

11

Consultas sencillas a la base de datos

Introducción

*En este capítulo empezaremos a trabajar con el lenguaje de manipulación de datos. Comenzaremos con la sentencia **SELECT** que nos permitirá realizar una de las operaciones fundamentales a la base de datos: la consulta.*

*La sentencia **SELECT** se utiliza para expresar consultas, con ella podemos realizar consultas sencillas sobre una única tabla o podemos realizar consultas más complejas combinando datos de varias tablas. **SELECT** es la sentencia más compleja y potente de todas las instrucciones **SQL**.*

Contenido

- 11.1. La sentencia **SELECT**
 - 11.2. Condiciones de búsqueda en la cláusula **WHERE**
 - 11.3. Funciones internas
- Ejercicios propuestos

Objetivos

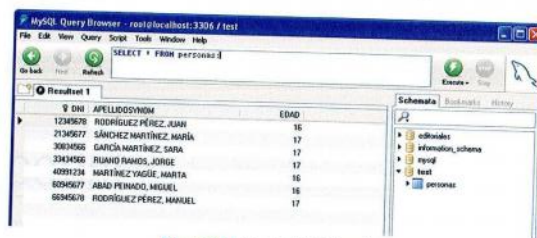
- ▶ Consultar determinadas columnas de una tabla
- ▶ Consultar determinadas filas de una tabla
- ▶ Consultar filas de una tabla utilizando operadores de comparación
- ▶ Consultar filas con diferentes condiciones de búsqueda
- ▶ Presentar el resultado de una consulta de forma ordenada
- ▶ Elegir las cláusulas adecuadas en una sentencia **SELECT**
- ▶ Identificar y aplicar las distintas funciones que se pueden usar con la cláusula **SELECT**
- ▶ Aplicar las diferentes funciones para obtener información de expresiones o de las columnas de las tablas
- ▶ Utilizar manuales en línea para obtener información adicional

11.1 La sentencia SELECT

Usaremos la sentencia SELECT para consultar los datos de una base de datos. En la sentencia SELECT el usuario especifica QUÉ es lo que quiere obtener, no DÓNDE ni CÓMO. La consulta puede obtener:

- Cualquier unidad de datos.
- Todos los datos.
- Cualquier subconjunto de datos.
- Cualquier conjunto de subconjuntos de datos.

El resultado de una consulta SQL es siempre una tabla de datos semejante a las tablas de la base de datos. Si se escribe una sentencia SELECT utilizando SQL interactivo, el SGBD visualiza los resultados de forma tabular. Véase la Figura 11.1.



DNI	APELLIDOSYNOM	EDAD
12345678	RODRIGUEZ PÉREZ, JUAN	16
21345677	SÁNCHEZ MARTÍNEZ, MARÍA	17
30834566	GARCÍA MARTÍNEZ, SARA	17
33434566	RUANO PANDOS, JORGE	17
40991234	MARTÍNEZ YAGUE, MARTA	16
60945677	ABAD PENADO, MIGUEL	16
60945678	RODRIGUEZ PEREZ, MANUEL	17

Figura 11.1. Uso de SQL interactivo

El formato básico de la sentencia SELECT es el siguiente:

```
SELECT * FROM nombredetabla;
```

Donde observamos los siguientes elementos:

- La primera palabra es SELECT; indica que se quiere realizar una consulta a la base de datos.
- El * a la derecha de SELECT indica que se quieren recuperar todas las columnas de la tabla.
- La siguiente palabra es FROM, que indica la tabla o tablas que se van a consultar.
- A la derecha de la cláusula FROM se escribe el nombre de la tabla que se quiere consultar.

11.1.1. Cláusula FROM

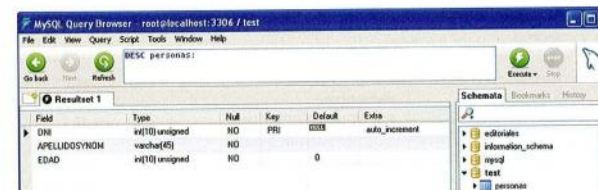
Especifica la tabla o lista de tablas de donde se recuperarán los datos. Si se escribe más de una tabla, se separarán por comas.

Selección de columnas

Con el * a la derecha de SELECT se consultan todas las columnas de la tabla. Para consultar determinadas columnas hemos de escribir sus nombres separados por coma. Es decir, escribiremos algo parecido a esto:

```
SELECT columna1, columna2, ..., columna FROM nombredetabla;
```

Para obtener información sobre la descripción de las columnas de una tabla usamos la orden DESCRIBE o DESC. Por ejemplo, para saber la descripción de las columnas de la tabla PERSONAS escribimos la siguiente orden: DESC personas; (Figura 11.2).



Field	Type	Null	Key	Default	Extra
DNI	varchar(9)	NO	PRI	NULL	auto_increment
APELLIDOSYNOM	varchar(45)	NO	PRI	NULL	auto_increment
EDAD	int(10) unsigned	NO		0	auto_increment

Figura 11.2. Descripción de una tabla

Donde el significado de las columnas es el siguiente:

Field: es el nombre de la columna.

Type: tipo de dato de la columna.

Null: indica si se pueden almacenar valores nulos. El valor NO indica que no se pueden almacenar. YES indica que sí se pueden almacenar valores nulos.

Key: indica si la columna está indexada. PRI señala que el campo es clave primaria.

Default: indica el valor por defecto asignado al campo.

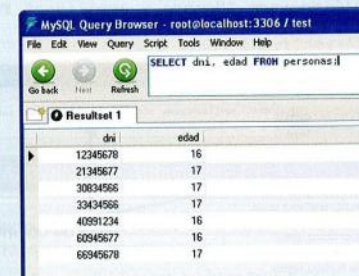
Extra: contiene cualquier información adicional acerca del campo dado. Esta columna en el campo DNI indica que fue creado con la palabra clave AUTO_INCREMENT.

Ejercicio Resuelto 11.1

Realizamos consultas sobre la tabla PERSONAS de la base de datos TEST.

1. Obtener el DNI y la EDAD de todas las personas. Escribimos la siguiente orden en el Área de Consulta SQL (véase la Figura 11.3):

```
SELECT dni, edad FROM personas;
```



dni	edad
12345678	16
21345677	17
30834566	17
33434566	17
40991234	16
60945677	16
60945678	17

Figura 11.3. Consulta 1. Ejercicio 11.1

- Obtener sólo los DNI (véase la Figura 11.4):

```
SELECT dni FROM personas;
```

- Obtener los apellidos y nombres y la edad (véase la Figura 11.5):

```
SELECT apellidosnom, edad FROM personas;
```

dni
12345678
21345677
30834566
33434566
40991234
60945677
66945678

Figura 11.4. Consulta 2. Ejercicio 11.1

apellidosnom	edad
RODRÍGUEZ PÉREZ, JUAN	16
SÁNCHEZ MARTÍNEZ, MARÍA	17
GARCÍA MARTÍNEZ, SARA	17
RUANO RAMOS, JORGE	17
MARTÍNEZ YAGÜE, MARTA	16
ABAD PEINADO, MIGUEL	16
RODRÍGUEZ PÉREZ, MANUEL	17

Figura 11.5. Consulta 3. Ejercicio 11.1

• ALIAS DE TABLAS:

Se puede asociar un nuevo nombre a una tabla usando *alias*. Suelen utilizarse cuando consultamos varias tablas y hay nombres de columna que coinciden. Ejemplo:

```
SELECT P.apellidosnom, P.edad FROM personas P;
```

A la tabla PERSONAS le asignamos el alias **P**. Las columnas de la tabla van acompañadas del alias **P** seguido de un punto.

• ALIAS DE COLUMNAS:

Cuando se consulta la base de datos, los nombres de las columnas se usan como cabeceras de presentación. Si éste resulta demasiado largo, corto o críptico, o es el resultado de un cálculo, puede cambiarse con la misma sentencia SQL de consulta creando un ALIAS. El ALIAS se pone entre comillas simples o dobles a la derecha de la columna deseada. Véase la Figura 11.6.

```
SELECT apellidosnom "Nombre y apellidos" FROM personas;
```

Nombre y apellidos
RODRÍGUEZ PÉREZ, JUAN
SÁNCHEZ MARTÍNEZ, MARÍA
GARCÍA MARTÍNEZ, SARA
RUANO RAMOS, JORGE
MARTÍNEZ YAGÜE, MARTA
ABAD PEINADO, MIGUEL
RODRÍGUEZ PÉREZ, MANUEL

Figura 11.6. Alias de columnas

• COLUMNAS CALCULADAS:

Una consulta SQL puede incluir columnas calculadas cuyos valores se obtienen a partir de los valores de datos almacenados en las columnas de las tablas. Esta columna calculada formará parte de la lista de selección. El siguiente ejemplo obtiene los apellidos y nombres y el año de nacimiento (se calcula restando al año actual la edad) de las personas de la tabla PERSONAS (véase la Figura 11.7):

```
SELECT apellidosnom, 2006-edad "Año Nacimiento" FROM personas;
```

apellidosnom	Año Nacimiento
RODRÍGUEZ PÉREZ, JUAN	1990
SÁNCHEZ MARTÍNEZ, MARÍA	1989
GARCÍA MARTÍNEZ, SARA	1989
RUANO RAMOS, JORGE	1989
MARTÍNEZ YAGÜE, MARTA	1990
ABAD PEINADO, MIGUEL	1990
RODRÍGUEZ PÉREZ, MANUEL	1989

Figura 11.7. Columnas calculadas

11.1.2. Cláusula WHERE

Obtiene aquellas filas que cumplan la condición expresada. La complejidad de la condición es prácticamente ilimitada. El formato de la sentencia SELECT con la cláusula WHERE es el siguiente:

```
SELECT columna1, columna2, ..., columnan | *
FROM nombredetabla
WHERE condición;
```

El formato de la condición es el siguiente: **expresión operador expresión**

Pueden construirse condiciones múltiples usando los operadores lógicos booleanos estándares: AND, OR, NOT. También se pueden emplear paréntesis para forzar el orden de evaluación.

Ejemplos de condiciones en la cláusula WHERE son los siguientes:

```
WHERE total = 5
WHERE total = 6 OR nombre_alumno LIKE '%Martínez%'
WHERE (total > 5) AND (nota2 > 5 OR curso = 1)
```

Actividad de Aplicación 11.1

Crea un nuevo esquema de base de datos de nombre UNIDAD11 para cargar las tablas con las que trabajaremos en esta unidad.

A continuación, carga el script TABLAS_UNIDAD11.SQL que se encuentran en la página web del libro en el Editor de Script y ejecútalo.





Ejercicio Resuelto 11.2

Realizamos consultas sobre la tabla NOTAS_ALUMNOS que está en el esquema: UNIDAD11.

Hacemos una descripción de la tabla NOTAS_ALUMNOS para inspeccionar sus columnas (véase la Figura 11.8). Esta tabla contiene para cada nombre de alumno tres notas de tres controles que hizo en una asignatura: NOTA1 es la nota del primer control, NOTA2 es la nota del segundo control y NOTA3 la del tercer control. El contenido se muestra en la Figura 11.9. Las notas con valor NULL significa que el alumno no hizo el control.

Field	Type	Null	Key	Default
nombre_alumno	varchar(25)	NO		
nota1	tinyint(2)	YES		0000
nota2	tinyint(2)	YES		0000
nota3	tinyint(2)	YES		0000
asignatura	varchar(25)	YES		
curso	varchar(4)	YES		

Figura 11.8. Tabla NOTAS_ALUMNOS

nombre_alumno	nota1	nota2	nota3	asignatura	curso
Alcalde García, M. Luisa	5	5	5	MATEMÁTICAS	4ESO
Benito Martín, Luis	7	6	8	MATEMÁTICAS	3ESO
Casas Martínez, Manuel	7	5	5	LENGUA	4ESO
Correidor Sánchez, Ana	6	9	8	LENGUA	3ESO
Díaz Sánchez, María	NULL	NULL	7	LENGUA	4ESO

Figura 11.9. Contenido de la tabla NOTAS_ALUMNOS

1. Consultamos aquellos nombres de alumnos cuya nota en el segundo control sea un 9:

```
SELECT nombre_alumno FROM notas_alumnos WHERE nota2 = 9;
```

nombre_alumno
Correidor Sánchez, Ana

Figura 11.10. Consulta 1. Ejercicio 11.2

2. Obtenemos por cada nombre de alumno la nota media de los tres controles (Figura 11.11):

```
SELECT Nombre_alumno, (nota1+nota2+nota3)/3 "Nota media"
FROM notas_alumnos;
```

Nombre_alumno	Nota media
Alcalde García, M. Luisa	5.0000
Benito Martín, Luis	7.0000
Casas Martínez, Manuel	5.6667
Correidor Sánchez, Ana	7.6667
Díaz Sánchez, María	0000

Figura 11.11. Nota media de los alumnos de la tabla NOTAS_ALUMNOS

3. Obtenemos los nombres de alumnos cuya nota media supere 6:

```
SELECT Nombre_alumno FROM notas_alumnos
WHERE (nota1+nota2+nota3)/3 > 6;
```

Nombre_alumno
Benito Martín, Luis
Correidor Sánchez, Ana

Figura 11.12. Consulta 3. Ejercicio 11.2

4. Obtenemos los nombres de alumnos cuya nota en el primer control sea 7 en la asignatura de 'LENGUA':

```
SELECT Nombre_alumno FROM notas_alumnos
WHERE nota1 = 7 AND asignatura = 'LENGUA';
```

Nombre_alumno
Casas Martínez, Manuel

Figura 11.13. Consulta 4. Ejercicio 11.2

5. Obtenemos los nombres de alumnos cuya nota en el primer control sea un 7 en la asignatura de 'LENGUA' o 'MATEMÁTICAS':

```
SELECT Nombre_alumno FROM notas_alumnos
WHERE nota1 = 7
AND (asignatura = 'LENGUA' OR asignatura = 'MATEMÁTICAS');
```

Nombre_alumno
Benito Martín, Luis
Casas Martínez, Manuel

Figura 11.14. Consulta 5. Ejercicio 11.2

6. Obtenemos los nombres de alumnos y la asignatura de todos los alumnos del curso de '3ESO' que tengan un 8 en alguna de las notas.

```
SELECT Nombre_alumno, Asignatura FROM notas_alumnos
WHERE curso = '3ESO' AND (nota1 = 8 OR nota2 = 8 OR nota3 = 8);
```

Nombre_alumno	Asignatura
Benito Martín, Luis	MATEMÁTICAS
Correidor Sánchez, Ana	LENGUA

Figura 11.15. Consulta 6. Ejercicio 11.2

11.1.3. Cláusula ORDER BY

Las filas de los resultados de una consulta no se ordenan por ningún criterio. Podemos decir a SQL que ordene los resultados incluyendo la cláusula ORDER BY al final de la sentencia SELECT. El formato es el siguiente:

```
SELECT columna1, columna2, ..., columna n | *
FROM nombredetabla
[WHERE condición]
ORDER BY expre_columna [DESC|ASC]
[, expre_columna [DESC|ASC] ...]
```


ASC especifica ordenación ascendente. Es la opción por defecto. No es necesario escribirla cuando se quiere ordenar ascendentemente.

DESC indica ordenación descendente.

Los corchetes en la cláusula WHERE indican que su presencia no es obligatoria.



Ejercicio Resuelto 11.3

Realizamos consultas sobre la tabla NOTAS_ALUMNOS ordenando los resultados.

1. Obtenemos todas las columnas de la tabla ordenando los resultados descendente por la columna NOMBRE_ALUMNO:

```
SELECT * FROM notas_alumnos ORDER BY Nombre_alumno DESC;
```

nombre_alumno	nota1	nota2	nota3	asignatura	curso
Díaz Sánchez, María	6	9	7	LENGUA	4ESO
Correidor Sánchez, Ana	6	9	8	LENGUA	3ESO
Casas Martínez, Manuel	7	5	5	LENGUA	4ESO
Benito Martín, Luis	7	6	8	MATEMÁTICAS	3ESO
Alcalde García, M. Luisa	5	5	5	MATEMÁTICAS	4ESO

Figura 11.16. Consulta 1. Ejercicio Resuelto 11.3

2. La cláusula ORDER BY puede contener expresiones con valores de columnas. Por ejemplo, ordenamos la consulta por la nota media del alumno ascendentemente:

```
SELECT Nombre_alumno, (nota1+nota2+nota3)/3 "Nota media"
FROM notas_alumnos ORDER BY (nota1+nota2+nota3)/3;
```

Nombre_alumno	Nota media
Díaz Sánchez, María	7.3333
Alcalde García, M. Luisa	5.0000
Casas Martínez, Manuel	5.6667
Benito Martín, Luis	7.0000
Correidor Sánchez, Ana	7.6667

Figura 11.17. Consulta 2. Ejercicio Resuelto 11.3

También se puede expresar la ordenación mediante un número, que indica la posición de la columna a la derecha de SELECT por la que se quiere ordenar el resultado. La consulta anterior se puede escribir de la siguiente manera:

```
SELECT Nombre_alumno, (nota1+nota2+nota3)/3 "Nota media"
FROM notas_alumnos ORDER BY 2;
```

Se escribe un 2 porque es por la segunda columna (o expresión) por la que se quiere ordenar.

3. Pueden anidarse también los criterios; el de más a la izquierda será el principal. Por ejemplo, ordenamos la tabla por la asignatura ascendentemente y dentro de asignatura por nombre de alumno descendente:

```
SELECT Asignatura, Nombre_alumno FROM notas_alumnos
ORDER BY Asignatura, Nombre_alumno DESC;
```

O bien:

```
SELECT Asignatura, Nombre_alumno FROM notas_alumnos
ORDER BY 1, 2 DESC;
```

Asignatura	Nombre_alumno
LENGUA	Díaz Sánchez, María
LENGUA	Correidor Sánchez, Ana
LENGUA	Casas Martínez, Manuel
MATEMÁTICAS	Benito Martín, Luis
MATEMÁTICAS	Alcalde García, M. Luisa

Figura 11.18. Consulta 3. Ejercicio Resuelto 11.3

11.1.4. Cláusulas DISTINCT y ALL

La cláusula ALL recupera todas las filas, aunque algunas estén repetidas; es la opción por defecto. Por eso no se suele escribir a la derecha de SELECT, véase la Figura 11.19.

La cláusula DISTINCT sólo recupera las filas que son distintas.

Por ejemplo: consultamos las asignaturas de la tabla NOTAS_ALUMNOS, véase la Figura 11.19. Aparecen todas las filas de la tabla NOTAS_ALUMNOS donde la columna ASIGNATURA no sea nula; por tanto aparecen nombres de asignaturas repetidas.

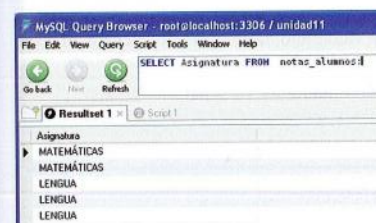


Figura 11.19. Asignaturas de la tabla NOTAS_ALUMNOS



Figura 11.20. Cláusula DISTINCT

Con DISTINCT se eliminan las filas repetidas, véase la Figura 11.20. Estas cláusulas se colocan a la derecha de SELECT.

11.2 Condiciones de búsqueda en la cláusula WHERE

SQL posee cinco condiciones de búsqueda básicas que se pueden incluir en la cláusula WHERE. Algunas de estas condiciones ya las hemos utilizado. Son las siguientes:

- Test de comparación.
- Test de correspondencia con patrón.
- Test de valor nulo.
- Test de pertenencia a un conjunto.
- Test de rango.

11.2.1. Test de comparación

Es la condición de búsqueda más utilizada. Compara el valor de una expresión con otra. El diagrama sintáctico del test de comparación es el siguiente:

expresión operador expresión

Las expresiones pueden ser: una constante, una expresión aritmética, un valor nulo o un nombre de columna. Los operadores de comparación o lógicos se resumen en esta tabla:

Operador Lógico	Función
=	Igual a
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
!= <>	Distinto de

Cuando SQL compara el valor de dos expresiones se pueden producir tres resultados:

- Si la comparación es cierta, el test devuelve TRUE.
- Si la comparación es falsa, el test devuelve FALSE.
- Si alguna de las expresiones produce un valor nulo, la comparación genera un resultado nulo (NULL).

11.2.2. Test de correspondencia con patrón. Operador LIKE

Este test determina si una expresión de tipo carácter se corresponde con un patrón especificado. El patrón es una cadena que puede incluir uno o más caracteres comodines. Los caracteres comodines son los siguientes:

- '%' representa cualquier cadena de 0 o más caracteres; se admite cualquier cosa en su lugar.
- '_' marcador de posición; representa un carácter cualquiera.

En la cláusula **WHERE** este operador se utilizará de la siguiente manera:

WHERE columna [NOT] **LIKE** 'caracteres_especiales'

Este operador nos sirve si queremos hacer consultas de este tipo: "obtener los datos de los alumnos cuyo apellido empiece por una letra 'P'", o bien: "obtener los nombres de alumnos que incluyan el apellido 'Martínez' dentro de la columna de apellidos". Por ejemplo, para obtener los datos de los alumnos cuya columna NOMBRE_ALUMNO empiece por la letra 'C' escribiríamos lo siguiente:

```
SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE 'C%';
```

Las mayúsculas y minúsculas son significativas: 'r' no es lo mismo que 'R'.

Las constantes alfanuméricas deben encerrarse siempre entre comillas simples.



Ejercicio Resuelto 11.4

Realizamos consultas sobre la tabla NOTAS_ALUMNOS:

1. Obtener aquellos nombres de alumnos que tengan un 7 en NOTA1 y la media sea mayor que 6. Ordenamos el resultado ascendentemente por nombre de alumno:

```
SELECT Nombre_alumno FROM notas_alumnos
WHERE nota1 = 7 AND (nota1 + nota2 + nota3)/3 > 6
ORDER BY 1;
```

Nombre_alumno
► Benito Martín, Luis

Figura 11.21. Consulta 1. Ejercicio Resuelto 11.4

2. Obtener todos los datos de alumnos que no tengan un 8 en la NOTA3 en 'MATEMÁTICAS':

```
SELECT * FROM notas_alumnos
WHERE nota3 <> 8 AND asignatura = 'MATEMÁTICAS';
```

nombre_alumno	nota1	nota2	nota3	asignatura	curso
► Alcalde García, M. Luisa	5	5	5	MATEMÁTICAS	4ESO

Figura 11.22. Consulta 2. Ejercicio Resuelto 11.4



Ejercicio Resuelto 11.5

Realizamos consultas sobre la tabla NOTAS_ALUMNOS:

1. Obtener los datos de los alumnos cuya columna NOMBRE_ALUMNO empiece por una letra 'C':

```
SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE 'C%';
```

nombre_alumno	nota1	nota2	nota3	asignatura	curso
► Casas Martínez, Manuel	7	5	5	LENGUA	4ESO
Corregidor Sánchez, Ana	6	9	8	LENGUA	3ESO

Figura 11.23. Consulta 1. Ejercicio Resuelto 11.5

2. Obtener los datos de los alumnos que tengan el apellido 'Sánchez' dentro de la columna NOMBRE_ALUMNO:

```
SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE '%Sánchez%';
```

nombre_alumno	nota1	nota2	nota3	asignatura	curso
► Corregidor Sánchez, Ana	6	9	8	LENGUA	3ESO
Díaz Sánchez, María	NULL	NULL	7	LENGUA	4ESO

Figura 11.24. Consulta 2. Ejercicio Resuelto 11.5

3. Qué obtienen las siguientes consultas:

- **SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE 'Sánchez';**
Aquellas filas cuya columna NOMBRE_ALUMNO es igual a 'Sánchez'.
- **SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE 'M%';**
Aquellas filas cuya columna NOMBRE_ALUMNO empieza por 'M'.

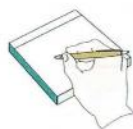
- **SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE '%M%';**
Aquellas filas cuya columna NOMBRE_ALUMNO contenga por lo menos una 'M'.
- **SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE '___M';**
Aquellas filas cuya columna NOMBRE_ALUMNO contenga 3 caracteres y termine en 'M'.
- **SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE 'n__';**
Aquellas filas cuya columna NOMBRE_ALUMNO contenga 2 caracteres y que el primero empiece por 'n'.
- **SELECT * FROM notas_alumnos WHERE nombre_alumno LIKE 'r%';**
Aquellas filas cuya columna NOMBRE_ALUMNO contenga cualquier cadena cuyo segundo carácter sea una 'r'.

11.2.3. Test de valor nulo (IS NULL)

Se dice que una columna de una fila es **NULL** si está completamente vacía. Para comprobar si el valor de una columna es nulo utilizamos la expresión: columna **IS NULL**. Si queremos comprobar si el valor de una columna no es nulo, usamos la expresión: columna **IS NOT NULL**. Cuando comparamos con valores nulos o no nulos no podemos utilizar los operadores de igualdad, mayor o menor.



Las operaciones aritméticas con valores nulos devuelven un valor nulo.



Ejercicio Resuelto 11.6

Realizamos consultas sobre la tabla NOTAS_ALUMNOS:

1. Obtener los datos de los alumnos cuya columna NOTA1 es nula:

```
SELECT * FROM notas_alumnos WHERE nota1 IS NULL;
```

nombre_alumno	nota1	nota2	nota3	asignatura	curso
Díaz Sánchez, María	NULL	NULL	7	LENGUA	4ESO

Figura 11.25. Consulta 1. Ejercicio Resuelto 11.6

2. Obtener los datos de los alumnos cuya columna NOTA1 no sea nula:

```
SELECT * FROM notas_alumnos WHERE nota1 IS NOT NULL;
```

nombre_alumno	nota1	nota2	nota3	asignatura	curso
Alcalde García, M. Luisa	5	5	5	MATEMÁTICAS	4ESO
Benito Martín, Luis	7	6	8	MATEMÁTICAS	3ESO
Casas Martínez, Manuel	7	5	5	LENGUA	4ESO
Corregidor Sánchez, Ana	6	9	8	LENGUA	3ESO

Figura 11.26. Consulta 2. Ejercicio Resuelto 11.6

11.2.4. Test de rango. Operador BETWEEN

El operador **BETWEEN** comprueba si un valor está comprendido o no (NOT) dentro de un rango de valores, desde un valor inicial a un valor final, ambos incluidos. Formato:

```
<expresión> [NOT] BETWEEN valor_inicial AND valor_final
```

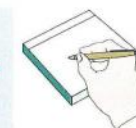
Por ejemplo, obtenemos todos los datos de la tabla PERSONAS cuya edad esté comprendida entre 14 y 17 años:

```
SELECT * FROM personas WHERE edad BETWEEN 14 AND 17;
```

Ejercicio Resuelto 11.7

Utilizamos la tabla LIBROS del esquema EDITORIALES creado en el Capítulo 9.

La descripción de la tabla se muestra en la Figura 11.27.



Field	Type	Null	Key	Default	Extra
codigo	int(4)	NO	PRI		
titulo	varchar(40)	NO			
autor	varchar(50)	NO			
editorial	int(2)	YES			
unidades_vendidas	int(5)	YES			
fecha_publicacion	date	YES			

Figura 11.27. Descripción de la tabla LIBROS

1. Obtener los datos de los libros cuyas UNIDADES_VENDIDAS estén comprendidas entre 1000 y 1500:

```
SELECT * FROM libros WHERE unidades_vendidas BETWEEN 1000 AND 1500;
```

codigo	titulo	autor	editorial	unidades_vendidas	fecha_publicacion
503	REDES IP	J.G. TOMÁS, J.L. RAYA y otros	20	1008	2002-03-15
530	DISEÑO Y DESARROLLO MULTIMEDIA	M.A. CASTRO, A. COLMENAR y otros	20	1356	2002-11-01
1647	VIRUS INFORMÁTICOS	RICHARD B. LEVIN	30	1500	1991-02-10
2232	SERVICIOS DE INFORMACIÓN ELECTRÓNICA	M. GONZÁLEZ, R. CHAMORRO y otros	10	1200	1995-04-11
7254	WINDOWS 2000 PROF.	A. GONZÁLEZ MANGAS	10	1500	2002-06-05
8609	INSTALACIONES ELÉCTRICAS	J.L. SANZ SERRANO	10	1004	2005-02-01

Figura 11.28. Consulta 1. Ejercicio Resuelto 11.7

2. Obtener los datos de los libros cuya columna FECHA_PUBLICACION comprenda los años 2000 y 2001 (es decir, desde el 1 de enero de 2000 al 31 de diciembre de 2001):

```
SELECT * FROM libros  
WHERE fecha_publicacion BETWEEN '2000-01-01' AND '2001-12-31';
```


codigo	título	autor	editorial	unidades_vend...	fecha_publicaci...
281	GUÍA COMPLETA PARA PC	RON GILSTER	30	1800	2001-11-01
455	XML A TRAVÉS DE EJEMPLOS	A.GUTIÉRREZ, R.MARTÍNEZ	20	1876	2001-02-01
1416	GUÍA DE SQL	J.R.GROFF, P.N.WEINBERG	30	2301	2000-06-01

Figura 11.29. Consulta 2. Ejercicio Resuelto 11.7

3. Obtener los datos de los libros cuya UNIDADES_VENDIDAS sean menores que 600 y mayores que 2000.

```
SELECT * FROM libros
WHERE unidades_vendidas NOT BETWEEN 600 AND 2000;
```

codigo	título	autor	editorial	unidades_vend...	fecha_publicaci...
506	SQL Y JAVA	J.MELTON, A.EISENBERG	20	2300	2002-01-24
1416	GUÍA DE SQL	J.R.GROFF, P.N.WEINBERG	30	2301	2000-06-01
3214	INGENIERÍA DEL SOFTWARE	ROGER S.PRESSMAN	30	2300	2002-04-04
4897	REDES DE ÁREA LOCAL	J.M.HUIDIBORO y otros	10	500	2006-02-01
9732	SQL PARA USUARIOS Y PROGRAMADORES	E.RIVERO, L.MARTÍNEZ, y otros	10	2500	2002-03-01

Figura 11.30. Consulta 3. Ejercicio Resuelto 11.7

11.2.5. Test de pertenencia a un conjunto. Operador IN

El operador **IN** nos permite comprobar si una expresión pertenece o no (NOT) a un conjunto de valores. Nos va a permitir realizar comparaciones múltiples. Formato:

<expresión> [NOT] **IN** (lista de valores separados por comas)

Por ejemplo, para obtener los datos de los libros cuyas editoriales sean 10 o 30 escribimos la siguiente consulta:

```
SELECT * FROM libros WHERE editorial IN (10,30);
```



Ejercicio Resuelto 11.8

Realizamos consultas sobre la tabla LIBROS.

1. Obtener los datos de los libros cuyas UNIDADES_VENDIDAS sean 1200, 1500 o 2000:

```
SELECT * FROM libros WHERE unidades_vendidas IN (1200, 1500, 2000);
```

codigo	título	autor	editorial	unidades_vend...	fecha_publicaci...
1647	VIRUS INFORMÁTICOS	RICHARD B.LEVIN	30	1500	1991-02-10
2232	SERVICIOS DE INFORMACIÓN ELECTRÓNICA	M.GONZÁLEZ, R.CHAMORRO y otros	10	1200	1995-04-11
7254	WINDOWS 2000 PROF.	A.GONZÁLEZ MANGAS	10	1500	2002-06-05

Figura 11.31. Consulta 1. Ejercicio Resuelto 11.8

2. Obtener los datos de los libros cuyas UNIDADES_VENDIDAS no sean 1200, 1500 o 2000

```
SELECT * FROM libros
WHERE unidades_vendidas NOT IN (1200, 1500, 2000);
```

3. Obtener los datos de los libros cuya FECHA_PUBLICACION sea alguna de las siguientes: '2005-02-01', '2002-11-01' o '2006-02-01' en la EDITORIAL 10:

```
SELECT * FROM libros
WHERE fecha_publicacion IN ('2005-02-01', '2002-11-01', '2006-02-01')
AND editorial = 10;
```

codigo	título	autor	editorial	unidades_vend...	fecha_publicaci...
4897	REDES DE ÁREA LOCAL	J.M.HUIDIBORO y otros	10	500	2006-02-01
5044	MANTO DE PORTALES DE INFORMACIÓN	E.QUERO, A.GARCÍA y otros	10	600	2006-02-01
9609	INSTALACIONES ELÉCTRICAS	J.L.SANZ SERRANO	10	1004	2005-02-01

Figura 11.32. Consulta 3. Ejercicio Resuelto 11.8

Actividad de Aplicación 11.2

Intenta escribir las consultas anteriores sin utilizar el operador **IN**. ¿Es posible hacerlo?



11.3 Funciones internas

Las funciones se usan dentro de expresiones y actúan con los valores de las columnas, variables o constantes. Generalmente producen dos tipos diferentes de resultados: unas producen un resultado que es una modificación de la información original (por ejemplo, poner en minúscula una cadena que está en mayúscula); el resultado de otras indica alguna cosa sobre la información (por ejemplo, el número de caracteres que tiene una cadena). Se utilizan en cláusulas **SELECT**, cláusulas **WHERE** y cláusulas **ORDER BY**.

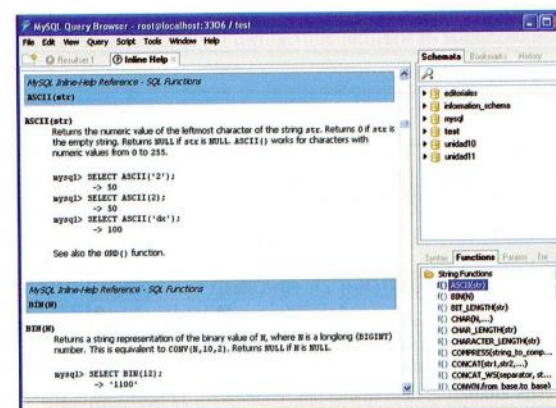


Figura 11.33. Pantalla de ayuda

Existen varios tipos de funciones: aritméticas, de cadenas de caracteres, de manejo de fechas y horas, de conversión, de comparación y otras funciones. En este apartado sólo veremos algunas de ellas.

No olvidemos que desde el Navegador de Información podemos acceder al navegador de funciones, pestaña **Functions**, véase la Figura 11.33, para consultar las diferentes categorías de funciones que podemos utilizar en las sentencias SQL. Al hacer clic en la categoría y doble clic en la función, se visualizará su cometido en el Área de Resultado en la pestaña *Inline Help*.

11.3.1. Funciones aritméticas

Las funciones aritméticas trabajan con datos de tipo numérico. Este tipo incluye los dígitos de 0 a 9, el punto decimal (.) y el signo menos, si es necesario. Los literales numéricos no se encierran entre comillas. Por ejemplo, son datos numéricos: -123.32, 1400. Estas funciones se muestran en la Tabla 11.1.

Función	Propósito
ABS (n)	Devuelve el valor absoluto de "n". El valor absoluto es siempre un número positivo.
MOD (m, n)	Devuelve el resto resultante de dividir "m" entre "n".
POWER (m, exponente)	Calcula la potencia de un número. Devuelve el valor de "m" elevado a un "exponente".
ROUND (número [,m])	Devuelve el valor de "número" redondeado a "m" decimales. Si se omite "m", devuelve "número" con 0 decimales y redondeado.
SQRT (n)	Devuelve la raíz cuadrada de "n". El valor de "n" no puede ser negativo.
TRUNCATE (número, m)	Trunca los números para que tengan un cierto número de dígitos de precisión. Devuelve "número" truncado a "m" decimales. Si "m" es 0, devuelve "número" con 0 decimales.

Tabla 11.1. Funciones aritméticas



Ejercicio Resuelto 11.9

Probamos las funciones expuestas anteriormente.

- La siguiente consulta calcula el valor absoluto de -20, el resto de dividir 25 entre 4, la potencia de 2 elevado a 4 y la raíz cuadrada de 81:

```
SELECT ABS (-20), MOD (25, 4), POWER (2,4), SQRT (81);
```

ABS(-20)	MOD (25, 4)	POWER(2,4)	SQRT(81)
20	1	16	9

Figura 11.34. Consulta 1. Ejercicio Resuelto 11.9

- Obtenemos por cada alumno la nota media sin redondear y la nota media redondeada a dos posiciones ordenando ascendente por la nota media sin redondear:

```
SELECT Nombre_alumno, (nota1+nota2+nota3)/3 "Media No Redondeada",
ROUND ((nota1+nota2+nota3)/3, 2) "Media Redondeada"
FROM notas_alumnos ORDER BY 2;
```

Nombre_alumno	Media No Redondeada	Media Redondeada
Díaz Sánchez, María	5.6667	5.67
Alcalde García, M. Luisa	5.0000	5.00
Casas Martínez, Manuel	5.6667	5.67
Benito Martín, Luis	7.0000	7.00
Corregidor Sánchez, Ana	7.6667	7.67

Figura 11.35. Consulta 2. Ejercicio Resuelto 11.9

- Obtenemos por cada alumno la nota media sin truncar y la nota media truncada a 0 decimales ordenando descendente por la nota media sin truncar:

```
SELECT Nombre_alumno, (nota1+nota2+nota3)/3 "Media No Truncada",
TRUNCATE ((nota1+nota2+nota3)/3, 0) "Media Truncada"
FROM notas_alumnos ORDER BY 2 DESC;
```

Nombre_alumno	Media No Truncada	Media Truncada
Corregidor Sánchez, Ana	7.6667	7
Benito Martín, Luis	7.0000	7
Casas Martínez, Manuel	5.6667	5
Alcalde García, M. Luisa	5.0000	5
Díaz Sánchez, María	5.6667	5

Figura 11.36. Consulta 3. Ejercicio Resuelto 11.9

Actividad de Aplicación 11.3

Consulta desde el navegador de funciones algunas funciones aritméticas que no se hayan visto y pruébalas.



11.3.2. Funciones de cadenas de caracteres

Estas funciones trabajan con datos de tipo carácter. Este tipo de datos incluyen cualquier carácter alfanumérico: letras, números y caracteres especiales. Se deben encerrar entre comillas simples. Ejemplo de una cadena de caracteres es 'El Quijote'.

Mediante estas funciones podremos calcular el número de caracteres de una cadena, convertir una cadena a mayúsculas o a minúsculas, suprimir o añadir caracteres a la izquierda o a la derecha, etc. Se muestran en la Tabla 11.2.

Función	Propósito
CONCAT(cad1, cad2)	Devuelve "cad1" concatenada con "cad2".
LOWER(cad)	Devuelve la cadena "cad" con todas sus letras convertidas a minúsculas.
UPPER(cad)	Devuelve la cadena "cad" con todas sus letras convertidas a mayúsculas.
LPAD(cad1, n, cad2) RPAD(cad1, n, cad2)	LPAD() añade caracteres a la izquierda de "cad1" hasta que alcance una cierta longitud "n"; "cad2" es la cadena con la que se rellena por la izquierda. "cad1" puede ser una columna de una tabla o cualquier literal. RPAD funciona como LPAD() pero añadiendo caracteres a la derecha.
LTRIM(cad) RTRIM(cad)	LTRIM() suprime los blancos a la izquierda de la cadena. RTRIM() suprime los blancos a la derecha de la cadena.
SUBSTR(cad, m [,n])	Devuelve la subcadena de "cad", que abarca desde la posición indicada en "m" hasta tantos caracteres como indique el número "n". Si se omite "n", devuelve la cadena desde la posición especificada por "m".
REPLACE(cad, cadena_búsqueda, cadena_sustitución)	Devuelve la cadena "cad" con todas las ocurrencias de "cadena_búsqueda" sustituidas por "cadena_sustitución".
LENGTH(cad)	Devuelve el número de caracteres de "cad".
INSTR(cad1, cad2)	Devuelve la posición de la primera ocurrencia de "cad2" en "cad1", empezando la búsqueda en la posición 1.

Tabla 11.2. Funciones de cadenas de caracteres

Ejercicio Resuelto 11.10

Probamos las funciones expuestas anteriormente con la tabla NOTAS_ALUMNOS.

- La siguiente consulta visualiza el NOMBRE_ALUMNO en mayúscula y la ASIGNATURA en minúscula de la tabla NOTAS_ALUMNOS:

```
SELECT UPPER(Nombre_Alumno), LOWER(Asignatura) FROM notas_alumnos;
```

UPPER(Nombre_Alumno)	LOWER(Asignatura)
ALCALDE GARCÍA, M. LUISA	matemáticas
BENITO MARTÍN, LUIS	matemáticas
CASAS MARTÍNEZ, MANUEL	lengua
CORREGIDOR SÁNCHEZ, ANA	lengua
DÍAZ SÁNCHEZ, MARÍA	lengua

Figura 11.37. Consulta 1. Ejercicio Resuelto 11.10

- La siguiente consulta añade * a la izquierda del NOMBRE_ALUMNO hasta formar una longitud de 30 caracteres y a la derecha de la ASIGNATURA hasta formar una longitud de 15:

```
SELECT LPAD(Nombre_Alumno,30,'*'), RPAD(Asignatura,15,'*')
FROM notas_alumnos;
```

LPAD(Nombre_Alumno,30,'*')	RPAD(Asignatura,15,'*')
*Alcalde García, M. Luisa	MATEMÁTICAS*****
*Benito Martín, Luis	MATEMÁTICAS*****
*Casas Martínez, Manuel	LENGUA*****
*Corregidor Sánchez, Ana	LENGUA*****
*Díaz Sánchez, María	LENGUA*****

Figura 11.38. Consulta 2. Ejercicio Resuelto 11.10

- La siguiente consulta concatena las columnas de NOMBRE_ALUMNO y ASIGNATURA y sustituye 'MATEMÁTICAS' por 'Mates':

```
SELECT CONCAT(Nombre_Alumno,Asignatura) "Concatenación",
REPLACE(Asignatura, 'MATEMÁTICAS', 'Mates') "Sustitución"
FROM notas_alumnos;
```

Concatenación	Sustitución
Alcalde García, M. LuisaMATEMÁTICAS	Mates
Benito Martín, LuisMATEMÁTICAS	Mates
Casas Martínez, ManuelLENGUA	LENGUA
Corregidor Sánchez, AnaLENGUA	LENGUA
Díaz Sánchez, MaríaLENGUA	LENGUA

Figura 11.39. Consulta 3. Ejercicio Resuelto 11.10

- La siguiente consulta devuelve la longitud de la cadena NOMBRE_ALUMNO y la posición donde comienza el apellido 'Sánchez' en la columna NOMBRE_ALUMNO:

```
SELECT Nombre_Alumno, LENGTH(Nombre_Alumno)"Longitud",
INSTR(Nombre_Alumno,'Sánchez') "Posición"
FROM notas_alumnos;
```

Nombre_Alumno	Longitud	Posición
Alcalde García, M. Luisa	24	0
Benito Martín, Luis	19	0
Casas Martínez, Manuel	22	0
Corregidor Sánchez, Ana	23	12
Díaz Sánchez, María	19	6

Figura 11.40. Consulta 4. Ejercicio Resuelto 11.10

Actividad de Aplicación 11.4

Consulta desde el navegador de funciones algunas funciones de caracteres que no se hayan visto y pruébalas.



11.3.3. Funciones de manejo de fechas y horas

MySQL proporciona gran cantidad de funciones para trabajar con las fechas y horas. Por defecto, la fecha se almacena en formato AAAA-MM-DD; es decir, los cuatro primeros dígitos corresponden al año, los dos siguientes al mes y los dos siguientes al día. El formato de la hora se expresa en forma de HH:MM:SS, es decir, horas:minutos:segundos. Las funciones que nos permiten descomponer la fecha y la hora se muestran en la Tabla 11.3.

Función	Propósito
CURTIME() CURRENT_TIME CURRENT_TIME()	Devuelve la hora actual en formato HH:MM:SS.
CURDATE() CURRENT_DATE CURRENT_DATE()	Devuelve la fecha actual en formato AAAA:MM:DD
SYSDATE()	Devuelve una cadena con la fecha y hora actual: AAAA:MM:DD HH:MM:SS.
DAYNAME(fecha)	Devuelve el nombre del día de la semana en inglés.
DAYOFMONTH(fecha)	Devuelve el número de día del mes (1,2,...,31).
MONTH(fecha)	Devuelve el número del mes de la fecha dada (1,2,...,12).
MONTHNAME(fecha)	Devuelve el nombre del mes de la fecha dada en inglés.
YEAR(fecha)	Devuelve un entero entre 1000 y 9999 que indica el año de la fecha dada.
HOURL(hora)	Devuelve la hora, un entero de 0 a 23.
MINUTE(hora)	Devuelve los minutos, un entero de 0 a 59.
SECOND(hora)	Devuelve los segundos, un entero de 0 a 59.

Tabla 11.3. Funciones de fecha y hora

Ejercicio Resuelto 11.11

Probamos las funciones expuestas anteriormente.

1. A partir de la tabla LIBROS de la base de datos EDITORIALES. La siguiente consulta muestra la FECHA_PUBLICACION, el año, el nombre del mes, el día del mes y el nombre del día de los libros de la tabla LIBROS de la editorial 20:

```
SELECT Titulo, Autor, Fecha_publicacion,
       YEAR(Fecha_publicacion) "Año",
       MONTHNAME(Fecha_publicacion) "Mes",
       DAYOFMONTH(Fecha_publicacion) "Día",
       DAYNAME(Fecha_publicacion) "Nombre Día"
FROM libros WHERE editorial = 20;
```

Título	Autor	Fecha_publicacion	Año	Mes	Día	Nombre Día
▶ XML A TRAVÉS DE EJEMPLOS	A.GUTIÉRREZ, R.MARTÍNEZ	2001-02-01	2001	February	1	Thursday
REDES IP	J.G.TOMÁS, J.L.RAYA y otros	2002-03-15	2002	March	15	Friday
SQL Y JAVA	J.MELTON, A.EISENBERG	2002-01-24	2002	January	24	Thursday
DISEÑO Y DESARROLLO MULTIMEDIA	M.A.CASTRO, A.COLMENAR y otros	2002-11-01	2002	November	1	Friday

Figura 11.41. Consulta 1. Ejercicio Resuelto 11.11

2. Descomponemos la hora actual en horas, minutos y segundos:

```
SELECT CURTIME(), HOUR(CURTIME()) "HH", MINUTE(CURTIME()) "MM",
       SECOND(CURTIME()) "SS";
```

CURTIME()	HH	MM	SS
▶ 15:15:24	15	15	24

Figura 11.42. Consulta 2. Ejercicio Resuelto 11.11

3. Descomponemos la fecha actual en día, mes y año:

```
SELECT CURDATE(), YEAR(CURDATE()) "YYYY", MONTH(CURDATE()) "MM", DAYOF-
MONTH(CURDATE()) "DD";
```

CURDATE()	YYYY	MM	DD
▶ 2006-11-25	2006	11	25

Figura 11.43. Consulta 3. Ejercicio Resuelto 11.11

Actividad de Aplicación 11.5

Consulta desde el navegador de funciones algunas funciones de fechas y horas que no se hayan visto y pruébalas.



11.3.4. Funciones de comparación

En este apartado veremos algunas funciones que comparan los valores de cada una de las columnas en el interior de una fila para obtener el mayor o el menor valor de ellos y una función que comprueba si el valor de una columna es nulo. Se muestran en la Tabla 11.4.

Función	Propósito
GREATEST(valor1, valor2, ...)	Obtiene el mayor valor de la lista.
LEAST(valor1, valor2, ...)	Obtiene el menor valor de la lista.
IFNULL(expr1, expr2)	Si "expr1" no es NULL, la función devuelve "expr1"; en caso contrario la función devuelve "expr2".

Tabla 11.4. Funciones de comparación

Ejercicio Resuelto 11.12

Probamos las funciones expuestas anteriormente con la tabla NOTAS_ALUMNOS de la base de datos UNIDAD11.

1. Obtenemos el nombre de alumno, la mayor nota de las tres que tiene y la menor:

```
SELECT Nombre_Alumno, GREATEST(notas1, notas2, notas3) "Mayor Nota",
       LEAST(notas1, notas2, notas3) "Menor Nota"
FROM notas_alumnos;
```

Nombre_Alumno	Mayor Nota	Menor Nota
▶ Alcalde García, M. Luisa	5	5
Berito Martín, Luis	8	6
Casas Martínez, Manuel	7	5
Corregidor Sánchez, Ana	9	6
Díaz Sánchez, María	NULL	NULL

Figura 11.44. Consulta 1. Ejercicio Resuelto 11.12

2. La siguiente consulta muestra las tres notas de los alumnos. Si alguna nota tiene valor nulo, se visualizará en su columna la palabra 'NP':

```
SELECT Nombre_Alumno, IFNULL(nota1, 'NP') "Nota 1",
IFNULL(nota2, 'NP') "Nota 2",
IFNULL(nota3, 'NP') "Nota 3"
FROM notas_alumnos;
```

Nombre_Alumno	Nota 1	Nota 2	Nota 3
Alcalde García, M. Luisa	5	5	5
Benito Martín, Luis	7	6	8
Casas Martínez, Manuel	7	5	5
Corregidor Sánchez, Ana	6	9	8
Díaz Sánchez, María	NP	NP	7

Figura 11.45. Consulta 2. Ejercicio Resuelto 11.12



Direcciones Web de interés:

Zona de descarga de documentación sobre MySQL:
<http://dev.mysql.com/doc/>

Manual de referencia de MySQL en castellano:
<http://dev.mysql.com/doc/refman/5.0/es/index.html>

Fuente de consulta y referencia para el aprendizaje de MySQL: artículos, noticias, eventos, etc.
<http://www.mysql-hispano.org/>

Tutorial básico de MySQL:
http://www.programacion.com/bbdd/tutorial/mysql_basico/

Ejercicios propuestos

- 11.1 Disponemos de la tabla ALUM2006 que contiene los datos de alumnos matriculados en el curso 2005/2006 para un centro de enseñanza:

Columna	Tipo de dato	Descripción
DNI	VARCHAR(10)	DNI del alumno
NOMBRE	VARCHAR(15)	Nombre del alumno
APELLIDOS	VARCHAR(20)	Apellidos del alumno
FECHA_NAC	DATE	Fecha de nacimiento
DIRECCION	VARCHAR(20)	Dirección del alumno
POBLACION	VARCHAR(20)	Población del alumno
PROVINCIA	VARCHAR(20)	Provincia del alumno

Columna	Tipo de dato	Descripción
CURSO	INT(2)	Curso del alumno (1, 2, 3, 4)
NIVEL	VARCHAR(3)	Nivel, puede ser: ESO, BAC, DAI, ASI, ADM, COM, ESI
CLASE	CHAR(1)	Aula en la que está el alumno, puede ser: A, B, C, D, E, F
FALTAS1	INT(2)	Faltas primer trimestre
FALTAS2	INT(2)	Faltas segundo trimestre
FALTAS3	INT(2)	Faltas tercer trimestre

Obtener las siguientes consultas:

- Consultar todos los datos de los alumnos.
- Consultar los siguientes datos de alumnos: DNI, NOMBRE, APELLIDOS, CURSO, NIVEL y CLASE.
- Consultar el NOMBRE y APELLIDOS de todos los alumnos cuya POBLACIÓN sea 'GUADALAJARA'.
- Consultar el DNI, NOMBRE, APELLIDOS, CURSO, NIVEL, CLASE y edad de todos los alumnos ordenado por APELLIDOS y NOMBRE ascendentemente.
- Consultar aquellos DNI cuya fecha de nacimiento sea nula.
- Consultar todos los datos de los alumnos cuya fecha de nacimiento no sea nula.
- Consultar el DNI, NOMBRE y APELLIDOS de todos aquellos alumnos que tengan entre sus apellidos el apellido 'Pérez'.
- Consultar el DNI, NOMBRE, APELLIDOS, CURSO, NIVEL y CLASE de todos los alumnos cuya población sea alguna de las siguientes: 'MARCHAMALO', 'CABANILLAS' o 'YUNQUERA'.
- Consultar el DNI, NOMBRE, APELLIDOS, CURSO, NIVEL y CLASE de todos aquellos alumnos cuya edad esté comprendida entre 17 y 20 años.
- Obtener el DNI, NOMBRE, APELLIDOS y el máximo de faltas de los tres trimestres para aquellos alumnos de 'ESO'.
- Obtener el DNI, NOMBRE, APELLIDOS, NIVEL y la media de faltas de los tres trimestres para aquellos alumnos de 'ESO' y 'ESI'.
- Consultar los datos de los alumnos que tengan faltas con valor nulo.
- Obtener el DNI, NOMBRE, APELLIDOS y NIVEL de todos los alumnos que nacieron en el año 1985 y en el mes de Febrero (February, en inglés).
- Obtener en una columna el DNI, y en otra, la concatenación de las columnas NOMBRE y APELLIDOS, de todos los alumnos de la tabla ordenando descendentemente por DNI.
- Obtener el NOMBRE y APELLIDOS de todos los alumnos de cuarto de 'ESO' de la clase 'B' ordenados por APELLIDOS y NOMBRE ascendentemente.
- Obtener el NOMBRE, APELLIDOS y el total de faltas de los tres trimestres de todos los alumnos cuarto de 'ESO' de la clase 'B' ordenados por APELLIDOS y NOMBRE ascendentemente.
- Consultar el DNI, NOMBRE, APELLIDOS, CURSO, NIVEL, CLASE y edad de todos los alumnos ordenado por CURSO, NIVEL, CLASE ascendentemente y APELLIDOS descendentemente cuyo nivel no sea ni 'ESO' ni 'BAC'.
- Obtener la consulta anterior para aquellos alumnos cuya edad esté comprendida entre 20 y 22.