Bloque 3: Datos e Información

Programación y Computación

2º Bachillerato

Ies Camilo José Cela

Prof: Jesús Núñez y Antonio Jesús Conejo

Bloque 3: Datos e Información.

Introducción

Un tema importante hoy en día es el almacenamiento de los datos, así que adquieren cada vez más importancia *las bases de datos, el almacenamiento en la nube y el Big Data.*
En este tema vamos a estudiar el **lenguaje** de bases de datos **Sql,** concretamente el **gestor MySql**, ya que permite la creación, definición y manipulación de bases de datos y sus elementos como ta- blas y consultas principalmente, tanto de forma local como online.

Comencemos con MySql

Sql está formado por un conjunto de instrucciones para la definición, manipulación y consultas de las bases de datos.

Lenguaje de definición de datos

Para crear una tabla debemos especificar diversos datos: El nombre que le queremos asignar, los nombres de los campos y sus características. Además, puede ser necesario especificar cuáles de estos campos van a ser índices y de qué tipo van a ser.

A continuación os explicamos la sintaxis de esta sentencia y os proponemos una serie de ejemplos prácticos:

Sintaxis

```
Create Table nombre_tabla (
nombre_campo_1 tipo_1
nombre_campo_2 tipo_2
nombre_campo_n tipo_n Key(campo_x,...)
)
```

Pongamos ahora como ejemplo la creación de la tabla pedidos que hemos empleado en capítulos previos:

```
Create Table pedidos (
id_pedido INT(4) NOT NULL AUTO_INCREMENT,
id_cliente INT(4) NOT NULL,
id_articulo INT(4)NOT NULL,
fecha DATE,
cantidad INT(4),
total INT(4),
KEY(id_pedido,id_cliente,id_articulo)
)
```

En este caso creamos los campos id los cuales son considerados de tipo entero de una longitud especificada por el número entre paréntesis. Para id_pedido requerimos que dicho campo se incremente automáticamente (AUTO_INCREMENT) de una unidad a cada introducción de un nuevo registro para, de esta forma, automatizar su creación. Por otra parte, para evitar un mensaje de error, es necesario requerir que los campos que van a ser definidos como índices no puedan ser nu- los

(NOT NULL). El campo fecha es almacenado con formato de fecha (DATE) para permitir su correcta explotación a partir de las funciones previstas a tal efecto. Finalmente, definimos los índi- ces enumerándolos entre paréntesis precedidos de la palabra KEY o INDEX.

Del mismo modo podríamos crear la tabla de *artículos* con una sentencia como ésta:

```
Create Table articulos (
id_articulo INT(4) NOT NULLAUTO_INCREMENT,
titulo VARCHAR(50),
autor VARCHAR(25),
editorial VARCHAR(25),
precio REAL, KEY(id_articulo))
```

En este caso puede verse que los campos alfanuméricos son introducidos de la misma forma que los numéricos. Volvemos a recordar que en tablas que tienen campos comunes es de vital importancia definir estos campos de la misma forma para el buen funcionamiento de la base.

Muchas son las opciones que se ofrecen al generar tablas. No vamos a tratarlas detalladamente pues sale de lo estrictamente práctico. Tan sólo mostraremos algunos de los tipos de campos que pueden ser empleados en la creación de tablas con sus características:

Tipo	Bytes	Descripción
INT **INTEGER**	4	Números enteros. Existen otros tipos de mayor o menor longitud específicos de cada base de datos.
DOUBLE **REAL**	8	Números reales (grandes y con decimales). Permiten almacenar todo tipo de número no entero.
CHAR	1/caracter	Alfanuméricos de longitud fija predefinida
VARCHAR	1/caracter+ 1	Alfanuméricos de longitud variable
DATE	3	Fechas, existen multiples formatos específicos de cada base de datos
BLOB	1/caracter+ 2	Grandes textos no indexables
BIT **BOOLEAN**	1	Almacenan un bit de información (verdadero o falso)

Lenguaje de Manipulación de Datos.

• Los comandos son los siguientes:

–INSERT, insertar o ingresar.

–UPDATE, actualizar o modificar.

–DELETE, borrar o eliminar.

Introducir Información

- **INSERT**

Permite agregar una sola tupla o fila a una tabla. Se debe especificar el nombre de la tabla y una lista de valores para las columnas de la fila.

INSERT INTO nombreDeLaTabla VALUES (valor_col1, valor_col2, ..., valor_coln)

Los valores deberán ingresarse en el mismo orden en que se especificaron los atributos en la instrucción **CREATE TABLE**.

Por ejemplo, si queremos agregar un registro en la tabla PERSONAS, podemos usar:

INSERT INTO PERSONAS VALUES (1234567, 'Fulana', 'de Tal', 'Propios', 9966, 'Av. Burgues', 'fdetal@algo.com','05/03/1979')

- **INSERT**

Una segunda sintaxis de la instrucción INSERT permite asignar valores a una lista de atributos y en que orden. En este caso, los atributos con valores NULL o DEFAULT se pueden omitir.

Por ejemplo, :.si queremos introducir un registro o fila para una nueva persona del cual sólo conocemos los atributos NOMBRE, APELLIDO y CI, podemos usar:

INSERT INTO PERSONAS (NOMBRE, APELLIDO,CI) VALUES ('Mengana','de Tal',221234)

Modificar Información

- **UPDATE**

Esta instrucción actualiza los valores de una fila, en su forma básica:

```
UPDATE nombreDeLaTabla
SET columna2 = valor2
columna3 = valor3
columnaN = valorN
WHERE columna1 = valor1--(col1 clave primaria)
```

• Modifica el NOMBRE y el APELLIDO de la PERSONA de CI = 1234567

```
UPDATE PERSONAS
SET NOMBRE = 'Menganito', APELLIDO = 'de Tal'
WHERE CI=1234567;
```

• El precio de TODOS los productos es 10 euros.

```
UPDATE PRODUCTOS SET PRECIO=10;
```

• Una segunda forma de la instrucción UPDATE permite modificar valores a un conjunto de filas, que cumplan con determinada condición.

Ejemplo: Otorgar a los CAJEROS que tienen menos de 2500€ de sueldo, un aumento del 10%.

```
UPDATE CAJEROS SET SUELDO= AUMENTO * 1.1 WHERE SUELDO< 2500;
```

Eliminar Información

• DELETE

Esta instrucción borra los valores de una tabla, en su forma básica. **DELETE FROM nombre_tabla;**

• **Borra todas las filas de la tabla PERSONAS**

```
DELETE FROM PERSONAS;
```

• También, esta instrucción borra los valores de una fila, determinando un valor para su clave primaria.

```
DELETE FROM PERSONAS WHERE CI = 1234567
```

Borra los datos de la persona CI = 1234567

• **Se puede eliminar todas las filas de una tabla que cumplan con una condición.**

Ejemplo: Borrar los PRODUCTOS que pertenecen al tipo Quesos y Fiambres

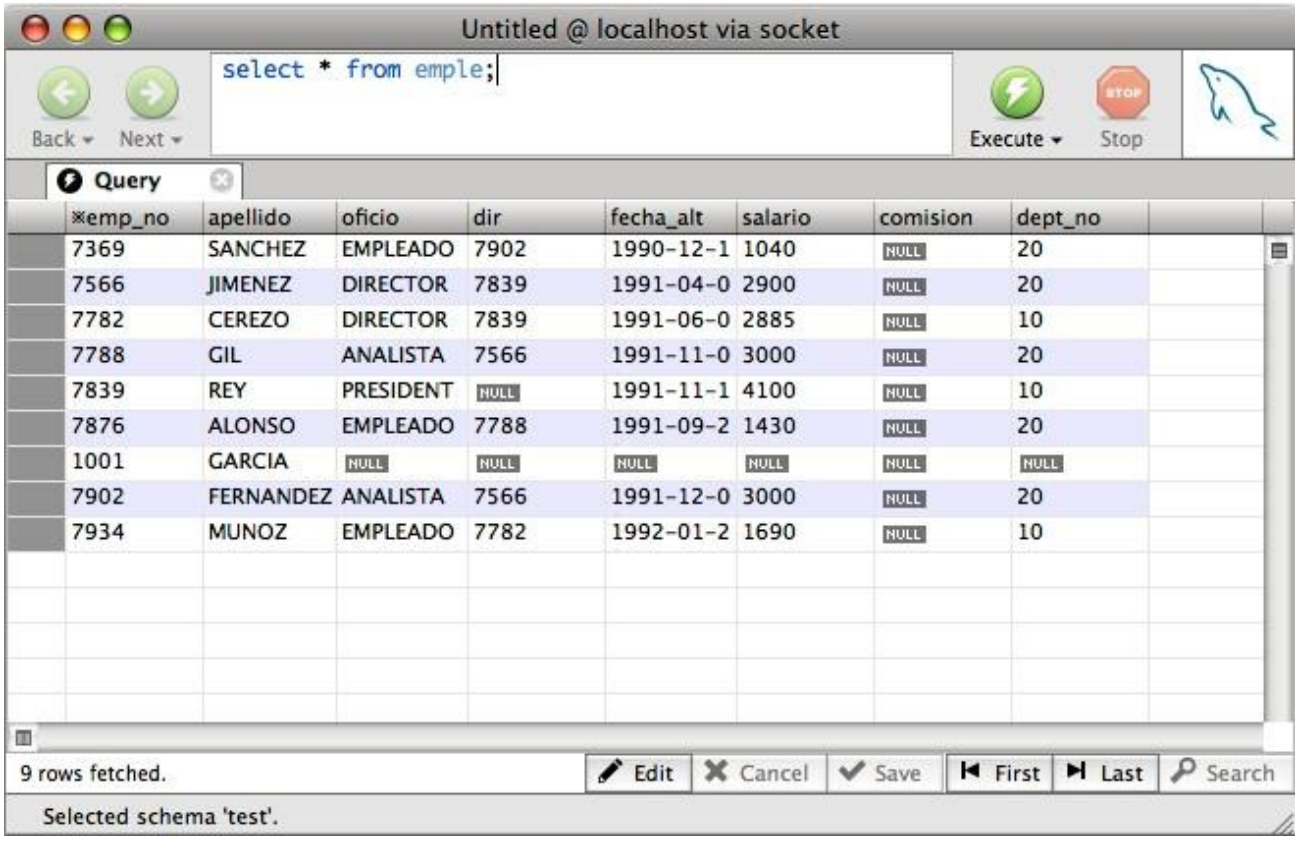
```
DELETE FROM PRODUCTOS WHERE TIPO='Quesos y Fiambres'
```

INTRODUCCIÓN A SQL. CONSULTAS SENCILLAS A LA BASE DE DATOS

La sentencia SELECT

• Se utiliza para realizar consultas sobre la base de datos
• Su formato básico es el siguiente**:** **SELECT * FROM nombredetabla;**

*La primera palabra es **SELECT** e indica que se quiere realizar una consulta. * quiere decir que se quieren recuperar todos los campos de la tabla. **FROM** indica la(s) tabla(s) sobre la que se realiza la consulta. A la derecha de FROM se escribe el nombre de la tabla sobre la que se realiza la consulta.*



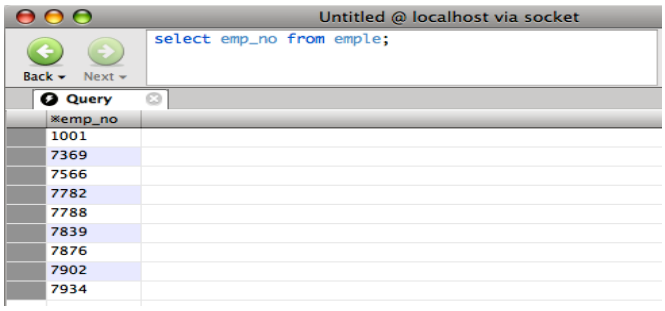
emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7369	SANCHEZ	EMPLEADO	7902	1990-12-1	1040	NULL	20
7566	JIMENEZ	DIRECTOR	7839	1991-04-0	2900	NULL	20
7782	CEREZO	DIRECTOR	7839	1991-06-0	2885	NULL	10
7788	GIL	ANALISTA	7566	1991-11-0	3000	NULL	20
7839	REY	PRESIDENT	NULL	1991-11-1	4100	NULL	10
7876	ALONSO	EMPLEADO	7788	1991-09-2	1430	NULL	20
1001	GARCIA	NULL	NULL	NULL	NULL	NULL	NULL
7902	FERNANDEZ	ANALISTA	7566	1991-12-0	3000	NULL	20
7934	MUNOZ	EMPLEADO	7782	1992-01-2	1690	NULL	10

Seleccion de columnas

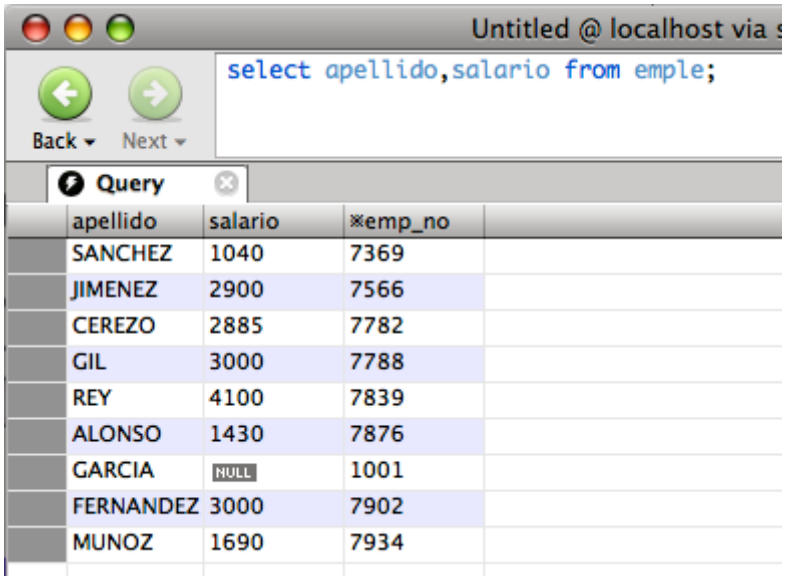
• Con ***** a la derecha del **SELECT** se visualizan todas las columnas de la tabla.
• Si queremos consultas ciertas columnas, escribiremos sus nombres separados por comas.
• **SELECT campo1,campo2,campo3...FROM nombretabla;**

Ejemplos:

• Obtener el número de todos los empleados**:** ***SELECT emp_no FROM emple;***



Obtener el apellido y salario de todos los empleados: ***SELECT apellido,salario FROM emple;***



apellido	salario	emp_no
SANCHEZ	1040	7369
JIMENEZ	2900	7566
CEREZO	2885	7782
GIL	3000	7788
REY	4100	7839
ALONSO	1430	7876
GARCIA	NULL	1001
FERNANDEZ	3000	7902
MUNOZ	1690	7934

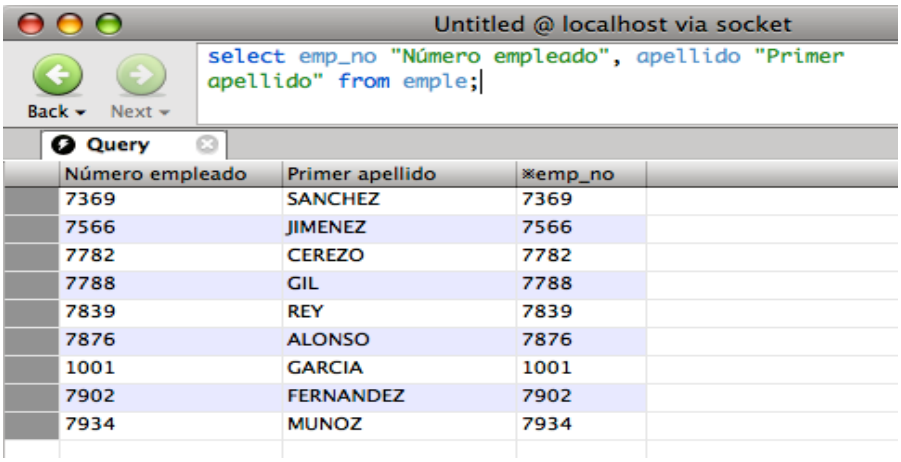
Alias de tablas

• Se puede asociar un nuevo nombre a la tabla utilizando alias.
• **SELECT E.apellido, E.salario FROM Temple E;**
• A la tabla emple se le asigna un nuevo nombre (E).

Alias de columnas

• Los nombres de las columnas se usan como cabeceras de presentación.
• Si son demasiados largos, se pueden cambiar utilizando alias de columnas.
• El alias se pone entre comillas simples o dobles a la derecha de la columna deseada.

SELECT apellido “Primer apellido”,emp_no “Número empleado”FROM emple;



Número empleado	Primer apellido	emp_no
7369	SANCHEZ	7369
7566	JIMENEZ	7566
7782	CEREZO	7782
7788	GIL	7788
7839	REY	7839
7876	ALONSO	7876
1001	GARCIA	1001
7902	FERNANDEZ	7902
7934	MUNOZ	7934

Columnas Calculadas

• Una consulta puede incluir columnas cuyos valores se obtienen a partir de los valores almacenados en columnas de la tabla.

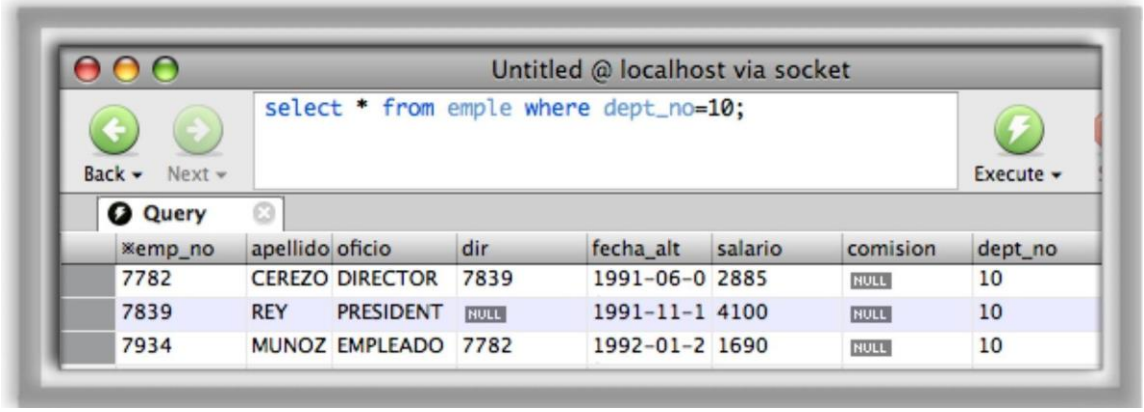
SELECT salario*2 “Salario duplicado”, comision/2 “Comisión dividida por 2” FROM emple;

Clausula Where

• La cláusula WHERE se utiliza para obtener aquellos datos que cumplan la condición expresada.
• Su formato es el siguiente:

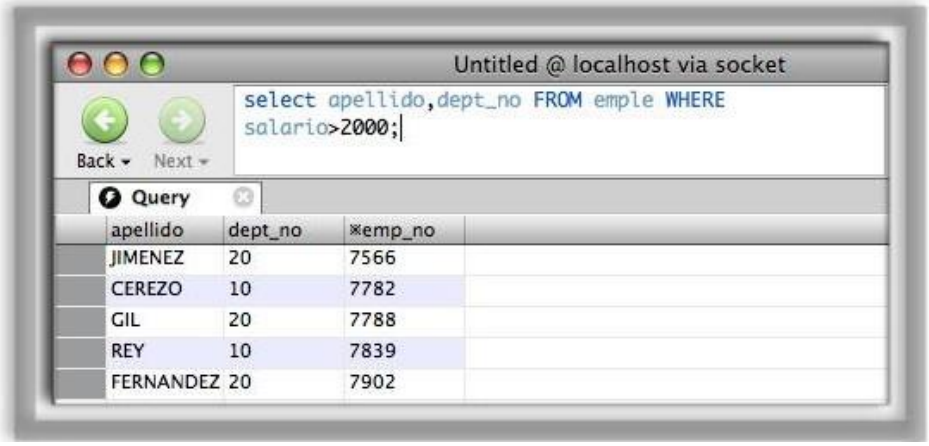
SELECT columna1, columna2, ...
FROM nombredetabla
WHERE condición;

SELECT * FROM emple WHERE dept_no=10;



emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7782	CEREZO	DIRECTOR	7839	1991-06-0	2885	NULL	10
7839	REY	PRESIDENT	NULL	1991-11-1	4100	NULL	10
7934	MUNOZ	EMPLEADO	7782	1992-01-2	1690	NULL	10

• **SELECT apellido, dept_no FROM emple WHERE salario>2000;**



apellido	dept_no	emp_no
JIMENEZ	20	7566
CEREZO	10	7782
GIL	20	7788
REY	10	7839
FERNANDEZ	20	7902

SELECT * FROM emple WHERE salario>2000 AND dept_no=20;

emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7566	JIMENEZ	DIRECTO	7839	1991-04-0	2900	NULL	20
7788	GIL	ANALISTA	7566	1991-11-0	3000	NULL	20
7902	FERNANDEZ	ANALISTA	7566	1991-12-0	3000	NULL	20

SELECT * FROM emple WHERE salario>2000 AND (dept_no=10 OR dept_no=20);

La cláusula ORDER BY

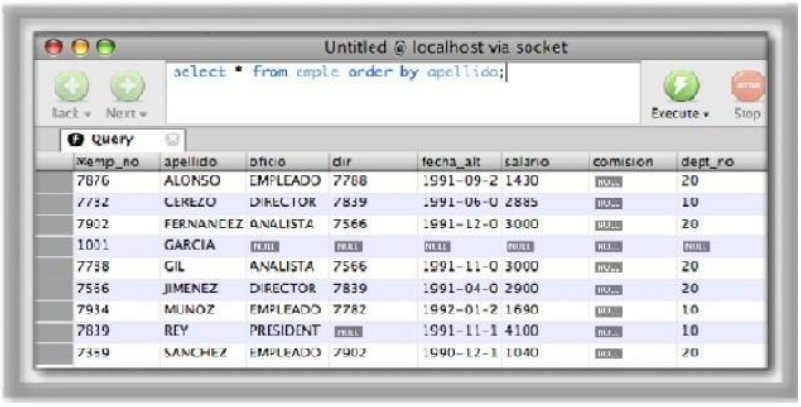
Permite ordenar los resultados de una consulta.

SELECT columna1, columna2, ...
FROM nombredetabla
WHERE condición
ORDER BY campo1 [ASC|DESC],
campo2 [ASC|DESC], ... ;

Order by ASC especifica criterio de ordenación ascendente. Es la opción por defecto.
Order by DESC indica ordenación descendente.

• Obtenemos los datos de los empleados ordenando el resultado por apellidos.

SELECT * FROM emple ORDER BY apellido;



SELECT * FROM emple ORDER BY oficio ASC, apellido DESC;

• Ordenamos el resultado por oficio, y dentro de oficio por nombre.
• SELECT * FROM emple ORDER BY oficio ASC, apellido DESC;

• También se puede especificar la ordenación por un número.
• Este número indica la posición de la columna a la derecha de SELECT por la que se quiere ordenar el resultado

```
SELECT * FROM emple ORDER BY 3 ASC, 2 DESC;
```

Distinct y All

• DISTINCT recupera las filas que son distintas.
• ALL recupera todas las filas aunque algunas estén repetidas. Es la opción por defecto.
• Ambas palabras vienen detrás de SELECT

Condición de búsqueda más utilizada en la cláusula WHERE

• Compara el valor de una expresión con otra.

expresión operador expresión

• Operadores: <, >, >=, <=, !=, <>
• Empleados cuyo apellido tenga 3 caracteres y termine en M.

```
SELECT * FROM emple WHERE apellido LIKE '_M';
```

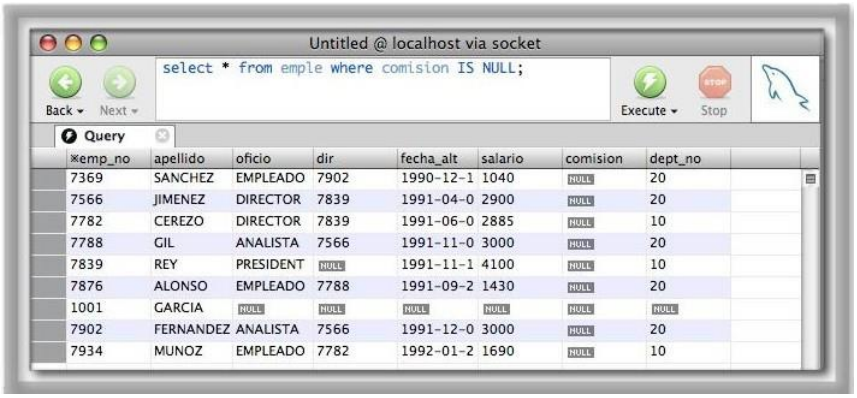
• Empleados cuyo apellido tenga 2 caracteres y empiece por la letra M.

```
SELECT * FROM emple WHERE apellido LIKE 'M_';
```

• El campo de una fila es NULL si no contiene ningún valor.
• Para comprobar si un campo tiene un valor nulo utilizamos la expresión IS NULL.

Empleados que no tienen comisión.

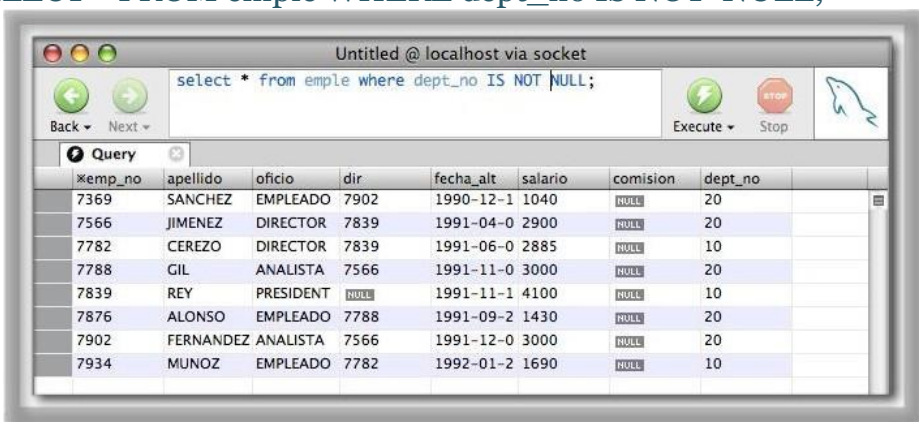
```
SELECT * FROM emple WHERE comision IS NULL;
```



emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7369	SANCHEZ	EMPLEADO	7902	1990-12-1	1040	NULL	20
7566	JIMENEZ	DIRECTOR	7839	1991-04-0	2900	NULL	20
7782	CEREZO	DIRECTOR	7839	1991-06-0	2885	NULL	10
7788	GIL	ANALISTA	7566	1991-11-0	3000	NULL	20
7839	REY	PRESIDENT	NULL	1991-11-1	4100	NULL	10
7876	ALONSO	EMPLEADO	7788	1991-09-2	1430	NULL	20
1001	GARCIA	NULL	NULL	NULL	NULL	NULL	NULL
7902	FERNANDEZ	ANALISTA	7566	1991-12-0	3000	NULL	20
7934	MUNOZ	EMPLEADO	7782	1992-01-2	1690	NULL	10

Empleados que están en algún departamento:

```
SELECT * FROM emple WHERE dept_no IS NOT  NULL;
```



emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7369	SANCHEZ	EMPLEADO	7902	1990-12-1	1040	NULL	20
7566	JIMENEZ	DIRECTOR	7839	1991-04-0	2900	NULL	20
7782	CEREZO	DIRECTOR	7839	1991-06-0	2885	NULL	10
7788	GIL	ANALISTA	7566	1991-11-0	3000	NULL	20
7839	REY	PRESIDENT	NULL	1991-11-1	4100	NULL	10
7876	ALONSO	EMPLEADO	7788	1991-09-2	1430	NULL	20
7902	FERNANDEZ	ANALISTA	7566	1991-12-0	3000	NULL	20
7934	MUNOZ	EMPLEADO	7782	1992-01-2	1690	NULL	10

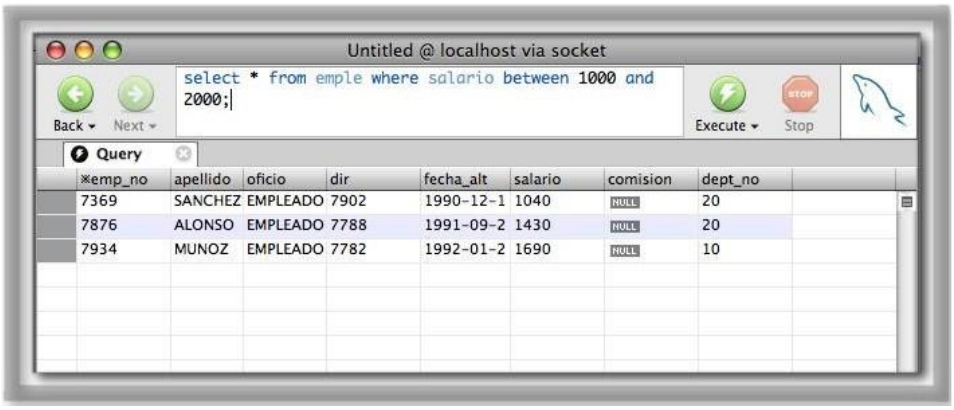
Between

• Comprueba si un valor está comprendido en un rango de valores.

<expresión> BETWEEN valor_inicial AND valor_final

• Empleados que tengan un salario entre 1000 y 2000 euros.

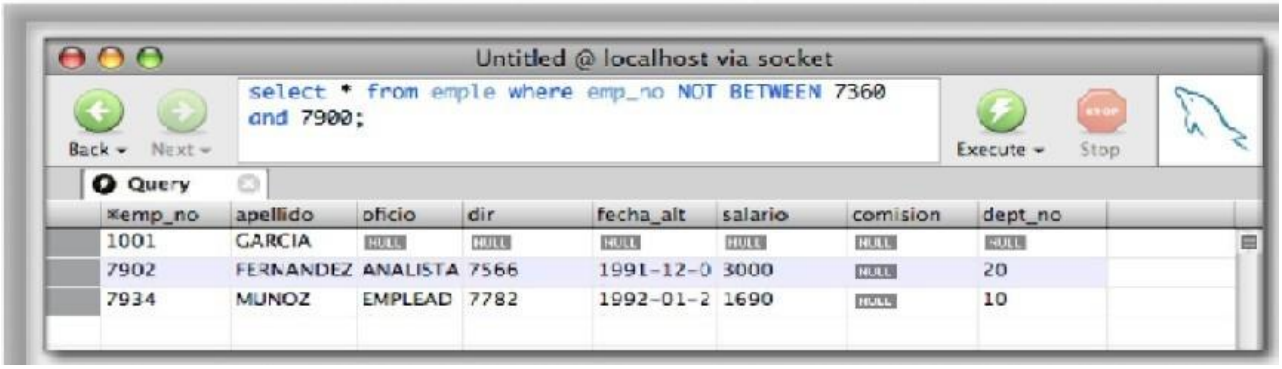
```
SELECT * FROM emple WHERE salario BETWEEN 1000 AND 2000;
```



emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7369	SANCHEZ	EMPLEADO	7902	1990-12-1	1040	NULL	20
7876	ALONSO	EMPLEADO	7788	1991-09-2	1430	NULL	20
7934	MUNOZ	EMPLEADO	7782	1992-01-2	1690	NULL	10

Empleados cuyo código no esté comprendido entre 7360 y 7900.

```
SELECT * FROM emple WHERE emp_no NOT BETWEEN 7360 AND 7900;
```



emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
1001	GARCIA	NULL	NULL	NULL	NULL	NULL	NULL
7902	FERNANDEZ	ANALISTA	7566	1991-12-0	3000	NULL	20
7934	MUNOZ	EMPLEAD	7782	1992-01-2	1690	NULL	10

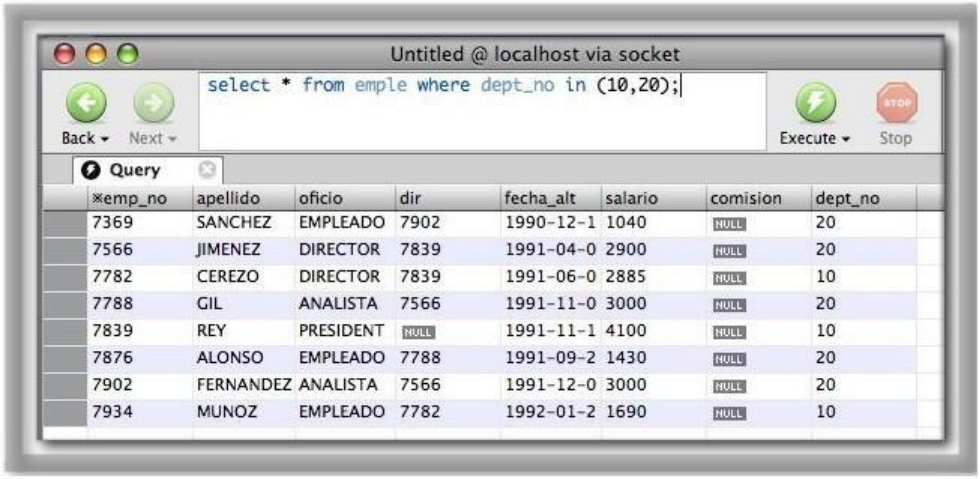
IN

Permite comprobar si una expresión pertenece a un conjunto de valores.

<expresión> IN (lista de valores separados por comas)

Obtener los empleados de los departamentos 10 ó 20:

```
SELECT * FROM emple WHERE dept_no IN (10,20);
```

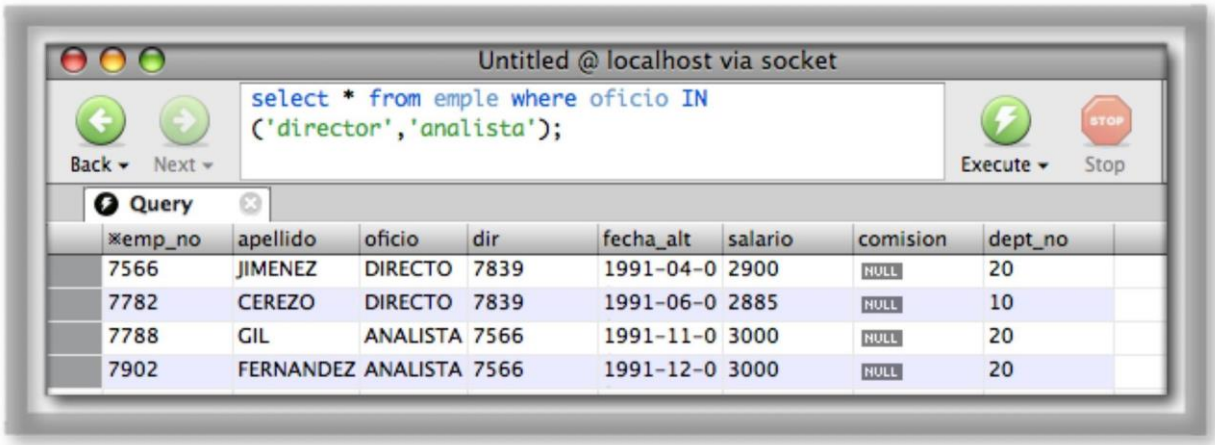


emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7369	SANCHEZ	EMPLEADO	7902	1990-12-1	1040	NULL	20
7566	JIMENEZ	DIRECTOR	7839	1991-04-0	2900	NULL	20
7782	CEREZO	DIRECTOR	7839	1991-06-0	2885	NULL	10
7788	GIL	ANALISTA	7566	1991-11-0	3000	NULL	20
7839	REY	PRESIDENT	NULL	1991-11-1	4100	NULL	10
7876	ALONSO	EMPLEADO	7788	1991-09-2	1430	NULL	20
7902	FERNANDEZ	ANALISTA	7566	1991-12-0	3000	NULL	20
7934	MUNOZ	EMPLEADO	7782	1992-01-2	1690	NULL	10

SELECT * FROM emple WHERE dept_no NOT IN (10);

emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7369	SANCHEZ	EMPLEADO	7902	1990-12-1	1040	NULL	20
7566	JIMENEZ	DIRECTOR	7839	1991-04-0	2900	NULL	20
7788	GIL	ANALISTA	7566	1991-11-0	3000	NULL	20
7876	ALONSO	EMPLEADO	7788	1991-09-2	1430	NULL	20
7902	FERNANDEZ	ANALISTA	7566	1991-12-0	3000	NULL	20

SELECT * FROM emple WHERE oficio IN ('DIRECTOR','ANALISTA');



emp_no	apellido	oficio	dir	fecha_alt	salario	comision	dept_no
7566	JIMENEZ	DIRECTO	7839	1991-04-0	2900	NULL	20
7782	CEREZO	DIRECTO	7839	1991-06-0	2885	NULL	10
7788	GIL	ANALISTA	7566	1991-11-0	3000	NULL	20
7902	FERNANDEZ	ANALISTA	7566	1991-12-0	3000	NULL	20