

2. Consultas simples

2.1. Objetivos

Los objetivos de esta actividad son:

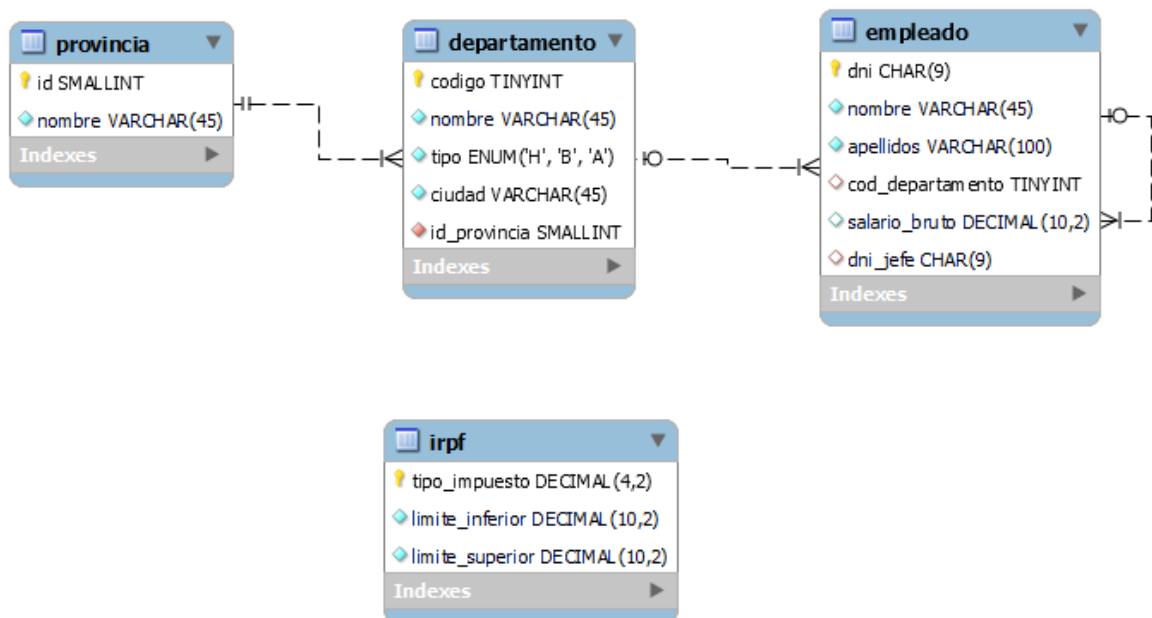
- Realizar consultas simples en una tabla de la base de datos utilizando la sentencia SELECT con las cláusulas FROM, WHERE, ORDER BY y LIMIT.
- Utilizar en las consultas funciones integradas en el gestor de base de datos.

Se utilizará **MySQL Workbench** como herramienta gráfica.

2.2. Base de datos de ejemplo

La base de datos *empleados* se utilizará para los ejemplos de esta unidad. Antes de empezar a probar los ejemplos, hay que ejecutar el script de creación de la base de datos en el servidor y ponerla en uso.

Esta base de datos está formada por un grupo de tablas, relacionadas entre sí, tal y como se muestra en el siguiente diagramado diseñado con MySQL Workbench:



- Tabla *empleado*: La columna *departamento* es una clave foránea que contiene el código del departamento en el que trabaja el empleado, y hace referencia a la columna *codigo* de la tabla *departamento*. Los valores que toma la columna *departamento* tienen que coincidir con los que toma la columna *codigo* de la tabla *departamento*, o ser NULL en el caso que el empleado no esté asignado a ningún departamento. La columna *dni_jefe* es otra clave foránea que contiene el dni de otro empleado que sería su jefe, o el valor NULL en el caso que no tuviera jefe.
- Tabla *departamento*: La columna *id_provincia* es una clave foránea que hace referencia a la columna *id* de la tabla *provincia*.

- Tabla *irpf*: Contiene el porcentaje de impuesto a aplicarle a cada empleado, en función de su salario bruto, dependiendo de los límites entre los que se encuentre. Esta tabla podría contener una información similar a esta:

Result Grid			Filter Rows:
	limite_inferior	limite_superior	tipo_impuesto
▶	0.00	17707.00	15.75
	17707.00	33007.00	21.00
	33007.00	53407.00	27.00
	53407.00	120000.00	30.00
	120000.00	175000.00	35.00
	175000.00	300000.00	42.00

Salario bruto entre 0 e 17107 euros corresponde un 15.75% de imposto

2.3. Actividad

2.3.1. Manipulación de datos con SQL

SQL corresponde al acrónimo de *Structured Query Language* (Lenguaje Estructurado de Consultas). Aunque en un principio fue creado para realizar consultas, se utiliza para controlar todas las funciones que suministra un SGBD a sus usuarios, incluyendo todas las funciones propias de los lenguajes diseñadas para el manejo de bases de datos: Lenguaje de Definición de Datos, Lenguaje de Manipulación de Datos y Lenguaje de Control de Datos. El lenguaje de manipulación de datos o LMD (en inglés *Data Management Language* o *DML*), permite realizar las operaciones necesarias para manejar los datos almacenados en una base de datos. Estas operaciones son: insertar filas de datos (sentencia INSERT), modificar el contenido de las filas de datos (sentencia UPDATE), borrar filas de datos (sentencia DELETE), y consultar los datos contenidos en las tablas de la base de datos (sentencia SELECT).

2.3.2. Herramientas para realizar consultas

Para realizar consultas se necesita el código con las sentencias SQL, un cliente conectado al servidor que permita enviar el código y un servidor que ejecute ese código. La escritura de guiones (*scripts*) SQL puede hacerse con cualquier editor de texto plano.

Las aplicaciones cliente con una interfaz gráfica (GUI) incorporan un editor especializado en la escritura de guiones y posibilitan, además, la ejecución de los guiones mediante un sistema de menús o combinación de teclas.

Las aplicaciones cliente más empleadas para conectar a un servidor MySQL y enviarle las sentencias SQL para que se ejecuten, son las siguientes:

- MySQL Workbench

Cliente gráfico que de forma fácil e intuitiva permite establecer conexión con un servidor, mostrar información sobre la conexión, el estado del servidor, bases de datos y tablas; escribir sentencias SQL mediante un editor; enviar sentencias al servidor para que sean ejecutadas; y dispone de ayuda en las tareas de edición y ejecución de sentencias. Para más información sobre su funcionamiento, debe utilizarse el manual de referencia.

- Consola modo texto *mysql.exe*

Cliente en modo texto que permite establecer la conexión con el servidor y escribir y ejecutar sentencias SQL.

2.3.3.Sentencia SELECT para consulta de datos

La sentencia SELECT permite realizar consultas y es la instrucción más potente y compleja de todas las instrucciones SQL. A pesar de la gran cantidad de opciones que ofrece, es posible empezar formulando consultas simples e ir construyendo consultas más complejas a medida que se tenga mayor conocimiento del lenguaje.

La sentencia SELECT recupera datos de las tablas de una base de datos y devuelve el resultado de la consulta en forma de tabla, con las filas y columnas seleccionadas. Un resumen de la sintaxis de la sentencia es el siguiente:

```
SELECT [ALL | DISTINCT | DISTINCTROW] lista_de_selección
[FROM tablas]
[WHERE expresión_condicional]
[GROUP BY {lista_columnas | expresión | posición}]
[HAVING expresión_condicional]
[ORDER BY {columna | expresión | posición} [ASC | DESC], ...]
[LIMIT intervalo]
[INTO OUTFILE 'nombre_archivo']
```

- Las opciones DISTINCT y ALL especifican si hay que mostrar las filas duplicadas o no. La opción ALL es el valor predeterminado si no se indica ninguna de las opciones, y significa que se muestran todas las filas, incluidas las duplicadas. La opción DISTINCT se utiliza cuando se quieren eliminar las filas duplicadas del conjunto de resultados. La opción DISTINCTROW es un sinónimo de DISTINCT.
- La lista de selección (*lista_de_selección*) es la relación de los datos que se quieren obtener en la consulta, separados por comas.
- La cláusula FROM se utiliza para escribir la relación de tablas utilizadas en la consulta y las relaciones que hay entre ellas.
- La cláusula WHERE se utiliza para escribir las condiciones que tienen que cumplir las filas para que se muestren en el conjunto de resultados.
- La cláusula GROUP BY permite agrupar las filas del conjunto de resultados.
- La cláusula HAVING se utiliza para escribir las condiciones que tienen que cumplir los grupos resultantes de agrupar las filas del conjunto de resultados con la cláusula GROUP BY.
- La cláusula LIMIT se utiliza para restringir el número de filas devueltas por la consulta.
- La cláusula INTO OUTFILE se utiliza para enviar el conjunto de resultados a un archivo.

En las especificaciones de la sintaxis se indica el orden en que se han de escribir las cláusulas opcionales. En el caso de escribir las cláusulas en un orden diferente del que figura en la sintaxis, se mostrará un mensaje de error.

Las sentencias SELECT, al igual que el resto de sentencias de SQL, terminan en un punto y coma, aunque MySQL permite escribir las consultas sin el punto y coma final por tratarse de una única sentencia. Es una buena práctica utilizar el punto y coma siempre, porque éste es el modo habitual de indicar el fin de una instrucción en programación.

Podemos encontrar la documentación de referencia de la sentencia SELECT en el siguiente enlace:

<https://dev.mysql.com/doc/refman/8.0/en/select.html>

2.3.3.1. Lista de selección

La *lista_de_selección* de la sentencia SELECT, indica los datos que se quieren obtener con la consulta, separados por comas. Cada dato representa una columna de la tabla que contiene el conjunto de resultados, y puede contener cualquiera de las funciones y operadores que soporta MySQL. La sentencia permite recuperar datos de cálculos o de información del servidor, sin hacer referencia a ninguna tabla, como por ejemplo:

```
select 2+2, version(), user();
```

Result Grid			
	2+2	VERSION()	USER()
▶	4	8.0.31	root@localhost

La lista de selección tiene que cumplir la siguiente sintaxis:

expresión [[AS] alias_columna] [, expresión [[AS] alias_columna]] ...

- La *expresión* puede ser el nombre de una columna, una expresión aritmética, una expresión de fecha, una constante, una variable de usuario, una función, o una combinación de todas esas cosas.
- Los *alias_columna* son nombres que se dan a las columnas de la tabla de resultados y son especialmente útiles en el caso de columnas que contienen una expresión. Si no se asigna un alias, el nombre de la columna coincide con la expresión. Por ejemplo:

```
select 2+2,  
       version() as 'Versión del servidor',  
       user() as Usuario_conectado;
```

La utilización de alias permite facilitar la lectura de los resultados, como se puede ver en la salida que produce la consulta anterior.

Result Grid			
	2+2	Versión del servidor	Usuario_conectado
▶	4	8.0.31	root@localhost

- En la lista de selección se puede emplear el símbolo * para representar a todas las columnas de las tablas seleccionadas, o emplear el formato *nombre_tabla.** para hacer referencia a todas las columnas de una tabla. Por ejemplo:

```
select * from departamento; /* Muestra todas las columnas de la tabla departamento*/
```

Result Grid					
	codigo	nombre	tipo	ciudad	id_provincia
▶	1	Central	H	Lugo	29
	2	Oficina1	H	Monforte	29
	3	Oficina2	B	Ferrol	19
	4	Oficina3	H	Vigo	39
	5	Oficina4	A	Ourense	36
	6	Oficina5	A	Villalba	29
	7	Oficina6	H	Ourense	36
	8	Oficina7	H	Lugo	29
	9	Oficina8	A	Coruña	19
	10	Oficina9	B	Villalba	29
*	NULL	NULL	NULL	NULL	NULL

2.3.3.2. Expresiones

Los elementos que pueden contener las expresiones son: nombres de columnas, constantes o literales, variables de usuario, funciones y operadores.

Constantes

Las constantes o valores literales representan valores fijos. Las más utilizadas en MySQL son las de tipo cadenas de caracteres, números, fecha y hora, booleanas o lógicas, y el valor NULL.

- Los valores constantes de tipo cadena de caracteres se representan por un conjunto de caracteres cerrados entre comiencos simples o dobles. Por ejemplo:

```
select 'cadena con comillas simples',
       "cadena con comillas dobles",
       'cadena que incluye un carácter " (comillas dobles)';
```

Result Grid			
	cadena con comillas simples	cadena con comillas dobles	cadena que incluye un carácter " (comillas dobles)
▶	cadena con comillas simples	cadena con comillas dobles	cadena que incluye un carácter " (comillas dobles)

- Los valores constantes de tipo numérico entero se representan como una secuencia de dígitos; los de tipo decimal utilizan el carácter (.) como separador decimal. Cualquier número puede ir precedido del signo menos (-) para indicar un valor negativo, o del signo más para indicar un valor positivo (+). Si no se pone ningún signo, se considera positivo. Por ejemplo:

```
select 258 as entero, 3658.25 as 'decimal', -25 as negativo;
```

Result Grid			
	entero	decimal	negativo
▶	258	3658.25	-25

- Los valores constantes de tipo DATE (fecha), TIME (hora) y DATETIME (fecha y hora) representan valores cerrados entre comillas simples o dobles con un determinado formato.

- En el caso de las fechas, el formato establecido es *'aaaa-mm-dd'* o *'aa-mm-dd'*. MySQL admite cualquier carácter como delimitador de año, mes y día; e incluso admite que se omitan los delimitadores. Por ejemplo:

```
select '2015-12-25', "15-12-25", '20151225';
```

- En el caso de las horas, el formato establecido es *'hh:mm:ss'*. Por ejemplo:

```
select '12:00:30', "12:00:30", '120030';
```

- Los valores constantes de tipo lógico en MySQL son TRUE (verdadero) y FALSE (falso) y se evalúan como 1 y 0 respectivamente. Los nombres de estas constantes pueden escribir en cualquier combinación de mayúsculas y minúsculas.

Ejemplos:

```
select TRUE, true, FALSE, false, 3>2, 5=6;
```

	TRUE	true	FALSE	false	3>2	5=6
▶	1	1	0	0	1	0

- El valor NULL significa 'no hay dato' o 'valor desconocido'. La palabra se puede escribir en cualquier combinación de mayúsculas y minúsculas. Hay que tener en cuenta que el valor NULL no es lo mismo que el valor 0 para tipos numéricos o que la cadena vacía para tipos cadena de caracteres. Por ejemplo:

```
select NULL, null;
```

	NULL	NULL
▶	NULL	NULL

Operadores

MySQL dispone de multitud de operadores diferentes para cada uno de los tipos de dato. Los operadores permiten construir expresiones que se utilizan en la lista de selección, o en las cláusulas de las sentencias de manipulación de datos, incluida la sentencia SELECT.

Operadores aritméticos

Permiten hacer operaciones aritméticas con datos de tipo numérico. Si alguno de los operandos toma el valor NULL el resultado de la operación siempre es el valor NULL.

*	Multiplicación
/	División
%	Resto de la división entera
+	Suma
-	Resta

Operadores relacionales

Permiten hacer comparaciones entre expresiones devolviendo el valor 1 (*true = verdadero*) o 0 (*false = falso*). Si uno de los valores a comparar es el valor NULL, el resultado es NULL, excepto cuando se utiliza el operador <=>, o el operador IS NULL.

=	Igual
!= <>	Distinto
>	Mayor
<	Menor
>=	Mayor o igual
<=	Menor o igual
<=>	Igual, pero devuelve 1 en lugar de NULL si ambos operandos son NULL, o 0 en lugar de NULL si uno de ellos es NULL
IS	Comparación con el valor NULL. Se explica con más detalle en el apartado: Cláusula WHERE
LIKE	Comparación con patrón de búsqueda. Se explica con más detalle en el apartado: Cláusula WHERE
IN	Comparación con un conjunto de valores. Se explica con más detalle en el apartado: Cláusula WHERE
BETWEEN	Comparación con un intervalo de valores. Se explica con más detalle en el apartado: Cláusula WHERE

2.3.4.Consultas sencillas

Son consultas que utilizan únicamente las cláusulas FROM, WHERE, ORDER BY y LIMIT, aunque no tienen porque utilizarlas todas. En esta primera actividad se realizan consultas de este tipo tomando los datos de una sola tabla.

2.3.4.1. Cláusula FROM

Esta cláusula se utiliza para escribir la relación de las tablas en las que están los datos que se utilizan en la consulta. Por ejemplo, si queremos recuperar el nombre y dni de los empleados:

```
select dni, nombre, apellidos
from empleado;
```

2.3.4.2. Cláusula WHERE

Esta cláusula permite filtrar las filas, mediante condiciones del tipo:

```
expresión operador_relacional expresión
expresión [NOT] BETWEEN expresión1 AND expresión2
expresión [NOT] IN (lista_de_valores)
nome_columna [NOT] LIKE 'patrón_de_busca'
nome_columna [NOT] REGEXP [BINARY] 'expresión_regular'
nome_columna IS [NOT] NULL
```

Las condiciones utilizadas en la cláusula pueden ser compuestas, combinando más de una condición simple con los operadores lógicos:

- AND o &&: para que la condición compuesta devuelva el valor verdadero todas las condiciones tienen que ser verdaderas.
- OR o ||: para que la condición compuesta devuelva el valor verdadero es suficiente con que sea verdadera alguna de las condiciones.

Ejemplo de condición compuesta: seleccionar los *apellidos*, *nombre*, *salario_bruto* y *departamento* de los empleados del Departamento 1 que tengan un salario superior a 40000 euros.

```
select apellidos, nombre, salario_bruto, cod_departamento
from empleado
where cod_departamento = 1 and salario_bruto > 40000;
```

Result Grid				
		Filter Rows:		
		Export:		
	apellidos	nombre	salario_bruto	cod_departamento
▶	Fernandez Lopez	Jose Luis	160000.00	1

Predicado BETWEEN

Permite comparar el valor de la expresión ubicada a la izquierda de la palabra BETWEEN con los valores comprendidos en el intervalo definido por la *expresión1* y la *expresión2*, ambas incluidas. La sintaxis es:

```
expresión [NOT] BETWEEN expresión1 AND expresión2
```

Ejemplo: seleccionar *apellidos*, *nombre* y *salario_bruto* de los empleados que tengan un salario bruto entre 50000 y 70000 euros, ambos incluidos.

```
select apellidos, nombre, salario_bruto
from empleado
where salario_bruto between 50000 and 70000;
```

Result Grid			
		Filter Rows:	
		Export:	
	apellidos	nombre	salario_bruto
▶	Iglesias Dominguez	Adolfo	52500.00

Predicado IN

Permite comparar el valor de la expresión ubicado a la izquierda de la palabra IN con la lista de valores que se encuentran entre paréntesis. Sintaxis:

```
expresión [NOT] IN (lista_de_valores)
```

Ejemplo:

- Seleccionar el *nombre*, *ciudad* e *id_provincia* de todos los departamentos ubicados en provincias gallegas, es decir, que la columna *id_provincia* tome los valores 19, 29, 36 ó 39.


```
select * from departamento
where id_provincia in (19,29,36,39);
```

	codigo	nombre	tipo	ciudad	id_provincia
▶	1	Central	H	Lugo	29
	2	Oficina1	H	Monforte	29
	3	Oficina2	B	Ferrol	19
	4	Oficina3	H	Vigo	39
	5	Oficina4	A	Ourense	36
	6	Oficina5	A	Villalba	29
	7	Oficina6	H	Ourense	36
	8	Oficina7	H	Lugo	29
	9	Oficina8	A	Coruña	19
	10	Oficina9	B	Villalba	29
✱	NULL	NULL	NULL	NULL	NULL

Predicado LIKE

Permite realizar una comparación de semejanza entre el valor de una expresión y el de una cadena de caracteres. Sintaxis:

```
nombre_columna [NOT] LIKE 'patrón_de_búsqueda'
```

En el patrón se pueden emplear los siguientes caracteres 'comodín':

% (porcentaje)	Sustituye a un grupo de caracteres. Ejemplos: 'A%' representa un grupo de caracteres que empieza por 'A' '%ez' representa un grupo de caracteres que acaba por 'ez' '%' representa a cualquier grupo de caracteres
_ (guión bajo)	Sustituye a un carácter. Ejemplo: '1_a' representa un grupo de caracteres que empieza por 1, seguido de dos caracteres cualquiera y termina con el carácter 'a'

Ejemplos:

- Seleccionar *apellidos*, *nombre* y *salario_bruto* de todos los empleados cuyos apellidos empiezan por 'M'.

```
select nombre, apellidos, salario_bruto
from empleado
where apellidos like 'M%';
```

	nombre	apellidos	salario_bruto
▶	Benito	Martinez Iglesias	25000.00
	Carlos	Martinez Diaz	31500.00

- Seleccionar el *nombre* y *apellidos* de todos los empleados que tienen un *nombre* formado por 6 letras, acabado en 'O'.

```
select nombre, apellidos
from empleado
where nombre like '____o';
```

Result Grid			Filter Rows:
	nombre	apellidos	
▶	Benito	Martinez Iglesias	
	Adolfo	Iglesias Dominguez	

Predicado IS [NOT] NULL

Sirve para comprobar si el contenido de una columna es un valor nulo o no. En SQL, el valor nulo representa un valor desconocido, diferente de 0 y de una cadena vacía. Sintaxis:

```
nombre_columna IS [NOT] NULL
```

Ejemplos:

- Seleccionar *nombre*, *apellidos* y *dni_jefe* de los empleados que no tienen jefe, o no se conoce quién es su jefe, es decir, los que tienen el valor NULL en la columna *dni_jefe*.

```
select nombre, apellidos, dni_jefe
from empleado
where dni_jefe is null;
```

Result Grid				Filter Rows:
	nombre	apellidos	dni_jefe	
▶	Jose Luis	Fernandez Lopez	NULL	
	Dario	Ruiz Macias	NULL	
	Carlos	Martinez Diaz	NULL	

- Seleccionar *nombre*, *apellidos* y *departamento* de los empleados que tienen asignado un departamento, es decir, que tienen un valor distinto de NULL en la columna *cod_departamento*.

```
select nombre, apellidos, cod_departamento
from empleado
where cod_departamento is not null;
```

Result Grid				Filter Rows:
	nombre	apellidos	cod_departamento	
▶	Benito	Martinez Iglesias	1	
	Jose Luis	Fernandez Lopez	1	
	Antonia	Nuñez Bernardes	2	
	Carlos	Martinez Diaz	2	
	Valentina	Hernandez Valin	3	
	Fernanda	Case Rodriguez	4	
	Teolindo	Villar Bernal	5	
	Dario	Ruiz Macias	10	

2.3.4.3. Cláusula ORDER BY

Esta cláusula permite ordenar los resultados de la consulta teniendo en cuenta el contenido de una o más columnas o expresiones. Sintaxis:

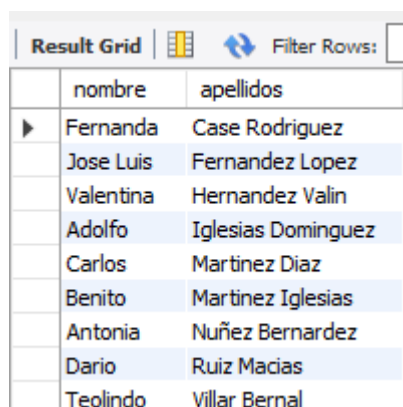
```
[ORDER BY {columna | expresión | posición} [ASC | DESC], ...]
```

- Los nombres de columna a los que se hace referencia en esta cláusula son los de la tabla del conjunto de resultados de la consulta, por lo tanto, si hay alias asociados a las columnas, pueden ser utilizados.
- Las opciones [ASC | DESC] indican el orden en la que se muestran los resultados (ASC = ascendente, DESC = descendente). Si no se indica nada, se toma ASC como valor por defecto.

Ejemplos:

- Seleccionar *nombre* y *apellidos* de todos los empleados ordenando el resultado alfabéticamente por los apellidos, y en el caso de haber empleados que tengan los mismos *apellidos*, se ordenan alfabéticamente por el *nombre*.

```
select nombre, apellidos  
from empleado  
order by apellidos, nombre;
```



	nombre	apellidos
▶	Fernanda	Case Rodriguez
	Jose Luis	Fernandez Lopez
	Valentina	Hernandez Valin
	Adolfo	Iglesias Dominguez
	Carlos	Martinez Diaz
	Benito	Martinez Iglesias
	Antonia	Nuñez Bernardes
	Dario	Ruiz Macias
	Teolindo	Villar Bernal

- Mostrar el nombre, apellidos y el 20% del *salario_bruto*, que representa la retención a hacer, de todos los empleados ordenando el resultado por la retención, de tal manera que se muestren en primer lugar los que tienen una retención más alta.

```
select nombre, apellidos, salario_bruto * 0.20 as retencion  
from empleado  
order by retencion desc;
```

Result Grid			
Filter Rows:			
	nombre	apellidos	retencion
▶	Jose Luis	Fernandez Lopez	32000.0000
	Adolfo	Iglesias Dominguez	10500.0000
	Antonia	Nuñez Bernardez	8400.0000
	Fernanda	Case Rodriguez	7104.0000
	Dario	Ruiz Macias	6400.0000
	Carlos	Martinez Diaz	6300.0000
	Valentina	Hernandez Valin	5200.0000
	Benito	Martinez Iglesias	5000.0000
	Teolindo	Villar Bernal	3500.0000

2.3.4.4. Cláusula LIMIT

Esta cláusula permite indicar el número de filas que ha de devolver la consulta y, de forma optativa, desde qué posición se van a mostrar. Puede tener uno o dos argumentos numéricos, que deben ser enteros positivos (incluyendo cero). En el caso de utilizar un solo argumento, éste indica el número de filas a mostrar; y en el caso de utilizar dos argumentos, el primero indica el número de fila a partir de la que hay que mostrar el conjunto de resultados, y el segundo representa el nº de filas.

Ejemplos:

- Seleccionar el *nombre*, *apellidos* y *retención* de los 5 empleados con la retención más alta.

```
select nombre, apellidos, salario_bruto * 0.20 as retencion
from empleado
order by retencion desc
limit 5;
```

Result Grid			
Filter Rows:			
	nombre	apellidos	retencion
▶	Jose Luis	Fernandez Lopez	32000.0000
	Adolfo	Iglesias Dominguez	10500.0000
	Antonia	Nuñez Bernardez	8400.0000
	Fernanda	Case Rodriguez	7104.0000
	Dario	Ruiz Macias	6400.0000

La consulta muestra las 5 primeras filas de la tabla de resultados, después de ordenar por la columna *retencion*.

- Seleccionar el *nombre*, *apellidos* y *retención* de los 3 empleados siguientes a los mostrados en el ejemplo anterior, teniendo en cuenta que están ordenados por las retenciones, de mayor a menor.

```
select nombre, apellidos, salario_bruto * 0.20 as retencion
from empleado
order by retencion desc
limit 5,3;
```

Result Grid			
	nombre	apellidos	retencion
▶	Carlos	Martinez Diaz	6300.0000
	Valentina	Hernandez Valin	5200.0000
	Benito	Martinez Iglesias	5000.0000

La consulta muestra 3 filas de la tabla de resultados, empezando por la 6ª, después de ordenar por la columna *retencion*.

2.3.5. Funciones incorporadas en MySQL

Cuando se necesita hacer alguna operación compleja, debe de consultarse el manual para saber si ya existe alguna función implícita en el servidor que la realice, antes de ponerse a crear una función nueva de usuario. MySQL incorpora una gran cantidad de funciones que pueden ser utilizadas en las expresiones contenidas en las consultas. El manual de referencia de MySQL posee información muy detallada de las funciones; aquí se muestra un resumen de las más utilizadas.

2.3.5.1. Funciones de fecha y hora

Permiten obtener información sobre la fecha y hora del sistema en distintos formatos.

- **CURRENT_DATE() o CURDATE()**
Devuelve la fecha actual del sistema, en formato 'aaaa-mm-dd'
- **CURRENT_TIME() o CURTIME()**
Devuelve la hora actual del sistema, en formato 'hh:mm:ss'
- **NOW()**
Devuelve la fecha y hora del sistema, en el formato 'aaaa-mm-dd hh:mm:ss'

2.3.5.2. Funciones de cadenas de caracteres

- **CONCAT**
Concatena una serie de cadenas y devuelve una cadena, o el valor NULL si algún argumento es NULL. Sintaxis:
`CONCAT(cadena1, cadena2 [,cadena3] ...)`
- **LENGTH**
Devuelve la longitud de la cadena en caracteres. Sintaxis:
`LENGTH (cadena)`
- **TRIM**
Elimina los espacios situados a la izquierda y a la derecha en la cadena. Sintaxis:
`TRIM(cadena)`

2.3.5.3. Funciones numéricas

- **ABS**
Obtiene el valor absoluto de un número. Cuando se pasa como parámetro un valor que no es numérico devuelve el valor 0. Sintaxis:
`ABS (número)`

- **ROUND**

Redondea un número decimal al entero más cercano. Se puede utilizar un segundo argumento para indicar el número de decimales con los que debe hacer el redondeo. Sintaxis:

`REDONDEAR (número [,decimales])`

- **TRUNCATE**

Retorna el número truncado con el número de decimales que se pasan como parámetro. Sintaxis:

`TRUNCATE (número, decimales)`

- **RAND**

Devuelve un valor decimal aleatorio en el rango $0 \leq \text{valor} \leq 1.0$. Sintaxis:

`RAND ()`

2.3.5.4. Funciones de agrupamiento o de columna

Permiten hacer cálculos con las columnas de la tabla de resultados de una consulta. Cuando se utilizan en la lista de selección, no pueden ir acompañadas de ninguna otra expresión que no contenga una función de este tipo. Sólo pueden ir acompañadas de nombres de columnas o de expresiones que contengan nombres de columnas cuando se agrupan filas utilizando la cláusula **GROUP BY**, que se verá en una actividad posterior.

- **COUNT**

Cuenta el número de líneas resultantes de la consulta, o el número de valores distintos que toma una expresión (normalmente, una columna). Sintaxis:

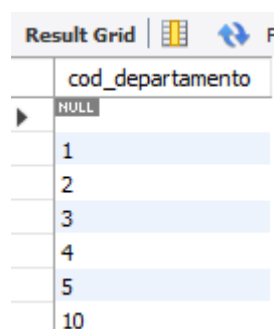
`COUNT (*)`

`COUNT ([DISTINCT] expresión)`

- Cuando se utiliza el símbolo asterisco (*) como parámetro, devuelve el número de filas que tiene la tabla de resultados.
- Cuando se pone como parámetro 'DISTINCT expresión', devuelve el número de valores distintos que toma la expresión en las filas de la tabla de resultados, sin tener en cuenta las que toman el valor NULL.
- Cuando se pone como parámetro sólo una 'expresión', devuelve el número de filas de la tabla de resultados en los que la expresión toma un valor distinto de NULL.

Ejemplos:

select distinct cod_departamento **from** empleado;



cod_departamento
NULL
1
2
3
4
5
10

select count(*), count(departamento), count(**distinct** departamento) **from** empleado;

Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell			
	count(*)	count(cod_departamento)	count(distinct cod_departamento)
▶	9	8	6

■ SUM

Suma los valores que toma la expresión (normalmente, una columna) especificada, para todas las filas resultantes de la consulta. La opción DISTINCT no tiene en cuenta los valores repetidos de la columna. Sintaxis:

SUM ([DISTINCT] expresión)

Ejemplo: Calcular la cantidad total de dinero que destina la empresa a los salarios.

select sum(salario_bruto) **from** empleado;

Result Grid Fil	
	sum(salario_bruto)
▶	422020.00

■ AVG

Calcula la media de los valores que toma la expresión en las filas de la tabla de resultados. La opción DISTINCT no tiene en cuenta los valores repetidos de la columna. Sintaxis:

AVG ([DISTINCT] expresión)

■ MAX

Devuelve el valor más alto que toma la expresión en las filas de la tabla de resultados. Sintaxis:

MAX (expresión)

■ MIN

Devuelve el valor más bajo de la expresión en las filas de la tabla de resultados. Sintaxis:

MIN (expresión)

Valores NULL y las funciones de agrupamiento:

El estándar ANSI/ISO establece unas reglas para el manejo de valores NULL en las funciones de agrupamiento:

- Si alguno de los valores de una columna es NULL, no se tiene en cuenta al hacer los cálculos en una función de agrupamiento.
- Si el valor de todas las columnas es NULL, las funciones SUM, AVG, MAX y MIN devuelven el valor NULL y la función COUNT el valor cero.
- COUNT(*) cuenta filas y no depende de la presencia de valores NULL en las columnas. Cuando se quiere tener en cuenta los valores NULL de una columna, se puede utilizar la fórmula COUNT(columna) que cuenta las filas en las que la columna no es NULL.

2.3.5.5. Otras funciones

A continuación, se muestran de forma muy resumida grupos de funciones que MySQL incorpora y que son utilizadas con menos frecuencia pero que pueden ser de utilidad.

Funciones de control de flujo

- IF

Examina la *expresión1*; y si es verdadera entonces devuelve *expresión2*; si es falsa, devuelve la *expresión3*. Sintaxis:

```
IF(expresión1, expresión2, expresión3)
```

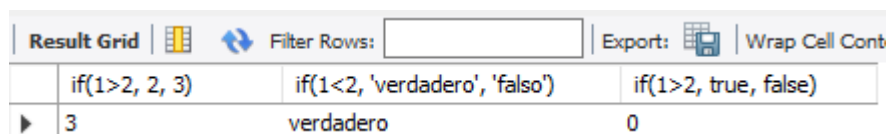
- IFNULL

Cuando la *expresión1* toma el valor NULL, la función devuelve el contenido de *expresión2*, en otro caso devuelve el contenido de *expresión1*. Sintaxis:

```
IFNULL(expresión1, expresión2)
```

Ejemplos:

```
select if(1>2, 2, 3), if(1<2, 'verdadero', 'falso'), if(1>2, true, false);
```



	if(1>2, 2, 3)	if(1<2, 'verdadero', 'falso')	if(1>2, true, false)
▶	3	verdadero	0

```
select apellidos, nombre, ifnull(cod_departamento, 'Sin departamento') as departamento
from empleado
order by cod_departamento;
```



	apellidos	nombre	departamento
▶	Iglesias Dominguez	Adolfo	Sin departamento
	Martinez Iglesias	Benito	1
	Fernandez Lopez	Jose Luis	1
	Núñez Bernardes	Antonia	2
	Martinez Diaz	Carlos	2
	Hernandez Valin	Valentina	3
	Case Rodriguez	Fernanda	4
	Villar Bernal	Teolindo	5
	Ruiz Macias	Dario	10

Funciones de información del sistema

- USER

Muestra información del usuario que hizo la conexión (nombre y host). Sintaxis:

```
USER ()
```

- VERSION

Muestra información de la versión del servidor MySQL. Sintaxis:

```
VERSION ()
```