

Normalización en Bases de Datos Relacionales

"Aprenderemos a organizar datos para que no se repitan innecesariamente y mantengan su coherencia".





INTEGRITY



ORGANIZATION



PERFORMANCE



Objetivos de Aprendizaje



Conocer las reglas de normalización (1FN, 2FN, 3FN).



Aplicar normalización paso a paso en tablas.







Entender cómo mejora la integridad y el rendimiento.



Practicar con ejemplos reales en VS Code + SQLite.

¿Qué es la Normalización?

-  Es el proceso de organizar datos en una base de datos.
-  Evita la redundancia (datos repetidos).
-  Previene inconsistencias y pérdida de integridad.
-  Hace las consultas más rápidas y claras.



Problema Sin Normalización

Ejemplo de tabla inicial:

ID	Nombre	Teléfonos
1	Ana	555-111, 555-222

✗ Problemas:

- Valores repetidos.
- Difícil buscar y mantener.
- Peligro de inconsistencia.

Beneficios de la Normalización



Menos redundancia

Eliminación de datos duplicados que ocupan espacio innecesario.



Mejor integridad de datos

Garantiza que los datos sean consistentes y precisos.



Consultas más rápidas

Mejora el rendimiento de las búsquedas en la base de datos.



Tablas más claras

Estructura más entendible y fácil de mantener.

Before

≡ Skillnest After



Primera Forma Normal (1FN)

Reglas:

- Cada columna debe tener valores indivisibles (atómicos).
- Nada de listas o valores separados por comas.
- Cada fila debe ser única.

Ejemplo:

✗ "Ana – 555-111, 555-222"

✓ Separar en dos filas:

ID	Nombre	Teléfono
----	--------	----------

Segunda Forma Normal (2FN)

Reglas:

- Cumple con 1FN.

Todos los atributos no clave dependen de toda la clave primaria.

- Elimina dependencias parciales.

Ejemplo sin 2NF:

| EstudianteID | CursoID |

NombreEstudiante |

NombreCurso |

Solución:

- Tabla Estudiantes
- Tabla Cursos
- Tabla Inscripciones



Tercera Forma Normal (3FN)

Reglas:

- Cumple con 2FN.
- Elimina dependencias transitivas.

Ejemplo:

Si "TeléfonoEditorial"

depende de

"NombreEditorial" y no del

ID, deben ir en otra tabla.

Ejemplo Guiado – Tabla sin Normalizar

ID	Cliente	Producto	Cantidad	PrecioUnitario	Dirección
1	Juan Pérez	Laptop	1	1000	Calle Falsa 123
2	Ana García	Mouse, Teclado	2,1	20,50	Av. Siempre Viva

Esta tabla presenta múltiples problemas de normalización que resolveremos paso a paso.

Paso 1 – Aplicar 1FN

Separar productos múltiples:

ID	Cliente	Producto	Cantidad	PrecioUnitario	Dirección
1	Juan Pérez	Laptop	1	1000	Calle Falsa 123
2	Ana García	Mouse	2	20	Av. Siempre Viva
2	Ana García	Teclado	1	50	Av. Siempre Viva

Paso 2 – Aplicar 2FN

Separar datos que dependen solo del cliente:

Cientes

ID	Cliente	Dirección
1	Juan Pérez	Calle Falsa 123
2	Ana García	Av. Siempre Viva

Compras

| ID | Producto | Cantidad | PrecioUnitario |

Paso 3 – Aplicar 3FN

Separar productos de precios:

Productos

Producto	PrecioUnitario
Laptop	1000
Mouse	20
Teclado	50

DetalleCompra

| ID | Producto | Cantidad |

Ejercicio Práctico en Clase



Crear base de datos SQLite tienda.db



Crear tabla sin normalizar



Insertar datos



Normalizar paso a paso hasta 3FN

 **Actividad grupal en VS Code**

Código SQL Inicial

```
CREATE TABLE compras (  
  id INTEGER,  
  cliente TEXT,  
  producto TEXT,  
  cantidad INTEGER,  
  precio_unitario REAL,  
  direccion TEXT  
);  
  
INSERT INTO compras VALUES  
(1, 'Juan Pérez', 'Laptop', 1, 1000, 'Calle Falsa 123'),  
(2, 'Ana García', 'Mouse, Teclado', '2,1', '20,50', 'Av. Siempre  
Viva');
```

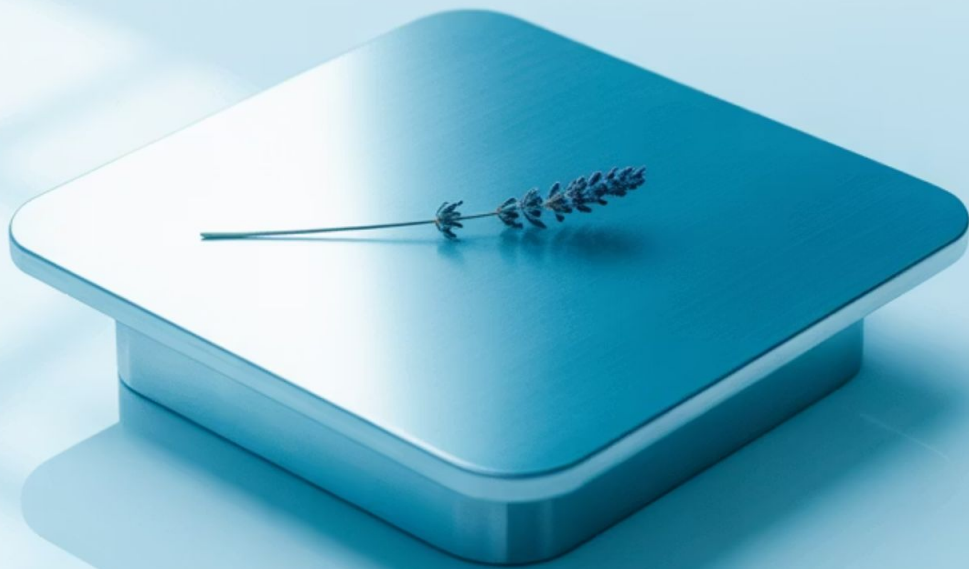
Este código crea la tabla inicial con los problemas de normalización que hemos identificado.



Código Normalización – Clientes

```
CREATE TABLE clientes (  
    id INTEGER PRIMARY KEY,  
    nombre TEXT,  
    direccion TEXT  
);  
  
INSERT INTO clientes VALUES  
(1, 'Juan Pérez', 'Calle Falsa 123'),  
(2, 'Ana García', 'Av. Siempre Viva');
```

Primera tabla normalizada que contiene solo la información de los clientes.



Código Normalización – Productos

```
CREATE TABLE productos (  
    producto TEXT PRIMARY KEY,  
    precio_unitario REAL  
);  
INSERT INTO productos VALUES  
( 'Laptop', 1000),  
( 'Mouse', 20),  
( 'Teclado', 50);
```

Segunda tabla normalizada que contiene solo la información de los productos y sus precios.



Código Normalización – DetalleCompra

```
CREATE TABLE detalle_compra (  
    id INTEGER,  
    producto TEXT,  
    cantidad INTEGER,  
    FOREIGN KEY (id) REFERENCES clientes(id),  
    FOREIGN KEY (producto) REFERENCES productos(producto)  
);  
INSERT INTO detalle_compra VALUES  
(1, 'Laptop', 1),  
(2, 'Mouse', 2),  
(2, 'Teclado', 1);
```

Tabla de relación que conecta clientes con productos mediante claves foráneas.

Actividad Activa

En grupos:

1. Diseñar una tabla de "Biblioteca" (libros, autores, préstamos).
2. Identificar redundancias.
3. Normalizar hasta 3FN.
4. Presentar su solución en la clase.

Resumen

1FN: datos atómicos

Cada columna contiene valores indivisibles y cada fila es única.

2FN: eliminar dependencias parciales

Todos los atributos no clave dependen de toda la clave primaria.

3FN: eliminar dependencias transitivas

No hay dependencias entre atributos no clave.

Normalización = menos redundancia + más integridad.

1 FN

2 FN

2 FN

3 FN

3 FN



Cierre y Reflexión

Preguntas para discutir:

¿Qué problemas resolvió la normalización?

Reflexiona sobre los beneficios que obtuvimos al aplicar las formas normales.

¿En qué casos no conviene normalizar demasiado?

Considera situaciones donde la desnormalización podría ser beneficiosa.

¿Cómo aplicarías normalización en un proyecto real?

Piensa en ejemplos concretos de tu experiencia o futuros proyectos.