

# INFORME PRÁCTICO ASO - CONSULTAS EN MYSQL WORKBENCH

## Introducción

Este informe detalla la creación de una base de datos en **MySQL Workbench**, la estructura de la tabla Empleados, la inserción de datos y la ejecución de consultas específicas sobre ella. Se incluyen capturas de pantalla y explicaciones detalladas de cada paso.

## 1. Creación e inserción de datos en la tabla Empleados

### Paso 1: Creación de la base de datos

Abrimos **MySQL Workbench** y ejecutamos:

```
CREATE DATABASE CP_UD5;
USE CP_UD5;
```

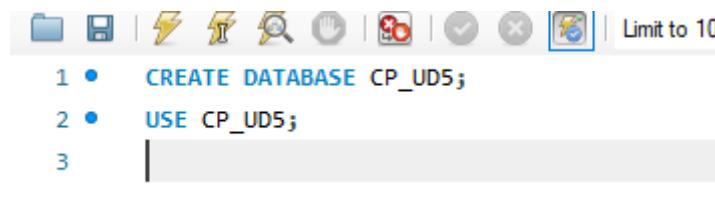


Ilustración 1: CREATE DATABASE CP\_UD5;

2 09:54:06	CREATE DATABASE CP_UD5	1 row(s) affected	0.015 sec
3 09:54:26	USE CP_UD5	0 row(s) affected	0.000 sec

Ilustración 2: Creación y uso de la base de datos

### Explicación:

- **CREATE DATABASE CP\_UD5;** → Crea la base de datos denominada CP\_UD5.
- **USE CP\_UD5;** → Nos aseguramos de que todas las operaciones posteriores se realicen dentro de esta base de datos.

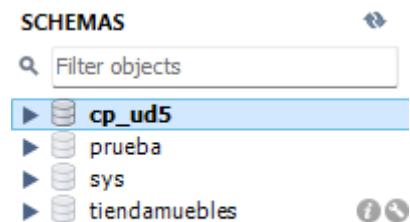


Ilustración 3: Visualización de la base de datos

## Paso 2: Creación de la tabla Empleados

Ejecutamos la siguiente consulta para crear la tabla con los campos especificados en la imagen:

```
CREATE TABLE Empleados (
    DNI INT PRIMARY KEY NOT NULL ,
    Nombre VARCHAR(45),
    Apellidos VARCHAR(45),
    Fecha_Nac DATE,
    N_hijos INT DEFAULT NULL,
    estatura DOUBLE );
```

### Explicación:

- **DNI INT PRIMARY KEY NOT NULL** → Define **DNI** como clave primaria y evita valores **NULL**.
- **Nombre VARCHAR(45), Apellidos VARCHAR(45)** → Almacenan los datos personales del empleado.
- **Fecha\_Nac DATE** → Registra la fecha de nacimiento en formato YYYY-MM-DD.
- **N\_hijos INT DEFAULT NULL** → Permite valores **NULL** para aquellos empleados que no han proporcionado esta información.
- **estatura DOUBLE** → Guarda la altura en metros con decimales.

Para verificar la estructura de la tabla, ejecutamos:

```
DESCRIBE empleados;
```

The screenshot shows a database result grid for the 'empleados' table. The grid has columns for Field, Type, Null, Key, Default, and Extra. The data is as follows:

Field	Type	Null	Key	Default	Extra
DNI	int	NO	PRI	NULL	
Nombre	varchar(45)	YES		NULL	
Apellidos	varchar(45)	YES		NULL	
Fecha_Nac	date	YES		NULL	
N_hijos	int	YES		NULL	
estatura	double	YES		NULL	

Ilustración 4: DESCRIBE empleados;

## Paso 3: Inserción de datos en la tabla

Insertamos los registros de los empleados con la siguiente consulta SQL:

```
INSERT INTO Empleados (DNI, Nombre, Apellidos, Fecha_Nac, N_hijos, estatura) VALUES
(1, 'Sergio', 'González', '2000-01-25', 1, 1.78),
(2, 'Carmen', 'González', '1970-05-22', 5, 2.00),
(3, 'Leoncio', 'Lorca', '2000-10-24', 4, 1.69),
(4, 'Zola', 'Ramos', '1978-01-30', NULL, NULL),
(5, 'José', 'Hierro', '1950-01-25', 0, 2.05),
(6, 'Paloma', NULL, '2000-08-08', 1, 1.50);
```

- `INSERT INTO Empleados (DNI, Nombre, Apellidos, Fecha_Nac, N_hijos, estatura) VALUES (1, 'Sergio', 'González', '2000-01-25', 1, 1.78), (2, 'Carmen', 'González', '1970-05-22', 5, 2.00), (3, 'Leoncio', 'Lorca', '2000-10-24', 4, 1.69), (4, 'Zola', 'Ramos', '1978-01-30', NULL, NULL), (5, 'José', 'Hierro', '1950-01-25', 0, 2.05), (6, 'Paloma', NULL, '2000-08-08', 1, 1.50);`

Ilustración 5: Registros de los empleados

#### Explicación:

- Los datos insertados coinciden exactamente con los proporcionados en la imagen de referencia.
- Se incluyen valores **NULL** en **N\_hijos** cuando la información no está disponible.

Para confirmar que la inserción se ha realizado correctamente, ejecutamos:

```
SELECT * FROM Empleados;
```

	DNI	Nombre	Apellidos	Fecha_Nac	N_hijos	estatura
1	Sergio	González	2000-01-25	1	1.78	
2	Carmen	González	1970-05-22	5	2	
3	Leoncio	Lorca	2000-10-24	4	1.69	
4	Zola	Ramos	1978-01-30	NULL	NULL	
5	José	Hierro	1950-01-25	0	2.05	
6	Paloma	NULL	2000-08-08	1	1.5	
NULL	NULL	NULL	NULL	NULL	NULL	

Ilustración 6: `SELECT * FROM Empleados;`

## 2. Consultas SQL realizadas

**Consulta 1:** Obtener empleados con 1, 2, 3 o 5 hijos.

```
SELECT * FROM Empleados WHERE N_hijos IN (1, 2, 3, 5);
```

#### Explicación:

- Se emplea **IN (1, 2, 3, 5)** en lugar de múltiples condiciones con **OR**, lo que mejora la legibilidad y eficiencia de la consulta.

2 • `SELECT * FROM Empleados WHERE N_hijos IN (1, 2, 3, 5);`

	DNI	Nombre	Apellidos	Fecha_Nac	N_hijos	estatura
1	Sergio	González	2000-01-25	1	1.78	
2	Carmen	González	1970-05-22	5	2	
6	Paloma	NULL	2000-08-08	1	1.5	
NULL	NULL	NULL	NULL	NULL	NULL	

Ilustración 7: `SELECT * FROM Empleados WHERE N_hijos IN (1, 2, 3, 5);`

**Consulta 2:** Obtener empleados nacidos entre el 01/01/1985 y el 31/12/2000

```
SELECT * FROM Empleados WHERE Fecha_Nac BETWEEN '1985-01-01' AND '2000-12-31';
```

**Explicación:**

- **BETWEEN** se usa para seleccionar fechas dentro del rango especificado.
- Se utiliza el formato **YYYY-MM-DD**, que es el estándar en MySQL.

The screenshot shows a MySQL Workbench interface. At the top, there is a code editor with the following SQL query:

```
3 •   SELECT * FROM Empleados
4     WHERE Fecha_Nac BETWEEN '1985-01-01' AND '2000-12-31';
5   |
```

Below the code editor is a "Result Grid" window. The grid has the following columns: DNI, Nombre, Apellidos, Fecha\_Nac, N\_hijos, and estatura. The data is as follows:

DNI	Nombre	Apellidos	Fecha_Nac	N_hijos	estatura
1	Sergio	González	2000-01-25	1	1.78
3	Leoncio	Lorca	2000-10-24	4	1.69
6	Paloma	NULL	2000-08-08	1	1.5
NULL	NULL	NULL	NULL	NULL	NULL

Ilustración 8: `SELECT * FROM Empleados WHERE Fecha_Nac BETWEEN '1985-01-01' AND '2000-12-31';`

**Consulta 3:** Obtener empleados sin información en N\_hijos (valores NULL)

```
SELECT * FROM Empleados WHERE N_hijos IS NULL;
```

**Explicación:**

- Los valores **NULL** no pueden compararse con `=`, por lo que se usa `IS NULL`.
- Esta consulta devuelve los empleados que no han registrado el número de hijos.

The screenshot shows a MySQL Workbench interface. At the top, there is a code editor with the following SQL query:

```
6 •   ALTER TABLE Empleados ADD COLUMN nombreApellidos VARCHAR(100);
7   |
```

Below the code editor is a "Result Grid" window. The grid has the following columns: DNI, Nombre, Apellidos, Fecha\_Nac, N\_hijos, and estatura. The data is as follows:

DNI	Nombre	Apellidos	Fecha_Nac	N_hijos	estatura
4	Zola	Ramos	1978-01-30	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Ilustración 9: `SELECT * FROM Empleados WHERE N_hijos IS NULL;`

## Creación de la nueva columna nombreApellidos

**Paso 1: Añadir la nueva columna**

```
ALTER TABLE Empleados ADD COLUMN nombreApellidos VARCHAR(100);
```

```
6 •   ALTER TABLE Empleados ADD COLUMN nombreApellidos VARCHAR(100);
7   |
```

**Explicación:**

- Se añade una nueva columna **nombreApellidos** con una longitud de **100 caracteres**, suficiente para almacenar la concatenación del nombre y los apellidos.

## Paso 2: Rellenar la columna con la información combinada

```
UPDATE Empleados SET nombreApellidos = CONCAT(Nombre, ' ', IFNULL(Apellidos, ''));
```

### Explicación:

- `CONCAT(Nombre, ' ', IFNULL(Apellidos, ''))` une el **Nombre** y los **Apellidos**, agregando un espacio entre ellos.
- **Se usa `IFNULL(Apellidos, '')`** para evitar valores NULL en los casos donde el apellido no ha sido registrado.
- Se ejecuta UPDATE para actualizar la columna con los valores existentes.

## Comprobación de los cambios

### Primera verificación: Revisión completa de la tabla

Para confirmar que la actualización se ha aplicado correctamente en toda la tabla, ejecutamos la siguiente consulta:

```
SELECT * FROM Empleados;
```

DNI	Nombre	Apellidos	Fecha_Nac	N_hijos	estatura	nombreApellidos
1	Sergio	González	2000-01-25	1	1.78	Sergio González
2	Carmen	González	1970-05-22	5	2	Carmen González
3	Leoncio	Lorca	2000-10-24	4	1.69	Leoncio Lorca
4	Zola	Ramos	1978-01-30	NULL	NULL	Zola Ramos
5	José	Hierro	1950-01-25	0	2.05	José Hierro
6	Paloma	NULL	2000-08-08	1	1.5	Paloma
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Ilustración 10: Creación de la nueva columna nombreApellidos

Esta consulta permite revisar todos los campos de cada empleado y verificar que la columna nombreApellidos ha sido actualizada correctamente.

### Segunda verificación: Revisión específica de la columna nombreApellidos

Para comprobar exclusivamente los valores de la nueva columna, ejecutamos:

```
SELECT nombreApellidos FROM Empleados;
```

nombreApellidos
Sergio González
Carmen González
Leoncio Lorca
Zola Ramos
José Hierro
Paloma

Ilustración 11: SELECT nombreApellidos FROM Empleados;

Esto nos permite centrarnos en el resultado final de la concatenación y verificar que los nombres y apellidos se han combinado adecuadamente, sin errores ni valores **NULL** inesperados.

## Conclusión

Este informe ha detallado paso a paso la creación de una base de datos en MySQL Workbench, la definición de la tabla Empleados, la inserción de datos y la ejecución de diversas consultas SQL.

Además, se ha añadido una nueva columna nombreApellidos, asegurando que los datos sean gestionados de manera eficiente y manteniendo la integridad de la información.

### Bibliografía y Fuentes de Información:

#### Documentación Oficial de MySQL

- Oracle. (2024). *MySQL 8.0 Reference Manual*.

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

Fuente oficial que proporciona información detallada sobre la sintaxis SQL, estructura de datos y mejores prácticas en MySQL.

#### Manual de MySQL en Español

- Cartagena99. (2021). *Manual de MySQL*.

<https://www.cartagena99.com/recursos/alumnos/apuntes/210927124637-Manual%20MySQL.pdf>

Documento en español con explicaciones detalladas sobre MySQL, comandos básicos y avanzados.

#### Libros y Recursos Académicos

- McGraw Hill. (s.f.). *Capítulo sobre bases de datos y SQL*.

<https://www.mheducation.es/bcv/guide/capitulo/8448148819.pdf>

Material educativo con explicaciones teóricas y ejemplos prácticos sobre SQL y bases de datos.

#### Blogs y Foros Especializados

- Stack Overflow. (2024). *Errores comunes y soluciones en MySQL*.

<https://stackoverflow.com/>

Resolución de problemas y buenas prácticas en la comunidad de desarrolladores.

- SQL Shack. (2023). *Mejores prácticas en SQL y MySQL*.

<https://www.sqlshack.com/es/>

Explica estrategias de optimización y uso eficiente de MySQL.