

# Tema 5

Realización de consultas



# Índice

## Tema 5 – Realización de consultas

---

1. Sentencia SELECT
  1. Comandos DML
  2. Cláusulas
  3. Operadores
  4. Funciones de agregado
2. Consultas básicas, filtros y ordenación
3. Subconsultas

# Índice

## Tema 5 – Realización de consultas

---

1. Sentencia SELECT
  1. Comandos DML
  2. Cláusulas
  3. Operadores
  4. Funciones de agregado
2. Consultas básicas, filtros y ordenación
3. Subconsultas

## 5.1 Sentencia SELECT

La sentencia SELECT pertenece al lenguaje de manipulación de datos (*Data Manipulation Language* en inglés).

El **DML** permite generar consultas para:

- Extraer información de la base de datos.
- Ordenar la información
- Filtrar la información
- Añadir, actualizar y eliminar información de la base de datos

## 5.1 Sentencia SELECT. Comandos DML

### Comandos DML

Comando	Descripción
<b>SELECT</b>	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
<b>INSERT</b>	Utilizado para cargar lotes de datos en la base de datos en una única operación
<b>UPDATE</b>	Utilizado para modificar los valores de los campos y registros especificados
<b>DELETE</b>	Utilizado para eliminar registros de una tabla de una base de datos

## 5.1 Sentencia SELECT. Cláusulas SQL

### Cláusulas SQL

Las cláusulas son elementos del lenguaje SQL que permiten definir los datos que se desea seleccionar o manipular y la forma en que se hará dicha selección y manipulación.

Cláusula	Descripción
<b>FROM</b>	Utilizada para especificar la tabla o tablas de las cuales se van a seleccionar los registros
<b>WHERE</b>	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
<b>GROUP BY</b>	Utilizada para separar los registros seleccionados en grupos específicos
<b>HAVING</b>	Utilizada para expresar la condición que debe satisfacer cada grupo
<b>ORDER BY</b>	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

## 5.1 Sentencia SELECT. Cláusulas SQL

### Orden de las cláusulas

Dada una sentencia SELECT que incluya todas las posibles cláusulas, el orden de ejecución de las mismas es el siguiente:

SELECT

- 1) Cláusula **FROM**
- 2) Cláusula **WHERE**
- 3) Cláusula **GROUP BY**
- 4) Cláusula **HAVING**
- 5) Extracción de campos indicados en el **SELECT**
- 6) Cláusula **ORDER BY**



```
SELECT h.producto
      FROM company c
     WHERE o.producto = c.producto
   ORDER BY 2;
```

## 5.1 Sentencia SELECT. Operadores

### Operadores SQL

Los operadores nos permitirán expresar los criterios o condiciones que deban cumplir los registros.

Los principales operadores de SQL se dividen en 2 grupos:

- Operadores **lógicos**
- Operadores **relacionales** o de comparación

**Más información**



## 5.1 Sentencia SELECT. Operadores

### Operadores lógicos

Operador	Descripción
<b>AND</b>	Es el "Y" lógico. Evalúa dos condiciones y devuelve un valor verdadero (true) sólo si ambas son ciertas
<b>OR</b>	Es el "O" lógico. Evalúa dos condiciones y devuelve un valor verdadero (true) si alguna de las dos es cierta
<b>NOT</b>	Negación lógica. Devuelve el valor contrario de la expresión. Si la expresión es true, devuelve false y viceversa

Más información



## 5.1 Sentencia SELECT. Operadores

### Operadores relacionales

[Más información](#)

Operador	Descripción
<	Menor que
>	Mayor que
< >	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual a
<b>BETWEEN</b>	Permite indicar un intervalo de valores
<b>LIKE</b>	Utilizado para comparar con un “patrón”
<b>IN</b>	Permite evaluar la pertenencia a un conjunto
<b>IS NULL</b>	Permite saber si un campo es nulo (no tiene valor)

## 5.1 Sentencia SELECT. Funciones de agregado

### Funciones de agregado

Las funciones de agregado se usan dentro de una cláusula SELECT, en grupos de registros, para devolver un único valor (sumas, promedios, máximos, mínimos, conteos).

Función	Descripción
<b>AVG</b>	Utilizada para calcular el promedio (media aritmética) de los valores de un campo determinado
<b>COUNT</b>	Utilizada para devolver el número de registros de la selección
<b>SUM</b>	Utilizada para devolver la suma de todos los valores de un campo determinado
<b>MAX</b>	Utilizada para devolver el valor mayor de un campo especificado
<b>MIN</b>	Utilizada para devolver el valor menor de un campo especificado

Más información



## Actividad 1

Hacer un **restore** de la base de datos **Neptuno** que se utilizará para el seguimiento de la formación y la realización de ejercicios

# Índice

## Tema 5 – Realización de consultas

---

1. Sentencia SELECT
  1. Comandos DML
  2. Cláusulas
  3. Operadores
  4. Funciones de agregado
2. Consultas básicas, filtros y ordenación
3. Subconsultas

## 5.2 Consultas básicas. Opciones del SELECT

### Opciones del SELECT

SELECT ALL / \* / DISTINCT / [campo o lista de campos]

- **ALL:** Indica que queremos seleccionar todos los campos de la tabla. Es el valor por defecto y no suele especificarse casi nunca de este modo.
- **\*:** equivalente a ALL. Otra forma de expresarlo mucho más habitual
- **DISTINCT :** Indica que queremos extraer sólo los valores distintos que hay almacenados en un determinado campo
- **Campo o lista de campos:** se indica el campo que queremos seleccionar de la tabla. Si queremos seleccionar más de uno, los sepáramos por comas.

Más información

## 5.2 Consultas básicas. Opciones del SELECT

### Ejemplo 01

Mostrar todos los campos de la tabla CATEGORÍAS

SQL

```
SELECT * FROM categorias
```

IdCategoria	NombreCategoria	Descripcion	Imagen
1	Bebidas	Gaseosas, café, té, cervezas y maltas	BLOB
2	Condimentos	Salsas dulces y picantes, delicias, comida para untar y aderezos	BLOB
3	Repostería	Postres, dulces y pan dulce	BLOB
4	Lácteos	Quesos	BLOB
5	Granos/Cereales	Pan, galletas, pasta y cereales	BLOB
6	Carnes	Carnes preparadas	BLOB
7	Frutas/Verduras	Frutas secas y queso de soja	BLOB
8	Pescado/Marisco	Pescados, mariscos y algas	BLOB

## 5.2 Consultas básicas. Opciones del SELECT

### Ejemplo 02

Mostrar todos los países distintos de la tabla de CLIENTES

SQL

```
SELECT DISTINCT Pais FROM clientes
```

Pais
Alemania
México
Reino Unido
Suecia
Francia
España
Canadá
Argentina
Suiza
Brasil
Austria
Italia
Portugal
Estados Unidos
Venezuela
Irlanda
Bélgica
Noruega
Dinamarca
Finlandia
Polonia

## 5.2 Consultas básicas. Criterios en campos numéricos/fecha

### Ejemplo 03

Mostrar el nombreProducto y precioUnidad de todos los PRODUCTOS cuyo precioUnidad sea superior a 100

SQL

```
SELECT nombreProducto, precioUnidad  
FROM productos  
WHERE precioUnidad>100
```

nombreproducto	preciounidad
Salchicha Thüringer	123.7900
Vino Côte de Blaye	263.5000

## 5.2 Consultas básicas. Criterios en campos numéricos/fecha

### Ejemplo 04

Mostrar el *nombreProducto*, *precioUnidad* y *unidadesEnExistencia* de todos los PRODUCTOS con *precioUnidad* comprendido entre 75 y 100

SQL

```
SELECT nombreProducto, precioUnidad, unidadesEnExistencia  
FROM productos  
WHERE precioUnidad BETWEEN 75 AND 100
```

nombreProducto	precioUnidad	unidadesEnExistencia
Buey Mishí Kobe	97.0000	29
Mermelada de Sir Rodney's	81.0000	40

## 5.2 Consultas básicas. Criterios en campos numéricos/fecha

### Ejemplo 05

Mostrar el nombre, apellidos y fechaContratacion de todos los EMPLEADOS contratados (*FechaContratacion*) antes del 01-02-1993

SQL

```
SELECT nombre, apellidos, fechaContratacion  
FROM empleados  
WHERE FechaContratacion<'1993-02-01'
```

nombre	apellidos	fechaContratacion
Nancy	Davolio	1992-05-01 00:00:00
Andrew	Fuller	1992-08-14 00:00:00
Janet	Leverling	1992-04-01 00:00:00

La representación de las fechas varía de un sistema de gestión de bases de datos (SGBD) a otro. En MySQL se expresan en formato:

año-mes-dia

y delimitadas por comillas

## 5.2 Consultas básicas. Criterios en campos numéricos/fecha

### Ejemplo 06

Mostrar destinatario, fechaPedido y cargo de todos los PEDIDOS del segundo trimestre del año 1997

SQL

```
SELECT destinatario, fechaPedido, cargo  
FROM pedidos  
WHERE fechaPedido BETWEEN '1997-04-01' AND '1997-06-30'
```

destinatario	fechaPedido	cargo
Bottom-Dollar Markets	1997-04-01 00:00:00	62.8900
La maison d'Asie	1997-04-02 00:00:00	10.6400
Comércio Mineiro	1997-04-02 00:00:00	65.9900
Laughing Bacchus Wine Cellars	1997-04-03 00:00:00	4.6500
Tradição Hipermercados	1997-04-04 00:00:00	46.7700
Lehmanns Marktstand	1997-04-04 00:00:00	36.2100
HILARIÓN-Abastos	1997-04-07 00:00:00	29.7500
LILA-Supermercado	1997-04-08 00:00:00	102.0200
La maison d'Asie	1997-04-09 00:00:00	42.6800

• • •

## 5.2 Consultas básicas. Búsqueda de patrones en cadenas

### Búsqueda de patrones en campos de texto

Los patrones SQL permiten:

- Emplear el carácter '\_' para representar coincidencia con un carácter individual.
- El carácter '%' para 0 ó más caracteres.
- Los patrones SQL no son case-sensitive (no distinguen mayúsculas de minúsculas).
- No se emplean los operadores = o <> para trabajar con patrones SQL, en lugar de eso se usan los operadores de comparación **LIKE** o **NOT LIKE**.

Más información

## 5.2 Consultas básicas. Búsqueda de patrones en cadenas

### Ejemplo 07

Mostrar todos los CLIENTES cuyo nombreCompañía comience por 'A'

SQL

```
SELECT *
FROM clientes
WHERE nombrecompania LIKE "a%";
```

IdCliente	NombreCompañía	NombreContacto	CargoContacto	Direccion	Ciudad
ALFKI	Alfreds Futterkiste	Maria Anders	Representante de ventas	Obere Str. 57	Berlín
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Propietario	Avda. de la Constitución 2222	México D.F.
ANTON	Antonio Moreno Taquería	Antonio Moreno	Propietario	Mataderos 2312	México D.F.
AROUT	Around the Horn	Thomas Hardy	Representante de ventas	120 Hanover Sq.	Londres

## 5.2 Consultas básicas. Búsqueda de patrones en cadenas

### Ejemplo 08

Mostrar todos los CLIENTES cuyo nombreCompañía tenga una 'A' en el segundo carácter

SQL

```
SELECT * FROM clientes WHERE nombreCompañía LIKE "_a%";
```

IdCliente	NombreCompañía	NombreContacto	CargoContacto	Direccion	Ciudad
CACTU	Cactus Comidas para llevar	Patricia Simpson	Agente de ventas	Cerrito 333	Buenos Aires
EASTC	Eastern Connection	Ann Devon	Agente de ventas	35 King George	Londres
FAMIA	Familia Arquibaldo	Aria Cruz	Asistente de marketing	Rua Orós, 92	Sao Paulo
GALED	Galería del gastrónomo	Eduardo Saavedra	Gerente de marketing	Rambla de Cataluña, 23	Barcelona
HANAR	Hanari Carnes	Mario Pontes	Gerente de contabilidad	Rua do Paço, 67	Rio de Janeiro
LACOR	La corne d'abondance	Daniel Tonini	Representante de ventas	67, avenue de l'Europe	Versalles
LAMAI	La maison d'Asie	Annette Roulet	Gerente de ventas	1 rue Alsace-Lorraine	Toulouse
LAUGB	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	Asistente de marketing	1900 Oak St.	Vancouver
LAZYK	Lazy K Kountry Store	John Steel	Gerente de marketing	12 Orchestra Terrace	Walla Walla
MAGAA	Magazzini Alimentari Riuniti	Giovanni Rovelli	Gerente de marketing	Via Ludovico il Moro 22	Bérgamo
MAISD	Maison Dewey	Catherine Dewey	Agente de ventas	Rue Joseph-Bens 532	Bruselas
PARIS	Paris spécialités	Marie Bertrand	Propietario	265, boulevard Charonne	París
RANCH	Rancho grande	Sergio Gutiérrez	Representante de ventas	Av. del Libertador 900	Buenos Aires
RATTC	Rattlesnake Canyon Grocery	Paula Wilson	Representante agente ventas	2817 Milton Dr.	Albuquerque
SANTG	Santé Gourmet	Jonas Bergulfsen	Propietario	Erling Skakkes gate 78	Stavern
SAVEA	Save-a-lot Markets	Jose Pavarotti	Representante de ventas	187 Suffolk Ln.	Boise
VAFFE	Vaffeljernet	Palle Ibsen	Gerente de ventas	Smagsløget 45	Århus
WARTH	Wartian Herkku	Pirkko Koskitalo	Gerente de contabilidad	Torikatu 38	Oulu

## 5.2 Consultas básicas. Búsqueda de patrones en cadenas

### Ejemplo 09

Mostrar todos los CLIENTES cuyo nombreCompañía finalice con el texto "store"

SQL

```
SELECT *
FROM clientes
WHERE nombrecompania LIKE "%store";
```

IdCliente	NombreCompañía	NombreContacto	CargoContacto	Direccion	Ciudad
HUNGC	Hungry Coyote Import Store	Yoshi Latimer	Representante de ventas	City Center Plaza 1516 Main St.	Elgin
LAZYK	Lazy K Kountry Store	John Steel	Gerente de marketing	12 Orchestra Terrace	Walla Walla

## 5.2 Consultas básicas. Búsqueda de patrones en cadenas

### Ejemplo 10

Mostrar todos los CLIENTES cuyo nombreCompañía NO comience por 'A'

SQL

```
SELECT * FROM clientes WHERE nombreCompañía NOT LIKE "a%";
```

IdCliente	NombreCompañía	NombreContacto	CargoContacto	Direccion	Ciudad
BERGS	Berglunds snabbköp	Christina Berglund	Administrador de pedidos	Berguvsvägen 8	Luleå
BLAUS	Blauer See Delikatessen	Hanna Moos	Representante de ventas	Forsterstr. 57	Mannheim
BLONP	Blondel père et fils	Frédérique Citeaux	Gerente de marketing	24, place Kléber	Estrasburgo
BOLID	Bólido Comidas preparadas	Martín Sommer	Propietario	C/ Araquil, 67	Madrid
BONAP	Bon app'	Laurence Lebihan	Propietario	12, rue des Bouchers	Marsella
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Gerente de contabilidad	23 Tsawassen Blvd.	Tsawassen
BSBEV	B's Beverages	Victoria Ashworth	Representante de ventas	Fauntleroy Circus	Londres
CACTU	Cactus Comidas para llevar	Patricia Simpson	Agente de ventas	Cerrito 333	Buenos Aires
CENTC	Centro comercial Moctezuma	Francisco Chang	Gerente de marketing	Sierras de Granada 9993	México D.F.
CHOPS	Chop-suey Chinese	Yang Wang	Propietario	Hauptstr. 29	Berna
COMM1	Comércio Mineiro	Pedro Afonso	Asistente de ventas	Av. dos Lusíadas, 23	São Paulo
CONSH	Consolidated Holdings	Elizabeth Brown	Representante de ventas	Berkeley Gardens 12 Brewery	Londres
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Administrador de pedidos	Walserweg 21	Aachen

• • •

## 5.2 Consultas básicas. Búsqueda de patrones en cadenas

### Ejemplo 11

Mostrar todos los PRODUCTOS cuyo nombreProducto contenga el texto "chocolate" en cualquier parte del nombreProducto

SQL

```
SELECT *
FROM productos
WHERE nombreproducto LIKE "%chocolate%";
```

IdProducto	NombreProducto	IdProveedor	IdCategoria	CantidadPorUnidad	PrecioUnidad
19	Pastas de té de chocolate	8	3	10 cajas x 12 piez...	9.2000
25	Crema de chocolate y nueces NuNuCa	11	3	20 - vasos 450 g	14.0000
27	Chocolate Schoggi	11	3	100 - piezas 100 g	43.9000
48	Chocolate holandés	22	3	10 paq.	12.7500
50	Chocolate blanco	23	3	12 - barras 100 g	16.2500

## 5.2 Consultas básicas. Operador IS

### Operador IS

Se utiliza habitualmente para comparar una expresión o valor con un valor booleano, el cual puede ser *TRUE* (verdadero), *FALSE* (falso) o *UNKNOWN* (sin valor asignado).

### Ejemplo 12

Mostrar *nombreproducto* y *preciounidad* de todos los PRODUCTOS que se encuentren suspendidos. El valor del campo suspendido tiene que ser distinto de 0 (0 = FALSE).

```
SELECT nombreproducto, preciounidad  
FROM productos  
WHERE suspendido IS TRUE;
```

Más información

nombreproducto	preciounidad
Mezcla Gumbo del chef Anton	21.3500
Buey Mishi Kobe	97.0000
Cordero Alice Springs	39.0000
Refresco Guaraná Fantástica	4.5000
Col fermentada Rössle	45.6000
Salchicha Thüringer	123.7900
Tallarines de Singapur	14.0000
Empanada de carne	32.8000

## 5.2 Consultas básicas. Operador IS

### Ejemplo 13

Mostrar nombrecompañia, ciudad, pais, fax de los CLIENTES de los cuales no tengamos su número de fax (el valor del campo fax sea nulo).

SQL

```
SELECT nombrecompañia, ciudad, pais, fax  
FROM clientes  
WHERE fax IS NULL;
```

nombrecompañia	ciudad	pais	fax
Antonio Moreno Taquería	México D.F.	México	NULL
B's Beverages	Londres	Reino Unido	NULL
Chop-suey Chinese	Berna	Suiza	NULL
Comércio Mineiro	São Paulo	Brasil	NULL
Familia Arquibaldo	Sao Paulo	Brasil	NULL
Folk och fä HB	Bräcke	Suecia	NULL
Godos Cocina Típica	Sevilla	España	NULL
Gourmet Lanchonetes	Campinas	Brasil	NULL
Great Lakes Food Market	Eugenia	Estados Unidos	NULL
Island Trading	Cowes	Reino Unido	NULL
Königlich Essen	Brandenburgo	Alemania	NULL
Let's Stop N Shop	San Francisco	Estados Unidos	NULL

• • •



## 5.2 Consultas básicas. Operador IS

### Ejemplo 14

Mostrar los *nombreCompañia*, *ciudad*, *region* y *pais* de todos los PROVEESORES de los que conozcamos su región.

**SQL**

```
SELECT nombrecompañia, ciudad, region, pais  
FROM proveedores  
WHERE region IS NOT NULL;
```

nombrecompañia	ciudad	region	pais
New Orleans Cajun Delights	New Orleans	LA	Estados Unidos
Grandma Kelly's Homestead	Ann Arbor	MI	Estados Unidos
Cooperativa de Quesos 'Las Cabras'	Oviedo	Asturias	España
Pavlova, Ltd.	Melbourne	Victoria	Australia
Bigfoot Breweries	Bend	OR	Estados Unidos
New England Seafood Cannery	Boston	MA	Estados Unidos
G'day, Mate	Sydney	NSW	Australia
Ma Maison	Montréal	Québec	Canadá
Forêts d'érables	Ste-Hyacinthe	Québec	Canadá

## 5.2 Consultas básicas. Operador IN (pertenencia a conjunto)

### Operador IN

Lo utilizamos para comparar una expresión o valor con una lista de valores. Nos da como resultado 1 (true) cuando la expresión o valor se encuentra en la lista de valores y nos da 0 (false) cuando no se encuentra.

[Más información](#)

### Ejemplo 15

Mostrar los *nombreCompañia*, *ciudad*, *region* y *pais* de todos los PROVEEDORES de Oviedo, Madrid o Berlín.

```
SELECT nombrecompañia, ciudad, region, pais  
FROM proveedores  
WHERE ciudad IN ('Oviedo', 'Madrid', 'Berlin');
```

nombrecompañia	ciudad	region	pais
Cooperativa de Quesos 'Las Cabras'	Oviedo	Asturias	España
Heli Süßwaren GmbH & Co. KG	Berlín	NULL	Alemania

## 5.2 Consultas básicas. Funciones SQL

### Funciones SQL

MySQL dispone de multitud de funciones que pueden utilizarse en las cláusulas WHERE, HAVING, SELECT, etc. No necesariamente se utilizan de forma exclusiva para expresar condiciones, como veremos más adelante.

Indicaremos algunas de ellas en función del tipo de datos al que se aplica:

- Funciones de cadenas de caracteres
- Funciones numéricas
- Funciones de fecha y hora
- Funciones varias

[Más información](#)

## 5.2 Consultas básicas. Funciones SQL de cadenas

### Funciones SQL de cadenas de caracteres

Algunas de las funciones más utilizadas de este tipo son:

- **CHAR\_LENGTH**: devuelve la longitud de la cadena que se le pasa como argumento.
- **CONCAT**: concatena los argumentos que recibe como parámetro.
- **CONCAT\_WS**: concatena como la anterior pero inserta un carácter separador.
- **INSERT**: permite insertar una cadena en otra.
- **INSTR**: comprueba si una cadena está contenida en otra.
- **LCASE** o **LOWER**: convierte una cadena a minúsculas.
- **LEFT / RIGHT**: extrae por la izquierda / derecha tantos caracteres de una cadena como indiquemos.
- **LENGTH**: retorna la longitud de la cadena.

## 5.2 Consultas básicas. Funciones SQL de cadenas

### Funciones SQL de cadenas de caracteres

- **LOAD\_FILE**: lee un fichero y lo retorna como cadena de caracteres
- **LOCATE**: similar a INSTR, busca la primera ocurrencia de una cadena dentro de otra.
- **LPAD / RPAD**: retorna la cadena alineada a la izquierda/derecha n caracteres.
- **LTRIM / RTRIM / TRIM**: retorna la cadena eliminando los espacios en blanco que hubiera a la izquierda / derecha / izquierda y derecha respectivamente.
- **MID / SUBSTRING**: permite extraer una parte de la cadena.
- **REPEAT**: permite repetir una cadena tantas veces como se desee.
- **REPLACE**: sustituye las apariciones de una cadena por otra dentro de una cadena.
- **REVERSE**: retorna la cadena en orden inverso.
- **SPACE**: retorna una cadena con tantos espacios en blanco como se indique.
- **UCASE**: convierte la cadena a mayúsculas.

[Más información](#)

## 5.2 Consultas básicas. Funciones SQL de cadenas

### Ejemplo 16

Mostrar los *nombreCompañia*, *ciudad* y *codpostal* de todos los CLIENTES cuyo código postal tenga una longitud de 5 caracteres.

SQL

```
SELECT nombrecompañia, ciudad, codpostal  
FROM clientes  
WHERE CHAR_LENGTH(codpostal)=5
```

nombrecompañia	ciudad	codpostal
Alfreds Futterkiste	Berlín	12209
Ana Trujillo Emparedados y helados	México D.F.	05021
Antonio Moreno Taquería	México D.F.	05023
Blauer See Delikatessen	Mannheim	68306
Blondel père et fils	Estrasburgo	67000
Bólido Comidas preparadas	Madrid	28023
Bon app'	Marsella	13008
Centro comercial Moctezuma	México D.F.	05022
Drachenblut Delikatessen	Aachen	52066
Du monde entier	Nantes	44000
FISSA Fabrica Inter. Salchichas S.A.	Madrid	28034
Folies gourmandes	Lille	59000

• • •

## 5.2 Consultas básicas. Funciones SQL de cadenas

### Ejemplo 17

Mostrar de la tabla de EMPLEADOS, el nombre, apellidos y cargo con el siguiente formato: Nombre Apellidos (Cargo)

SQL

```
SELECT CONCAT(nombre," ",apellidos," (",cargo,")") AS Empleado  
FROM empleados
```

Empleado
Nancy Davolio (Representante de ventas)
Andrew Fuller (Vicepresidente comercial)
Janet Leverling (Representante de ventas)
Margaret Peacock (Representante de ventas)
Steven Buchanan (Gerente de ventas)
Michael Suyama (Representante de ventas)
Robert King (Representante de ventas)
Laura Callahan (Coordinador ventas interno)
Anne Dodsworth (Representante de ventas)

El comando **AS** se usa para cambiar el nombre de una columna o tabla con un alias.  
Solo existe mientras se ejecuta la consulta.

## 5.2 Consultas básicas. Funciones SQL de cadenas

### Ejemplo 18

Mostrar los *nombreproducto*, *preciounidad* y el *nombreproducto* reemplazando el texto 'Queso' por 'CHEESE' de todos los PRODUCTOS que contengan el texto 'queso' en cualquier parte del nombre.

SQL

```
SELECT nombreproducto,  
       REPLACE(nombreproducto,'Queso','CHEESE') AS Producto,  
       preciounidad  
  FROM productos  
 WHERE nombreproducto LIKE '%queso%';
```

nombreproducto	Producto	preciounidad
Queso Cabrales	CHEESE Cabrales	21.0000
Queso Manchego La Pastora	CHEESE Manchego La Pastora	38.0000
Queso gorgonzola Telino	CHEESE gorgonzola Telino	12.5000
Queso Mascarpone Fabioli	CHEESE Mascarpone Fabioli	32.0000
Queso de cabra	CHEESE de cabra	2.5000
Raclet de queso Courdavault	Raclet de queso Courdavault	55.0000
Queso Gudbrandsdals	CHEESE Gudbrandsdals	36.0000

• • •



## 5.2 Consultas básicas. Funciones numéricas de SQL

### Algunas funciones SQL numéricas

- **ABS**: valor absoluto.
- **CEILING /CEIL**: redondea hacia arriba al entero más próximo.
- **DEGREES**: convierte el argumento de radianes a grados.
- **FLOOR**: redondea hacia abajo al entero más próximo.
- **MOD**: devuelve el módulo o resto de dividir sus dos argumentos.
- **POW / POWER**: permite calcular potencias.
- **RADIANS**: devuelve el argumento convertido de grados a radianes.
- **RAND**: devuelve un aleatorio en coma flotante en el intervalo [0.0... 1.0]
- **ROUND**: redondea al entero más cercano.
- **TRUNCATE**: devuelve el número truncado a los decimales que le digamos.

Más información

## 5.2 Consultas básicas. Funciones numéricas de SQL

### Ejemplo 19

Mostrar los *precioUnidad* de todos los PRODUCTOS aplicándoles todas las funciones de redondeo (ROUND, FLOOR, CEIL). Mostrar únicamente los productos que tienen *idCategoria* con valor 1.

SQL

```
SELECT nombreproducto, preciounidad,  
       CEIL(preciounidad) AS 'Hacia ARRIBA',  
       FLOOR(preciounidad) AS 'Hacia ABAJO',  
       ROUND(preciounidad) AS Redondeo  
FROM productos  
WHERE idcategoria=1;
```

## 5.2 Consultas básicas. Funciones numéricas de SQL

### Ejemplo 19. Resultados

nombreproducto	preciounidad	Hacia ARRIBA	Hacia ABAJO	Redondeo
Té Dharamsala	18.0000	18	18	18
Cerveza tibetana Barley	19.0000	19	19	19
Refresco Guaraná Fantástica	4.5000	5	4	5
Cerveza Sasquatch	14.0000	14	14	14
Cerveza negra Steeleye	18.0000	18	18	18
Vino Côte de Blaye	263.5000	264	263	264
Licor verde Chartreuse	18.0000	18	18	18
Café de Malasia	46.0000	46	46	46
Cerveza Laughing Lumberjack	14.0000	14	14	14
Cerveza Outback	15.0000	15	15	15
Cerveza Klosterbier Rhönbräu	7.7500	8	7	8
Licor Cloudberry	18.0000	18	18	18

## 5.2 Consultas básicas. Funciones SQL de fecha/hora

### Funciones de fecha/hora

- **ADDDATE**: permite añadir días a una fecha.
- **ADDTIME**: permite añadir tiempo a un dato de tipo fecha/hora.
- **CONVERT\_TZ**: convierte una hora a otra zona horaria.
- **CURDATE/CURRENT\_DATE / CURTIME/CURRENT\_TIME**: retorna la fecha/hora actual.
- **DATE**: extrae de algo de tipo fecha y hora la parte relativa a la fecha.
- **DATEDIFF**: permite calcular la diferencia en días entre dos fechas.
- **DATE\_ADD**: permite añadir un intervalo de tiempo (no necesariamente días) a una fecha.
- **DATE\_FORMAT**: permite formatear fechas.
- **DAY**: devuelve el día del mes.
- **DAYNAME**: devuelve el nombre del día de la semana (inglés)
- **DAYOFWEEK**: devuelve el índice del día de la semana de una fecha (1=domingo,...,6=sábado)

## 5.2 Consultas básicas. Funciones SQL de fecha/hora

### Ejemplo 20

Mostrar de la tabla de PEDIDOS: *idcliente*, *idpedido*, *fechapedido*, *fechaenvio*, *fechaentrega*, número de días transcurridos desde que se envió el pedido hasta que lo recibió el cliente y cargo del pedido de todos los pedidos cuyo *idCliente* sea 'WELLI'.

**SQL**

```
SELECT idcliente, idpedido, fechapedido, fechaenvio, fechaentrega,  
      DATEDIFF(fechaentrega, fechaenvio)  
            AS 'Tiempo transporte (días)', cargo  
      FROM pedidos  
     WHERE idcliente='WELLI';
```

**DATEDIFF**: permite calcular la diferencia en días entre dos fechas. Recibe como argumentos, primero la fecha más reciente, seguida de la fecha más antigua

## 5.2 Consultas básicas. Funciones SQL de fecha/hora

### Ejemplo 20. Resultados

idcliente	idpedido	fechapedido	fechaenvio	fechaentrega	Tiempo transporte (días)	cargo
WELLI	10256	1996-07-15 00:00:00	1996-07-17 00:00:00	1996-08-12 00:00:00	26	13.9700
WELLI	10420	1997-01-21 00:00:00	1997-01-27 00:00:00	1997-02-18 00:00:00	22	44.1200
WELLI	10585	1997-07-01 00:00:00	1997-07-10 00:00:00	1997-07-29 00:00:00	19	13.4100
WELLI	10644	1997-08-25 00:00:00	1997-09-01 00:00:00	1997-09-22 00:00:00	21	0.1400
WELLI	10803	1997-12-30 00:00:00	1998-01-06 00:00:00	1998-01-27 00:00:00	21	55.2300
WELLI	10809	1998-01-01 00:00:00	1998-01-07 00:00:00	1998-01-29 00:00:00	22	4.8700
WELLI	10900	1998-02-20 00:00:00	1998-03-04 00:00:00	1998-03-20 00:00:00	16	1.6600
WELLI	10905	1998-02-24 00:00:00	1998-03-06 00:00:00	1998-03-24 00:00:00	18	13.7200
WELLI	10935	1998-03-09 00:00:00	1998-03-18 00:00:00	1998-04-06 00:00:00	19	47.5900



## Actividad 2

Realiza las siguientes consultas

- 1) Mostrar todos los datos de los *clientes* cuyo campo *NombreCompañía* tenga una longitud inferior a 10 caracteres.
- 2) Mostrar los datos de todos los proveedores añadiendo un nuevo campo que muestre concatenados el *CodPostal* y la *Ciudad*. Este campo tendrá el alias '*Ubicación*'.
- 3) Mostrar el *NombreProducto* y el *PrecioUnidad* de todos los productos pero mostrando los nombres de los productos en mayúsculas.
- 4) Mostrar todos los datos de los pedidos indicando cuántos días tardó en llegar el pedido cliente (diferencia entre la *fechaEntrega* y la *fechaPedido*)

## 5.2 Consultas básicas. Funciones de agregado

### Funciones de agregado

SQL dispone de otras funciones llamadas **funciones de agregado**. Su nombre indica que son funciones que se aplicarán a **grupos de registros**.

Más adelante veremos cómo hacer grupos de registros.

Estas funciones también son válidas para hacer cálculos sobre todos los registros de la tabla o tablas.

Algunas de estas funciones son las siguientes:

- **AVG**: devuelve la **media** de los valores de un campo.
- **COUNT**: **cuenta** el número de registros.
- **COUNT(DISTINCT)**: **cuenta** el número de **valores distintos** en un campo.
- **MAX**: devuelve el valor **máximo** de un campo.
- **MIN**: devuelve el valor **mínimo** de un campo.
- **SUM**: devuelve la **suma** de un campo.

## 5.2 Consultas básicas. Funciones de agregado

### Ejemplo 21

¿Cuántos CLIENTES tenemos?

SQL

```
SELECT COUNT(*) AS 'Total clientes' FROM clientes;
```

Total clientes
91

## 5.2 Consultas básicas. Funciones de agregado

### Ejemplo 22

Mostrar el número de países diferentes que hay en la tabla de CLIENTES.

SQL

```
SELECT COUNT(DISTINCT pais) AS 'Paises' FROM clientes;
```

Paises
21

## 5.2 Consultas básicas. Funciones de agregado

### Ejemplo 23

Mostrar el cargo medio de los PEDIDOS realizados en el año 1996.

SQL

```
SELECT AVG(cargo) AS 'Cargo medio'  
FROM pedidos  
WHERE fechapedido BETWEEN '1996-01-01' AND '1996-12-31';
```

Cargo medio
-------------

67.63072368
-------------

## 5.2 Consultas básicas. Funciones de agregado

### Ejemplo 24

Calcular los *precioUnidad*: mínimo, máximo y medio de todos los productos.

**SQL**

```
SELECT MIN(precioUnidad) AS 'Precio mínimo',  
       MAX(precioUnidad) AS 'Precio máximo',  
       AVG(precioUnidad) AS 'Precio medio'  
FROM productos;
```

Precio mínimo	Precio máximo	Precio medio
2.5000	263.5000	28.86636364

## 5.2 Consultas básicas. Funciones de agregado

### Ejemplo 25

Calcular la suma de los cargos de todos los pedidos con *fechaPedido* del año 1996.

**SQL**

```
SELECT SUM(cargo) AS 'Ventas 1996'  
FROM pedidos  
WHERE fechapedido BETWEEN '1996-01-01' AND '1996-12-31';
```

Ventas 1996
10279.8700

## Actividad 3

Realiza las siguientes consultas

- 1) Calcular las *fechapedido* de los PEDIDOS más antiguo y más reciente.
- 2) Calcular cuántos PEDIDOS ha hecho el empleado con idEmpleado = 1
- 3) Queremos saber cuántas unidades hay en stock de media de los PRODUCTOS. Se trata de calcular la media del campo *UnidadesEnExistencia*.



## 5.2 Consultas básicas. Cálculos en consultas

### Cálculos en consultas

En los SELECT como en otras partes de las consultas están admitidos los cálculos. Por ejemplo:

#### Ejemplo 26

Queremos hacer un descuento del 25% en todos nuestros PRODUCTOS. Mostrar el nombreProducto, el precioUnidad sin y con descuento.

**SQL**

```
SELECT nombreProducto,  
       precioUnidad AS 'Precio actual',  
       (precioUnidad * 0.75) AS 'Precio con Dto.'  
FROM productos;
```

## 5.2 Consultas básicas. Cálculos en consultas

nombre	Producto	Precio actual	Precio con Dto.
Té Dharamsala		18.0000	13.500000
Cerveza tibetana Barley		19.0000	14.250000
Sirope de regaliz		10.0000	7.500000
Especias Cajun del chef Anton		22.0000	16.500000
Mezcla Gumbo del chef Anton		21.3500	16.012500
Mermelada de grosellas de la abuela		25.0000	18.750000
Peras secas orgánicas del tío Bob		30.0000	22.500000
Salsa de arándanos Northwoods		40.0000	30.000000
Buey Mishí Kobe		97.0000	72.750000
Pez espada		31.0000	23.250000
Queso Cabrales		21.0000	15.750000
Queso Manchego La Pastora		38.0000	28.500000
Algas Konbu		6.0000	4.500000
Cuajada de judías		23.2500	17.437500
Salsa de soja baja en sodio		15.5000	11.625000
Postre de merengue Pavlova		17.4500	13.087500
Cordero Alice Springs		39.0000	29.250000
Langostinos tigre Carnarvon		62.5000	46.875000

• • •

## 5.2 Consultas básicas. Cálculos en consultas

### Ejemplo 27

Queremos ver en una consulta sobre la tabla de PEDIDOS: el *idpedido*, *idcliente*, *idempleado*, *fechapedido*, *fechaentrega*, número de días\* transcurridos desde el pedido hasta la entrega del mismo al cliente, cargo del pedido, comisión del vendedor\* (15%) y beneficio neto\* (cargo-comisión: 85%). Solamente se contemplarán los pedidos con *FechaPedido* a partir del 1 de enero de 1997 y que se haya tardado menos de 30 días en entregar.

**SQL**

```
SELECT idpedido, idcliente, idempleado, fechapedido,  
       fechaentrega,  
       DATEDIFF(fechaentrega,fechapedido) AS 'Días envío',  
       cargo, (cargo * 0.15) AS Comisión, (cargo * 0.85) AS Neto  
FROM pedidos  
WHERE fechapedido > '1997-01-01' AND  
      DATEDIFF(fechaentrega,fechapedido)<30;
```

Los campos con \* son calculados

## 5.2 Consultas básicas. Cálculos en consultas

idpedido	idcliente	idempleado	fechapedido	fechaentrega	Días envío	cargo	Comisión	Neto
10403	ERNSH	4	1997-01-03 00:00:00	1997-01-31 00:00:00	28	73.7900	11.068500	62.721500
10404	MAGAA	2	1997-01-03 00:00:00	1997-01-31 00:00:00	28	155.9700	23.395500	132.574500
10405	LINOD	1	1997-01-06 00:00:00	1997-02-03 00:00:00	28	34.8200	5.223000	29.597000
10407	OTTIK	2	1997-01-07 00:00:00	1997-02-04 00:00:00	28	91.4800	13.722000	77.758000
10408	FOLIG	8	1997-01-08 00:00:00	1997-02-05 00:00:00	28	11.2600	1.689000	9.571000
10409	OCEAN	3	1997-01-09 00:00:00	1997-02-06 00:00:00	28	29.8300	4.474500	25.355500
10410	BOTTM	3	1997-01-10 00:00:00	1997-02-07 00:00:00	28	2.4000	0.360000	2.040000
10411	BOTTM	9	1997-01-10 00:00:00	1997-02-07 00:00:00	28	23.6500	3.547500	20.102500
10412	WARTH	8	1997-01-13 00:00:00	1997-02-10 00:00:00	28	3.7700	0.565500	3.204500
10413	LAMAI	3	1997-01-14 00:00:00	1997-02-11 00:00:00	28	95.6600	14.349000	81.311000
10414	FAMIA	2	1997-01-14 00:00:00	1997-02-11 00:00:00	28	21.4800	3.222000	18.258000
10415	HUNGC	3	1997-01-15 00:00:00	1997-02-12 00:00:00	28	0.2000	0.030000	0.170000
10416	WARTH	8	1997-01-16 00:00:00	1997-02-13 00:00:00	28	22.7200	3.408000	19.312000
10417	SIMOB	4	1997-01-16 00:00:00	1997-02-13 00:00:00	28	70.2900	10.543500	59.746500
10418	QUICK	4	1997-01-17 00:00:00	1997-02-14 00:00:00	28	17.5500	2.632500	14.917500
10419	RICSU	4	1997-01-20 00:00:00	1997-02-17 00:00:00	28	137.3500	20.602500	116.747500
10420	WELLI	3	1997-01-21 00:00:00	1997-02-18 00:00:00	28	44.1200	6.618000	37.502000
10422	FRANS	2	1997-01-22 00:00:00	1997-02-19 00:00:00	28	3.0200	0.453000	2.567000
10423	GOURL	6	1997-01-23 00:00:00	1997-02-06 00:00:00	14	24.5000	3.675000	20.825000

## 5.2 Consultas básicas. Cambiando el orden de los resultados

### Cambiar el orden de los resultados

SQL dispone de la cláusula ORDER BY que permite indicar el orden en que se desea que se muestren los resultados. Las opciones ASC y DESC permiten indicar el tipo de orden.

#### Ejemplo 27

Se desea ver el *nombreProducto*, *precioUnidad* y *unidadesEnExistencia* de todos los PRODUCTOS ordenados descendente por el *precioUnidad*.

SQL

```
SELECT nombreProducto, precioUnidad, unidadesEnExistencia  
FROM productos  
ORDER BY precioUnidad DESC;
```

## 5.2 Consultas básicas. Cambiando el orden de los resultados

nombre producto	precio unidad	unidades en existencia
Vino Côte de Blaye	263.5000	17
Salchicha Thüringer	123.7900	0
Buey Mishi Kobe	97.0000	29
Mermelada de Sir Rodney's	81.0000	40
Langostinos tigre Carnarvon	62.5000	42
Raclet de queso Courdavault	55.0000	79
Manzanas secas Manjimup	53.0000	20
Tarta de azúcar	49.3000	17
Café de Malasia	46.0000	17
Col fermentada Rössle	45.6000	26
Sandwich de vegetales	43.9000	24
Chocolate Schoggi	43.9000	49
Salsa de arándanos Northwoods	40.0000	6
Cordero Alice Springs	39.0000	0

• • •

## 5.2 Consultas básicas. Cálculos en consultas

### Ejemplo 28

Se pueden indicar ordenaciones por varios campos. El orden se establece de izquierda a derecha.

Se desea ver el *nombreCompañia*, *Ciudad* y *Pais* de todos los CLIENTES ordenados ascendenteamente primero por el *Pais*, después por la *Ciudad* y por último por el *nombreCompañia*.

**SQL**

```
SELECT nombrecompañia, ciudad, pais  
FROM clientes  
ORDER BY pais, ciudad, nombrecompañia;
```

## 5.2 Consultas básicas. Cálculos en consultas

nombrecompañia	ciudad	país
Drachenblut Delikatessen	Aachen	Alemania
Alfreds Futterkiste	Berlín	Alemania
Königlich Essen	Brandenburgo	Alemania
QUICK-Stop	Cunewalde	Alemania
Lehmanns Marktstand	Francfurt	Alemania
Ottilie's Käseladen	Köln	Alemania
Morgenstern Gesundkost	Leipzig	Alemania
Blauer See Delikatessen	Mannheim	Alemania
Frankenversand	München	Alemania
Toms Spezialitäten	Münster	Alemania
Die Wandernde Kuh	Stuttgart	Alemania
Cactus Comidas para llevar	Buenos Aires	Argentina
Océano Atlántico Ltda.	Buenos Aires	Argentina
Rancho grande	Buenos Aires	Argentina
Ernst Handel	Graz	Austria
Piccolo und mehr	Salzburgo	Austria
Maison Dewey	Bruselas	Bélgica

• • •

## 5.2 Consultas básicas. Limitar la salida de los resultados

### Limitando la salida de los resultados

MySQL dispone de una cláusula llamada **LIMIT** que permite limitar el número de resultados en la ejecución de una consulta. LIMIT permite indicar el número de registros que se quieren visualizar. Lo podemos hacer de 2 formas:

- Indicando el número de registros a visualizar desde el primero.
- Indicando el número de registros a visualizar desde uno en concreto.

### Ejemplo 29

Se desea ver el *nombreProducto*, *precioUnidad* y *unidadesEnExistencia* de los 8 PRODUCTOS más caros.

SQL

```
SELECT nombreproducto, preciounidad, unidadesenexistencia  
FROM productos  
ORDER BY preciounidad DESC  
LIMIT 8
```



## 5.2 Consultas básicas. Limitar la salida de los resultados

nombre producto	precio unidad	unidades en existencia
Vino Côte de Blaye	263.5000	17
Salchicha Thüringer	123.7900	0
Buey Mishí Kobe	97.0000	29
Mermelada de Sir Rodney's	81.0000	40
Langostinos tigre Carnarvon	62.5000	42
Raclet de queso Courdavault	55.0000	79
Manzanas secas Manjimup	53.0000	20
Tarta de azúcar	49.3000	17

## 5.2 Consultas básicas. Limitar la salida de los resultados

### Ejemplo 30

Se desea ver el *nombreProducto*, *precioUnidad* y *unidadesEnExistencia* de los 6 productos siguientes al 4º producto más caro.

**SQL**

```
SELECT nombreproducto, preciounidad, unidadesenexistencia  
FROM productos  
ORDER BY preciounidad DESC  
LIMIT 3, 6;
```

Debemos tener en cuenta que los registros (filas), MySQL los comienza a numerar en 0 (cero). Por ello, el 4º registro será el 3.

Esta funcionalidad es muy útil cuando se quieren **paginar resultados**.

## 5.2 Consultas básicas. Limitar la salida de los resultados

nombre producto	precio unidad	unidades en existencia
Mermelada de Sir Rodney's	81.0000	40
Langostinos tigre Carnarvon	62.5000	42
Raclet de queso Courdavault	55.0000	79
Manzanas secas Manjimup	53.0000	20
Tarta de azúcar	49.3000	17
Café de Malasia	46.0000	17

# Actividad 4

## Consultas de repaso

- 1) Se desea ver todos los datos de los 3 primeros CLIENTES que comiencen por 'B'. Los datos deben ordenarse por el nombreCompañía.

IdCliente	NombreCompañía	NombreContacto	CargoContacto	Direccion	Ciudad
BSBEV	B's Beverages	Victoria Ashworth	Representante de ventas	Fauntleroy Circus	Londres
BERGS	Berglunds snabbköp	Christina Berglund	Administrador de pedidos	Berguvsvägen 8	Luleå
BLAUS	Blauer See Delikatessen	Hanna Moos	Representante de ventas	Forsterstr. 57	Mannheim

- 2) Se desea ver todos los datos de aquellos PEDIDOS cuya fechaPedido pertenezca a la segunda mitad del mes de agosto de 1996. Los resultados deben mostrarse en orden descendente por el Cargo del pedido. Solamente deben mostrarse las 10 primeras filas.

IdPedido	IdCliente	IdEmpleado	FechaPedido	FechaEntrega	FechaEnvio	FormaEnvio	Cargo
10286	QUICK	8	1996-08-21 00:00:00	1996-09-18 00:00:00	1996-08-30 00:00:00	3	229.2400
10294	RATTIC	4	1996-08-30 00:00:00	1996-09-27 00:00:00	1996-09-05 00:00:00	2	147.2600
10283	LILAS	3	1996-08-16 00:00:00	1996-09-13 00:00:00	1996-08-23 00:00:00	3	84.8100
10290	COMMI	8	1996-08-27 00:00:00	1996-09-24 00:00:00	1996-09-03 00:00:00	1	79.7000
10285	QUICK	1	1996-08-20 00:00:00	1996-09-17 00:00:00	1996-08-26 00:00:00	2	76.8300
10284	LEHMS	4	1996-08-19 00:00:00	1996-09-16 00:00:00	1996-08-27 00:00:00	1	76.5600
10289	BSBEV	7	1996-08-26 00:00:00	1996-09-23 00:00:00	1996-08-28 00:00:00	3	22.7700
10293	TORTU	1	1996-08-29 00:00:00	1996-09-26 00:00:00	1996-09-11 00:00:00	3	21.1800
10287	RICAR	8	1996-08-22 00:00:00	1996-09-19 00:00:00	1996-08-28 00:00:00	3	12.7600
10282	ROMEY	4	1996-08-15 00:00:00	1996-09-12 00:00:00	1996-08-21 00:00:00	1	12.6900



# Actividad 4

## Consultas de repaso

- 3) Se desea ver de la tabla de *PEDIDOS*, el *idCliente*, *fechaPedido*, *Cargo*, diferencia en días entre la *fechaPedido* y la *fechaEntrega*; de todos aquellos pedidos cuyo *idCliente* comience por 'B' , la *fechaPedido* sea del año 1997.

Los resultados deben mostrarse ordenados descendente por Días de diferencia y por *Cargo*. Mostrar solamente las 15 primeras filas.



idcliente	fechapedido	cargo	Dias
BSBEV	1997-07-15 00:00:00	29.9800	42
BONAP	1997-04-18 00:00:00	350.6400	28
BERGS	1997-05-01 00:00:00	244.7900	28
BOTTM	1997-11-14 00:00:00	243.7300	28
BLONP	1997-02-05 00:00:00	156.6600	28
BERGS	1997-08-11 00:00:00	138.6900	28
BERGS	1997-06-18 00:00:00	116.4300	28
BERGS	1997-11-07 00:00:00	110.1100	28
BOLID	1997-12-29 00:00:00	97.0900	28
BLONP	1997-06-12 00:00:00	88.4000	28
BONAP	1997-03-11 00:00:00	64.5600	28
BOTTM	1997-04-01 00:00:00	62.8900	28
BLONP	1997-06-30 00:00:00	59.1400	28
BERGS	1997-09-02 00:00:00	55.2600	28
BLONP	1997-02-18 00:00:00	53.3000	28

## 5.2 Consultas básicas. Estructura condicional (IF)

### Estructura condicional IF

Permite evaluar una condición y devolver un resultado si la condición es cierta y otro si la condición es falsa.

#### Ejemplo 31

Se desea calcular el descuento aplicado a los PEDIDOS sabiendo que si el Cargo del pedido es mayor de 200 se le aplica un 25% y en caso contrario le aplicaremos un 10%. Mostrar el *idPedido*, *fechaPedido*, *Destinatario* y *Cargo* de los pedidos, así como dicho descuento.

SQL

```
SELECT idpedido, fechapedido, destinatario, cargo,  
       IF(cargo>200, cargo*0.25, cargo*0.1) AS Descuento  
  FROM pedidos;
```

## 5.2 Consultas básicas. Estructura condicional (IF)

idpedido	fechapedido	destinatario	cargo	Descuento
10248	1996-07-04 00:00:00	Wilman Kala	32.3800	3.23800
10249	1996-07-05 00:00:00	Toms Spezialitäten	11.6100	1.16100
10250	1996-07-08 00:00:00	Hanari Carnes	65.8300	6.58300
10251	1996-07-08 00:00:00	Victuailles en stock	41.3400	4.13400
10252	1996-07-09 00:00:00	Suprêmes délices	51.3000	5.13000
10253	1996-07-10 00:00:00	Hanari Carnes	58.1700	5.81700
10254	1996-07-11 00:00:00	Chop-suey Chinese	22.9800	2.29800
10255	1996-07-12 00:00:00	Richter Supermarkt	148.3300	14.83300
10256	1996-07-15 00:00:00	Wellington Importadora	13.9700	1.39700
10257	1996-07-16 00:00:00	HILARIÓN-Abastos	81.9100	8.19100
10258	1996-07-17 00:00:00	Ernst Handel	140.5100	14.05100
10259	1996-07-18 00:00:00	Centro comercial Moctezuma	3.2500	0.32500
10260	1996-07-19 00:00:00	Ottilies Käseladen	55.0900	5.50900
10261	1996-07-19 00:00:00	Que Delícia	3.0500	0.30500
10262	1996-07-22 00:00:00	Rattlesnake Canyon Grocery	48.2900	4.82900
10263	1996-07-23 00:00:00	Ernst Handel	146.0600	14.60600
10264	1996-07-24 00:00:00	Folk och fä HB	3.6700	0.36700
10265	1996-07-25 00:00:00	Blondel père et fils	55.2800	5.52800
10266	1996-07-26 00:00:00	Wartian Herkku	25.7300	2.57300
10267	1996-07-29 00:00:00	Frankenversand	208.5800	52.145000
10268	1996-07-30 00:00:00	GROSELLA-Restaurante	66.2900	6.62900
10269	1996-07-31 00:00:00	White Clover Markets	4.5600	0.45600

• • •

# Actividad 5

## Consulta con condicionales

Diseñar la siguiente consulta sobre la base de datos Neptuno:

La tabla de *Productos* tiene un campo llamado *UnidadesEnExistencia* que indica el número de unidades del producto que tenemos en stock. Hay otro campo (*NivelNuevoPedido*) que indica las unidades mínimas que podemos tener en stock para hacer otro pedido.

Hacer una consulta que muestre:

- *nombreproducto*,
- *preciounidad*,
- *unidadesenexistencia*,
- *Nivelnuevopedido*
- y un campo nuevo llamado *Alerta* donde aparezca 'SI' si las *unidadesenexistencia* son menores que *nivelnuevopedido*, en caso contrario que muestre 'NO'.



## 5.2 Consultas básicas. Solución actividad 5

SQL

```
SELECT nombreproducto, preciounidad,  
       unidadesenexistencia, nivelnuevopedido,  
       IF(unidadesenexistencia<nivelnuevopedido, 'SI', 'NO') AS  
          Alerta  
     FROM productos;
```

nombreproducto	preciounidad	unidadesenexistencia	nivelnuevopedido	Alerta
Té Dharamsala	18.0000	39	10	NO
Cerveza tibetana Barley	19.0000	17	25	SI
Sirope de regaliz	10.0000	13	25	SI
Especias Cajun del chef Anton	22.0000	53	0	NO
Mezcla Gumbo del chef Anton	21.3500	0	0	NO
Mermelada de grosellas de la abuela	25.0000	120	25	NO
Peras secas orgánicas del tío Bob	30.0000	15	10	NO
Salsa de arándanos Northwoods	40.0000	6	0	NO
Buey Mishi Kobe	97.0000	29	0	NO
Pez espada	31.0000	31	0	NO
Queso Cabrales	21.0000	22	30	SI
• • •				



## 5.2 Consultas básicas. Estructura selectiva (CASE)

### Estructura selectiva CASE

Es equivalente a la estructura **switch** de otros lenguajes. Permite representar multialternativas. Sintaxis:

```
CASE valor
    WHEN [valor_1]
        THEN resultado_1
    ...
    [WHEN [valor_i]
        THEN resultado_i ...]
    [ELSE resultado]
END
```

Si no hay coincidencias, se devuelve el valor asociado al ELSE, o NULL si no hay parte ELSE.

## 5.2 Consultas básicas. Estructura selectiva (CASE)

Si observamos la tabla *CompañíasEnvíos* (se muestra a continuación), apreciamos que solamente existen tres compañías de envío para los pedidos.

IdCompañiaEnvios	NombreCompañia	Telefono
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

### Ejemplo 32

Se quiere hacer una consulta sobre la tabla de PEDIDOS que muestre: *idPedido*, *Destinatario*, *fechaPedido*, *Cargo* y la compañía de envíos que se ha utilizado para enviarlo. No debe aparecer el código de la compañía (campo *FormaEnvio*), sino su nombre. Hacer la consulta anterior para los pedidos del mes de septiembre de 1996.

## 5.2 Consultas básicas. Estructura selectiva (CASE)

SQL

```
SELECT idpedido, destinatario, fechapedido, cargo,  
CASE formaenvio  
    WHEN 1 THEN 'Speedy Express'  
    WHEN 2 THEN 'United Package'  
    WHEN 3 THEN 'Federal Shipping'  
END AS 'Empresa Transporte'  
FROM pedidos  
WHERE fechapedido BETWEEN '1996-09-01' AND '1996-09-30';
```

idpedido	destinatario	fechapedido	cargo	Empresa Transporte
10295	Vins et alcools Chevalier	1996-09-02 00:00:00	1.1500	United Package
10296	LILA-Supermercado	1996-09-03 00:00:00	0.1200	Speedy Express
10297	Blondel père et fils	1996-09-04 00:00:00	5.7400	United Package
10298	Hungry Owl All-Night Grocers	1996-09-05 00:00:00	168.2200	United Package
10299	Ricardo Adocicados	1996-09-06 00:00:00	29.7600	United Package
10300	Magazzini Alimentari Riuniti	1996-09-09 00:00:00	17.6800	United Package
10301	Die Wandernde Kuh	1996-09-09 00:00:00	45.0800	United Package
10302	Suprêmes délices	1996-09-10 00:00:00	6.2700	United Package
10303	Godos Cocina Típica	1996-09-11 00:00:00	107.8300	United Package
10304	Tortuga Restaurante	1996-09-12 00:00:00	63.7900	United Package
10305	Old World Delicatessen	1996-09-13 00:00:00	257.6200	Federal Shipping

• • •



# Actividad 6

## Estructura selectiva CASE

Se desea visualizar de la tabla de *PEDIDOS*: el *idPedido*, *fechaPedido*, *Destinatario* y *NOMBRE* del empleado que ha realizado el pedido de todos los pedidos vendidos al cliente con *idCliente*='ANATR'.

Los datos de los empleados son los siguientes:

IdEmpleado	Apellidos	Nombre
1	Davolio	Nancy
2	Fuller	Andrew
3	Leverling	Janet
4	Peacock	Margaret
5	Buchanan	Steven
6	Suyama	Michael
7	King	Robert
8	Callahan	Laura
9	Dodsworth	Anne

Resultados:

idpedido	fechapedido	Destinatario	empleado
10308	1996-09-18 00:00:00	Ana Trujillo Emparedados y helados	Robert
10625	1997-08-08 00:00:00	Ana Trujillo Emparedados y helados	Janet
10759	1997-11-28 00:00:00	Ana Trujillo Emparedados y helados	Janet
10926	1998-03-04 00:00:00	Ana Trujillo Emparedados y helados	Margaret

## 5.2 Consultas básicas. Agrupando registros

### Agrupamientos de registros

Es posible agrupar filas en la salida de una sentencia SELECT, según los distintos valores de una columna, usando la cláusula **GROUP BY**. Esto, en principio, puede parecer redundante, ya que podíamos hacer lo mismo usando la opción **DISTINCT**. Sin embargo, la cláusula GROUP BY es más potente.

#### Ejemplo 33

SQL

```
SELECT pais  
FROM clientes  
GROUP BY pais;
```



## 5.2 Consultas básicas. Agrupando registros

### Agrupamientos de registros

SQL

```
SELECT pais FROM clientes GROUP BY pais;
```

El código anterior produciría la salida que se muestra a la derecha.

- Una diferencia que observamos, con respecto a DISTINCT, es que si se usa GROUP BY **la salida se ordena** según los valores de la columna indicada. En este caso, las columnas aparecen ordenadas por País.
- La diferencia principal es que el uso de la cláusula **GROUP BY permite usar funciones de resumen o agregado**.

país
Alemania
Argentina
Austria
Bélgica
Brasil
Canadá
Dinamarca
España
Estados Unidos
Finlandia
Francia
Irlanda
Italia
México
Noruega
Polonia
Portugal
Reino Unido
Suecia
Suiza
Venezuela

## 5.2 Consultas básicas. Agrupando registros

### Ejemplo 34

Calcular el número de CLIENTES que tenemos en cada país.

**SQL**

```
SELECT pais, COUNT(*) AS 'Nº clientes'  
FROM clientes  
GROUP BY pais;
```

pais	Nº clientes
Alemania	11
Argentina	3
Austria	2
Bélgica	2
Brasil	9
Canadá	3
Dinamarca	2
España	5
Estados Unidos	13
Finlandia	2

• • •

## 5.2 Consultas básicas. Agrupando registros

### Ejemplo 35

Mostrar a partir de la tabla de PEDIDOS: el nombre de la compañía de envío (utilizar el campo *FormaEnvio* en un CASE) y la suma de los cargos de los pedidos enviados con cada compañía, así como los cargos mínimo, medio y máximo.

**SQL**

```
SELECT CASE formaEnvio
        WHEN 1 THEN 'Speedy Express'
        WHEN 2 THEN 'United Package'
        WHEN 3 THEN 'Federal Shipping'
    END AS 'Empresa Transporte',
    SUM(cargo) AS 'Suma Pedido',
    MAX(cargo) AS 'Pedido Máximo',
    MIN(cargo) AS 'Pedido Mínimo',
    AVG(cargo) AS 'Pedido Medio'
FROM Pedidos
GROUP BY Formaenvio;
```

## 5.2 Consultas básicas. Agrupando registros

### Ejemplo 35

Empresa Transporte	Suma Pedido	Pedido Máximo	Pedido Mínimo	Pedido Medio
Speedy Express	16155.8700	458.7800	0.1200	65.14463710
United Package	28274.3100	890.7800	0.0200	86.46577982
Federal Shipping	20512.5100	1007.6400	0.4000	80.44121569

## 5.2 Consultas básicas. Agrupando registros

### Agrupamientos de registros. Especificar condiciones

Hasta ahora todas las condiciones que hemos indicado en nuestras consultas, las hemos especificado con la cláusula WHERE. Si hacemos uso de la cláusula GROUP BY, debemos sustituir el WHERE por **HAVING**.

#### Ejemplo 36

Mostrar a partir de la tabla de PEDIDOS: el nombre de la compañía de envío (utilizar el campo

**SQL**

```
SELECT pais, COUNT(*) AS 'Nº clientes'  
FROM clientes  
GROUP BY pais  
HAVING pais LIKE "A%";
```

pais	Nº clientes
Alemania	11
Argentina	3
Austria	2

## 5.2 Consultas básicas. Agrupando registros

### Ejemplo 37 (ampliación al ejemplo 35)

Mejorar el ejemplo 35 para que además de mostrar los datos de los cargos medios, mínimos, máximos y suma de cargos, agrupados por *formadeenvío*, también agrupe los pedidos por *idempleado*. Debe mostrar los datos como la tabla que aparece abajo.

La consulta afectará solamente a los PEDIDOS de los años 1996 y 1997 (*fechapedido*)

Empresa Transporte	Empleado	fechapedido	Suma Pedido	Pedido Medio	Pedido Mínimo	Pedido Máximo
Speedy Express	Nancy	1996-07-17 00:00:00	2194.7300	176.4800	0.4500	57.75605263
Speedy Express	Andrew	1996-07-25 00:00:00	2281.0900	348.1400	0.7500	65.17400000
Speedy Express	Janet	1996-07-08 00:00:00	2491.7400	328.7400	0.2000	69.21500000
Speedy Express	Margaret	1996-07-19 00:00:00	3129.3900	458.7800	0.1500	68.03021739
Speedy Express	Steven	1996-07-31 00:00:00	1218.2700	200.2400	4.5600	87.01928571
Speedy Express	Michael	1996-07-05 00:00:00	1020.3700	201.2900	0.1200	46.38045455
Speedy Express	Robert	1996-11-08 00:00:00	1204.6000	411.8800	3.2000	60.23000000
Speedy Express	Laura	1996-08-27 00:00:00	1889.2700	212.9800	0.3300	69.97296296
Speedy Express	Ann	1996-10-08 00:00:00	726.4100	214.2700	0.4800	72.64100000
United Package	Nancy	1996-08-20 00:00:00	3259.9900	544.0800	0.2100	74.09068182
United Package	Andrew	1996-09-02 00:00:00	4350.2000	810.0500	0.1700	120.83888889
United Package	Janet	1996-07-10 00:00:00	4258.9000	789.9500	0.1400	94.64222222
United Package	Margaret	1996-07-08 00:00:00	4926.7300	719.7800	0.0200	70.38185714
United Package	Steven	1996-07-11 00:00:00	1991.8400	890.7800	5.7400	132.78933333
United Package	Michael	1996-08-01 00:00:00	1299.8800	227.2200	1.1700	49.99538462

• • •

## 5.2 Consultas básicas. Agrupando registros

SQL

```
SELECT CASE formaEnvio
        WHEN 1 THEN 'Speedy Express'
        WHEN 2 THEN 'United Package'
        WHEN 3 THEN 'Federal Shipping'
    END AS 'Empresa Transporte',
CASE idempleado
    WHEN 1 THEN 'Nancy'
    WHEN 2 THEN 'Andrew'
    WHEN 3 THEN 'Janet'
    WHEN 4 THEN 'Margaret'
    WHEN 5 THEN 'Steven'
    WHEN 6 THEN 'Michael'
    WHEN 7 THEN 'Robert'
    WHEN 8 THEN 'Laura'
    WHEN 9 THEN 'Ann'
END AS Empleado,
fechapedido,
SUM(cargo) AS 'Suma Pedido',
MAX(cargo) AS 'Pedido Medio',
MIN(cargo) AS 'Pedido Mínimo',
AVG(cargo) AS 'Pedido Medio'
FROM pedidos
GROUP BY Formaenvio, idempleado
HAVING fechapedido BETWEEN '1996-01-01' AND '1997-12-31';
```



## 5.2 Consultas básicas. Agrupando registros

### Ejemplo 38

Mostrar el número de PEDIDOS realizados cada año. Recordad que existe una función llamada YEAR(fecha) que devuelve el año de una fecha. Agrupar utilizando esta función

**SQL**

```
SELECT YEAR(fechapedido) AS Año, COUNT(*) AS 'Num. Pedidos'  
FROM Pedidos  
GROUP BY YEAR(fechapedido);
```

Año	Num. Pedidos
1996	152
1997	408
1998	270

## 5.2 Consultas básicas. Agrupando registros

### Ejemplo 39

Mostrar cuánto hemos vendido (Cargo de PEDIDOS) agrupando los resultados por año y por *PaisDestinatario* de los pedidos.

**SQL**

```
SELECT YEAR(fechapedido) AS Año,  
       PaisDestinatario, SUM(cargo) AS Importe  
  FROM Pedidos  
 GROUP BY YEAR(fechapedido), paisdestinatario;
```

## 5.2 Consultas básicas. Agrupando registros

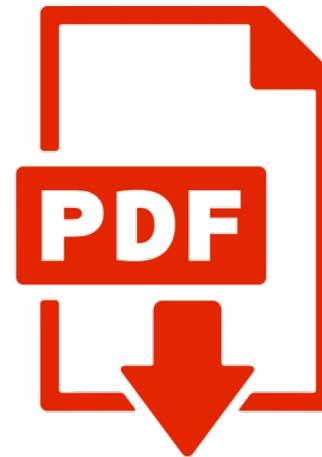
### Ejemplo 39

Año	PaisDestinatario	Importe
1996	Alemania	1957.1400
1996	Austria	1255.0900
1996	Bélgica	57.5700
1996	Brasil	1223.8900
1996	Canadá	136.3100
1996	Dinamarca	67.6900
1996	España	219.0800
1996	Estados Unidos	1972.3500
1996	Finlandia	229.8100
1996	Francia	631.2900
1996	Irlanda	416.7800
1996	Italia	52.0600
1996	México	244.3700
1996	Noruega	93.6300
1996	Polonia	3.9400
1996	Portugal	164.1000
1996	Reino Unido	566.3500
1996	Suecia	342.7800
1996	Suiza	172.4800
• • •		

Año	PaisDestinatario	Importe
1996	Venezuela	473.1600
1997	Alemania	6232.5500
1997	Argentina	117.6600
1997	Austria	3745.6500
1997	Bélgica	460.4800
1997	Brasil	2226.0100
1997	Canadá	1687.7600
1997	Dinamarca	1040.8500
1997	España	212.1000
1997	Estados Unidos	5819.3700
1997	Finlandia	636.1100
1997	Francia	2467.7600
1997	Irlanda	980.7700
1997	Italia	440.6700
1997	México	635.5100
1997	Noruega	52.0100
1997	Polonia	104.4400
1997	Portugal	274.4900
1997	Reino Unido	1177.5900
1997	Suecia	1798.6800

# Actividad 7

## Ejercicios de Repaso



Actividad7.pdf

# Índice

## Tema 5 – Realización de consultas

---

1. Sentencia SELECT
  1. Comandos DML
  2. Cláusulas
  3. Operadores
  4. Funciones de agregado
2. Consultas básicas, filtros y ordenación
3. Subconsultas
4. Consultas multitablea: composiciones internas y externas
5. Consultas reflexivas
6. Consultas con tablas derivadas

## 5.3 Subconsultas

### Subconsultas

Una **subconsulta** es una sentencia SELECT que aparece dentro de otra sentencia SELECT que llamaremos **consulta principal**.

Se puede encontrar **en la lista de selección, en la cláusula WHERE o en la cláusula HAVING** de la consulta principal.



## 5.3 Subconsultas

### Subconsultas

Ejemplo sobre la base de datos Neptuno:

SQL

```
SELECT empleados.idEmpleado, empleados.nombre,  
       (SELECT MAX(pedidos.cargo)  
        FROM pedidos  
       WHERE pedidos.idEmpleado=empleados.idEmpleado)  
              AS 'Mejor pedido'  
  FROM empleados
```

En esta subconsulta tenemos una referencia externa ( *idEmpleado* ) es un campo de la tabla empleados (origen de la consulta principal).

## 5.3 Subconsultas

### ¿Qué pasa cuando se ejecuta la consulta principal?

Se coge el primer empleado y se calcula la subconsulta sustituyendo *pedidos.idEmpleado* por el valor que tiene *idEmpleado* en la tabla de EMPLEADOS. La subconsulta obtiene el cargo máximo de los PEDIDOS de ese empleado.

A continuación se repite el proceso con el resto de los empleados.

Al final obtenemos una lista con el *idEmpleado*, *nombre* y el cargo (importe) máximo de los PEDIDOS gestionados por ese empleado.

<i>idEmpleado</i>	<i>nombre</i>	Mejor pedido
1	Nancy	544.0800
2	Andrew	810.0500
3	Janet	1007.6400
4	Margaret	719.7800
5	Steven	890.7800
6	Michael	367.6300
7	Robert	830.7500
8	Laura	398.3600
9	Anne	754.2600

## 5.3 Subconsultas

Otro uso habitual de las subconsultas consiste en definirlas en la cláusulas WHERE o HAVING de la consulta.

Por ejemplo, cuando los datos que queremos visualizar están en una tabla pero para seleccionar las filas de esa tabla necesitamos un dato que está en otra tabla.

### Ejemplo 40

Ver los datos de todos los empleados cuya fecha de contratación sea anterior a la fecha del primer pedido que ha recibido la empresa

**SQL**

```
SELECT nombre, apellidos, fechaContratacion  
FROM empleados  
WHERE fechaContratacion < (SELECT MIN(fechaPedido)  
                           FROM pedidos)
```



## 5.3 Subconsultas

La consulta anterior produce los siguientes resultados:

nombre	apellidos	fechacontratacion
Nancy	Davolio	1992-05-01 00:00:00
Andrew	Fuller	1992-08-14 00:00:00
Janet	Leverling	1992-04-01 00:00:00
Margaret	Peacock	1993-05-03 00:00:00
Steven	Buchanan	1993-10-17 00:00:00
Michael	Suyama	1993-10-17 00:00:00
Robert	King	1994-01-02 00:00:00
Laura	Callahan	1994-03-05 00:00:00
Anne	Dodsworth	1994-11-15 00:00:00

## 5.3 Subconsultas. **Condiciones de selección**

### Condiciones de selección en subconsultas

Las **condiciones de selección** son las condiciones que pueden aparecer en la cláusula **WHERE** o **HAVING**. La mayoría se han visto en el tema anterior, ahora incluiremos las condiciones que utiliza una subconsulta como operando.

En SQL tenemos cuatro nuevas condiciones:

- el **test de comparación con subconsulta**
- el **test de comparación cuantificada**
- el **test de pertenencia a un conjunto**
- el **test de existencia**

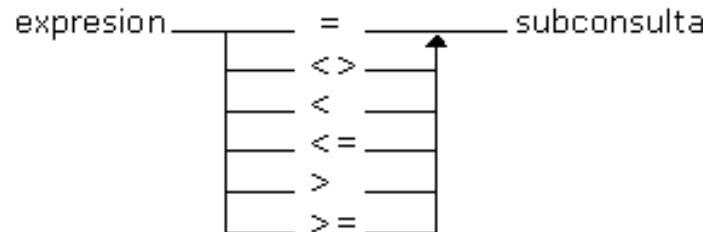
## 5.3 Subconsultas. Condiciones de selección

### Test de comparación con subconsulta

Es el **equivalente al test de comparación simple**. Se utiliza para comparar un valor de la fila que se está examinando con un único valor producido por la subconsulta. La subconsulta debe devolver una **única columna**, sino se produce un error.

Si la subconsulta no produce **ninguna fila** o devuelve el valor **nulo**, el test devuelve el **valor nulo**, si la subconsulta produce **varias filas**, SQL devuelve una **condición de error**.

Sintaxis:



## 5.3 Subconsultas. Condiciones de selección

### Ejemplo teórico (no trabaja con la BD Neptuno)

Lista las oficinas cuyo objetivo sea superior a la suma de las ventas de sus empleados.

**SQL**

```
SELECT oficina, ciudad  
FROM oficinas  
WHERE objetivo > (SELECT SUM(ventas) FROM empleados  
                     WHERE empleados.oficina = oficinas.oficina)
```

En este caso la subconsulta devuelve una única columna y una única fila (es un consulta de resumen sin GROUP BY)

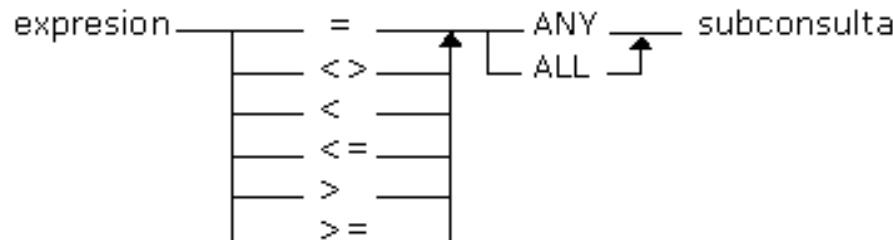
## 5.3 Subconsultas. Condiciones de selección

### Test de comparación cuantificada

Este test es una extensión del test de comparación y del test de conjunto. **Compara el valor** de la expresión **con cada uno de los valores** producidos por la subconsulta. La subconsulta debe devolver una **única columna** sino se produce un error.

Tenemos el test **ANY** (algún, alguno en inglés) y el test **ALL** (todos en inglés).

Sintaxis:



## 5.3 Subconsultas. Condiciones de selección

### El test ANY

La subconsulta debe devolver una única columna sino se produce un error. Se evalúa la comparación con cada valor devuelto por la subconsulta.

- Si **alguna de las comparaciones individuales produce el resultado verdadero, el test ANY devuelve el resultado verdadero.**
- Si la subconsulta no devuelve **ningún valor**, el test **ANY** devuelve **falso**.
- Si el test de comparación es **falso** para **todos los valores** de la columna, **ANY** devuelve **falso**.
- Si el test de comparación **no es verdadero** para ninguno valor de la columna, y **es nulo** para al menos alguno de los valores, **ANY** devuelve **nulo**.

## 5.3 Subconsultas. Condiciones de selección

### Ejemplo teórico (no trabaja con la BD Neptuno)

En este caso la subconsulta devuelve una única columna con las sumas de las cuotas de los empleados de cada oficina.

SQL

```
SELECT oficina, ciudad  
FROM oficinas  
WHERE objetivo > ANY (SELECT SUM(cuota) FROM empleados  
GROUP BY oficina)
```

Lista las oficinas cuyo objetivo sea superior a alguna de las sumas obtenidas.

## 5.3 Subconsultas. Condiciones de selección

### El test ALL

La subconsulta debe devolver una única columna sino se produce un error. Se evalúa la comparación con cada valor devuelto por la subconsulta.

- Si **todas** las comparaciones individuales, producen un resultado **verdadero**, el test devuelve el valor **verdadero**.
- Si la subconsulta no devuelve **ningún valor** el test **ALL** devuelve el valor **verdadero**. (¡Ojo con esto!).
- Si el test de comparación es **falso** para algún valor de la columna, el resultado es **falso**.
- Si el test de comparación **no es falso** para ningún valor de la columna, pero es **nulo** para alguno de esos valores, el test **ALL** devuelve valor **nulo** (¡Ojo con esto!).

## 5.3 Subconsultas. Condiciones de selección

### Ejemplo teórico (no trabaja con la BD Neptuno)

En este caso se listan las oficinas cuyo objetivo sea superior a todas las sumas.

SQL

```
SELECT oficina, ciudad  
FROM oficinas  
WHERE objetivo > ALL (SELECT SUM(cuota) FROM empleados  
GROUP BY oficina)
```

## 5.3 Subconsultas. Condiciones de selección

### Test de pertenencia a conjunto (IN)

Examina si el **valor** de la expresión es uno de los valores **incluidos en la lista de valores producida por la subconsulta**.

La subconsulta debe generar una única columna y las filas que sean.

Si la subconsulta no produce ninguna fila, el test da falso.

Tiene la siguiente sintaxis:

expresion — IN — subconsulta

## 5.3 Subconsultas. Condiciones de selección

### Ejemplo teórico (no trabaja con la BD Neptuno)

Con la subconsulta se obtiene la lista de los números de oficina del este y la consulta principal obtiene los empleados cuyo número de oficina sea uno de los números de oficina del este.

Por lo tanto lista los empleados de las oficinas del este.

**SQL**

```
SELECT numemp, nombre, oficina  
FROM empleados  
WHERE oficina IN (SELECT oficina FROM oficinas WHERE region = 'este')
```

## 5.3 Subconsultas. Condiciones de selección

### Ojo con NOT IN

Hay que tener especial cuidado con los valores nulos cuando utilizamos el operador NOT IN porque el resultado obtenido no siempre será el deseado. Cuando la subconsulta devuelve algún nulo el resultado es falso o nulo pero nunca verdadero.

Ejemplos para entenderlo:

**SQL**

```
SELECT oficina, ciudad  
FROM oficinas  
WHERE oficina IN (SELECT oficina FROM empleados)
```

Obtenemos las oficinas que tienen algún empleado.

## 5.3 Subconsultas. Condiciones de selección

SQL

```
SELECT oficina, ciudad  
FROM oficinas  
WHERE oficina NOT IN (SELECT oficina FROM empleados)
```

Al contrario de lo que podríamos pensar esta consulta NO devuelve las oficinas que no tienen ningún empleado porque la subconsulta devuelve algún NULL (por los empleados que no están asociados a ninguna oficina).

SQL

```
SELECT oficina, ciudad  
FROM oficinas  
WHERE oficina NOT IN (SELECT oficina  
                      FROM empleados  
                      WHERE oficina IS NOT NULL)
```

Así, sí

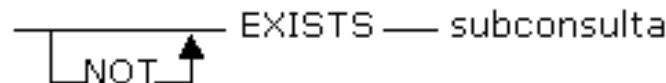
## 5.3 Subconsultas. Condiciones de selección

### Test de existencia (EXISTS)

Examina si la subconsulta produce alguna fila de resultados.

- Si la subconsulta **contiene filas**, el test adopta el valor **verdadero**, si la subconsulta no contiene ninguna fila, el test toma el valor falso, nunca puede tomar el valor nulo.
- Con este test la subconsulta **puede tener varias columnas**, no importa ya que el test se fija no en los valores devueltos sino en si hay o no fila en la tabla resultado de la subconsulta.
- Cuando se utiliza el test de existencia en la mayoría de los casos habrá que utilizar una referencia externa. Si no se utiliza una referencia externa la subconsulta devuelta siempre será la misma para todas las filas de la consulta principal y en este caso se seleccionan todas las filas de la consulta principal (si la subconsulta genera filas) o ninguna (si la subconsulta no devuelve ninguna fila)

La sintaxis es la siguiente:



## 5.3 Subconsultas. Condiciones de selección

**Ejemplo teórico** (no trabaja con la BD Neptuno)

**SQL**

```
SELECT numemp, nombre, oficina  
FROM empleados  
WHERE EXISTS (SELECT * FROM oficinas  
              WHERE region = 'este'  
              AND empleados.oficina = oficinas.oficina)
```

- Este ejemplo obtiene lo mismo que el ejemplo del test **IN**.
- Observa que delante de **EXISTS** no va ningún nombre de columna.
- En la subconsulta se pueden poner las columnas que queramos en la lista de selección (hemos utilizado el **\***).
- Hemos añadido una condición adicional al **WHERE**, la de la referencia externa para que la oficina que se compare sea la oficina del empleado.

**Cuando se trabaja con tablas muy voluminosas el test EXISTS suele dar mejor rendimiento que el test IN.**

# Resumen

## Tema 5 – Realización de consultas

---

1. Sentencia SELECT
  1. Comandos DML
  2. Cláusulas
  3. Operadores
  4. Funciones de agregado
2. Consultas básicas, filtros y ordenación
3. Subconsultas