

Tema 8

Realización de consultas.

"Torture the data, and it will confess to anything."

– Ronald Coase

El objetivo de una BdD no es tanto almacenar datos como que se puedan consultar (para lo que es imprescindible lo primero).

8.1. La sentencia SELECT.

Para obtener información almacenada en una base de datos.

Es muy versátil ya que incluye muchas partes opcionales. ¹

¹Para limitar el número de resultados diferentes SGBD usan soluciones distintas, así en MySQL se utiliza LIMIT, en Oracle se usa FETCH y en SQL Server o MS Access se utiliza TOP, que no se incluye en la sintaxis anterior por utilizarse en otro punto de la instrucción.

```
SELECT [ALL | DISTINCT | DISTINCTROW]
    expresion_select,...
FROM referencias_de_tablas
WHERE condiciones
[GROUP BY {nombre_col | expresion | posicion}
[ASC | DESC], ... [WITH ROLLUP]]
[HAVING condiciones]
[ORDER BY {nombre_col | expresion | posicion}
[ASC | DESC] ,...]
[LIMIT | FETCH {[desplazamiento,] contador |
    contador OFFSET desplazamiento}]
```

la forma más básica (obtener todos los datos de una tabla) es:

```
SELECT * FROM empleados;
```

la clausula FROM sirve para indicar en qué tabla (o tablas) realizamos la búsqueda.

Es preferible determinar las columnas y su orden:

```
SELECT id_empleado, nombre, apellido FROM empleados;
```

Importante

En un SGBD puede haber varias BdD por lo que antes de realizar cualquier operación hay que decir que base de datos usaremos:

```
use empleados;
```

donde empleados es el nombre de la BdD.

8.1.1. DISTINCT y ALL.

ALL es la opción por defecto (opcional). Muestra todo incluyendo repetidos.

Permite obtener valores únicos de un campo.

Debe ir obligatoriamente a continuación de SELECT.

```
SELECT id_trabajo FROM empleados;

/* Equivalente a */
SELECT ALL id_trabajo FROM empleados;
```

AC_ACCOUNT	IT_PROG	SA_REP	ST_CLERK
AC_MGR	MK_MAN	SA_REP	ST_CLERK
AD_ASST	MK_REP	SA_REP	ST_CLERK
AD_ASST	MK_REP	SA_REP	ST_CLERK
AD_ASST	SA_MAN	SA_REP	ST_CLERK
AD_ASST	SA_REP	ST_CLERK	ST_CLERK
AD_PRES	SA_REP	ST_CLERK	ST_CLERK
AD_PRES	SA_REP	ST_CLERK	ST_CLERK
AD_VP	SA_REP	ST_CLERK	ST_MAN
IT_PROG	SA_REP	ST_CLERK	
IT_PROG	SA_REP	ST_CLERK	

devuelve valores repetidos mientras que con DISTINCT no:

```
SELECT DISTINCT id_trabajo FROM empleados;
```

AC_ACCOUNT	MK_MAN
AC_MGR	MK_REP
AD_ASST	SA_MAN
AD_PRES	SA_REP
AD_VP	ST_CLERK
IT_PROG	ST_MAN

Si filtramos por más de un campo, se buscan los conjuntos diferentes:

```
SELECT DISTINCT id_trabajo, salario FROM empleados;
```

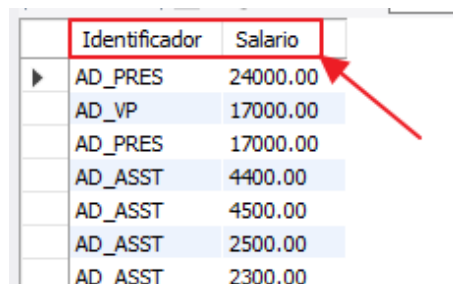
AD_PRES 24000.00	ST_CLERK 2600.00
AD_VP 17000.00	ST_CLERK 2500.00
AD_PRES 17000.00	IT_PROG 9000.00
AD_ASST 4400.00	IT_PROG 6000.00
AD_ASST 4500.00	IT_PROG 4200.00
AD_ASST 2500.00	SA_MAN 10500.00
AD_ASST 2300.00	SA_REP 11000.00
MK_MAN 13000.00	SA_REP 8600.00
MK_REP 6000.00	SA_REP 7000.00
ST_MAN 5800.00	SA_REP 10500.00
ST_CLERK 3500.00	AC_MGR 12000.00
ST_CLERK 3100.00	AC_ACCOUNT 8300.00

8.1.2. Alias.

Se pueden asignar alias a nombres de columnas:

```
SELECT DISTINCT id_trabajo AS Identificador, salario AS Salario
FROM empleados;
```

Se usan en el resultado. (Figura 8.1).



Identificador	Salario
AD_PRES	24000.00
AD_VP	17000.00
AD_PRES	17000.00
AD_ASST	4400.00
AD_ASST	4500.00
AD_ASST	2500.00
AD_ASST	2300.00

Figura 8.1: Resultado con alias para nombres de columnas.

y para nombres de tablas (lo usaremos más adelante):

```
SELECT id_trabajo FROM empleados AS emp;
```

8.1.3. Nombres de tablas.

Al indicar el nombre de una columna se puede especificar el nombre de la tabla a la que pertenece. Ahora parece innecesario pero **más adelante nos hará falta**:

```
SELECT DISTINCT empleados.id_trabajo FROM empleados;
```

Podemos combinar todo:

```
SELECT DISTINCT emp.id_trabajo AS Identificador, emp.salario AS Salario  
FROM empleados AS emp;
```

Realmente el uso de AS es opcional, pero eliminarlo a veces puede hacer que la consulta no quede muy clara:

```
SELECT DISTINCT emp.id_trabajo Identificador, emp.salario Salario  
FROM empleados emp;
```

Ejercicios 8.1

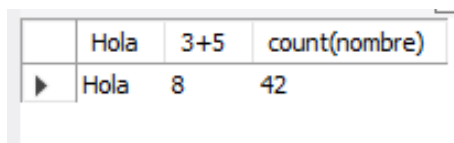
En MySQL Workbench, si quieres ejecutar una sentencia de un archivo, pon el cursor encima y pulsa *Ctrl + Enter*.

1. Selecciona todos los elementos de la tabla *departamentos*.
2. Selecciona las columnas *nombre_departamento* e *id_localizacion* de la tabla *departamentos*. Usa el nombre de la tabla como prefijo de los nombres de las columnas para especificar a qué tabla pertenecen.
3. Selecciona los diferentes *id_localizacion* que hay en la tabla *departamentos*.
4. Selecciona los valores diferentes (en conjunto) de *min_salario* y *max_salario* de la tabla *trabajos*.
5. Añade alias a la consulta anterior tanto para las columnas como para el nombre de tabla. Utiliza el alias de la tabla como prefijo para los nombres de columnas.

8.1.4. Constantes, expresiones y funciones.

No solo podemos seleccionar valores de columnas. También podemos usar valores constantes, resultados de expresiones y funciones:

```
SELECT "Hola", 3+5, count(nombre) FROM empleados;
```



	Hola	3+5	count(nombre)
▶	Hola	8	42

Figura 8.2: Usar elementos diferentes de nombres de columnas.

(Figura 8.2).

```
SELECT "Nombre: ", nombre, "Apellido: ", apellido FROM empleados;
```

Se pueden usar con Alias:

```
SELECT count(nombre) AS "Total Empleados" FROM empleados;
```

Se pueden mezclar:

```
SELECT concat(nombre, " ", apellido) AS Nombre, salario FROM empleados;
```

Nota

Aprenderemos más sobre funciones a lo largo del curso, ya que para usar algunas necesitamos elementos que todavía no hemos visto.

8.2. Filtrar los resultados: la cláusula WHERE.

Mostrar solo las filas que cumplan una determinada condición.

```
SELECT * FROM empleados WHERE id_trabajo = "ST_CLERK";
```

No es necesario que la (o las) columna que se usa al filtrar aparezca entre las seleccionadas:

```
SELECT nombre, apellido FROM empleados WHERE id_trabajo = "ST_CLERK";
```

8.2.1. Operadores.

Aunque algunos se pueden usar en otras partes, aquí es donde tienen más relevancia.

Se pueden usar paréntesis para alterar la precedencia.

8.2.1.1. Operadores lógicos.

Cuando queremos aplicar más de una condición.

Álgebra booleana: *verdadero o falso*.

En MySQL:

- Verdadero: TRUE o 1.
- Falso: FALSE o 0.
- Añade un tercer valor: desconocido o NULL.

Operador	Significado
AND o &&	Y
OR o	O
NOT o !	NO (negación).
XOR	O exclusivo. No es estándar SQL pero muchos SGBDs lo incluyen.

```
SELECT nombre, apellido FROM empleados WHERE id_trabajo = "ST_CLERK"
AND salario > 2500;

SELECT nombre, apellido FROM empleados WHERE apellido = "López"
OR apellido = "González";
```

Al introducir NULL se pueden dar resultados sorprendentes: vamos a [verlo](#).

La precedencia de operadores lógicos es:

1. Comparaciones. (Ver punto a continuación).
2. NOT.
3. AND.
4. OR.

8.2.1.2. Operadores de comparación.

Comparar valores de columnas, resultados de operaciones, constantes...

Devuelven valores lógicos: *verdadero* o *falso*.

Operador	Significado
=	Igualdad.
!=, <>, ^=	Desigualdad.
<	Menor que.
>	Mayor que.
<=	Menor o igual que.
>=	Mayor o igual que.
IN	Igual que cualquiera de los miembros entre paréntesis.
NOT IN	Distinto que cualquiera de los miembros entre paréntesis.
BETWEEN	Entre. Contenido dentro del rango.
NOT BETWEEN	Fuera del rango.
LIKE	Se utiliza sobre todo con textos y permite obtener columnas cuyo valor en un campo cumpla una condición textual. Utiliza una cadena que puede contener los símbolos " %" que sustituye a un conjunto de caracteres o " _ " que sustituye a un carácter.
NOT LIKE	Lo contrario al anterior. Los casos que no lo cumplan.
IS NULL	Devuelve verdadero si el valor del campo de la fila que examina es nulo.
IS NOT NULL	Devuelve verdadero si el valor del campo de la fila que examina NO es nulo.

Nota

Realmente los casos en los que aparece NOT son variaciones en las que se está usando dicho operador. Se incluyen para explicitar que su uso es correcto.


```

SELECT nombre, apellido FROM empleados WHERE id_trabajo <> "ST_CLERK";

SELECT nombre, apellido FROM empleados WHERE id_trabajo <> "ST_CLERK"
    AND (apellido = "López" OR apellido = "González");

SELECT nombre, apellido FROM empleados WHERE comision IS NOT NULL;

SELECT nombre, apellido FROM empleados WHERE salario
    BETWEEN 2000 AND 3000;

SELECT nombre, apellido FROM empleados WHERE
    apellido IN ("Smith", "Schroeder", "Martín", "Ferrerías");

```

8.2.1.3. Operadores aritméticos.

Operador	Significado
+	Suma.
-	Resta. Delante de un número cambia el signo.
*	Multiplicación.
/	División.
DIV	División entera.
% o MOD	Módulo. Resto de la división entera.

```

SELECT 3+5;

SELECT id_departamento * 100 + id_director FROM empleados;

SELECT id_director % id_departamento FROM empleados;

SELECT nombre, apellido FROM empleados WHERE salario * comision > 1000;

SELECT nombre, apellido, salario * comision AS "Comisión calculada"
    FROM empleados WHERE salario * comision > 1000;

```

Nota

Es importante destacar que las dos últimas consultas no modifican los valores de los campos. El cálculo se realiza para mostrar el resultado. Se denominan *campos calculados*.

8.2.1.4. Operador LIKE.

Comparar cadenas con patrones.

Dos comodines básicos:

- `_` : un único carácter.
- `%` : cualquier número de caracteres, incluso ninguno.

También para hacer comparaciones sin diferenciar entre mayúsculas y minúsculas.

Dependiendo del SGBD puede ignorar también "modificadores" de letras como las tildes.

```
SELECT nombre, apellido FROM empleados WHERE apellido LIKE "Go%";

SELECT nombre, apellido FROM empleados WHERE nombre LIKE "JUAN MANUEL";

SELECT nombre, apellido FROM empleados WHERE apellido LIKE "Gonza__";

SELECT nombre, apellido FROM empleados WHERE apellido LIKE "Ha__is%";

SELECT nombre, apellido FROM empleados WHERE salario LIKE "2%";
```

¿Qué sucede si quisiéramos buscar un `_` o un `%`? para solucionarlo se usa un carácter de escape que suele ser `\`

Así:

```
SELECT nombre, apellido FROM empleados WHERE email LIKE "jm\_da@";
```

buscaría los empleados con correo que empezara por jm_da@.

MySQL permite usar otros caracteres de escape con la cláusula opcional `ESCAPE`:

```
SELECT nombre, apellido FROM empleados WHERE email  
LIKE 'jm!_da@%' ESCAPE '!';
```

Ejercicios 8.2

1. Selecciona todos los empleados del departamento 40.
2. Repite la consulta anterior evitando los empleados sin comisión.
3. Selecciona los departamentos con identificador del director mayor que 300 y menor que 600.
4. Selecciona la dirección y el código postal de las localizaciones que no sean ni de España ni de Estados Unidos.
5. Selecciona los salarios mínimo y máximo para los trabajos Contable, Director de contabilidad, Vicepresidente, Programador y Director de ventas.
6. Selecciona todos los empleados que tengan comisión por su trabajo.
7. Selecciona todos los empleados cuyo apellido termine por z.
8. Selecciona todos los empleados cuyo apellido empiece por g y termine por z.
9. Selecciona todos los empleados cuyo teléfono empiece por 9546 y termine por 54 teniendo en medio otros tres dígitos.

8.3. Ordenar los resultados: la cláusula ORDER BY.

Se pueden ordenar los resultados por uno o más campos y de forma ascendente o descendente.

No es necesario que los campos por los que se ordena aparezcan en la selección.

Por defecto se ordena de forma ascendente.

```
SELECT * FROM empleados ORDER BY apellido;  
  
SELECT * FROM empleados WHERE id_trabajo='ST_CLERK'  
ORDER BY apellido DESC;  
  
SELECT nombre, apellido FROM empleados WHERE id_trabajo='ST_CLERK'  
ORDER BY salario ASC, fecha_contratacion DESC, telefono;
```

8.4. Limitar el número de resultados.

Aunque no es necesario usarlo con `ORDER BY` muchas veces se hace.

Nota

Otros gestores de bases de datos utilizan palabras reservadas diferentes como `FETCH`, `ROWNUM`, `START`, `SKIP`, etc.

Dos formatos:

- Sencillo: los X primeros resultados.
- Con desplazamiento: los X primeros resultados a partir de Y.

```
/* Mostrar los 10 primeros */  
SELECT * FROM empleados LIMIT 10;  
  
/* Mostrar del 6 al 15 */  
SELECT * FROM empleados LIMIT 5, 10;  
  
SELECT * FROM empleados ORDER BY apellido LIMIT 8;
```

Ejercicios 8.3

1. Selecciona todos los empleados del departamento 40 ordenados por orden de lista de apellido.
2. Selecciona todos los empleados del departamento 40 ordenados por salario decreciente y comisión creciente.
3. Repite la consulta anterior evitando los empleados sin comisión.
4. Selecciona los departamentos con identificador del director mayor que 300 y menor que 600 ordenados de mayor a menor identificador del director.
5. Selecciona el nombre, apellido y salario de los 6 empleados que más cobran.
6. Selecciona el nombre y apellidos de los empleados 3 empleados con sueldos más bajos de entre los que cobran comisión.
7. Selecciona los 10 empleados más antiguos.
8. Selecciona los 10 empleados más antiguos entre los contratados después del 1 de enero del 2000.

8.5. Funciones.

Aunque el estándar SQL define solo unas pocas funciones, los SGBDs incluyen muchas más. Veremos las estándar y algunas de las que [añade](#) MySQL.

8.5.1. Funciones numéricas.

[Lista](#) completa.

Útiles:

- ABS()
- CEIL()
- FLOOR()
- ROUND()
- TRUNCATE()

```
SELECT ROUND(salario * comision / 100) AS calculo FROM empleados
WHERE comision IS NOT NULL;

SELECT nombre, apellido FROM empleados WHERE comision IS NOT NULL
AND FLOOR(salario * comision) <= 2000;
```

8.5.2. Funciones de cadenas de caracteres.

[Lista](#) completa.

Útiles:

- CHAR_LENGTH()
- CONCAT()
- FORMAT()
- LOWER()
- UPPER()
- TRIM()

```
SELECT nombre, CHAR_LENGTH(nombre) FROM empleados;

SELECT CONCAT(nombre, " ", apellido) AS Nombre FROM empleados;

SELECT FORMAT(salario * comision / 100, 2) AS calculo FROM empleados
WHERE comision IS NOT NULL;

SELECT UPPER(apellido) FROM empleados;

SELECT UPPER(CONCAT(nombre, " ", apellido)) AS Nombre FROM empleados;
```

Ejercicios 8.4

1. Obtén el entero que esté justo por encima del resultado de dividir tu día de nacimientos entre tu mes de nacimiento.
2. Muestra el resultado de dividir tu día de nacimientos entre tu mes de nacimiento con tres decimales.
3. Selecciona el nombre y el teléfono de cada empleado de tal forma que aparezca por pantalla como *El teléfono de Manuel es 934343456*. Esto es un ejemplo, debes sacar el de todos los empleados.
4. Repite la consulta anterior pero ahora el nombre debe aparecer en mayúsculas.

8.5.3. Funciones de fecha y hora.

[Lista completa.](#)

Útiles:

- Generales:
 - NOW()
 - CURDATE()
 - CURTIME()
 - TIMESTAMP()
- Extraer partes:
 - DATE()
 - TIME()

- YEAR()
- MONTH()
- WEEKOFYEAR()
- DAYOFMONTH()
- LAST_DAY
- DAYOFWEEK()
- DAYOFYEAR()
- HOUR()
- MINUTE()
- SECOND()
- Cálculos:
 - TIMESTAMP()
 - TIMESTAMPADD()
 - TIMESTAMPDIF()
 - DATEDIFF()
 - ADDDATE()
 - ADDTIME()
- Formatos:
 - GET_FORMAT()
 - DATE_FORMAT()
 - STR_TO_DATE()
- Crear fechas y horas:
 - MAKEDATE()
 - MAKETIME()
 - SEC_TO_TIME()

```
SELECT NOW();

SELECT CURDATE();

SELECT MONTH(NOW()) AS Mes;

SELECT nombre, YEAR(fecha_contratacion) AS año FROM empleados;

SELECT TIMESTAMP(NOW(), '12:00:00');

/* Diferencia en días */
SELECT apellido, DATEDIFF(CURDATE(), fecha_contratacion) AS 'Antigüedad'
FROM empleados;

/* Diferencia en años */
SELECT apellido, TIMESTAMPDIFF(YEAR, fecha_contratacion, NOW())
AS 'Antigüedad'
FROM empleados;

SELECT nombre, ADDDATE(fecha_contratacion, 30) AS 'Fin prácticas'
FROM empleados;

SELECT DATE_FORMAT(fecha_contratacion, 'Día: %d, Mes: %m, Año: %Y')
FROM empleados;

SELECT DATE_FORMAT(fecha_contratacion, GET_FORMAT(DATE, 'EUR'))
FROM empleados;
```

8.5.4. Funciones de agregados.

Toman un grupo de datos (una columna) y producen un único dato a partir de los que hay.

Si alguna fila contiene el valor NULL en esa columna, no se tiene en cuenta.

No se pueden mezclar en la selección con nombres de columna.

Modificadores:

- ALL (por defecto).
- DISTINCT.

[Lista completa.](#)

Útiles:

- AVG()
- COUNT()
- MAX()
- MIN()
- SUM()

```
SELECT COUNT(nombre) FROM empleados;
```

```
SELECT COUNT(comision) FROM empleados;
```

```
SELECT COUNT(DISTINCT nombre) FROM empleados;
```

```
SELECT SUM(salario) FROM empleados;
```

Ejercicios 8.5

1. Selecciona la hora actual.
2. Selecciona el día del mes de la fecha actual.
3. Selecciona el día de la semana en el que fue contratado cada empleado.
4. Selecciona el nombre y el apellido de todos los empleados ordenados por día del mes en el que fueron contratados de mayor a menor. Como criterio secundario de ordenación usa orden alfabético de apellido.
5. Selecciona la fecha de contratación con el formato *Día 7 del mes 5 del año 2022*.
6. Selecciona cuántos días quedan para tu cumpleaños. STR_TO_DATE() te puede ser útil.
7. Selecciona el salario medio.
8. Selecciona el salario más alto de los empleados que cobran comisión.

8.6. Agrupando resultados. GROUP BY.

Por una o varias columnas.

Se pueden seleccionar las columnas agrupadas, funciones de agregados sobre esa u otras columnas o todo (*).

Las funciones de agregados se aplican a cada grupo.

```
/* Poco sentido ya que hace lo mismo que  
   SELECT DISTINCT id_trabajo FROM empleados; */  
SELECT id_trabajo FROM empleados GROUP BY id_trabajo;  
  
SELECT id_trabajo, AVG(salario) FROM empleados GROUP BY id_trabajo;  
  
SELECT id_trabajo, AVG(salario) FROM empleados WHERE salario > 2000  
      GROUP BY id_trabajo;  
  
SELECT id_trabajo, COUNT(*), AVG(salario) FROM empleados  
      WHERE salario > 2000 GROUP BY id_trabajo;  
  
SELECT id_trabajo, AVG(salario) FROM empleados  
      GROUP BY id_trabajo, id_departamento;
```

A medida que se complica la consulta hay gente que prefiere este estilo de código. En Workbench *Edit > Format > Beautify query* o *Ctrl + b*:

```
SELECT  
    id_trabajo, ROUND(AVG(salario))  
FROM  
    empleados  
WHERE  
    salario > 2000  
GROUP BY id_trabajo , id_departamento;
```

8.6.1. Filtrar grupos. HAVING.

Determinar qué grupos nos interesan.

```
/* En este caso se podría filtrar con WHERE */
SELECT
    id_trabajo, AVG(salario)
FROM
    empleados
GROUP BY id_trabajo
HAVING id_trabajo = 'ST_CLERK'
    OR id_trabajo = 'SA_REP';
```

```
/* Filtrar por agregado */
SELECT
    id_trabajo, AVG(salario)
FROM
    empleados
GROUP BY id_trabajo
HAVING AVG(salario) > 6000;
```

```
/* Un más completa */
SELECT
    id_trabajo, id_departamento, AVG(salario)
FROM
    empleados
WHERE
    fecha_contratacion > STR_TO_DATE('2000-01-01', '%Y-%m-%d')
GROUP BY id_trabajo , id_departamento
HAVING AVG(salario) > 6000
ORDER BY id_trabajo DESC, id_departamento;
```

Orden en el que se ejecutan las cláusulas:

1. WHERE: filtra las filas según las condiciones que pongamos.
2. GROUP BY: crea una tabla de grupos nueva.
3. HAVING: filtra los grupos.
4. ORDER BY: ordena o clasifica la salida.

8.6.2. Resúmenes. WITH ROLLUP.

Añade una o más filas con datos en conjunto. Depende de por cuántos elementos se agrupe.

```
/* Agrupamos por un elemento, un resumen al final */
SELECT
    id_trabajo, COUNT(*), AVG(salario)
FROM
    empleados
WHERE
    salario > 2000
GROUP BY id_trabajo WITH ROLLUP;
```

```
/*Agrupamos por dos elementos, un resumen por id_trabajo y otro
en total */
SELECT
    id_trabajo, id_departamento, AVG(salario)
FROM
    empleados
GROUP BY id_trabajo , id_departamento WITH ROLLUP
HAVING AVG(salario) > 6000;
```

En el segundo ejemplo de los anteriores se mostrarán dos niveles de resúmenes (Figura 8.3):

1. Un resumen por el grupo creado con el primer argumento (id_trabajo). Vemos cómo se hace un resumen juntando los datos de ambos departamentos, por eso el campo id_departamento para esa fila es NULL.
2. Un resumen total con todos los campos a NULL excepto el agregado.

MK_MAN	30	13000.000000
MK_MAN	NULL	13000.000000
SA_MAN	60	10000.000000
SA_MAN	NULL	10000.000000
SA_REP	NULL	7000.000000
SA_REP	60	9344.444444
SA_REP	NULL	8181.818181
NULL	NULL	6190.000000

Figura 8.3: With Rollup.

Ejercicios 8.6

1. Obtén el coste total en salarios por departamento.
 2. Obtén el primer y el último apellido por orden de lista de cada identificador de trabajo.
 3. Selecciona el salario más alto y el más bajo de cada departamento, para los empleados que tengan el mismo director.
 4. Añade resúmenes a la consulta anterior.
-

8.7. Bibliografía.

- José Luis Comesaña (2011). Curso [MEFP-IFCS03-BD](#). Ministerio de Educación y Formación Profesional.
- [Structured Query Language](#). Wikibook. Wikipedia.
- [Curso de MySQL](#). Con Clase.
- [MySQL 8.0 Reference Manual](#).
- José Juan Sánchez Hernández (2021/2022). [Bases de datos / Gestión de Bases de datos](#)
- Iván López Montalbán, Manuel de Castro Vázquez, John Ospino Rivas (2022). Bases de Datos (2ª Edición) . Garceta.