TUIA
BASES DE DATOS I – 1SEM2025 – COM R1

- Docentes:
- Fernando Roldan
- Cristian Gallo
- Lara Della Ceca

- FCEIA-UNR

TUIA - BASES DE DATOS I
JOIN

Combinar campos de 2 tablas:

Ejemplo BD Northwind

```
SELECT *
FROM Customers C, Orders O
WHERE C.CustomerID = O.CustomerID
```

TUIA - BASES DE DATOS I

JOIN

JOIN

La cláusula JOIN **nos permite combinar 2 o más tablas, en base a una columna** que tengan en común

TUIA - BASES DE DATOS I
JOIN

JOIN

Veremos 2 tipos combinaciones, **internas** y **externas**

TUIA - BASES DE DATOS I
JOIN

El ejemplo visto antes es una forma de escribir una consulta de combinación interna en la sintaxis de SQL SERVER.

Hay otra forma de escribirla que corresponde al estándar ANSI y es más habitual,

TUIA - BASES DE DATOS I
JOIN

Especificar las **condiciones** de la combinación en la cláusula **FROM** ayuda a separarlas de cualquier otra condición de búsqueda que se pueda especificar en una cláusula **WHERE**; es el método recomendado para especificar combinaciones.

La sintaxis simplificada de combinación de la cláusula **FROM** de ISO es:

TUIA - BASES DE DATOS I
JOIN

FROM **first_table** < join_type > **second_table** [ON (join_condition)]

TUIA - BASES DE DATOS I
JOIN

INNER JOIN – combinación interna

FROM first_table < join_type > second_table [ON (join_condition)]

```
SELECT *
FROM Customers C
INNER JOIN Orders O ON C.CustomerID = O.CustomerID
```

TUIA - BASES DE DATOS I
JOIN

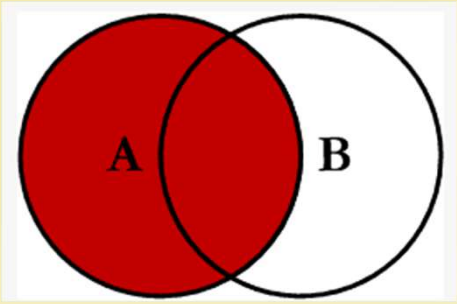
INNER JOIN – combinación interna

Realiza el producto cartesiano y luego filtra según la cláusula ON

TUIA - BASES DE DATOS I
JOIN

LEFT JOIN – combinación externa izquierda

- Combina todos los registros de la **primer** tabla con los de la **segunda** que cumplan la condición.
- Cuando no haya coincidencia en la segunda tabla se completa con NULOS



TUIA - BASES DE DATOS I
JOIN

LEFT JOIN – combinación externa izquierda

```
SELECT *
FROM Customers C
LEFT JOIN Orders O
ON C.CustomerID = O.CustomerID
```

TUIA - BASES DE DATOS I
JOIN

LEFT JOIN – combinación externa izquierda

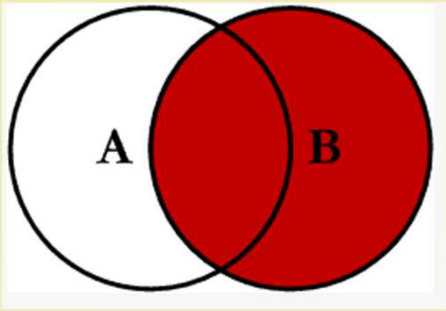
OUTER es opcional en la clausula

```
SELECT *
FROM Customers C
LEFT OUTER JOIN Orders O
ON C.CustomerID = O.CustomerID
```

TUIA - BASES DE DATOS I
JOIN

RIGHT JOIN – combinación externa derecha

- Combina todos los registros de la **segunda** tabla con los de la **primera** que cumplan la condición.
- Cuando no haya coincidencia en la primera tabla se completa con NULOS



TUIA - BASES DE DATOS I
JOIN

RIGHT JOIN – combinación externa derecha

OUTER es opcional en la clausula

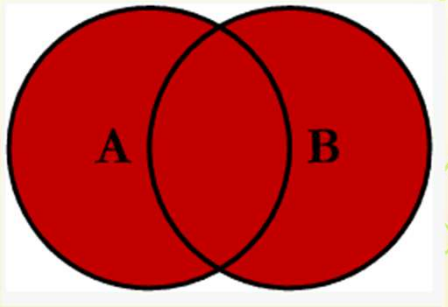
```
SELECT *
FROM Orders O
RIGHT OUTER JOIN Customers C
ON C.CustomerID = O.CustomerID
```

TUIA - BASES DE DATOS I
JOIN

FULL JOIN – combinación externa completa

- Devuelve un conjunto de resultados que incluye filas de las tablas izquierda y derecha.
- Cuando no existen filas coincidentes para la fila de la tabla de la izquierda, las columnas de la tabla de la derecha tendrán nulos.
- Cuando no existen filas coincidentes para la fila de la tabla de la derecha, la columna de la tabla de la izquierda tendrá NULL

TUIA - BASES DE DATOS I
JOIN



FULL JOIN – combinación externa completa

- Devuelve un conjunto de resultados que incluye filas de las tablas izquierda y derecha.
- Cuando no existen filas coincidentes para la fila de la tabla de la izquierda, las columnas de la tabla de la derecha tendrán nulos.
- Cuando no existen filas coincidentes para la fila de la tabla de la derecha, la columna de la tabla de la izquierda tendrá NULL

TUIA - BASES DE DATOS I
VARIABLES LOCALES

Una variable local de Transact-SQL es un objeto que contiene un valor individual de datos de un tipo específico. Normalmente, las variables se utilizan en lotes y scripts:

- Como contadores, para contar el número de veces que se realiza un bucle o controlar cuántas veces debe ejecutarse.
- Para contener un valor de datos que desea probar mediante una instrucción de control de flujo.
- Para guardar el valor de un dato que se va a devolver en un código de retorno de un procedimiento almacenado o un valor devuelto de una función.

TUIA - BASES DE DATOS I
VARIABLES LOCALES

La instrucción DECLARE inicializa una variable de Transact-SQL al:

- Asignar un nombre. El nombre debe tener un único @ como primer carácter.
- Asignar un tipo de datos suministrado por el sistema o definido por el usuario y una longitud. Para las variables numéricas, se asignan también una precisión y una escala si corresponde.
- Establecer el valor a NULL.

TUIA - BASES DE DATOS I
VARIABLES LOCALES

Establecer un valor en una variable de Transact-SQL

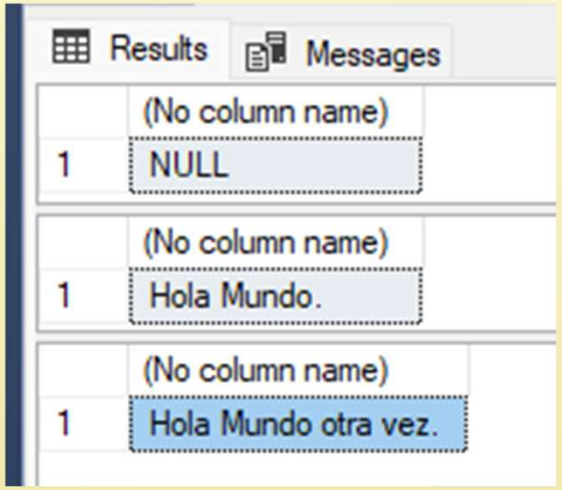
- Cuando una variable se declara por primera vez, su valor se establece en NULL.
- Para asignar un valor a una variable se utiliza
  - la instrucción SET
  - El comando SELECT

TUIA - BASES DE DATOS I
VARIABLES LOCALES

```
DECLARE @MyMsg VARCHAR(50)
SELECT @MyMsg

set @MyMsg = 'Hola Mundo.'
SELECT @MyMsg

SELECT @MyMsg = 'Hola Mundo otra vez.'
SELECT @MyMsg
```



TUIA - BASES DE DATOS I
VARIABLES LOCALES

Ejemplo

En la BD `Northwind` dado un ID de empleado seleccionar todas sus ordenes y los datos de dicho empleado

TUIA - BASES DE DATOS I
VARIABLES LOCALES

Ejemplo

En la BD `Northwind` mostrar los datos del empleado con mas ordenes.

TUIA - BASES DE DATOS I
COMANDO GO

- SQL Server proporciona comandos que no son instrucciones de Transact-SQL, pero que las utilidades **sqlcmd** y **osql**, y el Editor de código de SQL Server Management Studio sí reconocen.
- Estos comandos se pueden usar para facilitar la legibilidad y la ejecución de **lotes** y **scripts**.
- **GO** indica a las utilidades de SQL Server el final de un lote de instrucciones Transact-SQL.

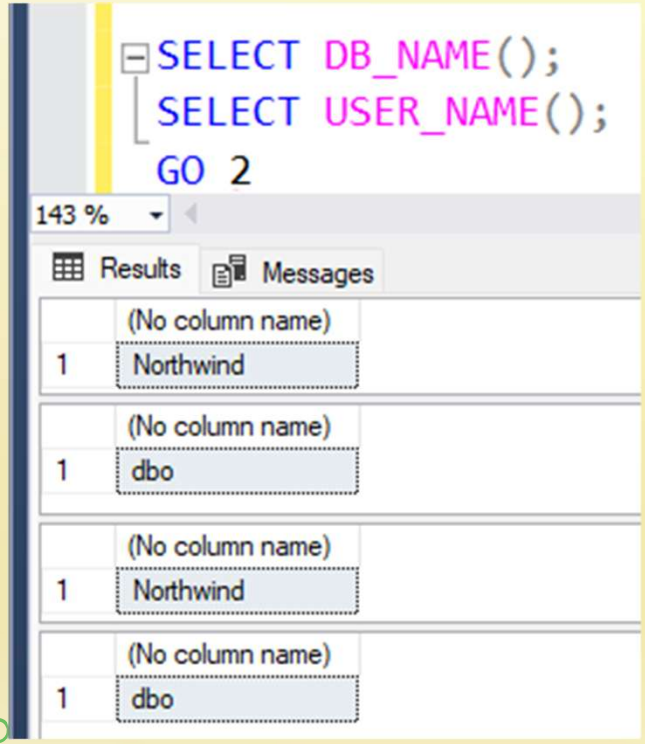
TUIA - BASES DE DATOS I
COMANDO GO

Argumentos

count

Es un entero positivo. El lote que precede a GO se ejecutará el número especificado de veces.

TUIA - BASES DE DATOS I
COMANDO GO



TUIA - BASES DE DATOS I
COMANDO GO

- Las utilidades de SQL Server interpretan GO como una señal de que deben enviar el lote actual de instrucciones Transact-SQL a una instancia de SQL Server.
- El lote actual de instrucciones está formado por todas las instrucciones especificadas desde el último comando GO o desde el comienzo de la sesión o script si se trata del primer comando GO.

TUIA - BASES DE DATOS I
COMANDO GO

- Una instrucción Transact-SQL no puede ocupar la misma línea que un comando **GO**. Sin embargo, la línea sí puede contener comentarios.
- El comando **GO** es un límite para el **alcance** de las variables locales
- Las instrucciones del lote se compilan en un único plan de ejecución.

TUIA - BASES DE DATOS I
COMANDO GO - EJEMPLO

```
USE Northwind;
GO
DECLARE @NumEmp INT
SELECT @NumEmp = COUNT(*)
FROM Employees;
PRINT 'La cantidad de empleados al ' +
      CAST(GETDATE() AS CHAR(20)) + ' es ' +
      CAST(@NumEmp AS CHAR(10));
GO
```

TUIA - BASES DE DATOS I
COMANDO GO - EJEMPLO

```
USE Northwind;
GO
DECLARE @MyMsg VARCHAR(50)
SELECT @MyMsg = 'Hola Mundo.'
GO -- @MyMsg no es valido despues que  GO finalice el lote.


-- Lanza un error porque @MyMsg no está declarado en este lote.
PRINT @MyMsg
GO
```

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Un procedimiento almacenado de SQL Server es un grupo de una o varias instrucciones Transact-SQL.

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.
- 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.

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Ventajas de usar procedimientos almacenados

En la siguiente lista se describen algunas de las ventajas que brinda el uso de procedimientos.

- Tráfico de red reducido entre el cliente y el servidor
- Mayor seguridad
- Reutilización del código
- Mantenimiento más sencillo
- Rendimiento mejorado (Se compila el plan de ejecución solo una vez)

Mas info: Procedimientos almacenados (motor de base de datos) - SQL Server | Microsoft Learn

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Crear un SP

Crear un procedimiento almacenado - SQL Server | Microsoft Learn

Se pueden crear mediante SQL Server Managment Studio o mediante T-SQL

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Crear un SP - CREATE PROCEDURE

```
USE [Northwind]
GO
CREATE PROCEDURE usp_GetEmployeesTest
    @LastName nvarchar(50),
    @FirstName nvarchar(50)
AS

    SET NOCOUNT ON;
    SELECT Title, FirstName, LastName
    FROM Employees
    WHERE FirstName = @FirstName AND LastName = @LastName;
GO
```

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Llamada a un SP - EXECUTE

Se utiliza la palabra clave EXECUTE o EXEC y deben especificarse los parametros

```
USE [Northwind]
GO
EXECUTE usp_GetEmployeesTest 'Davolio', 'Nancy'
GO
EXECUTE usp_GetEmployeesTest 'Nancy', 'Davolio'
GO
EXECUTE usp_GetEmployeesTest @FirstName='Nancy', @LastName='Davolio’
GO
```

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Modificar un SP – ALTER PROCEDURE

```
USE [Northwind]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[usp_GetEmployeesTest]
    @LastName nvarchar(50),
    @FirstName nvarchar(50)
AS
    SET NOCOUNT ON;
    SELECT FirstName, LastName, Title,  HomePhone
    FROM Employees
    WHERE FirstName = @FirstName AND LastName = @LastName;
```

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Eliminar un SP – DROP PROCEDURE

DROP PROCEDURE [<stored procedure name>];

GO

```
USE [Northwind]
GO
DROP PROCEDURE [dbo].[usp_GetEmployeesTest]
```

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Parámetros

Los parámetros se usan para intercambiar datos entre las funciones y los procedimientos almacenados y la aplicación o la herramienta que llamó a la función o al procedimiento almacenados:

- Los parámetros de entrada permiten a quien realiza la llamada pasar un valor de datos a la función o al procedimiento almacenado.
- Los parámetros de salida permiten al procedimiento almacenado devolver un valor de datos o variable de cursor a quien realizó la llamada. Las funciones definidas por el usuario no pueden especificar parámetros de salida.
- Cada procedimiento almacenado devuelve un código de retorno de tipo entero a quien realiza la llamada. Si el procedimiento almacenado no establece explícitamente un valor para el código de retorno, éste es 0.

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Parámetros

```
CREATE PROCEDURE usp_SampleProcedure
@EmployeeIDParm INT,
        @Total INT OUTPUT
AS
DECLARE @ErrorSave INT
SET @ErrorSave = 0

-- SELECT usando el parametro de entrada
SELECT FirstName, LastName, Title
FROM Employees
WHERE EmployeeID = @EmployeeIDParm
```

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Parámetros

```
-- Guardar @@ERROR distinto de 0.
IF (@@ERROR <> 0)
   SET @ErrorSave = @@ERROR

-- Setear un valor en el parametron de salida.
SELECT @Total = COUNT(*)
FROM Orders
WHERE EmployeeID = @EmployeeIDParm;

SELECT @Total
IF (@@ERROR <> 0)
   SET @ErrorSave = @@ERROR

-- Devuelve 0 si nigun SELECT dio error; de otra forma devuelve el ultimo error.
RETURN @ErrorSave
GO
```

TUIA - BASES DE DATOS I
PROCEDIMIENTOS ALMACENADOS - STORED PROCEDURES

Parámetros

```
DECLARE @ReturnCode INT

DECLARE @Total1 INT

EXEC @ReturnCode = usp_SampleProcedure 5, @Total1 OUTPUT


SELECT @ReturnCode

SELECT @Total1
```