

BASES DE DATOS

Lenguaje SQL

Agenda



- ❑ Introducción
- ❑ Select,
- ❑ Funciones agregadas
- ❑ Otras Funciones
- ❑ Group By
- ❑ Ejercicios

Structured Query Language

Codd, IBM, System R, Oracle

DDL: Create, Alter, Drop

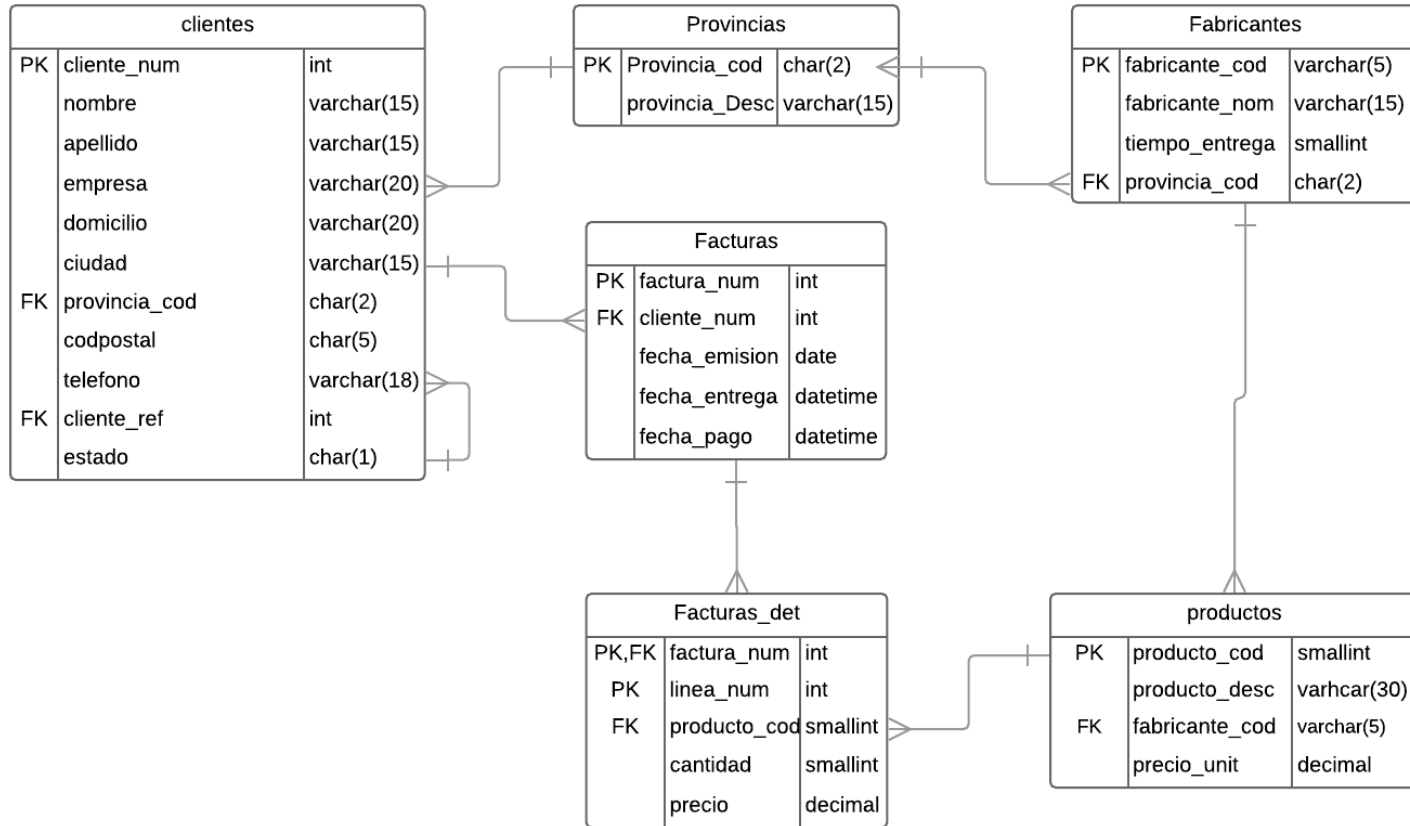
DML: Insert, Update, Delete, Select

DCL: Grant, Revoke

TCL: Begin Transaction, Commit, Rollback



DER



Operador SELECT

SELECT columna [,columna]

FROM tabla [**JOIN** tabla ...]

[**WHERE** condición]

[**GROUP BY** columna [, columna]]

[**HAVING** condición]

[**ORDER BY** columna [, columna]]

Operador SELECT

“

```
SELECT count(*)  
FROM clientes
```

```
SELECT *  
FROM clientes
```

Operador SELECT

“

```
SELECT cliente_num, nombre, apellido,  
        empresa, telefono  
FROM clientes
```

```
SELECT cliente_num, nombre, apellido,  
        empresa, telefono  
FROM clientes  
ORDER BY empresa, apellido, nombre DESC
```

Operador SELECT

“

```
SELECT cliente_num, nombre, apellido,  
        empresa, telefono  
FROM clientes  
WHERE ciudad = 'Rosario'  
ORDER BY empresa, apellido, nombre DESC
```


Operador SELECT

“ Concatenación de atributos

```
SELECT cliente_num, apellido + ', ' + nombre NombreCompleto,  
        empresa, telefono  
FROM clientes  
WHERE ciudad = 'Rosario'  
ORDER BY empresa, apellido, nombre DESC
```

Operador SELECT

“ Alias de columnas

```
SELECT cliente_num, apellido + ', ' + nombre AS apeynom,  
FROM clientes
```

```
SELECT cliente_num, apellido + ', ' + nombre apeynom,  
FROM clientes
```

Operador SELECT

“ Alias de columnas

```
SELECT cliente_num,  
       apellido + ', ' + nombre AS 'Nombre y Apellido'  
FROM clientes
```

```
SELECT cliente_num,  
       apellido + ', ' + nombre 'Nombre y Apellido'  
FROM clientes
```

Operador SELECT

“ Manejo de atributos nulos

```
SELECT cliente_num, nombre, apellido,  
       ciudad, COALESCE(ciudad, 'Desconocida') ciudad  
FROM clientes
```

Operador SELECT

“ Columnas Constantes

```
SELECT 'Producto:', producto_cod, producto_desc  
FROM productos
```

Operador SELECT

“ Columnas calculadas

```
SELECT 'Producto: ', producto_cod, producto_desc,  
       precio_unit,  
       precio_unit * 1.15  
FROM productos
```

Operador SELECT

“ Limitar cantidad de filas

```
SELECT TOP 3 producto_cod, producto_desc,  
           precio_unit * 1.15 NuevoPrecio  
FROM productos  
Order by NuevoPrecio DESC
```

Operador SELECT

“ Limitar cantidad de filas. Otra forma.

```
SELECT producto_cod, producto_desc,  
        precio_unit * 1.15 NuevoPrecio  
FROM productos  
ORDER BY 3 DESC  
OFFSET 2 ROWS  
FETCH NEXT 4 ROWS ONLY
```

Cuál sería la sintaxis equivalente a la sentencia del slide anterior?

Operador SELECT

“ Cláusula DISTINCT

```
SELECT DISTINCT producto_cod, cantidad  
FROM facturas_det
```

Cuidado !! Discrimina la fila entera, no el atributo. Salvo que se encuentre dentro de una función agregada que veremos más adelante.

Cláusula WHERE

“ OPERADORES

<, >, <=, >=, <>, !=

BETWEEN

IN, NOT IN

IS NULL, IS NOT NULL

OR, AND

WHERE - Operadores



```
SELECT factura_num  
FROM facturas  
WHERE fecha_emisión >= '2021-01-01' AND  
       fecha_emisión <= '2021-01-31'
```

```
SELECT factura_num  
FROM facturas  
WHERE fecha_emisión BETWEEN '2021-01-01' AND  
                             '2021-01-31'
```

WHERE - Operadores

“

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE teléfono IN ('1111-2222',  
                  '2222-3333')
```

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE teléfono NOT IN ('1111-2222',  
                      '2222-3333')
```

Operadores

“

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE teléfono IS NULL
```

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE teléfono IS NOT NULL
```

Operadores condicionales

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE telefono IS NULL AND Provincia_cod = 'CB'  
OR teléfono IS NOT NULL AND provincia_cod = 'TF'
```

Operadores condicionales

“

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE teléfono IS NULL AND  
    (Provincia_cod = 'CB' OR 'TF')
```

Operadores condicionales

“

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE telefono IN ('1111-2222',  
                  '2222-3333') OR  
       Provincia_cod = 'CB'
```


Uso del operador LIKE

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE nombre LIKE 'A%'
```

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE nombre LIKE '%LAN%'
```

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE nombre LIKE '%TO'
```

Uso del operador LIKE

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE nombre LIKE '[JM]%'
```

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE nombre LIKE '_[A-D]%'
```

```
SELECT cliente_num, nombre, apellido  
FROM clientes  
WHERE nombre LIKE '____DO'
```

SQL – Funciones



Funciones de Cadenas

`Len(var)`: Longitud de la cadena

`Substring(var, ini, long)`: Subcadena

`Upper(var)`, `lower(var)`: Convierte a Mayúsculas/ minúsculas

`Ltrim(var)`, `rtrim(var)`, `trim(var)`: Saca espacios.

`Left(var, long)`, `Right(var, long)`: Corta la cadena.

[* Sql Server - Funciones de cadenas](#)

SELECT – Funciones

“ Funciones de Cadenas. Ejemplos.

```
select len('abcdefgh'), SUBSTRING('abcdefgh', 3, 4),  
       left('abcdefgh', 2), right('abcdefgh', 5)|
```

6			
Results	Messages		
(No column name)	(No column name)	(No column name)	(No column name)
8	cdef	ab	defgh

SQL – Funciones



Funciones de Cadenas

ASCII(char): Nro. ASCII correspondiente al carácter pasado como parámetro.

CHAR(int): Obtiene el character correspondiente al nro. ASCII ingresado.

CONCAT(string, string, ...): Concatena cadenas.

REPLACE(string, patrón, reemplazo): Busca el **patrón** en **string** y lo reemplaza por **reemplazo**

[* Sql Server - Funciones de cadenas](#)

SELECT – Funciones

“ Funciones de Cadenas. Ejemplos.

```
select ascii('1'),  ascii('2'),  ascii('A'),  ascii('a'),  
        char(49), char(50), char(65), char(97)
```

Results Messages								
	(No column...	(No column na...	(No column na...	(No column...	(No column...	(No column ...	(No colu...	(No column name)
1	49	50	65	97	1	2	A	a

SELECT – Funciones

“ Funciones de Cadenas. Ejemplos.

```
select concat('Martin', 'Manuel', 'de', 'Güemes')
```

```
select REPLACE ('Martin Manuel de Güemes', 'Manuel', 'Miguel')
```

Results		Messages
(No column name)		
MartinManueldeGüemes		
(No column name)		
Martin Miguel de Güemes		

SQL – Funciones



Funciones de Cadenas

SPACE(int): Retorna **int** caracteres blancos

REPLICATE(char, int): Repite el caracter **char** **int** veces.

STR(numero, int1, int2): Convierte el valor **numero** a una cadena de **int1** caracteres e **int2** decimales.

SELECT – Funciones

“ Funciones de Cadenas. Ejemplos.

```
SQLQuery2.sql - L...BM59SU\herni (55))* X
select '|' + space('5') + '*' Espacios,
       REPLICATE('*', 7) Asteriscos,
       STR(1234.56) StringNumero,
       STR(1234.56, 8, 3) StringNumero2
```

Results		Messages		
		Espacios	Asteriscos	StringNumero
		5 *	*****	1235
				1234.560

SQL – Funciones



Funciones de Números

ABS(num): Retorna el valor absolute de **num**.

CEILING(num): Retorna el primer valor entero mayor o igual a **num**.

FLOOR(num): Retorna el primer valor entero menor o igual a **num**.

ROUND(num, dec): Retorna el valor entero de **num** redondeado a **dec** dígitos.

SELECT – Otras funciones



Funciones de Número

SQLQuery3.sql - L...BM59SU\herni (53))* X SQLQuery2.sql - L...BM59SU\herni (54) SQLQuery1.sql - L...BM59SU\herni (52)

```
SELECT ABS(-98.3), ABS(0.8), ABS(11.0)
```

```
SELECT CEILING(223.01), CEILING(-223.65),  
       FLOOR(223.01), FLOOR(-223.65),  
       ROUND(223.01, 1), ROUND(-223.65, 1)
```

150 %

Results Messages

	(No column name)	(No column name)	(No column name)
1	98.3	0.8	11.0

	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)
1	224	-223	223	-224	223.00	-223.70

SQL – Funciones



Más funciones de Números

POWER(num, exp): Retorna el valor **num** elevado a la **exp** potencia.

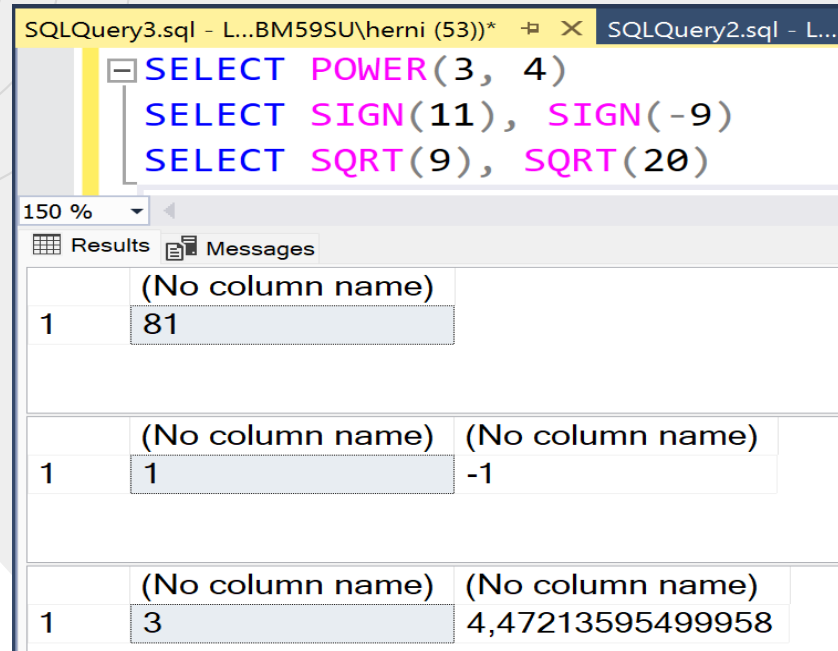
SIGN(num): Retorna 1 o -1 de acuerdo al signo de **num**.

SQRT(num): Retorna la raíz cuadrada del valor de **num**.

SELECT – Otras funciones

“

Funciones de Número



The screenshot shows a SQL query window with three statements and their results. The first statement calculates the power of 3 to the 4th power, resulting in 81. The second statement calculates the sign of 11 (1) and -9 (-1). The third statement calculates the square root of 9 (3) and 20 (4.47213595499958).

```
SQLQuery3.sql - L...BM59SU\herni (53))* SQLQuery2.sql - L...B
SELECT POWER(3, 4)
SELECT SIGN(11), SIGN(-9)
SELECT SQRT(9), SQRT(20)
```

150 %

Results Messages

(No column name)
1 81

(No column name)	(No column name)
1 1	-1

(No column name)	(No column name)
1 3	4,47213595499958

SELECT – Otras funciones

“

Funciones de FECHAS

```
select getdate(), day(Getdate()), month(getDate()), year(getDate())
```

Its Messages			
(No column name)	(No column name)	(No column name)	(No column name)
2022-03-06 11:05:27.673	6	3	2022

SQL – Funciones



Más funciones de FECHAS

`DATEADD(per, cant, fecha)`: Retorna el valor de `fecha` al cual se le suma la cantidad `cant` de periodos `per`.

`DATEDIFF(per, fecha1, fecha2)`: Retorna el resultado en periodos `per` de la resta `fecha2-fecha1`.

SELECT – Otras funciones



Funciones de FECHAS

SQLQuery3.sql - L...BM59SU\herni (53)* SQLQuery2.sql - L...BM59SU\herni (54) SQLQuery1.sql - L...BM59SU\herni (52)

```
-- SELECT DATEADD(day, 10, '2023-03-25') 'Mas 10 días',  
--         DATEADD(month, 11, '2023-03-25') 'Mas 11 Meses'  
  
-- SELECT  
--         DATEDIFF(day, '2022-03-25', '2023-01-25') 'Resta días',  
--         DATEDIFF(month, '2022-03-25', '2021-03-25') 'Resta Meses'
```

150 %

Results Messages

	Mas 10 días	Mas 11 Meses
1	2023-04-04 00:00:00.000	2024-02-25 00:00:00.000

	Resta días	Resta Meses
1	306	-12

SELECT – Otras funciones

“

Funciones de Usuario

- SYSTEM_USER
- CURRENT_USER

```
select SYSTEM_USER, CURRENT_USER
```

Messages	
(No column name)	(No column name)
LAPTOP-16RU31H6\Herni	dbo

SELECT – Otras funciones



Funciones de conversión

- **CAST(var AS tipo):** Convierte varios tipos
- **STR(var):** Convierte a cadena

```
select CAST(1234 as varchar), CAST('1234.56' AS decimal(10,2)),  
       CAST('2022-03-06' as date)
```

Messages		
(No column name)	(No column name)	(No column name)
1234	1234.56	2022-03-06

SELECT – Otras funciones

Manejo de valores NULOS

- **ISNULL(var, valor):** Devuelve **valor** si **var** es nulo.
- **COALESCE(var, valor, valor, ...):** Asigna el primer **valor** no nulo si **var** es **nulo**.

```
SQLQuery1.sql - L...RU31H6\Herni (52))*  X
select isnull(1,2), isnull(null, 2),
       coalesce(null, 3), coalesce(null, null, 4)
```

100 %

Results Messages

	(No column name)	(No column name)	(No column name)	(No column name)
1	1	2	3	4

TSql – Sentencias de control

```
declare @dia varchar(15), @numero int
set @numero = 3
SET @dia = CASE @numero when 1 then 'Lunes'
                    when 2 then 'Martes'
                    when 3 then 'Miercoles'
                    when 4 then 'Jueves'
                    when 5 then 'Viernes'
                    when 6 then 'Sabado'
                    when 7 then 'Domingo'
                    ELSE 'Invalido'
END
print @dia
```

El **CASE** tiene dos formatos

TSql – Sentencias de control

“

```
CASE when @valor < 1000 then 'Barato'  
      when @valor >= 1000 and @valor < 8000 then 'Medio'  
      when @valor >= 8000 then 'Caro'  
END
```

Operador SELECT

Funciones de grupo o agregadas.

“

Count()	COUNT(DISTINCT)
SUM()	AVG()
MAX()	MIN()

```
SELECT MIN(tiempo_entrega)
FROM fabricantes
```

SELECT – Funciones agregadas

“

Count()	COUNT(DISTINCT)
SUM()	AVG()
MAX()	MIN()

```
SELECT COUNT(*),  
       COUNT(cliente_num),  
       COUNT(DISTINCT cliente_num)  
FROM facturas
```

SELECT – Funciones agregadas

“

Count()	COUNT(DISTINCT)
SUM()	AVG()
MAX()	MIN()

```
SELECT SUM(cantidad), AVG(cantidad)
FROM facturas_det
```


SELECT – Funciones agregadas

“

Count()	COUNT(DISTINCT)
SUM()	AVG()
MAX()	MIN()

```
SELECT MAX(cliente_num), MIN(cliente_num)
FROM clientes
```

SELECT – GROUP BY, HAVING

```
SELECT columna [,columna]  
  FROM tabla [JOIN tabla ...]  
  [WHERE condición]  
  [GROUP BY columna [, columna]]  
  [HAVING condición]  
  [ORDER BY columna [, columna]]
```

SELECT – GROUP BY, HAVING

```
SELECT cliente_num, count(factura_num) cant  
FROM facturas  
GROUP BY cliente_num
```

factura_num	Cliente_num	...
1	1	
2	2	
3	2	
4	1	
5	2	
6	3	

Cliente_num	Cant
1	2
2	3
3	1

SELECT – GROUP BY, HAVING

```
SELECT cliente_num, count(factura_num) cant  
FROM facturas  
GROUP BY cliente_num  
HAVING COUNT(*) > 2
```

factura_num	Cliente_num	...
1	1	
2	2	
3	2	
4	1	
5	2	
6	3	

Cliente_num	Cant
2	3

SELECT – GROUP BY, HAVING



Ejercicio ...

Cuántas cantidades se vendieron de cada producto?

```
SELECT producto_cod, sum(cantidad)
FROM facturas_det
group by producto_cod
```



**Preguntas o
vamos a los
ejercicios ?**



Ejercicios