Guía manual de SQL

Qué es SQL

SQL es un lenguaje para **trabajar con bases de datos**.

Una base de datos guarda información en **tablas** (como hojas de Excel), pero con **reglas** para evitar errores.

- **Tabla**: "Clientes", "Habitaciones", "Reservas"
- **Fila/registro**: un cliente concreto, una habitación concreta
- **Columna/campo**: nombre, precio, fecha, email…
- **Relación**: una tabla se conecta con otra (reserva → cliente + habitación)

ESTRUCTURA (DDL)

- DDL
  - Crea
  - Modifica
  - borrar la estructura.

Paso 1. Crear una base de datos
CREATE DATABASE HotelDB;

Paso 2. Elegir la base de datos
USE HotelDB;

Paso 3. Elegir tipos de datos

- **INT** : números enteros (IDs)
- **VARCHAR(n)** : texto (n = máximo de caracteres)
- **DECIMAL(p,s)** : números con decimales exactos (dinero)
- **DATE** : fechas
- **BOOLEAN** : true/false

Consejo importante: teléfonos y códigos suelen ser VARCHAR, no INT.

Paso 4. Crear una tabla simple

Ejemplo: tabla **Clientes**.

```
CREATE TABLE Clientes (
  id_cliente INT PRIMARY KEY,
  nombre VARCHAR(80) NOT NULL,
  telefono VARCHAR(15),
  correo VARCHAR(100) UNIQUE
);
```

Qué significa cada regla (restricción)

- **PRIMARY KEY:** identifica de forma única cada fila.
- **NOT NULL:** obligatorio (no puede estar vacío).
- **UNIQUE**: no se repite (por ejemplo, emails).

Paso 5. Crear otra tabla: Habitaciones

```
CREATE TABLE Habitaciones (
  id_habitacion INT PRIMARY KEY,
  tipo VARCHAR(50),
  precio DECIMAL(6,2),
  disponible BOOLEAN DEFAULT TRUE
);
```

- **DEFAULT TRUE:** si no dices nada, por defecto está disponible.

Paso 6. Crear una tabla relacionada: Reservas

Aquí entra lo más importante de bases de datos: **relaciones** con claves foráneas.

```
CREATE TABLE Reservas (
  id_reserva INT PRIMARY KEY,
  id_cliente INT NOT NULL,
  id_habitacion INT NOT NULL,
  fecha_entrada DATE NOT NULL,
  fecha_salida DATE NOT NULL,
  FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente),
```

```
  FOREIGN KEY (id_habitacion) REFERENCES
Habitaciones(id_habitacion)
);
```

Qué es una FOREIGN KEY

Una **clave foránea** obliga a que:

- id_cliente exista en Clientes
- id_habitacion exista en Habitaciones

Evita reservas “fantasma”.

DATOS (DML)

- DML
    - Insertar
    - Consultar
    - Actualizar
    - borrar datos.

Paso 7. Insertar datos (INSERT)

Insertar clientes

```
INSERT INTO Clientes VALUES
(1, 'Ana López', '600123456', 'ana@correo.com'),
(2, 'Javier Ruiz', '699874512', 'javier@correo.com');
```

Insertar habitaciones

```
INSERT INTO Habitaciones VALUES
(101, 'Suite', 120.00, TRUE),
(102, 'Doble', 90.00, TRUE),
(103, 'Individual', 70.00, TRUE);
```

Insertar una reserva

```
INSERT INTO Reservas VALUES
(1001, 1, 101, '2025-10-15', '2025-10-18');
```

Regla práctica: primero insertas en tablas “padre” (Clientes/Habitaciones) y después en la “hija” (Reservas).

Paso 8.Consultar datos (SELECT)

Ver toda la tabla
SELECT * FROM Clientes;

Ver columnas concretas
SELECT nombre, correo FROM Clientes;

Filtrar resultados con WHERE
SELECT * FROM Habitaciones WHERE disponible = TRUE;

Ordenar resultados
SELECT * FROM Habitaciones ORDER BY precio DESC;

Paso 9. Actualizar datos (UPDATE)

Cambiar un dato existente.

UPDATE Clientes
SET telefono = '611222333'
WHERE id_cliente = 1;

Regla de oro: UPDATE sin WHERE cambia TODAS las filas.

Hábito profesional:

1. prueba el filtro con SELECT
2. luego haces UPDATE

SELECT * FROM Clientes WHERE id_cliente = 1;

Paso 10. Borrar datos (DELETE)
Borrar una fila concreta.
DELETE FROM Reservas
WHERE id_reserva = 1001;
DELETE sin WHERE borra TODO.

CONSULTAS RELACIONALES (JOIN)

Paso 11. Unir tablas (JOIN)

Listar reservas mostrando nombre del cliente y tipo de habitación:

```
SELECT C.nombre, H.tipo, R.fecha_entrada, R.fecha_salida
FROM Reservas R
JOIN Clientes C ON R.id_cliente = C.id_cliente
JOIN Habitaciones H ON R.id_habitacion = H.id_habitacion;
```

Cómo leerlo

- Empiezo en Reservas (R)
- La conecto con Clientes (C) por el id_cliente
- La conecto con Habitaciones (H) por el id_habitacion
- Elijo qué columnas quiero mostrar

SEGURIDAD Y BUENAS PRÁCTICAS

Paso 12. Transacciones (evitar errores graves)

Sirve para que varias órdenes se ejecuten como “un paquete”.

Ejemplo: crear reserva y marcar habitación como ocupada:

```
START TRANSACTION;

INSERT INTO Reservas VALUES
(1002, 2, 102, '2025-10-20', '2025-10-23');

UPDATE Habitaciones
SET disponible = FALSE
WHERE id_habitacion = 102;

COMMIT;
```

Si algo sale mal:

```
ROLLBACK;
```

CAMBIOS EN LA ESTRUCTURA (ALTER/DROP)

Paso 13. Modificar una tabla (ALTER)

Añadir columna:

```
ALTER TABLE Clientes
ADD direccion VARCHAR(120);
```

Cambiar tipo:

```
ALTER TABLE Clientes
MODIFY telefono VARCHAR(20);
```

Paso 14. Borrar estructura (DROP)

Borrar una tabla completa:

```
DROP TABLE Reservas;
```

Borrar base de datos:

```
DROP DATABASE HotelDB;
```

Esto borra TODO de verdad.

Resumen

- Crear estructura
  - CREATE
  - ALTER
  - DROP

- Trabajar datos
  - INSERT
  - SELECT
  - UPDATE
  - DELETE

- UPDATE y DELETE
  - **SIEMPRE con WHERE**

- Varias acciones importantes
    - **TRANSACTION + COMMIT/ROLLBACK**
- Unir tablas : **JOIN**