalares o no deterministas.
- Simplificar la lectura y escritura de las consultas.
- Sustituir y redefinir objetos temporales.
- Dan la habilidad de evaluar las subconsultas que se ejecutan varias veces dentro de la consulta padre.

Explicación



Consulta con CTE.

Universidad Tecmilenio

Explicación

Consultas sobre tablas derivadas

Las tablas derivadas son conjuntos de resultados que se usan como valor de entrada o como un origen de datos de una tabla para una consulta.

```
DECLARE @Productos AS TABLE (
idProducto INT,
 Dia1     DATE,
 Dia2      DATE,
Cualidad     VARCHAR (60))
```

Crear tabla.

```
INSERT INTO @Productos VALUES(1,'2018-05-12','2018-05-13','Extensible')
INSERT INTO @Productos VALUES(2,'2018-05-12','2018-05-13','Fibroso')
INSERT INTO @Productos VALUES(3,'2018-05-12','2018-05-13','Fragil')
INSERT INTO @Productos VALUES(4,'2018-05-12','2018-05-13','Alcalino')
INSERT INTO @Productos VALUES(5,'2018-05-12','2018-05-13','Rectangular')
```

Insertar datos en la tabla.

```
SELECT * FROM
(SELECT IdProducto,Cualidad FROM @Productos WHERE
Cualidad = 'Fragil'
) AS [Cualidad Fragil]
WHERE idProducto = 1 OR idProducto = 2
```

Consulta con base en la tabla derivada.

Actividad

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

01 ¿Es recomendable usar objetos temporales en una base de datos?, ¿por qué?

02 ¿Qué aspectos determinan la decisión de utilizar objetos temporales?

Universidad Tecmilenio

Cierre

El tema de la temporalidad de las bases de datos resulta relevante para todo profesionista que quiera llevar sus habilidades técnicas usando el lenguaje T-SQL al siguiente nivel .

Sin embargo, gran parte del trabajo necesario para hacer realidad estas técnicas y/o reglas de T-SQL no se encuentra únicamente en memorizar y aplicar la sintaxis correspondiente, sino en encontrar esa lógica de SQL y empatarla con la nuestra.

Bibliografía

• Microsoft. (2022a). *tempdb database*. Recuperado de https://docs.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database?view=sql-server-ver16

• Microsoft. (2022b). *Tablas temporales*. Recuperado de https://docs.microsoft.com/es-es/sql/relational-databases/tables/temporal-tables?view=sql-server-ver16

• Microsoft. (2022c). *WITH common_table_expression (Transact-SQL)*. Recuperado de https://docs.microsoft.com/en-us/sql/t-sql/queries/with-common-table-expression-transact-sql?view=sql-server-ver16

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Combinación de resultados por medio de operadores de conjuntos

Semana 7

Actualmente se están impulsando ciertas transformaciones que orillan a las organizaciones a convertirse en empresas basadas en datos (*data-driven companies*).

Es decir, las organizaciones, sin importar su giro, deben propiciar la tendencia a tomar decisiones basadas en datos propios.

Resulta fundamental, no solo manejar las bases de datos, sino ser realmente precisos en conocer y saber aplicar todas aquellas cláusulas y mecanismos que lograrán hacer más perspicaces los resultados que se consultan.

Universidad Tecmilenio

Explicación

Uso del operador UNION

La cláusula **UNION** se usa para concatenar resultados de dos consultas en una sola exhibición de resultados (Microsoft, 2022a).

Condiciones:

- Cada vez que se use la cláusula **UNION** en una sentencia **SELECT**, se debe contar con la misma cantidad de columnas.
- Las columnas deben tener tipos de datos similares.
- Las columnas deben estar en el mismo orden.

Explicación

```
SELECT columnas FROM tabla1
UNION
SELECT columnas FROM tabla2;
```

Sintaxis para el uso del operador UNION.

Diferencia entre UNION y UNION ALL.

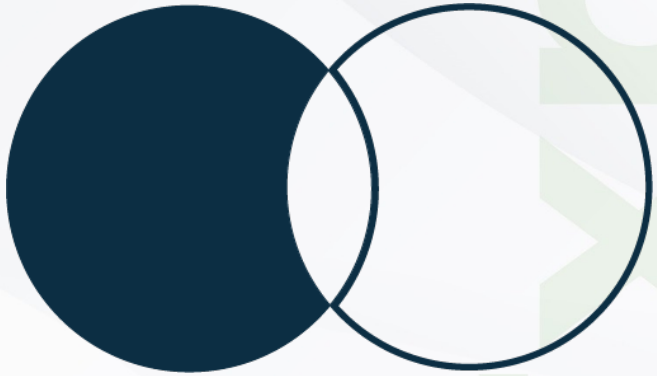
Explicación

Uso de los operadores INTERSECT y EXCEPT

El caso operador **EXCEPT** tiene la funcionalidad de regresar como resultado las filas que son diferentes con base en dos consultas separadas (Microsoft, 2022b).

```
SELECT columnas FROM tabla1
EXCEPT
SELECT columnas FROM tabla2
```

Sintaxis de la cláusula EXCEPT.



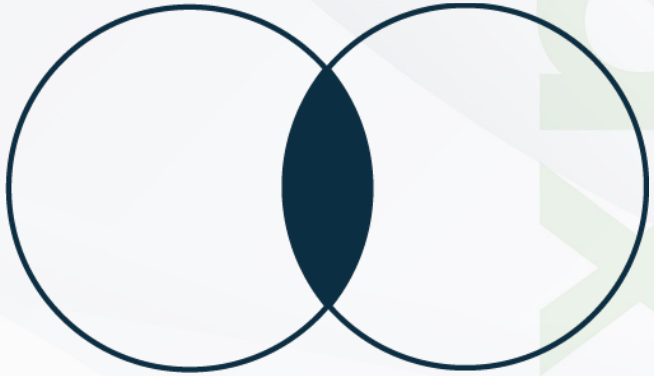
Ejemplo de la cláusula EXCEPT.

Explicación

La cláusula **INTERSECT** regresa aquellos datos que son iguales tanto en una tabla como en la otra.

```
SELECT Tipo FROM Plasticos
INTERSECT
SELECT Tipo FROM Maderas;
```

Aplicación de la cláusula INTERSECT en las tablas “Plásticos” y “Maderas”.



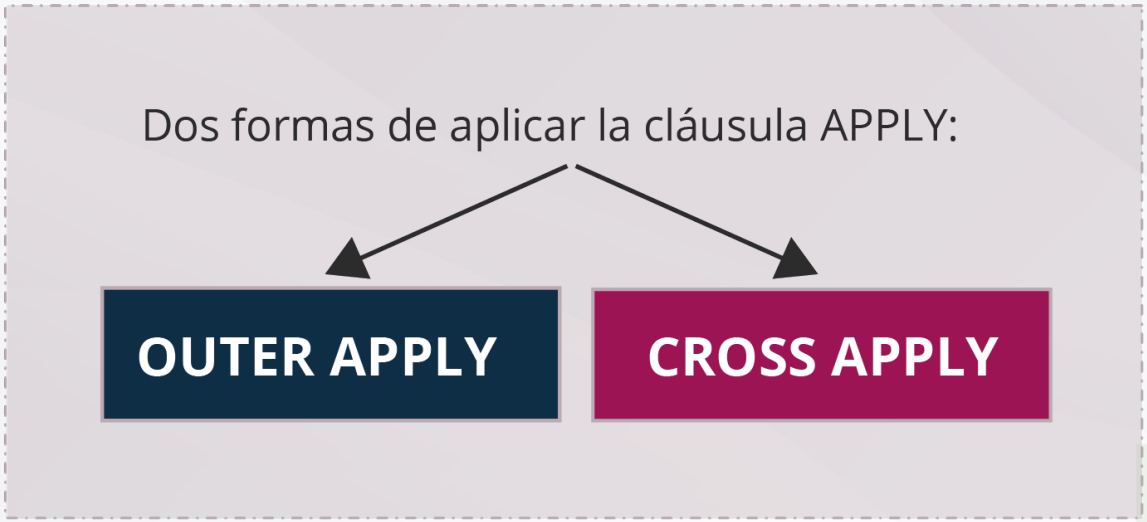
Ejemplo de la cláusula INTERSECT.

Explicación

Uso del operador APPLY

De acuerdo con Microsoft (2022c), el operador **APPLY** se usa para lógicas de concatenación y unión.

En otras palabras, se usa para unir dos expresiones de una tabla. La expresión de la primera tabla se procesa cada vez para cada fila de la expresión de la segunda tabla.



Actividad

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

01 ¿Cuáles son las diferencias entre UNION y UNION ALL?

02 ¿Bajo que circunstancias aplicarías el comando APPLY?

Es muy importante comprender que las diferentes cláusulas vistas en este tema tienen mucho parentesco en lógica con el funcionamiento que ofrece la cláusula JOIN.

Sin embargo, una similitud que vemos entre **UNION, EXCEPT** e **INTERSECT** es que estas cláusulas están diseñadas para propósitos de desplegar resultados de tablas completas y similares en características, sin la necesidad de crear filas ni hacer cálculos complejos.

En el caso del operador **APPLY**, este sí funciona como un **JOIN**, pero es únicamente utilizado para funciones con valores de salida, ya que, si se crea una función y se le aplica un JOIN, el sistema mostraría un error.

Bibliografía

• Microsoft. (2022a). *Set Operators - UNION (Transact-SQL)*. Recuperado de https://docs.microsoft.com/en-us/sql/t-sql/language-elements/set-operators-union-transact-sql?view=sql-server-ver16

• Microsoft. (2022b). *Set Operators - EXCEPT and INTERSECT (Transact-SQL)*. Recuperado de https://docs.microsoft.com/en-us/sql/t-sql/language-elements/set-operators-except-and-intersect-transact-sql?view=sql-server-ver16

• Microsoft. (2022c). *FROM clause plus JOIN, APPLY, PIVOT (Transact-SQL)*. Recuperado de https://docs.microsoft.com/en-us/sql/t-sql/queries/from-transact-sql?view=sql-server-ver16