

Base de Datos 1

Unidad 5
SQL





Trabajo de Regularidad 2

- Está disponible en la plataforma el segundo trabajo de regularidad.
- Consiste en el leer un artículo sobre el origen de SQL y responder unas preguntas
- Tiempo: Hasta el 1 de junio a las 8:00

The background of the slide features a series of concentric circles that create a tunnel-like effect. The color gradient transitions from a light blue on the left to a light green on the right. The text is centered within this circular pattern.

Trabajo Práctico

Combinando WHERE y ORDER BY

- Cuando los utilizamos juntos, la cláusula WHERE debe ir antes de la cláusula ORDER BY.

- **EJEMPLO**

```
SELECT FirstName, LastName  
FROM Employees  
WHERE LastName >= 'N'  
ORDER BY LastName DESC;
```

FirstName	LastName
Michael	Suyama
Margaret	Peacock

Operadores de Condición

- La cláusula WHERE y las palabras operadores

BETWEEN	Devuelve valores en un rango inclusivo
IN	Devuelve valores que se encuentran en un conjunto específico
LIKE	Devuelve valores que coinciden con un patrón simple
NOT	Niega una operación

- EJEMPLOS

```
/* Seleccionar nombre y apellido de todos los empleados cuyos nombres  
comiencen con una letra entre "J" y "M". */
```

```
SELECT FirstName, LastName  
FROM Employees  
WHERE LastName BETWEEN 'J' AND 'M';
```

-- Lo anterior es equivalente a lo siguiente.

```
SELECT FirstName, LastName  
FROM Employees  
WHERE LastName >= 'J' AND LastName <= 'M';
```

FirstName	LastName
Janet	Leverling
Robert	King

```
/* Seleccionar saludo (TitleOfCourtesy), nombre (FirstName) y apellido  
(LastName) de todos los empleados cuyo saludo sea "Mrs." o "Ms.". */
```

```
SELECT TitleOfCourtesy, FirstName, LastName  
FROM Employees  
WHERE TitleOfCourtesy IN ('Ms.', 'Mrs.');
```

-- o su equivalente

```
SELECT TitleOfCourtesy, FirstName, LastName  
FROM Employees  
WHERE TitleOfCourtesy = 'Ms.' OR TitleOfCourtesy = 'Mrs.';
```

TitleOfCourtesy	FirstName	LastName
Ms.	Nancy	Davolio
Ms.	Janet	Leverling
Mrs.	Margaret	Peacock
Ms.	Laura	Callahan
Ms.	Anne	Dodsworth

El operador LIKE

- Se utiliza para verificar si un campo coincide con una determinada expresión o patrón.
- En estas expresiones se utilizan caracteres comodines.
 - % (porcentaje): representa una cadena de cualquier tamaño, incluyendo la cadena vacía.
 - _ (guion bajo): representa un único carácter
 - **[lista de caracteres]**: le puede definir una lista de caracteres entre corchetes, que representan a un único carácter.
 - **[carácter_desde-carácter_hasta]**: representa un único carácter, comprendido en el rango establecido entre corchetes.
 - **[^carácter_desde-carácter_hasta]**: representa un único carácter, que no esté en el rango establecido. Es la negación de la expresión anterior.

EJEMPLOS

```
/* Seleccionar saludo (TitleOfCourtesy), nombre (FirstName) y  
apellido (LastName) de todos los empleados cuyo saludo  
comience con "M". */
```

```
SELECT TitleOfCourtesy, FirstName, LastName  
FROM Employees  
WHERE TitleOfCourtesy LIKE 'M%';
```

TitleOfCourtesy	FirstName	LastName
Ms.	Nancy	Davolio
Ms.	Janet	Leverling
Mrs.	Margaret	Peacock
Mr.	Steven	Buchanan
Mr.	Michael	Suyama
Mr.	Robert	King
Ms.	Laura	Callahan
Ms.	Anne	Dodsworth

```
/* Seleccionar saludo (TitleOfCourtesy), nombre  
(FirstName) y apellido (LastName) de todos los empleados  
cuyo saludo comience con una "M" seguida de cualquier  
carácter y luego un punto (.) */
```

```
SELECT TitleOfCourtesy, FirstName, LastName  
FROM Employees  
WHERE TitleOfCourtesy LIKE 'M_.';
```

TitleOfCourtesy	FirstName	LastName
Ms.	Nancy	Davolio
Ms.	Janet	Leverling
Mr.	Steven	Buchanan
Mr.	Michael	Suyama
Mr.	Robert	King
Ms.	Laura	Callahan
Ms.	Anne	Dodsworth

El operador NOT

- Se utiliza para negar una condición o una operación
/* Seleccionar saludo (TitleOfCourtesy), nombre (FirstName) y apellido (LastName) de todos los empleados cuyo saludo no sea "Ms." o "Mrs.". */

```
SELECT TitleOfCourtesy, FirstName, LastName  
FROM Employees  
WHERE NOT TitleOfCourtesy IN ('Ms.', 'Mrs.');
```

```
-- 0  
SELECT TitleOfCourtesy, FirstName, LastName  
FROM Employees  
WHERE TitleOfCourtesy NOT IN ('Ms.', 'Mrs.');
```

TitleOfCourtesy	FirstName	LastName
Dr.	Andrew	Fuller
Mr.	Steven	Buchanan
Mr.	Michael	Suyama
Mr.	Robert	King

Verificando múltiples condiciones

- **AND**
- Se utiliza en la cláusula WHERE para encontrar los registros que coincidan con más de una condición

```
/* Seleccionar nombre (FirstName) y apellido (LastName) de todos  
los representantes de ventas  
(Sales Representative) cuyo saludo (TitleOfCourtesy) es "Mr.". */
```

```
SELECT FirstName, LastName  
FROM Employees  
WHERE Title = 'Sales Representative' AND TitleOfCourtesy = 'Mr.';
```

FirstName	LastName
Michael	Suyama
Robert	King

Cláusula OR

- **OR**
- Se utiliza en la cláusula WHERE para encontrar los registros que coincidan con al menos una condición.

```
/* Seleccionar nombre (FirstName), apellido (LastName) y  
ciudad (City) de todos los empleados de la ciudad de Seattle  
o Redmond. */
```

```
SELECT FirstName, LastName, City  
FROM Employees  
WHERE City = 'Seattle' OR City = 'Redmond';
```

FirstName	LastName	City
Nancy	Davolio	Seattle
Margaret	Peacock	Redmond
Laura	Callahan	Seattle

Orden de evaluación

- Por defecto, SQL procesa los operadores AND antes de los operadores OR. Para ilustrar cómo funciona esto, veamos el siguiente ejemplo.



```
/* Seleccionar nombre (FirstName), apellido (LastName),  
ciudad (City) y puesto (Title) de todos los representantes de  
ventas (Sales Representative) que sean de la ciudad de  
Seattle o Redmond. */
```

```
SELECT FirstName, LastName, City, Title  
FROM Employees  
WHERE City = 'Seattle' OR City = 'Redmond' AND Title = 'Sales Representative';
```

FirstName	LastName	City	Title
Nancy	Davolio	Seattle	Sales Representative
Margaret	Peacock	Redmond	Sales Representative
Laura	Callahan	Seattle	Inside Sales Coordinator

- Noten que Laura no es representante de ventas (Sales Representative) y sin embargo está incluida en los resultados.
- Esto es porque la consulta está devolviendo empleados de Seattle O representantes de ventas de Redmond.


- Para solucionar esto, colocaremos la condición OR entre paréntesis.

```
/* Seleccionar nombre (FirstName), apellido (LastName), ciudad  
(City) y puesto (Title) de todos los representantes de ventas  
(Sales Representative) que sean de la ciudad de Seattle o Redmond.  
*/
```

```
SELECT FirstName, LastName, City, Title  
FROM Employees  
WHERE (City = 'Seattle' OR City = 'Redmond') AND Title = 'Sales  
Representative';
```

FirstName	LastName	City	Title
Nancy	Davolio	Seattle	Sales Representative
Margaret	Peacock	Redmond	Sales Representative

- De esta manera se evalúa primero la expresión OR y el resultado logrado es el buscado.
- Se recomienda entonces el uso de paréntesis en todo caso en el que la precedencia de operadores puede resultar ambigua.



Seleccionando registros sin repetición

- La palabra clave DISTINCT se utiliza para seleccionar combinaciones distintas de los valores de las columnas de una tabla. Por ejemplo, a continuación, se muestran sin repetición las ciudades que tienen empleados.

```
/* Hallar las distintas ciudades en las que viven los empleados. */
```

```
SELECT DISTINCT City  
FROM Employees  
ORDER BY City;
```

```
City  
-----  
Kirkland  
London  
Redmond  
Seattle  
Tacoma
```

CAMBIO DEL NOMBRE DE LAS COLUMNAS

```
USE northwind
SELECT  firstname AS First, lastname AS Last
        ,employeeid AS 'Employee ID:'
FROM employees
GO
```



<i>First</i>	<i>Last</i>	<i>Employee ID:</i>
Nancy	Davolio	1
Andrew	Fuller	2
Janet	Leverling	3
Margaret	Peacock	4
Steven	Buchanan	5
Michael	Suyama	6
Robert	King	7
Laura	Callahan	8
Anne	Dodsworth	9

Alias

- Las columnas correspondientes a las funciones no tienen título o nombre de columna en SQL Server y en MySQL les da como nombre la expresión. Mediante la palabra clave AS se puede agregar un encabezado a estas columnas.
- Tengan en cuenta que los alias no pueden ser utilizados en la cláusula WHERE.



CAMPOS CALCULADOS

- Los campos calculados son campos que no existen en las tablas, son creados en la sentencia SELECT. Por ejemplo, uno podría crear el campo “Nombre Completo” concatenando “Nombre” y “Apellido”.

Uso de literales

```
SELECT FirstName, LastName, 'Número de identificación:', EmployeeID  
FROM Employees;
```

FirstName	LastName	EmployeeID
Nancy	Davolio	Número de identificación: 1
Andrew	Fuller	Número de identificación: 2
Janet	Leverling	Número de identificación: 3
Margaret	Peacock	Número de identificación: 4
Steven	Buchanan	Número de identificación: 5
Michael	Suyama	Número de identificación: 6
Robert	King	Número de identificación: 7
Laura	Callahan	Número de identificación: 8
Anne	Dodsworth	Número de identificación: 9



CONCATENACIÓN

- La concatenación consiste en encadenar diferentes palabras o caracteres.
- T-SQL proporciona el operador de signo más (+), la función CONCAT y la función CONCAT_WS para concatenar cadenas.
- MySQL solo permite concatenar usando CONCAT y CONCAT_WS

CONCATENACIÓN


```
SELECT FirstName + ' ' + LastName  
FROM Employees;
```

```
SELECT CONCAT(FirstName, ' ', LastName)  
FROM Employees;
```

```
SELECT CONCAT_WS(' ', FirstName, LastName)  
FROM Employees;
```

```
-----  
Nancy Davolio  
Andrew Fuller  
Janet Leverling  
Margaret Peacock  
Steven Buchanan  
Michael Suyama  
Robert King  
Laura Callahan  
Anne Dodsworth
```

- La concatenación funciona sólo con cadenas de texto. Para concatenar otros tipos de datos, hay que convertirlos primero a cadena de texto.



CÁLCULOS MATEMÁTICOS

+

Suma

-

Resta

*

Multiplicación

/

División

%

Módulo

Campo calculado + Operadores + Alias

```
/* Si el costo de flete  
(Freight) es mayor o igual a  
$500.00, hay que aplicarle un  
impuesto del 10%. Crear un  
reporte que muestre ID de la  
orden (OrderID), costo de  
flete (Freight), y costo de  
flete con el impuesto para las  
órdenes con costo de flete de  
$500 o más. */
```

```
SELECT OrderID, Freight,  
Freight * 1.1 AS FleteTotal  
FROM Orders  
WHERE Freight >= 500;
```

OrderID	Freight	FleteTotal
-----	-----	-----
10372	890,78	979.85800
10479	708,95	779.84500
10514	789,95	868.94500
10540	1007,64	1108.40400
10612	544,08	598.48800
10691	810,05	891.05500
10816	719,78	791.75800
10897	603,54	663.89400
10912	580,91	639.00100
10983	657,54	723.29400
11017	754,26	829.68600
11030	830,75	913.82500
11032	606,19	666.80900

The background features several abstract geometric elements: a large orange circle on the right, a blue circle in the upper left, a yellow circle in the top right, a green L-shaped line in the top center, a green square outline on the left, and several yellow dashed lines scattered on the left side.

Funciones de uso
Común

FUNCIONES DE MANIPULACIÓN INCLUIDAS

Descripción	SQL Server	Oracle	MySQL
Valor Absoluto	ABS	ABS	ABS
Menor entero que sea \geq	CEILING	CEIL	CEILING
Mayor entero que sea \leq	FLOOR	FLOOR	FLOOR
Potencia	POWER	POWER	POWER
Redondeo	ROUND	ROUND	ROUND
Raíz Cuadrada	SQRT	SQRT	SQRT
Formatear números con dos decimales	CAST(num AS decimal(8,2))	CAST(num AS decimal(8,2))	FORMAT(num,2) O CAST(num AS decimal(8,2))

```

/* Obtener el costo de flete, como está y redondeado al primer decimal.
Utilizar ceiling y floor y ver resultados*/
SELECT Freight, ROUND(Freight,2) AS FleteRedondeado, ABS(Freight) AS
Flete_ABS, CEILING(Freight) AS Flete_Ceiling, FLOOR(Freight) AS
Flete_Floor
FROM Orders;

```

Freight	FleteRedondeado	Flete_ABS	Flete_Ceiling	Flete_Floor
-----	-----	-----	-----	-----
32,38	32,38	32,38	33,00	32,00
11,61	11,61	11,61	12,00	11,00
65,83	65,83	65,83	66,00	65,00
41,34	41,34	41,34	42,00	41,00
51,30	51,30	51,30	52,00	51,00
58,17	58,17	58,17	59,00	58,00
22,98	22,98	22,98	23,00	22,00
148,33	148,33	148,33	149,00	148,00
13,97	13,97	13,97	14,00	13,00
81,91	81,91	81,91	82,00	81,00
..				

```
/* Obtener el precio unitario como está y como CHAR(10) */  
SELECT UnitPrice, CAST(UnitPrice AS CHAR(10))  
FROM Products;
```

UnitPrice

UnitPrice	UnitPrice
18.00	18.00
19.00	19.00
10.00	10.00
22.00	22.00
21.35	21.35
25.00	25.00
30.00	30.00
...	
7.75	7.75
18.00	18.00
13.00	13.00

Agregar Concatenación

```
SELECT UnitPrice, CONCAT('$', CAST(UnitPrice AS char(10)))  
FROM Products;
```

UnitPrice

UnitPrice		
18.00	\$	18.00
19.00	\$	19.00
10.00	\$	10.00
22.00	\$	22.00
21.35	\$	21.35
25.00	\$	25.00
30.00	\$	30.00
...		
7.75	\$	7.75
18.00	\$	18.00
13.00	\$	13.00



Descripción	SQL Server	Oracle	MySQL
Convertir a minúsculas	LOWER	LOWER	LOWER
Convertir a mayúsculas	UPPER	UPPER	UPPER
Remover espacios de atrás	RTRIM	RTRIM	RTRIM
Remover espacios de adelante	LTRIM	LTRIM	LTRIM
Subcadena	SUBSTRING	SUBSTR	SUBSTRING

FUNCIONES DE MANIPULACIÓN DE CADENAS

```
/* Seleccionar nombre (FirstName) y apellido (LastName) de los  
empleados y mostrar en mayúsculas */
```

```
SELECT UPPER(FirstName), UPPER(LastName)  
FROM Employees;
```

```
-----  
NANCY      DAVOLIO  
ANDREW     FULLER  
JANET      LEVERLING  
MARGARET   PEACOCK  
STEVEN     BUCHANAN  
MICHAEL    SUYAMA  
ROBERT     KING  
LAURA     CALLAHAN  
ANNE       DODSWORTH
```

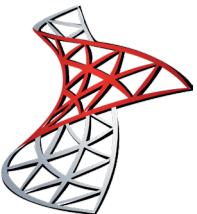

FUNCIONES DE FECHAS

Descripción	SQL Server	Oracle	MySQL
Suma	DATEADD	+	DATE_ADD
Resta	DATEDIFF	-	DATEDIFF
Convertir fecha a cadena	DATENAME	TO_CHAR	DATE_FORMAT
Convertir fecha a número	DATEPART	TO_NUMBER(TO_CHAR)	EXTRACT
Obtener fecha y hora actual	GETDATE	SYSDATE	NOW

-- Obtener la edad a la que fueron contratados los empleados

```
SELECT LastName, BirthDate, HireDate, DATEDIFF(year,  
BirthDate, HireDate) AS EdadAlAlta  
FROM Employees  
ORDER BY EdadAlAlta;
```

LastName	BirthDate	HireDate	EdadAlAlta
Dodsworth	1966-01-27 00:00:00.000	1994-11-15 00:00:00.000	28
Leverling	1963-08-30 00:00:00.000	1992-04-01 00:00:00.000	29
Suyama	1963-07-02 00:00:00.000	1993-10-17 00:00:00.000	30
King	1960-05-29 00:00:00.000	1994-01-02 00:00:00.000	34
Callahan	1958-01-09 00:00:00.000	1994-03-05 00:00:00.000	36
Buchanan	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	38
Fuller	1952-02-19 00:00:00.000	1992-08-14 00:00:00.000	40
Davolio	1948-12-08 00:00:00.000	1992-05-01 00:00:00.000	44
Peacock	1937-09-19 00:00:00.000	1993-05-03 00:00:00.000	56



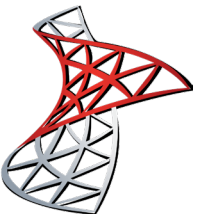
-- Obtener la edad a la que fueron contratados los empleados

```
SELECT LastName, BirthDate, HireDate,  
TIMESTAMPDIFF(YEAR, BirthDate, HireDate) AS  
EdadAlAlta  
FROM Employees  
ORDER BY EdadAlAlta;
```

-- Hallar el mes de nacimiento de cada empleado

```
SELECT FirstName, LastName, DATENAME(month,  
BirthDate) AS MesNacimiento  
FROM Employees;
```

FirstName	LastName	MesNacimiento
Laura	Callahan	January
Anne	Dodsworth	January
Andrew	Fuller	February
Steven	Buchanan	March
Robert	King	May
Michael	Suyama	July
Janet	Leverling	August
Margaret	Peacock	September
Nancy	Davolio	December



-- Hallar el mes de nacimiento de cada empleado

```
SELECT FirstName, LastName, MONTHNAME(BirthDate)  
AS MesNacimiento  
FROM Employees;
```

SELECT

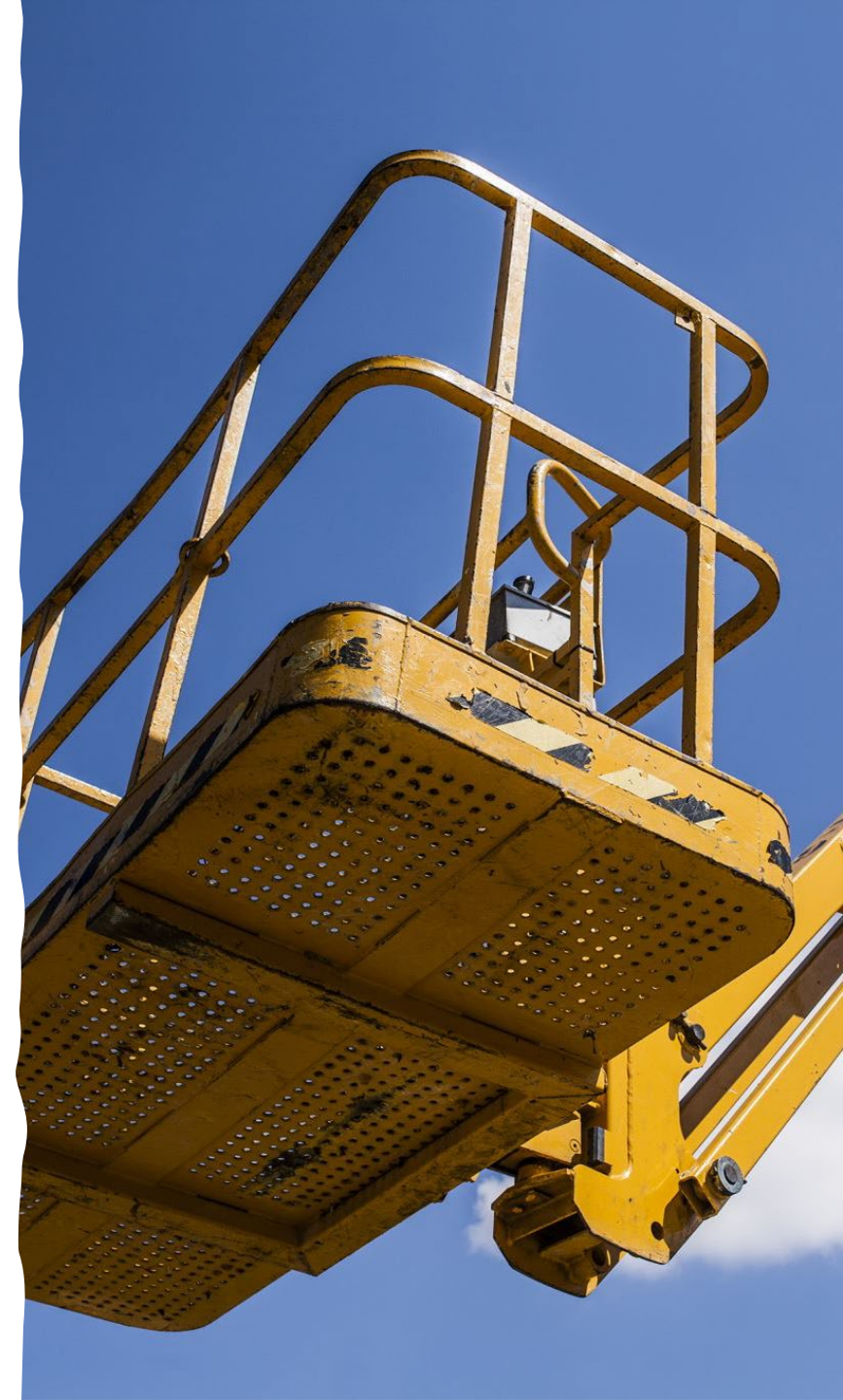
SELECT lista_select

[FROM tabla_origen]

[WHERE condición_filtro]

[ORDER BY expresión_orden [ASC | DESC]]

Manipulación de Datos (Insert/Update/Delete/Truncate)



INSERT

INSERT INTO tabla

[(columna1, columna2, ...)]

[VALUES]

[(expresión1, expresión2, ...)]

INSERT - Ejemplos

```
INSERT INTO Production.UnitMeasure  
VALUES ('FT', 'Feet', '20080414');
```

```
INSERT INTO Production.UnitMeasure VALUES  
( 'FT2', 'Square Feet ', '20080923' ),  
( 'Y', 'Yards', '20080923' ),  
( 'Y3', 'Cubic Yards', '20080923' );
```

```
INSERT INTO Production.UnitMeasure  
(Name, UnitMeasureCode, ModifiedDate)  
VALUES ('Square Yards', 'Y2', GETDATE());
```

INSERT
usando
SELECT

INSERT INTO tabla

[(columna1, columna2, ...)]

SELECT expresión1, expresión2, ...

[FROM tabla_origen]

[WHERE condición_filtro]

Inserción de datos parciales

Agregar datos nuevos

Ejemplo 1

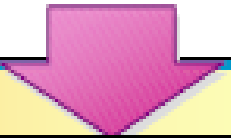
```
USE northwind
INSERT shippers (companyname)
VALUES ('Fitch & Mather')
GO
```

Comprobar datos nuevos

Ejemplo 2

```
USE northwind
SELECT *
FROM shippers
WHERE companyname = 'Fitch & Mather'
GO
```

Permite valores Null



<i>shipperid</i>	<i>companyname</i>	<i>phone</i>
4	Fitch & Mather	Null

INSERT - Ejemplos

```
DROP TABLE IF EXISTS EmployeeTitles;
```

```
CREATE TABLE EmployeeTitles  
( EmployeeID    INT NOT NULL,  
  LastName      nvarchar(20) NOT NULL,  
  Title         nvarchar(30) NOT NULL  
);
```

```
INSERT INTO EmployeeTitles  
  SELECT EmployeeID, LastName, Title  
  FROM Employees  
  WHERE Region IS NULL;
```

UPDATE

UPDATE tabla

SET columna1 = expresión1

[, columna2 = expresión2]

[, ...]

[WHERE condición_filtro]

Actualización de filas basadas en datos de la tabla

- La cláusula **WHERE** especifica las filas que se van a cambiar
- La palabra clave **SET** especifica los datos nuevos
- Los valores de entrada deben tener los mismos tipos de datos que las columnas
- No se actualizarán las filas que infrinjan alguna restricción de integridad

```
USE northwind
UPDATE products
  SET unitprice = (unitprice * 1.1)
GO
```

DELETE

DELETE

FROM tabla

[WHERE
condición_filtro]

Uso de la instrucción DELETE

- La instrucción DELETE quita una o más filas en una tabla a menos que utilice una cláusula WHERE
- Cada fila eliminada se almacena en el registro de transacciones

```
USE northwind
DELETE orders
  WHERE DATEDIFF(MONTH, shippeddate, GETDATE()) >= 6
GO
```


Uso de la instrucción TRUNCATE TABLE

- La instrucción **TRUNCATE TABLE** elimina todas las filas de una tabla
- SQL Server conserva la estructura de la tabla y los objetos asociados
- Sólo registra la cancelación de la asignación de las páginas de datos en el registro de transacciones

```
USE northwind
TRUNCATE TABLE orders
GO
```

Manejando Fechas

SQL

Fecha y Hora por separado

- Cuando no utiliza la parte de hora se almacena como hora la medianoche y cuando no se utiliza la parte de fecha se almacena como fecha 1 de enero de 1900 en SQL Server

```
SELECT CAST('12:30:15.123' AS DATETIME);
```

```
-----
```

```
1900-01-01 12:30:15.123
```

- En MySQL lo interpreta como error y devuelve NULL. Agregando el modificador TIME, devuelve una fecha, pero la de hoy

```
SELECT CAST(TIME '12:30:15.123' AS DATETIME);
```

```
-----
```

```
2024-05-18 12:30:15
```

Fecha y Hora por separado

- Si están almacenados con hora, hay que filtrar por rango
SELECT OrderID, CustomerID, EmployeeID, OrderDate
FROM Orders
WHERE OrderDate >= '19970212'
AND OrderDate < '19970213';

- Listar las órdenes del año 1997

```
SELECT OrderID, CustomerID, EmployeeID, OrderDate  
FROM Orders  
WHERE OrderDate >= '19970101' AND OrderDate <  
'19980101';
```

```
SELECT OrderID, CustomerID, EmployeeID, OrderDate  
FROM Orders  
WHERE YEAR(OrderDate)=1997;
```

Filtrando rangos

Filtrando rangos

- Listar las órdenes de febrero del año 1997

```
SELECT OrderID, CustomerID,  
EmployeeID, OrderDate  
FROM Orders  
WHERE OrderDate >= '19970201'  
AND OrderDate < '19970301';
```

```
SELECT OrderID, CustomerID,  
EmployeeID, OrderDate  
FROM Orders  
WHERE YEAR(OrderDate)=1997 AND  
MONTH(OrderDate)=2;
```

Presentación de los primeros n valores

- Presenta sólo las n primeras filas de un conjunto de resultados
- Especifica el intervalo de valores con la cláusula ORDER BY
- Devuelve las filas iguales si se utiliza WITH TIES

Ejemplo 1

```
USE northwind
SELECT TOP 5 orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
GO
```

Ejemplo 2

```
USE northwind
SELECT TOP 5 WITH TIES orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
GO
```

Agrupar y Resumir



Agrupamiento en SQL


- Uso de las funciones de agregado
- Fundamentos del GROUP BY
- Generación de valores de agregado dentro de los conjuntos de resultados




Funciones de agregación y agrupamiento

COUNT()	Devuelve la cantidad de filas que contienen valores NO nulos en un campo específico
SUM()	Devuelve la suma
AVG()	Devuelve el promedio
MAX()	Devuelve el valor máximo
MIN()	Devuelve el valor mínimo

- Las funciones de agregación son utilizadas para calcular resultados utilizando campos de múltiples registros. Las cinco más comunes son:

- 
- SQL Server cuenta además con las siguientes funciones:

CHECKSUM_AGG	Devuelve un checksum del grupo. Puede ser utilizado para detectar cambios
COUNT_BIG	Como el COUNT, pero devuelve un valor bigint en lugar int
STDEV/STDEVP	Devuelve el desvío estándar estadístico o poblacional
VAR/VARP	Devuelve la varianza estadística o poblacional

- 
- MySQL cuenta además con las siguientes funciones:

BIT_AND()	Devuelve AND bit a bit
BIT_OR()	Devuelve OR bit a bit
BIT_XOR()	Devuelve XOR bit a bit
GROUP_CONCAT()	Devuelve una cadena concatenada
JSON_ARRAYAGG()	Devuelve resultado como un array JSON
JSON_OBJECTAGG()	Devuelve resultado como un objeto JSON
STDDEV()/STDDEV_POP()	Devuelve el desvío estándar estadístico o poblacional
VARIANCE()/VAR_POP()	Devuelve la varianza estadística o poblacional

Uso de las funciones de agregado con valores nulos

- La mayoría de las funciones de agregado pasan por alto los valores nulos
- La función COUNT(*) cuenta las filas con valores nulos

```
USE northwind  
SELECT COUNT (*)  
FROM employees  
GO
```

Ejemplo 1

```
USE northwind  
SELECT COUNT(reportsto)  
FROM employees  
GO
```

Ejemplo 2

- DISTINCT puede utilizarse con funciones agregadas.

```
/* Hallar cuantas ciudades diferentes tienen empleados. */  
SELECT COUNT(DISTINCT City) AS NumCiudades  
FROM Employees;
```

NumCiudades

5

Seleccionando registros sin repetición

- Total de unidades ordenadas del producto 3

```
SELECT SUM(Quantity) AS
TotalUnidades
FROM [Order Details]
WHERE ProductID = 3;
```

TotalUnidades

328

- Precio Unitario Promedio de los productos

```
SELECT AVG(UnitPrice)
AS PrecioPromedio
FROM Products;
```

PrecioPromedio

28,8663

-- Encontrar la fecha de la primera y última contratación (HireDate) de empleados

```
SELECT MIN(HireDate) AS PrimeraFechaAlta,  
MAX(HireDate) AS UltimaFechaAlta  
FROM Employees;
```

PrimeraFechaAlta

UltimaFechaAlta

-----	-----
1992-04-01 00:00:00.000	1994-11-15 00:00:00.000

- Con GROUP BY las funciones de agrupación pueden ser aplicadas a grupos basados en los valores de sus campos. Por ejemplo, el número de empleados por ciudad:

```
SELECT City, COUNT(EmployeeID) AS NumEmpleados  
FROM Employees  
GROUP BY City;
```

City	NumEmpleados
Kirkland	1
London	4
Redmond	1
Seattle	2
Tacoma	1

Agrupando Datos

Uso de la cláusula GROUP BY

```
USE northwind
SELECT productid, orderid, quantity
FROM orderhist
GO
```

productid	orderid	quantity
1	1	5
1	1	10
2	1	10
2	2	25
3	1	15
3	2	30

```
USE northwind
SELECT productid, SUM(quantity) AS total_quantity
FROM orderhist
GROUP BY productid
GO
```

productid	total_quantity
1	15
2	35
3	45

Sólo se agrupan las filas que cumplan la cláusula WHERE

productid	total_quantity
2	35

```
USE northwind
SELECT productid, SUM(quantity) AS total_quantity
FROM orderhist
WHERE productid = 2
GROUP BY productid
GO
```



Having

- Mediante HAVING se pueden filtrar los resultados una vez que las funciones fueron calculadas.

`/* Obtener el número de empleados de cada ciudad, en la que haya al menos dos empleados */`

```
SELECT City, COUNT(EmployeeID) AS  
NumEmpleados  
FROM Employees  
GROUP BY City  
HAVING COUNT(EmployeeID) > 1;
```


City	NumEmpleados
-----	-----
London	4
Seattle	2

Uso de la cláusula GROUP BY con la cláusula HAVING

```
USE northwind
SELECT productid, orderid,
       quantity
FROM orderhist
GO
```

productid	orderid	quantity
1	1	5
1	1	10
2	1	10
2	2	25
3	1	15
3	2	30

```
USE northwind
SELECT productid, SUM(quantity)
       AS total_quantity
FROM orderhist
GROUP BY productid
HAVING SUM(quantity)>=30
GO
```



productid	total_quantity
2	35
3	45

Orden de las cláusulas

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. HAVING
6. ORDER BY

```
/* Hallar el número (cantidad) de representantes de  
ventas en cada ciudad que cuenta con al menos dos.  
Ordenar según el número de empleados. */
```

```
SELECT City, COUNT(EmployeeID) AS NumEmpleados  
FROM Employees  
WHERE Title = 'Sales Representative'  
GROUP BY City  
HAVING COUNT(EmployeeID) > 1  
ORDER BY NumEmpleados;
```

ciudad	NumEmpleados
-----	-----
London	3

Reglas de Agrupamiento



Cada columna no calculada que aparece en el SELECT debe aparecer también en el GROUP BY.



No se pueden utilizar alias en el HAVING



Se pueden utilizar alias en el ORDER BY



Sólo se pueden utilizar campos calculados en el HAVING



Se deben utilizar alias de campos calculados o campos reales en el ORDER BY



Subconsultas

BBDD1 - UNPAZ

¿Que son las subconsultas?

- Las subconsultas son consultas embebidas dentro de otras consultas.
- Se utilizan para
 - Obtener información de una tabla basada en información de otra tabla.
 - Para dividir una consulta compleja en varios pasos
- Por lo general las tablas deben tener algún tipo de relación entre ellas.

En dónde se pueden utilizar

- FROM
 - Tablas derivadas
- SELECT
 - Una expresión en SQL
- WHERE o HAVING
 - Como una subconsulta correlacionada o autocontenida

Uso de una Subconsulta como una tabla derivada

- Es un conjunto de registros dentro de una consulta que funciona como una tabla
- Ocupa el lugar de la tabla en la cláusula FROM
- Se optimiza con el resto de la consulta

```
SELECT T.OrderID, T.CustomerID  
FROM ( SELECT OrderID, CustomerID  
        FROM Orders ) AS T
```

Uso de una Subconsulta como una expresión

- Se evalúa y trata como una expresión
- Se ejecuta una vez para la instrucción entera

```
SELECT ProductName, UnitPrice,  
       ( SELECT AVG(UnitPrice) FROM Products) AS  
Promedio,  
       UnitPrice - ( SELECT AVG(UnitPrice) FROM Products) AS  
Diferencia  
FROM Products  
WHERE Discontinued = 1;
```

- Obtener el clienteid de una orden específica es muy sencillo.

```
/* Hallar el CustomerID de la compañía que realizó  
la orden 10290. */
```

```
SELECT CustomerID  
FROM Orders  
WHERE OrderID = 10290;
```

CustomerID

COMMI

- COMMI es probable que no signifique mucho para quien pueda estar leyendo el resultado del reporte. Mediante la siguiente consulta, que utiliza subconsulta podemos ver un resultado más útil.

```
/* Hallar el Nombre de la compañía que realizó la  
orden 10290. */
```

```
SELECT CompanyName
```

```
FROM Customers
```

```
WHERE CustomerID = (SELECT CustomerID  
                     FROM Orders  
                     WHERE OrderID = 10290);
```

Subconsultas

- La subconsulta puede contener cualquier SELECT válido, pero debe retornar en este caso una única columna con el número esperado de resultados.
- Si la subconsulta devuelve un solo valor se puede comparar por igualdad, desigualdad, mayor, menor, etc.
- Pero si la subconsulta devuelve más de un registro, la consulta deberá preguntar si el campo está dentro (IN) o no (NOT IN) del conjunto de valores devueltos.

- -- Hallar los nombres de las compañías que efectuaron órdenes en 1997

```
SELECT CompanyName
FROM Customers
WHERE CustomerID IN (SELECT CustomerID
                     FROM Orders
                     WHERE YEAR(OrderDate)=1997);
```


Clasificación formal de las subconsultas

- De acuerdo a la cantidad esperada de valores que puede devolver una subconsulta, se las puede clasificar en:
 - Escalar (Un solo valor)
 - Multivaluada (Múltiples Valores)
 - Expresión de Tabla (Múltiples valores en una Tabla Derivada)
- De acuerdo a la dependencia de la consulta principal, se las puede clasificar en:
 - Autocontenida
 - Correlacionada



Subconsultas Autocontenidas

- Las subconsultas autocontenidas son independientes de la consulta principal a la que pertenecen, es decir, pueden ser ejecutadas de manera independiente. Por lo tanto, son simples para probar, ya que pueden probarse por separado

Subconsultas Autocontenidas Escalares

- Dado que devuelve un único valor y es independiente de la consulta principal, puede aparecer en cualquier lugar de la consulta principal en que se necesite como en el SELECT o WHERE.

- Por ejemplo, si necesitamos obtener los datos de la orden con el mayor id (ordenid) podemos ejecutar la siguiente consulta:

```
SELECT OrderID, OrderDate, EmployeeID, CustomerID  
FROM Orders
```

```
WHERE OrderID = (SELECT MAX(O.OrderID)  
                  FROM Orders AS O);
```

OrderID	OrderDate	EmployeeID	CustomerID
11077	1998-05-06 00:00:00.000	1	RATTC

- Para que una subconsulta escalar sea válida debe retornar a lo sumo un valor. Si llega a devolver más de un valor se producirá un error en tiempo de ejecución.
- La siguiente consulta se ejecuta sin problema:

```
SELECT OrderID
```

```
FROM Orders
```

```
WHERE EmployeeID = (SELECT E.EmployeeID  
                     FROM Employees AS E  
                     WHERE E.LastName LIKE 'C%');
```

- Si una subconsulta escalar no devuelve ningún valor, devuelve NULL, por lo tanto, el predicado evalúa una comparación con NULL, que da como resultado Desconocido y por lo tanto la consulta principal no devuelve resultados.
- Por ejemplo, si buscamos empleado cuyo apellido empieza con A:

```
SELECT OrderID
```

```
FROM Orders
```

```
WHERE EmployeeID = (SELECT E.EmployeeID  
                     FROM Employees AS E  
                     WHERE E.LastName LIKE 'A%');
```

Subconsultas Autocontenidas Multivaluadas

- Son consultas que devuelven múltiples valores para una misma columna. Los resultados de estas subconsultas deben evaluarse con predicados como IN.
- El formato de un predicado utilizando IN es:
<expresión escalar> [NOT] IN (<subconsulta multivaluada>)
- El predicado devuelve verdadero si la expresión escalar coincide con alguno de los valores devueltos por la subconsulta.

- Si reescribimos uno de los ejemplos anteriores utilizando IN, ya no tendríamos problemas en tiempo de ejecución:

```
SELECT OrderID
```

```
FROM Orders
```

```
WHERE EmployeeID IN (SELECT E.EmployeeID  
                      FROM Employees AS E  
                      WHERE E.LastName LIKE 'D%');
```


Subconsultas Correlacionadas

- Las subconsultas correlacionadas son subconsultas en las que se hace referencia a atributos que forman parte de la consulta principal. Esto significa que la subconsulta es dependiente de la consulta principal y no puede ser ejecutada de manera independiente. Lógicamente, es como si la subconsulta es evaluada por cada fila de la consulta principal.

Evaluación de una SubConsulta correlacionada

1) La consulta externa pasa un valor de columna a la consulta interna

```
SELECT OrderID, CustomerID
FROM Orders AS O
WHERE 20 < ( SELECT Quantity
              FROM [Order Details] as OD
              WHERE O.OrderID =
                  OD.OrderID
              AND OD.ProductID = 23)
```

2) La consulta interna utiliza los valores que pasa la consulta externa

3) La consulta interna devuelve un valor a la consulta externa

4) Este proceso se repite para fila siguiente de la consulta externa



Volver al paso 1

- Por ejemplo, la siguiente consulta que devuelve las órdenes con el máximo número **para cada cliente** (CustomerID):

```
SELECT CustomerID, OrderID, OrderDate, EmployeeID
FROM Orders AS 01
WHERE OrderID = (SELECT MAX(02.OrderID)
                  FROM Orders AS 02
                  WHERE 02.CustomerID = 01.CustomerID);
```

- La consulta principal se realiza sobre una instancia de la tabla Ordenes que llamamos O1, que devuelve los valores para los que el campo ordenid coinciden con el resultado de la subconsulta.
- La subconsulta filtra los resultados de una segunda instancia de la tabla Ordenes que llamamos O2, en donde el clienteid de la tabla de la subconsulta es igual al clienteid de la tabla principal (O1).

EXISTS

- También contamos en SQL con el predicado EXISTS, que devuelve verdadero (true) en caso de que la subconsulta devuelva alguna fila, de lo contrario devuelve falso (false).
- El formato de un predicado utilizando EXISTS es:
[NOT] EXISTS (<subconsulta multivaluada>)

- Por ejemplo, una consulta para obtener los clientes de España (Spain) que han realizado órdenes:

```
SELECT CustomerID, CompanyName  
FROM Customers AS C  
WHERE Country = 'Spain'  
AND EXISTS  
(SELECT * FROM Orders AS O  
WHERE O.CustomerID = C.CustomerID);
```

Not Exists vs Not In

- Necesitamos los clientes de España que **no** realizaron órdenes:

```
SELECT CustomerID, CompanyName  
FROM Customers AS C  
WHERE Country = 'Spain'  
AND NOT EXISTS  
(SELECT * FROM Orders AS O  
WHERE O.CustomerID = C.CustomerID);
```

Con NOT IN

```
SELECT CustomerID, CompanyName  
FROM Customers AS C  
WHERE Country = 'Spain'  
AND CustomerID NOT IN  
(SELECT CustomerID FROM Orders AS O);
```


¿Entonces
da lo
mismo?

```
INSERT INTO Orders
(CustomerID, EmployeeID, OrderDate,
RequiredDate, ShippedDate,
ShipVia, Freight, ShipName,
ShipAddress, ShipCity, ShipRegion,
ShipPostalCode, ShipCountry)
VALUES(NULL, 1, '20090212', '20090212',
'20090212', 1, 123.00, 'abc',
'abc', 'abc', 'abc', 'abc', 'abc')
```

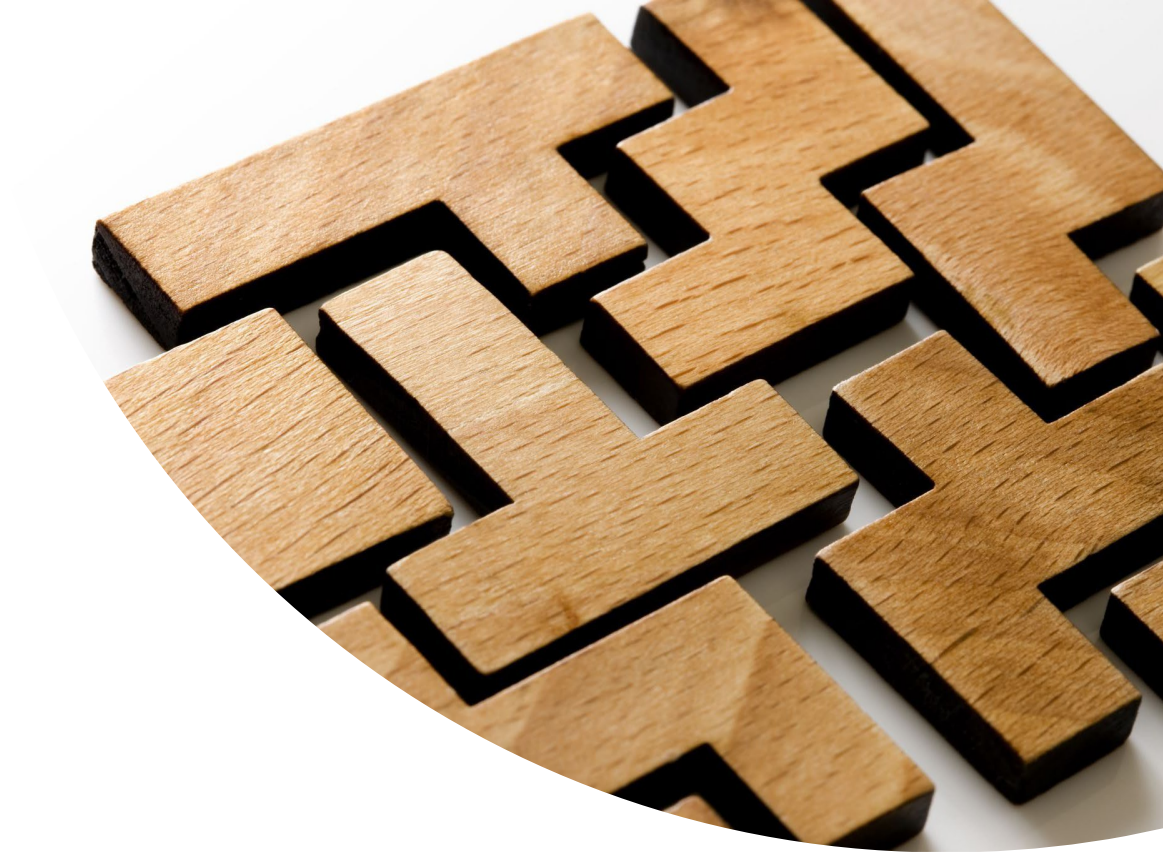
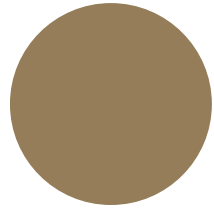
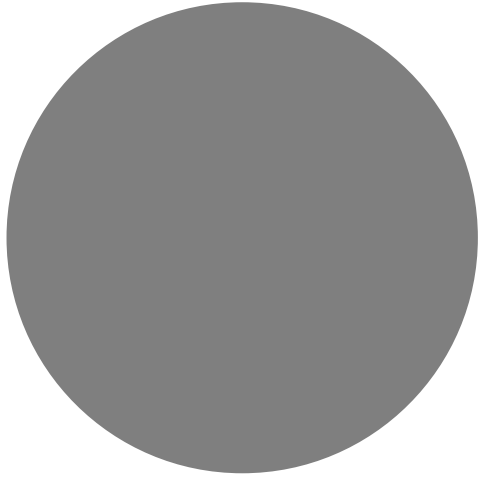
Not Exists vs Not In

- Si ejecutamos nuevamente las consultas, la de NOT EXISTS devuelve el mismo resultado
- La de NOT IN no, ¿por qué?
- La diferencia se da por la lógica de tres estados.
- La expresión val IN (val1, val2, ..., NULL) nunca puede devolver FALSE. Sólo puede devolver TRUE o DESCONOCIDO
- Por lo tanto, la expresión val NOT IN (val1, val2, ..., NULL) sólo puede devolver NOT TRUE o NOT DESCONOCIDO, ninguno de los cuales es TRUE

Not Exists vs Not In

- Para equiparar las respuestas debemos filtrar los valores NULL

```
SELECT CustomerID, CompanyName  
FROM Customers AS C  
WHERE Country = 'Spain'  
AND CustomerID NOT IN  
(SELECT CustomerID FROM Orders AS O  
WHERE CustomerID IS NOT NULL);
```



Combinación de varias Tablas JOINS

Introducción

- Uso de alias en los nombres de las tablas
- Combinación de datos de varias tablas
- Combinación de varios conjuntos de resultados



Uso de combinaciones cruzadas

```
USE joindb
SELECT buyer_name, qty
FROM buyers
CROSS JOIN sales
GO
```

Ejemplo 1

buyers

<i>buyer_id</i>	<i>buyer_name</i>
1	Adam Barr
2	Sean Chai
3	Eva Corets
4	Erin O'Melia

sales

<i>buyer_id</i>	<i>prod_id</i>	<i>qty</i>
1	2	15
1	3	5
4	1	37
3	5	11
4	2	1003

Resultado

<i>buyer_name</i>	<i>qty</i>
Adam Barr	15
Adam Barr	5
Adam Barr	37
Adam Barr	11
Adam Barr	1003
Sean Chai	15
Sean Chai	5
Sean Chai	37
Sean Chai	11
Sean Chai	1003
Eva Corets	15
...	...

JOINS

- Los joins permiten obtener información de múltiples tablas, para poder responder a preguntas como:
 - ¿Qué productos son suministrados por qué proveedores?
 - ¿Qué clientes realizaron qué órdenes?
 - ¿Qué clientes están comprando qué productos?

- **SINTAXIS**

```
SELECT tabla1.columna1, tabla2.columna2  
FROM tabla1 JOIN tabla2
```

```
ON
```

```
(tabla1.columna1=tabla2.columna1)
```

```
WHERE condiciones
```

■ Ejemplo 1 (sin un nombre de alias)

```
USE joindb
SELECT buyer_name, sales.buyer_id, qty
FROM buyers INNER JOIN sales
ON buyers.buyer_id = sales.buyer_id
GO
```

■ Ejemplo 2 (con un nombre de alias)

```
USE joindb
SELECT buyer_name, s.buyer_id, qty
FROM buyers AS b INNER JOIN sales AS s
ON b.buyer_id = s.buyer_id
GO
```


Introducción a las combinaciones

- **Selección de columnas específicas de varias tablas**

- La palabra clave JOIN especifica qué tablas se van a combinar y cómo
- La palabra clave ON especifica la condición de combinación

- **Consultas de dos o más tablas para producir un conjunto de resultados**

- Use claves principales y externas como condiciones de combinación
- Para combinar tablas, utilice columnas comunes a las tablas especificadas

- Realizar un reporte para obtener el EmployeeID y el OrderID de la tabla Ordenes no es complicado:

```
SELECT EmployeeID, OrderID  
FROM Orders;
```

- Pero para que pueda brindar información más útil, podemos hacer lo siguiente:

-- Crear un reporte que muestre las ordenes de un empleado.

```
SELECT Employees.EmployeeID, Employees.FirstName,  
Employees.LastName,  
Orders.OrderID, Orders.OrderDate  
FROM Employees JOIN Orders ON  
    (Employees.EmployeeID = Orders.EmployeeID)  
ORDER BY Orders.OrderDate;
```