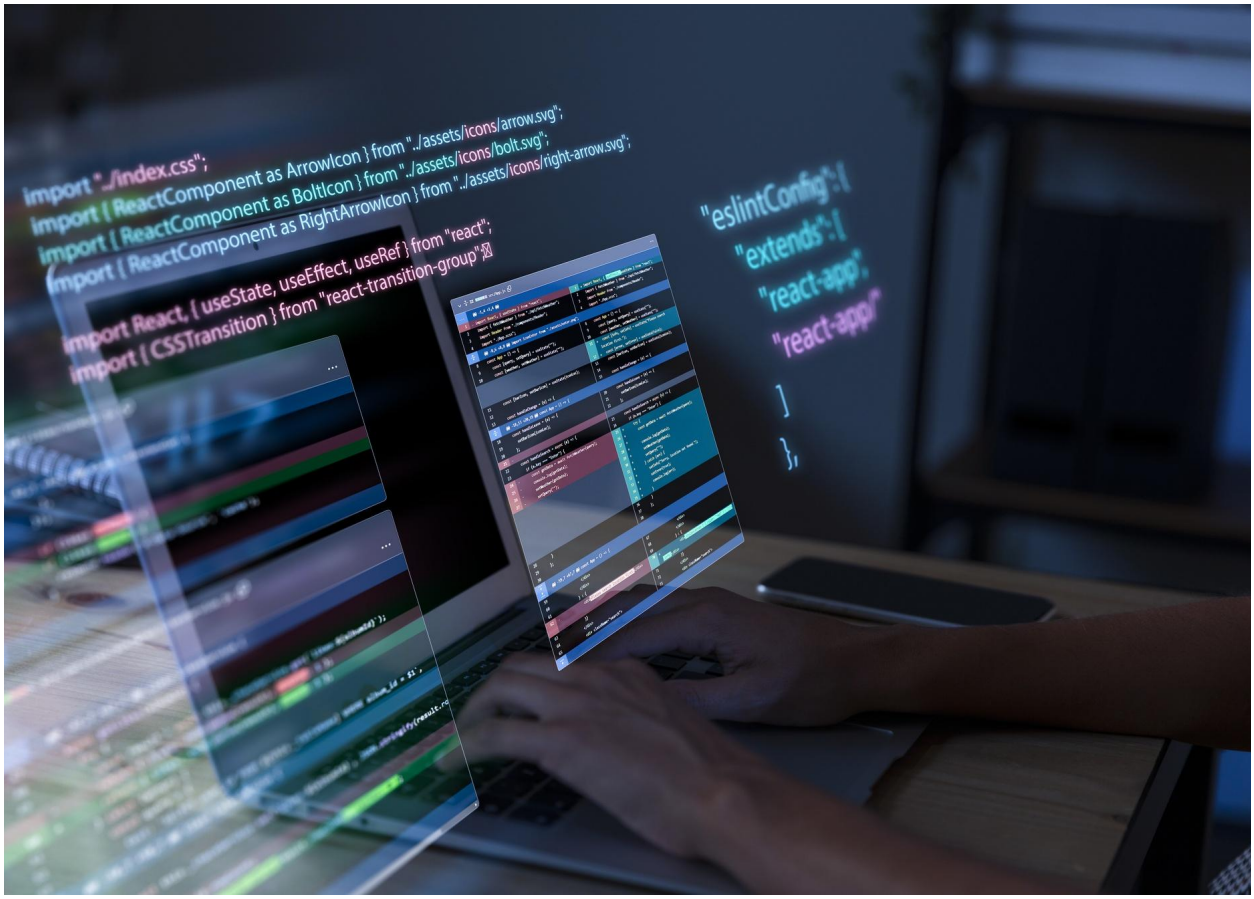
PROMETEO

1º DAM/DAW - BASES DATOS

UNIDAD 4

Normalización



Autor: Luz María Álvarez Moreno

Fecha: 09/12/2025

SESIÓN 12 y 13: DEL MODELO E-R AL MODELO RELACIONAL
Introducción a la Normalización 1FN y 2FN
3FN y Boyce-Codd (BCNF)

1. Recordatorio rápido: ¿Qué es el Modelo E-R?

Es un modelo conceptual que permite representar un problema del mundo real mediante:

Concepto	Qué es	Ejemplo
Entidad	Objeto real	Alumno, Profesor, Producto
Atributo	Propiedad	nombre, precio, fecha_nac
Relación	Asociación entre entidades	Alumno–Matrícula–Asignatura

2. Reglas de transformación E-R: Modelo Relacional

1. ENTIDADES → TABLAS

Cada entidad fuerte del E-R se convierte en una tabla.
Su **identificador único**: **Clave Primaria (PK)**.

Ejemplo
Entidad **Alumno**
Atributos: id, nombre, apellidos, fecha_nacimiento

Tabla resultante:

alumno	**Tipo**
id_alumno (PK)	INT
nombre	VARCHAR
apellidos	VARCHAR
fecha_nacimiento	DATE

Se transforma en:

```
CREATE TABLE Alumno (
  id_alumno INT PRIMARY KEY,
  nombre VARCHAR(50),
  apellidos VARCHAR(100),
  fecha_nacimiento DATE
);
```

La **PK** identifica unívocamente a cada alumno.

2. RELACIONES 1 : N _ CLAVE FORÁNEA

En una relación **uno a muchos**:
La clave foránea se coloca en el lado "muchos".

Ejemplo
Un **Profesor** imparte **muchas Asignaturas**.
Relación 1 → N

Tabla **asignatura**:
Profesor (1)
Asignatura (N)

asignatura	**Tipo**
id_asignatura (PK)	INT
nombre	VARCHAR
creditos	INT
id_profesor (FK → profesor.id_profesor)	INT

3. RELACIONES N : M _ TABLA INTERMEDIA

Una relación muchos–a–muchos **SIEMPRE** genera una tabla adicional.

Ejemplo

- Un alumno puede cursar varias asignaturas
- Una asignatura tiene varios alumnos

Alumnos ⇄ Asignaturas

Tabla intermedia: **matricula**

matricula	**Tipo**
id_alumno (PK, FK)	INT
id_asignatura (PK, FK)	INT
nota	DECIMAL
fecha_matricula	DATE

Se crea tabla **matricula**:

matricula
id_alumno (FK)
id_asignatura (FK)
nota
fecha

PK ⇒ (id_alumno, id_asignatura)

```
CREATE TABLE Matricula (
  id_alumno INT,
  id_asignatura INT,
  nota DECIMAL(3,1),
  fecha DATE,
  PRIMARY KEY (id_alumno, id_asignatura),
  FOREIGN KEY (id_alumno) REFERENCES Alumno(id_alumno),
  FOREIGN KEY (id_asignatura) REFERENCES Asignatura(id_asignatura)
);
```

4. ATRIBUTOS MULTIVALORADOS _ NUEVA TABLA

Si un atributo puede tener varios valores: tabla aparte.

Ejemplo:
Un cliente puede tener varios teléfonos.

Correcto:

Tabla telefonos_cliente:
id_cliente telefono

Ejemplo incorrecto

cliente	telefono
1	666111222, 639444555

Violación 1FN (se corrige luego), pero ya desde aquí:

Crear tabla:

telefonos_cliente
id_cliente (FK)
telefono

Restricciones de integridad

Tipo	Qué asegura	Ejemplo
Entidad	Una PK nunca es NULL ni repetida	No puede haber 2 alumnos con id 15
Referencial	Toda FK debe apuntar a un registro existente	No puedes matricular a un alumno inexistente
Dominio	Cada columna admite solo valores válidos	edad ≥ 0

5. RESTRICCIONES

Tipo	Qué evita
Integridad de entidad (PK)	IDs duplicados o nulos
Integridad referencial (FK)	Referencias incorrectas

Integridad de dominio	Valores incorrectos (edad<0, fecha inválida…)

EJEMPLO COMPLETO "Universidad"

E-R simplificado:

- Alumno
- Profesor
- Asignatura
- Relación: Imparte (1:N)
- Relación: Matricula (N:M)

Esquema relacional final

Tabla	**Contenido**
alumno	id_alumno PK, nombre, apellidos, fecha_nac
profesor	id_profesor PK, nombre, especialidad
asignatura	id_asignatura PK, nombre, creditos, id_profesor FK
matricula	id_alumno FK, id_asignatura FK, nota, fecha_matricula, PK compuesta

Ventajas del modelo:

- No hay alumnos asignados a asignaturas inexistentes.
- Las FK mantienen coherencia.
- Se pueden ejecutar consultas reales del sistema.

Consultas típicas:

- ¿Qué alumnos están matriculados en Programación?
- ¿Qué profesor imparte Matemáticas?
- Nota media por asignatura.

RESUMEN SESIÓN 6

1. Entidades → tablas
2. Atributos → columnas
3. Identificadores → PK
4. Relaciones 1:N → FK
5. Relaciones N:M → tabla intermedia

6. Atributos multivalorados → tabla propia
7. Aplicar integridad (PK, FK y dominio)

NORMALIZACIÓN (1FN, 2FN, 3FN)

¿Qué es la normalización?

Proceso para **evitar redundancias, inconsistencias y problemas de actualización**.

Es decir: "que la base de datos no se rompa con el tiempo".

Tres problemas clásicos:

- **Anomalía de inserción**
- **Anomalía de actualización**
- **Anomalía de eliminación**

La normalización los resuelve.

1FN Primera Forma Normal: Atomicidad

Una tabla está en 1FN si:

- No hay listas
- No hay valores repetidos en una columna
- Cada celda contiene **un solo valor**

Ejemplo NO normalizado

cliente	telefonos
1	600111222, 600333444

Resultado correcto

cliente(id_cliente, nombre…)
telefono_cliente(id_cliente, telefono)

2FN: Segunda Forma Normal: Dependencia TOTAL de la clave

Solo aplica cuando la **PK es compuesta**.

Regla:
Todo atributo no clave debe depender **de toda la clave**.

Ejemplo malo
tabla Pedidos(id_pedido, id_producto, nombre_cliente, cantidad)
nombre_cliente depende solo de id_pedido → **violación 2FN**

Solución
Separar entidades:
cliente(id_cliente, nombre_cliente)
pedido(id_pedido, id_cliente, fecha)
detalle_pedido(id_pedido, id_producto, cantidad)

3FN: Tercera Forma Normal: sin dependencias transitivas

Regla:
Ningún atributo no clave depende de otro atributo no clave.

Ejemplo malo
empleado(id_empleado, nombre, id_departamento, nombre_departamento)
El nombre del departamento depende de id_departamento, no de la clave.

Solución
departamento(id_departamento, nombre_departamento)
empleado(id_empleado, nombre, id_departamento)

BCNF (Boyce–Codd Normal Form / Forma Normal de Codd)

Es una versión más estricta de 3FN.
Se aplica cuando incluso cumpliendo 3FN siguen existiendo dependencias donde un atributo que no es clave determina parte clave.
BCNF corrige problemas que aparecen cuando:

- La tabla tiene **múltiples claves candidatas**
- Un atributo que no es *clave primaria* actúa como *determinante*

Regla BCNF
Una tabla está en BCNF si:

Para toda dependencia funcional X → Y, el determinante X debe ser una clave candidata.

- **Si un atributo determina a otro, ese atributo debe ser clave.**

Ejemplo clásico que cumple 3FN pero NO BCNF

Reserva de salas

sala	profesor	hora

Dependencias:

- Un profesor solo puede usar **una sala asignada fija** → profesor → sala
- Una sala solo puede usarse en **una hora concreta** → sala → hora

Aquí:

- profesor no es clave
- Pero **determina a sala** → viola BCNF

Solución BCNF

Dividir en dos tablas para romper la dependencia problemática:
1 **profesor_sala(profesor, sala)**
2 **sala_hora(sala, hora)**
Ahora cada determinante es clave → **cumple BCNF**.

Forma Normal	Qué evita	Regla principal	Ejemplo
1FN	Grupos repetidos, listas	Cada celda un valor → atomicidad	Teléfonos en una sola celda
2FN	Dependencia parcial	Solo para claves compuestas. Todo atributo no clave depende de TODA la clave	nombre_cliente depende solo de id_pedido
3FN	Dependencias transitivas	Atributos no clave no deben depender de otros no clave	nombre_departamento depende de id_departamento
BCNF	Determinantes no clave	Si X → Y, X debe ser clave candidata	profesor → sala sin que profesor sea clave

CASO PRÁCTICO COMPLETO "Tienda online"

Tabla inicial:

id_pedido
id_cliente
nombre_cliente
dirección
id_producto
nombre_prod
precio
cantidad
total

Problemas:

- Cliente repetido, redundancia
- Producto repetido, inconsistencia de precios
- total se puede calcular, dato derivado

Aplicación 1FN → 2FN → 3FN

- 1FN → no hay listas
- 2FN → separar clientes
- 3FN → separar productos

Esquema final

cliente(id_cliente, nombre, direccion)
producto(id_producto, nombre_producto, precio_unitario)
pedido(id_pedido, id_cliente, fecha)
detalle_pedido(id_pedido, id_producto, cantidad)

Beneficios:

- Actualizar dirección solo una vez
- Precio del producto coherente
- Datos consistentes para facturación

Comparación entre formas normales

Característica	1FN	2FN	3FN
Atomicidad	si	si	si
Sin dependencias parciales	no	si	si
Sin dependencias transitivas	no	no	si
¿Evita redundancia?	Poco	Medio	Mucho
Aplicación típica	Comienzo	Claves compuestas	Modelos finales

ERRORES TÍPICOS

1. Crear una tabla enorme “para todo”.
2. No poner claves primarias.
3. Usar nombres ambiguos (tabla1, campoA…).
4. No identificar relaciones N:M.
5. Dejar datos repetidos porque “funciona en Excel”.
6. Confundir atributo multivalorado con atributo compuesto.
7. No separar entidades lógicas.

EJERCICIOS PARA PRACTICAR

Ejercicio 1: Transformación E-R

Dado el E-R:
Cliente (1) —— (N) Pedido
Pedido (1) —— (N) ProductoPedido —— (N) Producto

- Crear tablas
- Definir PK y FK
- Indicar si alguna relación es N:M

Ejercicio 2: Normalización

Dada la tabla:

matricula	alumno_nombre	asignatura_nombre	profesor_nombre	curso

1. ¿Qué problemas tiene?
2. Separar en 3FN.
3. Identificar PK y FK.

Caso 2: Universidad

Entidades: **Alumno**, **Profesor**, **Asignatura**, **Matrícula**.
Existe una relación **N:M** entre Alumno y Asignatura mediante la entidad **Matrícula**.

1. Identificación de entidades y atributos

Entidad	Atributos	Clave primaria
Alumno	id_alumno, nombre, apellidos, email	id_alumno
Profesor	id_profesor, nombre, apellidos, departamento	id_profesor
Asignatura	id_asignatura, nombre, créditos, id_profesor	id_asignatura
Matricula	id_matricula, id_alumno, id_asignatura, fecha_matricula, nota	id_matricula

2. Relaciones y cardinalidades

- **Alumno N:M Asignatura**, mediante **Matrícula**.
- **Profesor 1:N Asignatura** (un profesor puede impartir muchas asignaturas).

3. SQL resultante

```
CREATE TABLE profesores (
  id_profesor INT PRIMARY KEY,
  nombre VARCHAR(50),
  apellidos VARCHAR(50),
  departamento VARCHAR(50)
);

CREATE TABLE asignaturas (
  id_asignatura INT PRIMARY KEY,
  nombre VARCHAR(50),
  creditos INT,
  id_profesor INT,
  FOREIGN KEY (id_profesor) REFERENCES profesores(id_profesor)
);

CREATE TABLE alumnos (
  id_alumno INT PRIMARY KEY,
  nombre VARCHAR(50),
  apellidos VARCHAR(50),
  email VARCHAR(100)
);

CREATE TABLE matriculas (
  id_matricula INT PRIMARY KEY,
  id_alumno INT,
  id_asignatura INT,
  fecha_matricula DATE,
  nota DECIMAL(4,2),
```

```
  FOREIGN KEY (id_alumno) REFERENCES alumnos(id_alumno),
  FOREIGN KEY (id_asignatura) REFERENCES asignaturas(id_asignatura)
);
```

4. Observaciones

- La tabla **matriculas** funciona como **entidad puente** que resuelve la relación N:M.
- Los atributos fecha_matricula y nota pertenecen a la relación, no a las entidades principales.
- Esta estructura facilita consultas como:
- SELECT a.nombre, s.nombre, m.nota
- FROM matriculas m
- JOIN alumnos a ON m.id_alumno = a.id_alumno
- JOIN asignaturas s ON m.id_asignatura = s.id_asignatura;

Caso 3: Tienda Online

Entidades: **Cliente**, **Pedido**, **Producto**, **Categoría**.
Relaciones:

- Un cliente puede tener varios pedidos (1:N).
- Un pedido puede contener varios productos (N:M).
- Cada producto pertenece a una categoría (N:1).

1. Identificación de entidades y atributos

Entidad	**Atributos**	**Clave primaria**
Cliente	id_cliente, nombre, email, direccion	id_cliente
Pedido	id_pedido, fecha, total, id_cliente	id_pedido
Producto	id_producto, nombre, precio, id_categoria	id_producto
Categoría	id_categoria, nombre_categoria	id_categoria
DetallePedido *(tabla intermedia)*	id_detalle, id_pedido, id_producto, cantidad, subtotal	id_detalle

2. Relaciones y cardinalidades

- Cliente 1:N Pedido
- Pedido N:M Producto, se resuelve con **DetallePedido**
- Producto N:1 Categoría

3. SQL resultante

```
CREATE TABLE categorias (
  id_categoria INT PRIMARY KEY,
  nombre_categoria VARCHAR(50)
);
```

```
CREATE TABLE productos (
  id_producto INT PRIMARY KEY,
  nombre VARCHAR(50),
  precio DECIMAL(10,2),
  id_categoria INT,
  FOREIGN KEY (id_categoria) REFERENCES categorias(id_categoria)
);

CREATE TABLE clientes (
  id_cliente INT PRIMARY KEY,
  nombre VARCHAR(50),
  email VARCHAR(100),
  direccion VARCHAR(100)
);

CREATE TABLE pedidos (
  id_pedido INT PRIMARY KEY,
  fecha DATE,
  total DECIMAL(10,2),
  id_cliente INT,
  FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
);

CREATE TABLE detalle_pedido (
  id_detalle INT PRIMARY KEY,
  id_pedido INT,
  id_producto INT,
  cantidad INT,
  subtotal DECIMAL(10,2),
  FOREIGN KEY (id_pedido) REFERENCES pedidos(id_pedido),
  FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
);
```