

ESPECIFICACIÓN DE DATOS SQL SERVER

Hola bienvenidos a una nueva sesión de la especialidad en Microsoft SQL Server de sistemas del sur, en esta sesión veremos los tipos de datos o datatypes que son más frecuentes a la hora de utilizar una base de datos SQL.

Comenzamos preguntándonos

¿Qué son los datatypes?

Datatype es el tipo de dato que va a recibir una columna en una tabla de una base de datos

Recuerda que cuando creamos una tabla el sistema nos pide los nombres de la columna y el tipo de dato que ese campo o columna va a recibir, entonces si ya sabemos que son los datatypes, vamos a conocer unos cuantos de ellos.

Existen muchísimos, pero vamos a ver los más utilizados, para que cuando usted este creando su base de datos, maneje los tipos de datos comunes y que no se complique al usarlo

Tipo de dato

1. Binary

Almacena y acepta solo números binarios 50 bits (tiene que ser exacta la data) si excede el límite, el sistema va a dar un error, recuerda que pide una cantidad exacta.

2. Char

Es similar a binary pero este almacena letras rango: 1 a 8000 caracteres (almacena caracteres) es solo texto y también es cantidad exacta (10) caracteres tiene que haber 10 caracteres (no se puede realizar operaciones aritméticas) tenemos ejemplo como el dni que tiene 8 dígitos, entonces no se puede ingresar más ni menos de 8 dígitos. Ahora, si quieres un campo con un tipo de dato que permita ingresar menos cantidad de dígitos, tenemos los tipos de datos.

3. Varchar

En este tipo de dato se limitan la cantidad de caracteres que puede recibir, pero puede ser menos caracteres del máximo establecido. Un Ejemplo: tengo un campo llamado descripción y el límite es de (50), si ingresas 20 caracteres, este te va a registrar sin algún problema.

4. Varbinary

Al igual que varchar se limitan la cantidad máxima de bits pero tienes ese campo puede llenarse con menos cantidad de bits del máximo establecido.

Ahora, tenemos los tipos de dato datetime y datetime 2.

5. Datetime

Permite almacenar las fechas en un rango de 1/1/1753 a 31/12/9999.

6. Datetime2

Permite almacenar las fechas desde el 1/1/0001 hasta 31/12/9999.

7. Image

Almacena una gran cantidad de datos variables en formato binario su máximo de almacenamiento es de 2 GB

Ahora tenemos 2 de los tipos de datos que va a estar presente en casi todas las tablas, y estas nos van a servir comúnmente como su identificador único, el cual nos servirá para saber que numero de registro estamos utilizando. También sirve para enumerar la cantidad de ítems que tenemos por ejemplo de un producto. Ahora tenemos 2 de los tipos de datos que va a estar presente en casi todas las tablas, y estas nos van a servir comúnmente como su identificador único, el cual nos servirá para saber que numero de registro estamos utilizando. Además, sirve para enumerar la cantidad de ítems que tenemos por ejemplo de un producto.

8. Int

Maneja tipo de datos enteros, de: -2³¹ hasta 2³¹-1

9. Bigint

Permite guardar datos de tipo enteros pero con un rango mucho mayor al Int que sería: -9²²³ hasta 9²²³

Recuerda que si ingresas un número que exceda el rango establecido por el tipo de dato, no te va a permitir registrar en la base de datos.

10. Money

Permite almacenar datos de tipos monetarios y tiene un rango de

-922'337'203'685'477,5808 hasta 922'337'203'685'477,5807

Si excedes esa cantidad de montos, no vas a poder registrar dentro de este tipo de dato

11. Numeric y Decimal

Estos tipos de datos almacena datos de tipo decimal, pero bajo 2 criterios que son:

- Precisión ->que es la cantidad de números incluyendo los decimales
- Escala ->que establece el número de decimales que contiene el tipo de dato

Ejemplo: decimal(5,2) 123,45

Ahora veremos los tipos de datos small

12. Smalldatetime

Almacena datos de tipo fecha, pero el rango reducido que es de 1/1/1900 – 6/6/2079

13. Smallint

Almacena datos de tipo enteros pero el rango es de -32,768 hasta 32,767

14. Smallmoney

Almacena datos de tipo money pero el rango es de -214'748.3648 hasta 214'748.3648

15. Tinyint

solo permite numeros positivos desde el 0 hasta 255

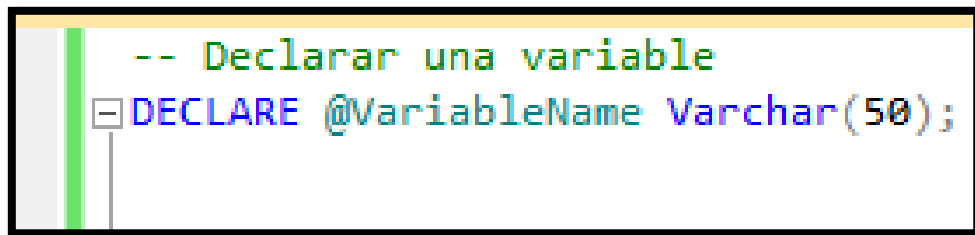
Ahora que conoces las especificaciones de estos tipos de datos, vas a darte cuenta de los limites que establezcas a cada campo de una tabla dentro de una base de datos.

VARIABLES Y TIPOS DE DATOS EN SQL SERVER

En SQL Server, cada columna, variable local, expresión y parámetro tiene un tipo de datos relacionado. Un tipo de datos es un atributo que especifica el tipo de datos que el objeto puede contener: datos de enteros, datos de caracteres, datos de moneda, datos de fecha y hora, cadenas binarias, etc.

SQL Server proporciona un conjunto de tipos de datos del sistema que define todos los tipos de datos que pueden utilizarse con SQL Server. También puede definir sus propios tipos de datos en Transact-SQL o Microsoft .NET Framework.

Recuerda que para declarar una variable es así:

A screenshot of a SQL Server code editor window. The window has a yellow title bar and a black border. Inside, there is a code editor with a light gray background. On the left side of the editor, there is a vertical green bar and a small square icon. The code being edited is:

```
-- Declarar una variable  
DECLARE @VariableName Varchar(50);
```

Hay reglas y sugerencias que se va a utilizar para los nombres :

- Un nombre puede empezar con un guion o una carta.

Algunos ejemplos son @_n, @act, o la @Segundo

- Después de que el primer carácter como un carácter de subrayado o una letra, el nombre tendrá combinaciones de relieve, las letras y dígitos.

Algunos ejemplos: @_n24 ó @act_52_t

Eso sí:

- Un nombre no incluir caracteres especiales como!, @, #, \$, %, ^, &, O *
 - Si el nombre es una combinación de palabras, cada palabra comenzará en mayúsculas. Algunos ejemplos: @FechaContratación, @_JuegoReal, o @NumeroLicencia
-

```

DECLARE @_n Varchar(50);
DECLARE @act Varchar(50);
DECLARE @Segundo Varchar(50);

DECLARE @_n24 Varchar(50);
DECLARE @act_52_t Varchar(50);

DECLARE @FechaContratación Date;
DECLARE @_JuegoReal INT;
DECLARE @NumeroLicencia Varchar(50);

```

Inicializar una variable

Después de declarar una variable, el intérprete se reserva un espacio en la memoria del ordenador, pero el espacio no necesariamente tiene un valor reconocible. Esto significa que, en este momento, la variable es nula. Una manera de cambiar esto es para darle un valor a la variable.

Esto se conoce como la inicialización de la variable.

Recuerde que el nombre de una variable comienza con @ y siempre que lo necesite para hacer referencia a la variable, debe asegurarse de incluir el símbolo @.

Para inicializar una variable, en la sección es necesario, escriba la instrucción SELECT o la palabra clave SET seguida por el nombre de la variable, seguido por el operador de asignación "=", seguido por un valor adecuado.

La fórmula utilizada es:

```

-- Inicializar una variable
DECLARE @NombreVariable Varchar(50) ;

SELECT @NombreVariable = 'Inicializar';
SET @NombreVariable = 'Inicializar' ;

```

Un nombre de variable no puede ser igual a las palabras reservadas de Transact-SQL que son palabras de clave interna

ADD	CONTAINSTABLE	ESCAPE	INNER
ALL	CONTINUE	EXCEPT	INSERT
ALTER	CONVERT	EXEC	INTERSECT
AND	CREATE	EXECUTE	INTO
ANY	CROSS	EXISTS	IS
AS	CURRENT	EXIT	JOIN
ASC	CURRENT_DATE	EXTERNAL	KEY
AUTHORIZATION	CURRENT_TIME	FETCH	KILL
BACKUP	CURRENT_TIMESTAMP	FILE	LEFT
BEGIN	CURRENT_USER	FILLFACTOR	LIKE
BETWEEN	CURSOR	FOR	LINENO
BREAK	DATABASE	FOREIGN	LOAD
BROWSE	DBCC	FREETEXT	MERGE
BULK	DEALLOCATE	FREETEXTTABLE	NATIONAL
BY	DECLARE	FROM	NOCHECK
CASCADE	DEFAULT	FULL	NONCLUSTERED
CASE	DELETE	FUNCTION	NOT
CHECK	DENY	GOTO	NULL
CHECKPOINT	DESC	GRANT	NULLIF
CLOSE	DISK	GROUP	OF
CLUSTERED	DISTINCT	HAVING	OFF
COALESCE	DISTRIBUTED	HOLDLOCK	OFFSETS
COLLATE	DOUBLE	IDENTITY	ON
COLUMN	DROP	IDENTITY_INSERT	OPEN
COMMIT	DUMP	IDENTITYCOL	OPENDATASOURCE
COMPUTE	ELSE	IF	OPENQUERY
CONSTRAINT	END	IN	OPENROWSET
CONTAINS	ERRLVL	INDEX	OPENXML

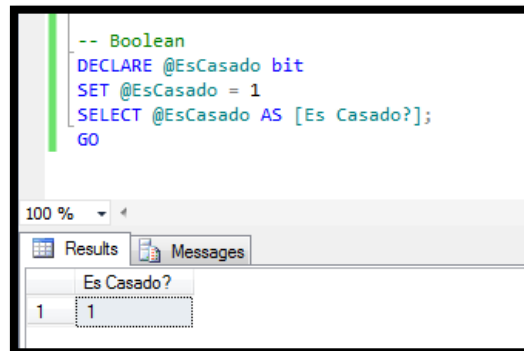
OPTION	ROWGUIDCOL	UNION
OR	RULE	UNIQUE
ORDER	SAVE	UNPIVOT
OUTER	SCHEMA	UPDATE
OVER	SECURITYAUDIT	UPDATETEXT
PERCENT	SELECT	USE
PIVOT	SEMANTICKEYPHRASETABLE	USER
PLAN	SEMANTICSIMILARITYDETAILSTABLE	VALUES
PRECISION	SEMANTICSIMILARITYTABLE	VARYING
PRIMARY	SESSION_USER	VIEW
PRINT	SET	WAITFOR
PROC	SETUSER	WHEN
PROCEDURE	SHUTDOWN	WHERE
PUBLIC	SOME	WHILE
RAISERROR	STATISTICS	WITH
READ	SYSTEM_USER	WITHIN GROUP
READTEXT	TABLE	WRITETEXT
RECONFIGURE	TABLESAMPLE	
REFERENCES	TEXTSIZE	
REPLICATION	THEN	
RESTORE	TO	
RESTRICT	TOP	
RETURN	TRAN	
REVERT	TRANSACTION	
REVOKE	TRIGGER	
RIGHT	TRUNCATE	
ROLLBACK	TRY_CONVERT	
ROWCOUNT	TSEQUAL	

Más sobre tipos de datos

Números Exactos

Boolean (bit)

Tipo de datos entero que puede aceptar los valores 1, 0 o NULL



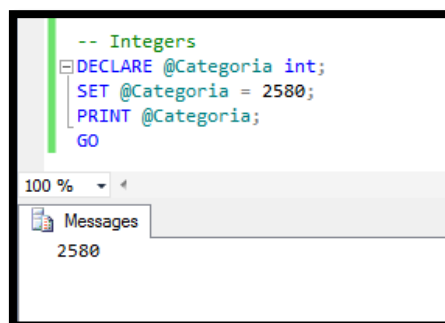
```
-- Boolean
DECLARE @EsCasado bit
SET @EsCasado = 1
SELECT @EsCasado AS [Es Casado?];
GO
```

Es Casado?
1

Integers (int)

Tipos de datos numéricos exactos que utilizan datos enteros

El rango de -2.147.483.648 a 2.147.483.647, se puede declarar con la palabra clave int como tipo de datos.

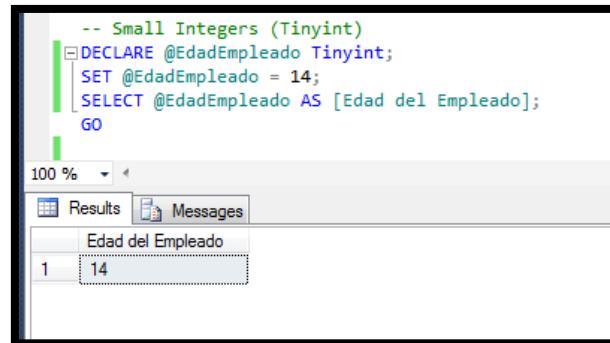


```
-- Integers
DECLARE @Categoria int;
SET @Categoria = 2580;
PRINT @Categoria;
GO
```

2580

Small Integers (Tinyint)

Si deseas utilizar números muy pequeños, como las edades del estudiante, o el número de páginas de un folleto o periódico, utilizar el tipo de datos tinyint. Una variable con el tipo de datos tinyint puede contener números positivos que van de 0 a 255.

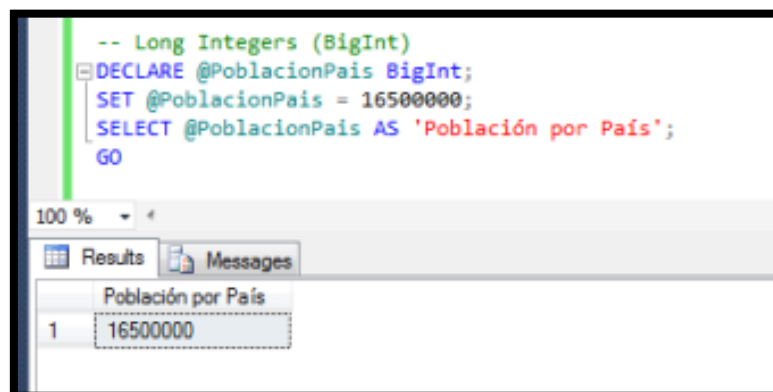


```
-- Small Integers (Tinyint)
DECLARE @EdadEmpleado Tinyint;
SET @EdadEmpleado = 14;
SELECT @EdadEmpleado AS [Edad del Empleado];
GO
```

	Edad del Empleado
1	14

Long Integers (BigInt)

El tipo de datos bigint sigue las mismas reglas y principios como el tipo de datos int, excepto que puede contener un gran número de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807.



```
-- Long Integers (BigInt)
DECLARE @PoblacionPais BigInt;
SET @PoblacionPais = 16500000;
SELECT @PoblacionPais AS 'Población por País';
GO
```

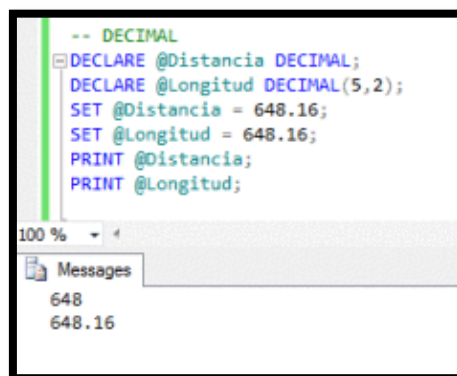
	Población por País
1	16500000

Decimal

Son tipos de datos numéricos que tienen precisión y escala fijas.

El número total máximo de dígitos decimales que se puede almacenar, tanto a la izquierda como a la derecha del separador decimal. El número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal.

Precisión	Bytes de Almacenamiento
1-9	5
10-19	9
20-28	13
29-38	17



```
-- DECIMAL
DECLARE @Distancia DECIMAL;
DECLARE @Longitud DECIMAL(5,2);
SET @Distancia = 648.16;
SET @Longitud = 648.16;
PRINT @Distancia;
PRINT @Longitud;
```

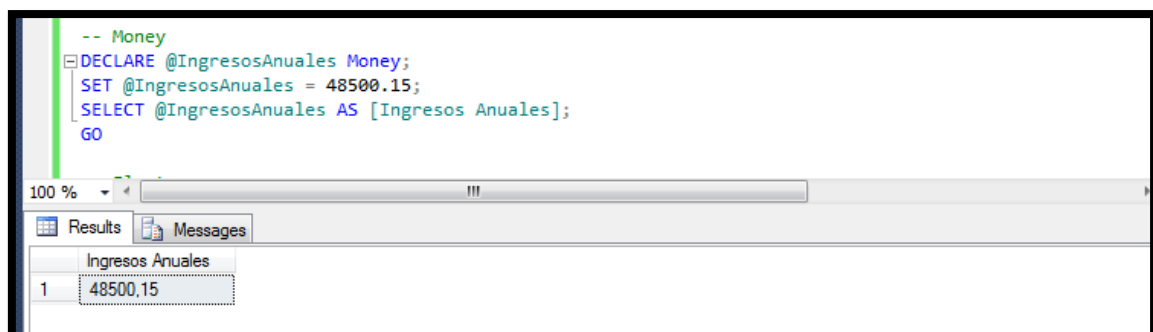
Messages

648
648.16

Money - smallmoney

Tipos de datos que representan valores monetarios o de moneda.

Tipos de datos	Intervalo	Almacenamiento
Money	De -922'337'203'685'477,2808	8 bytes
Smallmoney	De -214'748,3548 a 214'748,3647	4 bytes



```
-- Money
DECLARE @IngresosAnuales Money;
SET @IngresosAnuales = 48500.15;
SELECT @IngresosAnuales AS [Ingresos Anuales];
GO
```

Results

	Ingresos Anuales
1	48500.15

[illegible]

Cadenas de caracteres – Cadenas de caracteres Unicode

Char – Varchar

Un campo de carácter de longitud fija, puede consistir en cualquier tipo de símbolos alfabéticos legible o no.

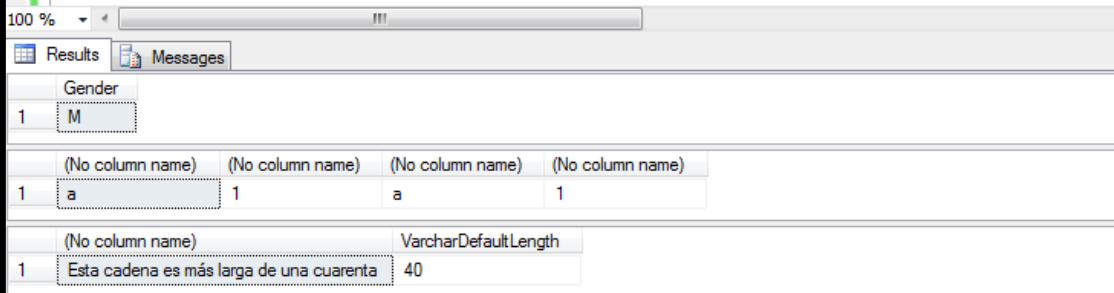
char (n) , n define la longitud de la cadena y debe ser un valor entre 1 y 8.000.

varchar [(n | max)], n define la longitud de la cadena y puede ser un valor entre 1 y 8.000. max indica que el tamaño máximo de almacenamiento es de 2³¹-1 bytes (2 GB)

```
-- Char - Varchar
DECLARE @Genero char;
SET @Genero = 'M';
SELECT @Genero AS Gender;
GO

-- el valor predeterminado de n es 1 para los tipos de datos char y varchar cuando se
-- utilizan en una declaración de variable.
DECLARE @VariableVarChar AS varchar = 'abc';
DECLARE @VariableChar AS char = 'abc';
-- El Resultado es 1
SELECT @VariableVarChar, LEN(@VariableVarChar), @VariableChar, LEN(@VariableChar);
GO

DECLARE @myVariable AS varchar(40);
SET @myVariable = 'Esta cadena es más larga de una cuarenta caracteres';
SELECT @myVariable, LEN(@myVariable) AS 'VarcharDefaultLength';
```



The screenshot displays the results of three SQL queries in SQL Server Enterprise Manager. The first query shows the value 'M' for the variable @Genero. The second query shows the values 'abc' and 1 for the variables @VariableVarChar and @VariableChar. The third query shows the value 'Esta cadena es más larga de una cuarenta caracteres' and 40 for the variable @myVariable.

Gender
1 M

(No column name)	(No column name)	(No column name)	(No column name)
1 a	1	a	1

(No column name)	VarcharDefaultLength
1 Esta cadena es más larga de una cuarenta	40

nchar y nvarchar

Este tipo de datos pueden almacenar caracteres Unicode.

sysname es un tipo de datos definido por el usuario y suministrado por el sistema que es funcionalmente equivalente a nvarchar(128), excepto que no acepta valores NULL. sysname se usa para hacer referencia a nombres de objetos de base de datos.

```
-- nchar y nvarchar
DECLARE @PruebaSysname AS sysname;
DECLARE @PruebaSysnameMayor128 AS sysname;
DECLARE @PruebaSysnamenchar AS nchar(100);
SET @PruebaSysname = 'sysname es un tipo de datos definido por el usuario';
SET @PruebaSysnameMayor128 = 'sysname es un tipo de datos definido por el usuario y suministrado por el sistema que es funcionalmente equivalente a nvarchar(128)';
SET @PruebaSysnamenchar = 'sysname es un tipo de datos definido por el usuario y suministrado por el sistema que es funcionalmente equivalente a nchar(100)';
SELECT @PruebaSysname, DATALENGTH(@PruebaSysname) AS 'bytes utilizados', LEN(@PruebaSysname) AS 'caracteres';
SELECT @PruebaSysnameMayor128, DATALENGTH(@PruebaSysnameMayor128) AS 'bytes utilizados', LEN(@PruebaSysnameMayor128) AS 'caracteres';
SELECT @PruebaSysnamenchar, DATALENGTH(@PruebaSysnamenchar) AS 'bytes utilizados', LEN(@PruebaSysnamenchar) AS 'caracteres';
```

The screenshot shows the results of the SQL queries in SQL Server Enterprise Manager. The 'Results' tab is active, displaying three tables of data. Each table has columns for the variable name, 'bytes utilizados' (bytes used), and 'caracteres' (characters).

(No column name)	bytes utilizados	caracteres
1 sysname es un tipo de datos definido por el usuario	102	51

(No column name)	bytes utilizados	caracteres
1 sysname es un tipo de datos definido por el usuario y suministrado por el sistema que es funcionalmente equivalente a nvarchar(128)	256	128

(No column name)	bytes utilizados	caracteres
1 sysname es un tipo de datos definido por el usuario y suministrado por el sistema que es funcionalmente equivalente a nchar(100)	200	100

Todas las respuestas hasta ahora indican que varchar es de un solo byte, nvarchar es de doble byte.

```
-- nchar y nvarchar
DECLARE @Pruebas TABLE
(
    C1 VARCHAR(20) COLLATE Chinese_Traditional_Stroke_Order_100_CS_AS_KS_WS,
    C2 NVARCHAR(20) COLLATE Chinese_Traditional_Stroke_Order_100_CS_AS_KS_WS
);
INSERT INTO @Pruebas VALUES (N'中华人民共和国', N'中华人民共和国');
SELECT LEN(C1) AS [LEN (C1)],
       DATALENGTH(C1) AS [DATALENGTH (C1)],
       C1,
       LEN(C2) AS [LEN (C2)],
       DATALENGTH(C2) AS [DATALENGTH (C2)],
       C2
FROM @Pruebas
```

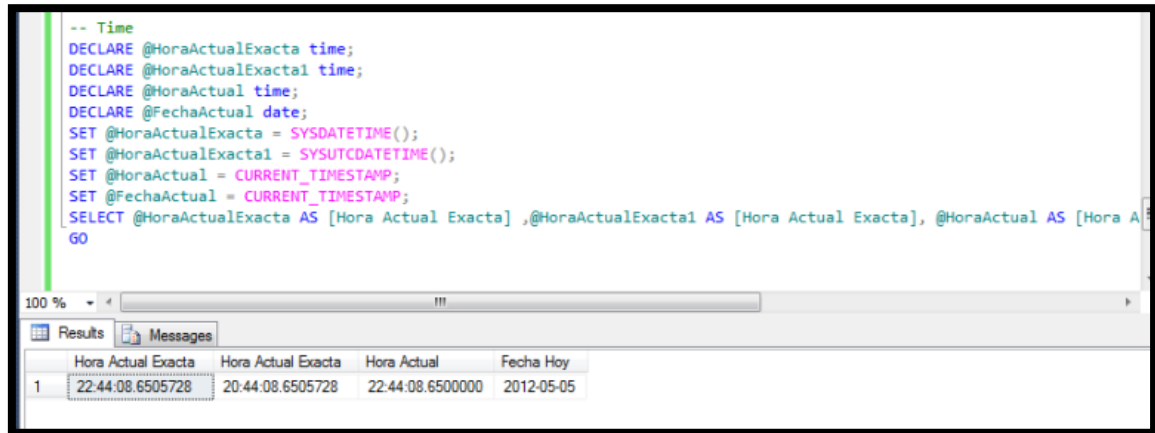
The screenshot shows the results of the SQL queries in SQL Server Enterprise Manager. The 'Results' tab is active, displaying a table with columns for 'LEN (C1)', 'DATALENGTH (C1)', 'C1', 'LEN (C2)', 'DATALENGTH (C2)', and 'C2'.

LEN (C1)	DATALENGTH (C1)	C1	LEN (C2)	DATALENGTH (C2)	C2
7	12	中?人民共和?	7	14	中华人民共和国

Fechas

Time

Define una hora de un día. La hora no distingue la zona horaria y está basada en un reloj de 24 horas.



```
-- Time
DECLARE @HoraActualExacta time;
DECLARE @HoraActualExacta1 time;
DECLARE @HoraActual time;
DECLARE @FechaActual date;
SET @HoraActualExacta = SYSDATETIME();
SET @HoraActualExacta1 = SYSUTCDATETIME();
SET @HoraActual = CURRENT_TIMESTAMP;
SET @FechaActual = CURRENT_TIMESTAMP;
SELECT @HoraActualExacta AS [Hora Actual Exacta] ,@HoraActualExacta1 AS [Hora Actual Exacta], @HoraActual AS [Hora Actual] ,@FechaActual AS [Fecha Actual]
```

	Hora Actual Exacta	Hora Actual Exacta	Hora Actual	Fecha Hoy
1	22:44:08.6505728	20:44:08.6505728	22:44:08.6500000	2012-05-05

SYSDATETIME y **SYSUTCDATE** tienen más precisión de fracciones de segundo que **GETDATE** y **GETUTCDATE**. **SYSDATETIMEOFFSET** incluye el ajuste de zona horaria del sistema.

SYSDATETIME, **SYSUTCDATE** y **SYSDATETIMEOFFSET** pueden asignarse a una variable de cualquier tipo de fecha y hora.

Date

Define una fecha.

Para inicializar una variable de fecha, utilice uno de las siguientes fórmulas:

Descripcion	Formato	Ejemplo
Año Mes Día	YYYYMMDD	20201201
Año-Mes-Día	YYYY-MM-DD	2020-12-01
Mes-Día-Año	MM-DD-YY	12-01-20
Mes-Día-Año	MM-DD-YYYY	12-01-2020
Mes/Día/Año	MM/DD/YY	12/01/20
Mes/Día/Año	MM/DD/YYYY	12/01/2020

```
-- Date
DECLARE @DiaUno Date;
SET @DiaUno = N'10360610';
SELECT @DiaUno AS [Día uno];

DECLARE @date Date = '12-10-25';
DECLARE @datetime datetime = @date;
SELECT @date AS '@date', @datetime AS '@datetime';
GO
```

Results	
Día uno	
1	1036-06-10

	@date	@datetime
1	2025-12-10	2025-12-10 00:00:00.000

En el ejemplo siguiente se comparan los resultados de los distintos tipos de datos relacionados con tipos de datos de fecha y hora

```
DECLARE @time time = CURRENT_TIMESTAMP;
DECLARE @date date = CURRENT_TIMESTAMP;
DECLARE @smalldatetime smalldatetime = CURRENT_TIMESTAMP;
DECLARE @datetime datetime = CURRENT_TIMESTAMP;
DECLARE @datetime2 datetime2 = CURRENT_TIMESTAMP;
DECLARE @datetimeoffset datetimeoffset = CURRENT_TIMESTAMP;
SELECT @time, @date, @smalldatetime, @datetime, @datetime2, @datetimeoffset;
GO
```

	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)
1	23:25:30.0470000	2012-05-05	2012-05-05 23:26:00	2012-05-05 23:25:30.047	2012-05-05 23:25:30.0470000	2012-05-05 23:25:30.0470000 +00:00

Otros tipos de datos

sql_variant

Transact-SQL proporciona el tipo de datos sql_variant.

Este se puede utilizar en lugar de cualquiera de los tipos de datos que hemos visto hasta ahora. Esto significa que usted puede declarar su variable como cualquier otra. Cuando se inicializa la variable, se debe utilizar el formato adecuado, si se trata de una cadena, un número o una fecha, etc.

sql_variant puede usarse en columnas, parámetros, variables y valores devueltos de funciones definidas por el usuario. sql_variant permite que estos objetos de base de datos admitan valores de otros tipos de datos.

```
-- SQL Variant
DECLARE @NombreCompleto SQL_VARIANT, @Fecha sql_variant, @EsCasado SQL_variant, @SueldoAnual sql_variant;
SET @NombreCompleto = N'Ricardo Huamán Suárez';
SET @Fecha = N'19760412';
SET @EsCasado = 1;
SET @SueldoAnual = 48500.15;
SELECT @NombreCompleto, @Fecha, @EsCasado, @SueldoAnual ;
GO
```

	(No column name)	(No column name)	(No column name)	(No column name)
1	Ricardo Huamán Suárez	19760412	1	48500.15

```
DECLARE @Pruebas TABLE
( C1 VARCHAR(MAX), C2 SQL_VARIANT)

INSERT INTO @Pruebas VALUES ('Texto', N'Ricardo Huamán Suárez')
INSERT INTO @Pruebas VALUES ('Fecha', N'19760412')
INSERT INTO @Pruebas VALUES ('Bit', 1)
INSERT INTO @Pruebas VALUES ('Decimal', 48500.15)
SELECT * FROM @Pruebas
```

	C1	C2
1	Texto	Ricardo Huamán Suárez
2	Fecha	19760412
3	Bit	1
4	Decimal	48500.15

Tipos de datos definidos por el usuario

Si usted ha programado en lenguajes como C / C + + o Pascal, probablemente está familiarizado con la capacidad de definir un propio tipo de dato.

Transact-SQL también le da esta opción, (UDT) es una técnica de creación de un tipo de datos basada en una existente de Transact-SQL.

