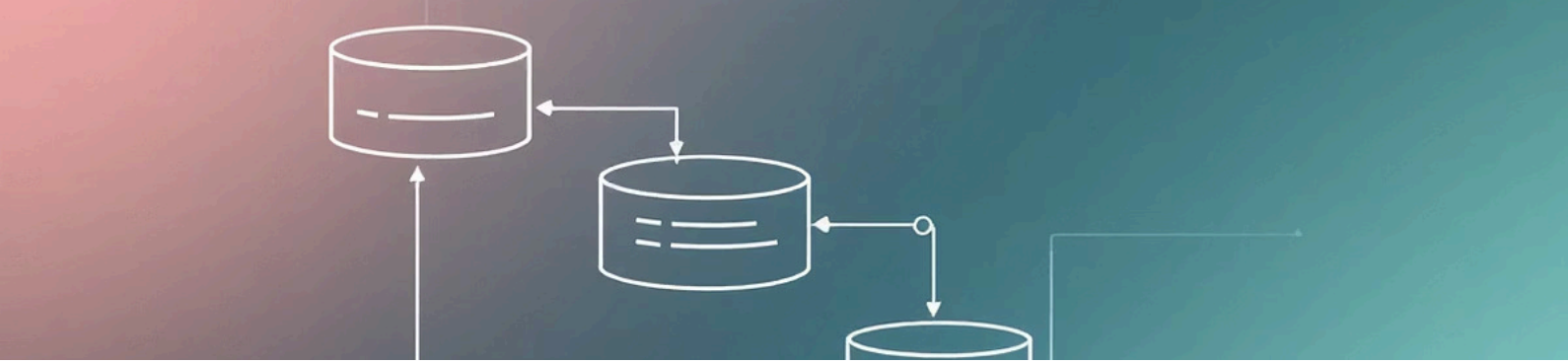
PROMETEO

Unidad 4:
Normalización

Bases de datos

Técnico Superior de DAM / DAW



Sesión 12 — Introducción a la Normalización: 1FN y 2FN

La normalización es el proceso sistemático mediante el cual se organiza una base de datos relacional para reducir redundancia, mejorar la integridad y facilitar su mantenimiento. En esencia, se trata de aplicar un conjunto de reglas que aseguran que los datos se almacenen una sola vez y de forma coherente, evitando errores y duplicidades.

Cuando una base de datos no está normalizada, aparecen problemas como:

Redundancia

la misma información se repite en varias filas.

Anomalías de actualización

modificar un dato en un registro no actualiza automáticamente otros registros que contienen el mismo valor.

Anomalías de inserción

no se pueden añadir datos nuevos sin incluir información innecesaria.

Anomalías de eliminación

al borrar un registro se pierden datos que deberían conservarse.

La normalización busca eliminar estas anomalías dividiendo las tablas grandes y complejas en estructuras más pequeñas, interrelacionadas mediante claves. Cada tabla debe representar una entidad o relación concreta, y los vínculos entre ellas deben reflejar dependencias lógicas, no repeticiones.

El proceso se divide en formas normales (FN), que representan niveles progresivos de refinamiento:

- La **Primera Forma Normal (1FN)** elimina los grupos repetidos dentro de una tabla, asegurando que cada campo contenga valores atómicos (indivisibles).
- La **Segunda Forma Normal (2FN)** elimina dependencias parciales, es decir, aquellas en las que un atributo depende de una parte de una clave compuesta, y no de toda ella.

Primera Forma Normal (1FN)

Para que una tabla cumpla la 1FN:

- Cada celda debe contener un único valor, no listas ni conjuntos.
- Cada registro debe ser único e identificable por una clave primaria.
- No deben existir columnas repetidas que indiquen grupos de datos (por ejemplo, "Teléfono1", "Teléfono2"...).

Cumplir la 1FN convierte la tabla en una estructura ordenada donde cada dato tiene su propio espacio y relación clara con la entidad que describe.

Segunda Forma Normal (2FN)

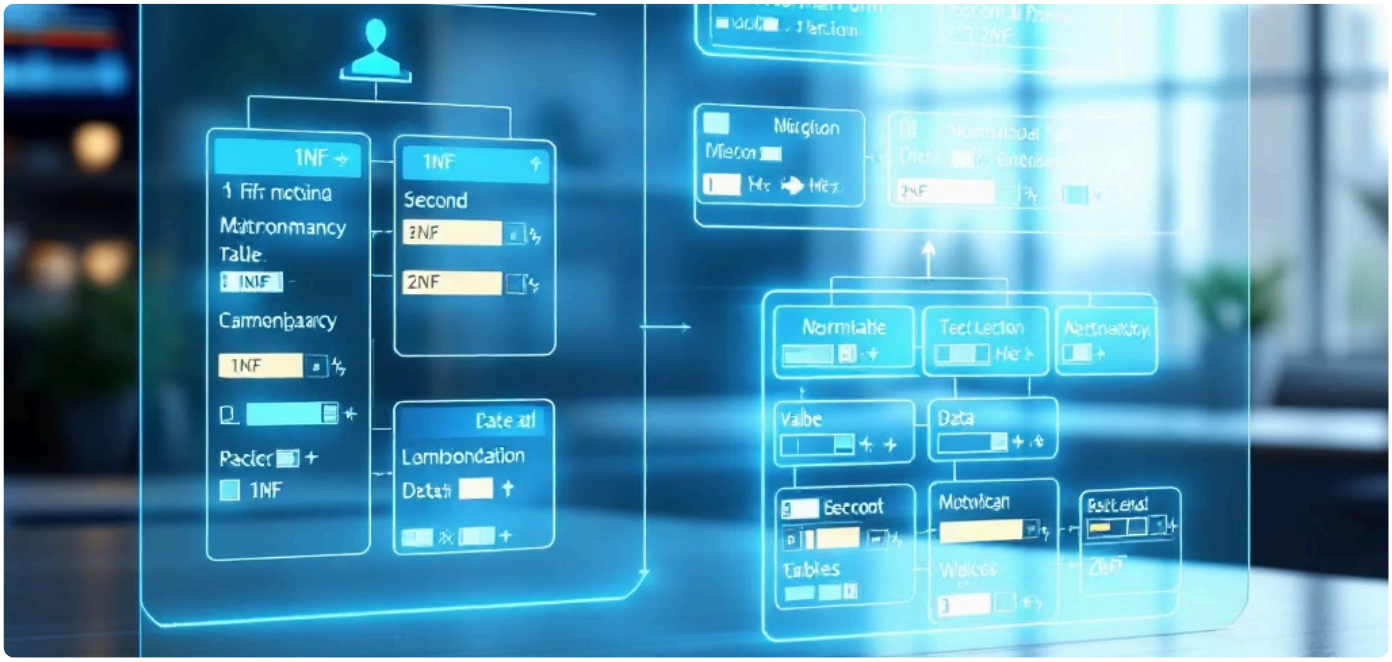
Una tabla está en 2FN si:

- Cumple la 1FN.
- Todos los atributos no clave dependen de toda la clave primaria (y no solo de una parte).

La 2FN es especialmente relevante cuando la tabla tiene clave primaria compuesta. Si un campo depende solo de una parte de esa clave, debe separarse en una nueva tabla. Esto evita duplicaciones y asegura una estructura más lógica.

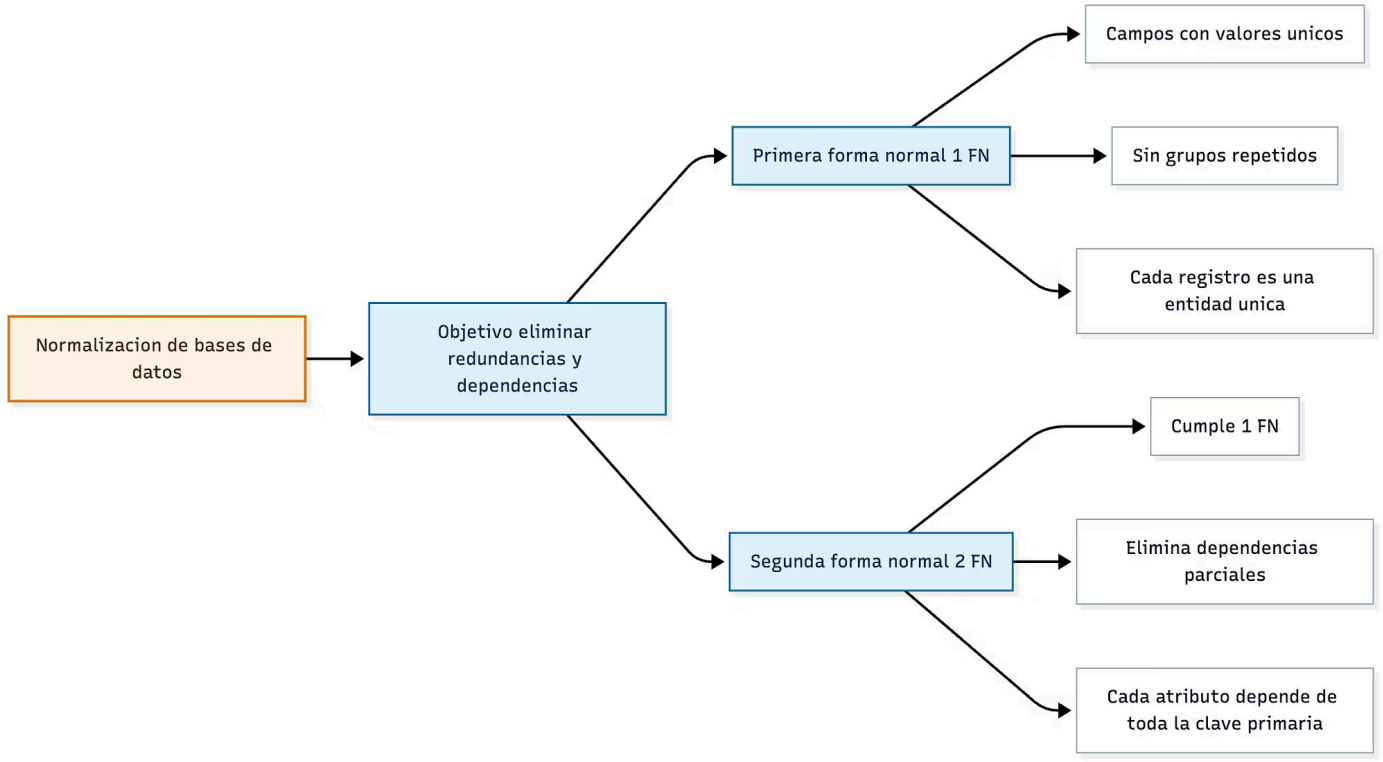
> **Ejemplo conceptual:** Una tabla con clave compuesta (PedidoID, ProductoID) que además incluye "NombreCliente" no cumple 2FN, porque "NombreCliente" depende de "PedidoID" (el pedido), no de "ProductoID". En este caso, se debe separar en una tabla de clientes.

La normalización, por tanto, no es un fin teórico: es una práctica esencial en todo diseño de base de datos profesional. Permite que los sistemas sean más fáciles de mantener, escalar y auditar, y evita pérdidas de información en aplicaciones empresariales complejas.



Esquema Visual

Esquema conceptual en formato Mermaid:



Descripción del esquema:

El diagrama comienza with el nodo principal "Normalización de Bases de Datos", cuyo propósito es eliminar redundancias y dependencias. Desde él parten dos ramas que representan las primeras fases del proceso:

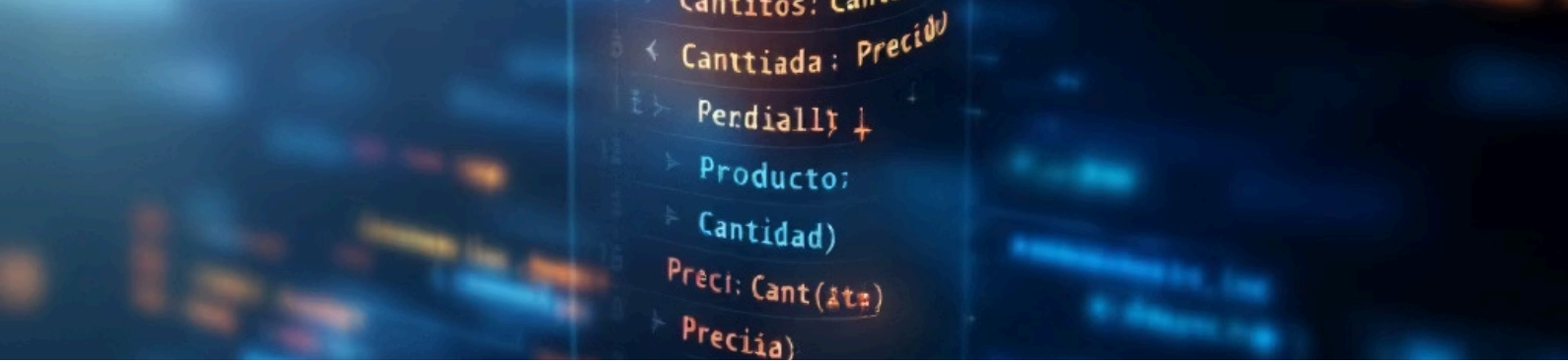
1ª Forma Normal (1FN)

define la estructura básica donde cada campo contiene un valor indivisible y no existen grupos repetidos.

2ª Forma Normal (2FN)

parte de la base de la 1FN y refina el diseño eliminando dependencias parciales, garantizando que cada atributo dependa completamente de la clave primaria.

Ambas formas conforman el bloque fundamental del diseño relacional, sobre el que se construirán niveles más avanzados (3FN, BCNF, etc.).



Caso de Estudio: Tienda Online "TechNow" y la Normalización Práctica

Contexto

La empresa TechNow, una tienda de productos tecnológicos, decide almacenar sus pedidos en una única tabla llamada Pedidos. En ella se registran los datos del cliente, los productos comprados y la información de cada pedido.

PedidoID	Cliente	Productos	Cantidad	Precio
1001	Juan	Ratón, Teclado	2,1	10,20

A primera vista parece funcional, pero presenta graves problemas de diseño:

- **Violación de la 1FN:** el campo "Productos" contiene dos valores ("Ratón, Teclado").
- **Dificultad de actualización:** si el precio del "Teclado" cambia, habría que editar manualmente cada fila donde aparezca.
- **Problemas de búsqueda:** no se pueden filtrar fácilmente los pedidos por producto, ya que están en una sola celda.

Estrategia de Normalización

01

Aplicación de la 1FN

Se separan los valores múltiples en filas independientes, garantizando que cada celda tenga un solo valor atómico.

02

Aplicación de la 2FN

Se identifica que el campo "Cliente" depende solo de PedidoID, no de Producto. Se separa la información en tres tablas especializadas.

Nueva tabla Pedidos:

PedidoID	Cliente	Producto	Cantidad	Precio
1001	Juan	Ratón	2	10
1001	Juan	Teclado	1	20

Ahora los datos cumplen la Primera Forma Normal, ya que no hay grupos repetidos ni campos con listas de valores.

Aplicación de la 2FN: La clave primaria es compuesta (PedidoID, Producto). Sin embargo, el campo "Cliente" depende solo de PedidoID, no de Producto. Esto viola la 2FN. Se decide separar la información en tres tablas:

- Clientes (ClienteID, NombreCliente)
- Pedidos (PedidoID, ClienteID, FechaPedido)
- DetallePedidos (PedidoID, Producto, Cantidad, Precio)

Este rediseño elimina las dependencias parciales y organiza los datos según su naturaleza.

Resultado

Gracias a la normalización:

Se eliminan redundancias

el cliente "Juan" ya no se repite en cada línea de producto.

Se facilita el mantenimiento

si cambia el nombre del cliente, se actualiza en una sola tabla.

Se mejora el rendimiento

de las consultas y se evita la pérdida de datos.

TechNow pasa de tener una estructura frágil y redundante a un modelo relacional profesional, fácil de ampliar y mantener.

Herramientas y Consejos

Consejos Prácticos

Identifica siempre las claves primarias antes de normalizar

No se puede eliminar una dependencia si no se sabe de qué depende cada campo.

Visualiza las relaciones mediante diagramas ERD

Un esquema entidad-relación (ERD) ayuda a detectar dependencias ocultas entre atributos.

Evita la sobre-normalización

Excederse fragmentando tablas puede dificultar las consultas y degradar el rendimiento.

Documenta cada paso

Mantén un registro de cómo evoluciona el modelo con cada forma normal aplicada.

Revisa dependencias funcionales con ejemplos reales

Intenta modificar un dato y observa si afecta indebidamente a otros campos.

Herramientas Profesionales

MySQL Workbench

permite diseñar esquemas, definir relaciones y generar SQL automáticamente.

dbdiagram.io

herramienta online para dibujar modelos relacionales simples y exportarlos a código SQL.

Lucidchart / Draw.io

ideales para representar diagramas ERD visuales colaborativos.

DBeaver

gestor universal de bases de datos que facilita pruebas y consultas en diferentes motores.

ERDPlus

útil para representar dependencias funcionales y practicar las formas normales.

Estas herramientas no solo facilitan la normalización, sino que ayudan a mantener la trazabilidad del diseño y detectar posibles dependencias parciales o redundancias antes de que se conviertan en errores costosos.

Mitos y Realidades

❌ Mito 1

"La normalización solo se usa en entornos académicos."

→ **FALSO.** La normalización es una práctica industrial esencial. Las bases de datos de banca, logística, sanidad o comercio electrónico se diseñan siguiendo formas normales para asegurar integridad y escalabilidad. Sin ella, los datos se vuelven inconsistentes y las consultas se ralentizan drásticamente.

❌ Mito 2

"Cuantas más formas normales, mejor."

→ **FALSO.** La normalización tiene un límite práctico. En entornos de producción, se suele trabajar hasta la 3FN o BCNF. Más allá, puede aumentar la complejidad de las consultas y reducir el rendimiento. En ocasiones, se aplican desnormalizaciones controladas para mejorar tiempos de respuesta, siempre sin comprometer la integridad de los datos.

Resumen final

- **Normalización:** proceso para eliminar redundancias y dependencias.
- **1FN:** sin grupos repetidos, cada campo contiene un único valor.
- **2FN:** cumple 1FN y elimina dependencias parciales en claves compuestas.
- **Claves primarias claras y dependencias bien definidas** = base sólida para la integridad de datos.
- **Equilibrio:** normalizar lo necesario, sin sacrificar rendimiento.

Sesión 13 — Tercera Forma Normal (3FN) y Forma de Boyce-Codd (BCNF)

En el diseño de bases de datos, llegar a la Tercera Forma Normal (3FN) y, en algunos casos, a la Forma Normal de Boyce-Codd (BCNF) representa alcanzar un nivel de madurez estructural que garantiza la integridad lógica y la consistencia de los datos. Estas formas avanzadas corrigen un problema muy frecuente incluso en modelos que ya cumplen la 2FN: las **dependencias transitivas**.

Una dependencia funcional se produce cuando el valor de un atributo determina el valor de otro. Por ejemplo, si en una tabla de cursos el campo Profesor depende del Curso, y el Curso depende del Alumno, entonces Profesor depende indirectamente del Alumno. Esa relación encadenada es lo que llamamos dependencia transitiva. Este tipo de relación genera redundancia y puede provocar anomalías de actualización, ya que un cambio en un profesor obliga a modificar múltiples filas.

Tercera Forma Normal (3FN)

Una tabla está en Tercera Forma Normal si cumple las siguientes condiciones:

- Cumple la Segunda Forma Normal (2FN).
- Ningún atributo no clave depende de otro atributo no clave.
- Todos los atributos dependen únicamente de la clave primaria.

En otras palabras, la 3FN garantiza que los datos se organizan de modo que cada tabla describa una sola entidad o relación, y cada atributo aporte información exclusivamente sobre ella. Si un campo depende de otro campo que no es clave, la tabla debe dividirse.

Forma Normal de Boyce-Codd (BCNF)

La Forma de Boyce-Codd, propuesta por Raymond Boyce y Edgar F. Codd, lleva la normalización un paso más allá. Una tabla está en BCNF si para cada dependencia funcional $X \rightarrow Y$, el determinante X es una clave candidata.

Esto significa que no puede existir ningún atributo determinante que no sea clave, eliminando así dependencias residuales más sutiles que la 3FN a veces no detecta.

En términos prácticos, la BCNF corrige aquellos casos en los que una tabla cumple la 3FN pero aún presenta dependencias funcionales entre atributos candidatos.

Ejemplo conceptual: Supongamos una tabla Empleados con las columnas: (EmpleadoID, Nombre, Departamento, JefeDepartamento)

Aquí, JefeDepartamento depende del Departamento, no del EmpleadoID. Por tanto, esta tabla viola la 3FN.

Beneficios Profesionales

Coherencia total

evita la propagación de errores cuando cambian valores dependientes.

Facilidad de mantenimiento

las modificaciones se centralizan.

Escalabilidad estructural

las tablas son modulares, lo que facilita añadir nuevos campos sin alterar el modelo.

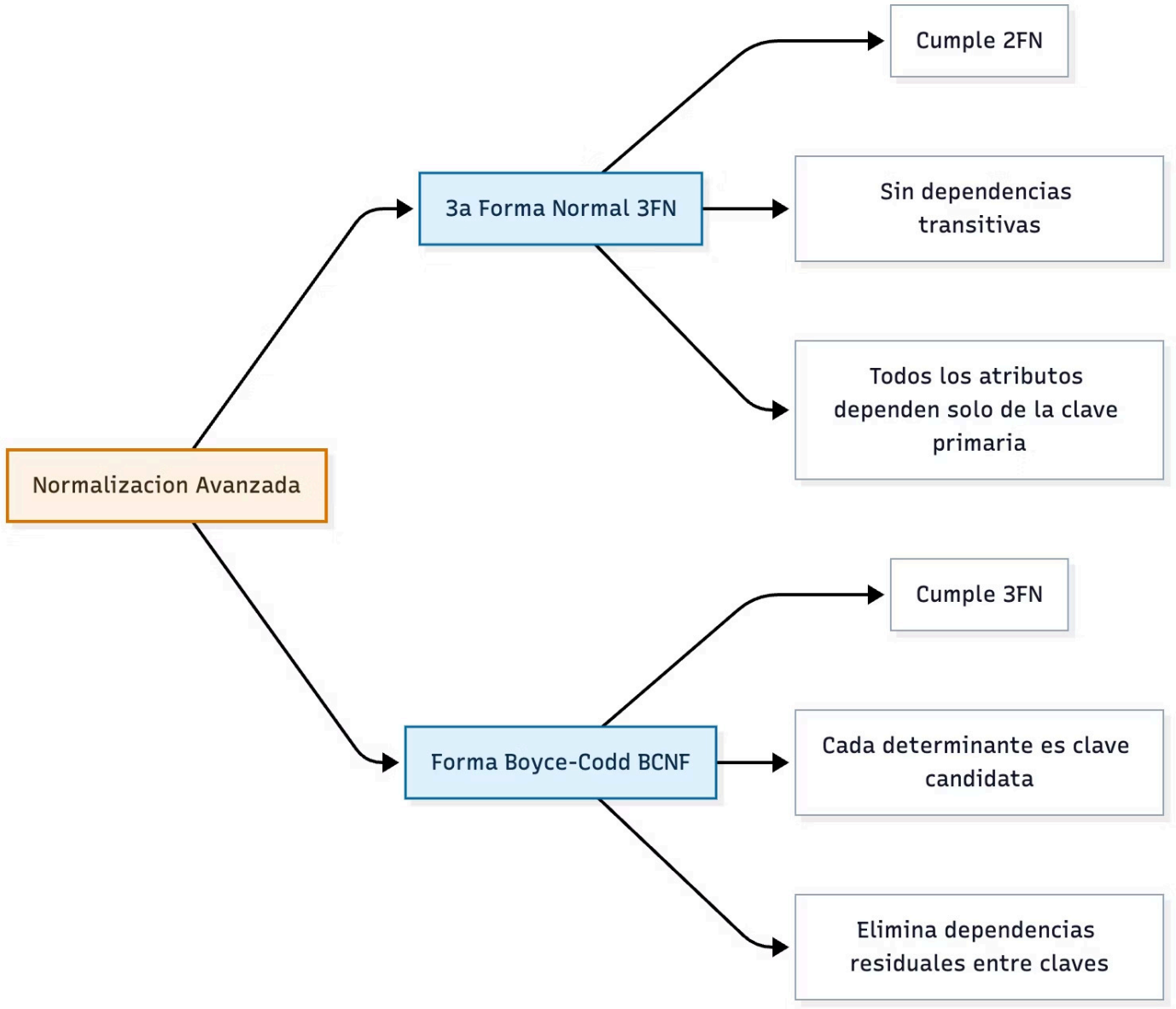
Integridad referencial más sólida

las relaciones entre entidades se vuelven lógicas, no circunstanciales.

En entornos empresariales —desde sistemas de facturación hasta plataformas de e-learning o CRM— la 3FN y la BCNF aseguran que las bases de datos crezcan ordenadamente sin comprometer la calidad del dato, lo que es clave para analítica y toma de decisiones.

Esquema Visual

Esquema conceptual en formato Mermaid:



Descripción del esquema:

El nodo principal "Normalización Avanzada" se divide en dos niveles. La 3FN consolida la integridad eliminando dependencias transitivas, asegurando que todos los atributos dependan directamente de la clave principal. La BCNF, por su parte, añade una capa de rigor adicional: toda dependencia funcional debe tener como determinante una clave candidata.

Este nivel evita redundancias más complejas, especialmente en tablas con múltiples claves posibles o relaciones no lineales. Visualmente, la 3FN representa la estabilidad estructural; la BCNF, la pureza lógica del modelo relacional.

Caso de Estudio: Academia GlobalTech – De la 2FN a la BCNF

Contexto

La academia GlobalTech ofrece cursos online de programación. Inicialmente, almacenaban la información de sus matrículas en una tabla única llamada Cursos_Alumnos:

Alumno	Curso	Profesor
Ana	SQL	Marta
Juan	Python	Luis
Ana	Python	Luis

A primera vista parece eficiente, pero presenta dependencias transitivas:

- El Profesor depende del Curso (no del Alumno).
- Por tanto, la tabla viola la 3FN, ya que un atributo no clave (Profesor) depende de otro atributo no clave (Curso).

Estrategia de Normalización

Aplicar la 3FN

Se divide la información en dos tablas:

- Cursos(Curso, Profesor)
- Matriculas(Alumno, Curso)

Con esto, si el profesor de un curso cambia, basta modificarlo en la tabla Cursos una sola vez. Se eliminan las redundancias y las inconsistencias potenciales.

Revisión para BCNF

Tras aplicar la 3FN, surge una nueva situación. En la academia, cada profesor puede impartir varios cursos, pero nunca dos profesores pueden enseñar el mismo curso. Además, los cursos dependen del profesor.

Existe una dependencia funcional Profesor → Curso. Sin embargo, la clave de la tabla Cursos es Curso, no Profesor. Esto viola la BCNF, ya que el determinante (Profesor) no es clave candidata.

Cumplir la BCNF

Para cumplir la BCNF, se crea una nueva estructura:

- Profesores(Profesor, Curso)
- Cursos(Curso)
- Matriculas(Alumno, Curso)

Ahora cada dependencia funcional se asocia a una clave candidata, eliminando completamente redundancias.

Resultado

El modelo final presenta varias ventajas:

Integridad total

los cambios en el profesorado no afectan registros de alumnos.

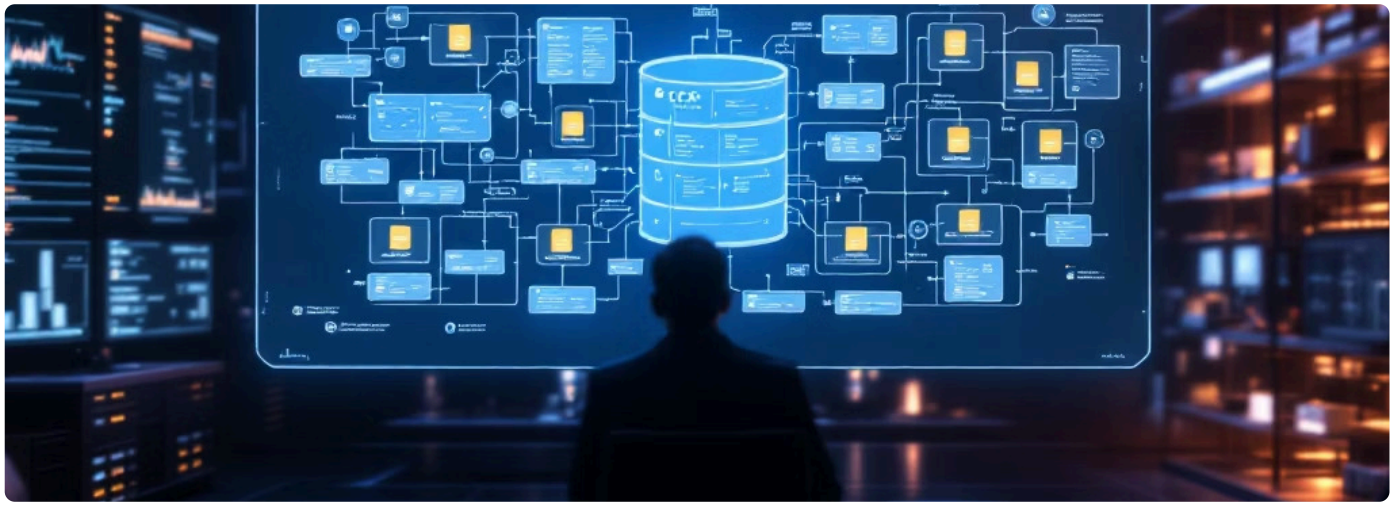
Reducción de errores

al actualizar un curso, el cambio se propaga de manera controlada.

Facilidad de expansión

si la academia añade un nuevo módulo o programa, solo se agregan nuevas filas, no nuevas columnas.

GlobalTech consiguió una base de datos limpia, escalable y lista para analítica avanzada (seguimiento de rendimiento por curso, métricas de asistencia o rotación de profesores).



Herramientas y Consejos

Consejos Profesionales

Usa diagramas de dependencias funcionales

Visualizar qué atributos determinan a otros ayuda a detectar dependencias transitivas y errores lógicos.

Comprueba dependencias con ejemplos reales

Antes de dividir tablas, prueba si modificar un campo afecta otros registros que no deberían cambiar.

No asumas que cumplir 2FN implica estar en 3FN

Es común encontrar dependencias transitivas ocultas incluso después de eliminar las parciales.

Aplica BCNF solo cuando sea necesario

En sistemas de alta carga (por ejemplo, comercio electrónico) una normalización extrema puede afectar el rendimiento. Prioriza la coherencia, pero evalúa el impacto en la velocidad de consultas.

Documenta el modelo lógico

Mantén un registro con las dependencias detectadas y las decisiones de diseño: es vital para mantenimiento y auditorías.

Herramientas Recomendadas

ERDPlus

ideal para dibujar dependencias funcionales y formas normales.

MySQL Workbench

permite definir claves candidatas y probar restricciones.

SQL Fiddle

entorno online para validar estructuras normalizadas y ejecutar consultas de prueba.

DbVisualizer o DBeaver

útiles para comparar esquemas antes y después de la normalización.

Lucidchart / Draw.io

excelentes para visualizar relaciones de tablas en modelos complejos.

Estas herramientas son ampliamente utilizadas por analistas de datos, arquitectos de sistemas y administradores de bases de datos en entornos empresariales reales.

Mitos y Realidades

❌ Mito 1

"Llegar a la 3FN garantiza que no habrá más anomalías."

→ **FALSO.** La 3FN elimina dependencias transitivas, pero no siempre corrige dependencias funcionales entre claves candidatas. En esos casos, la BCNF es necesaria. Además, pueden existir otras formas superiores (4FN, 5FN) destinadas a resolver dependencias multivaluadas o uniones más complejas.

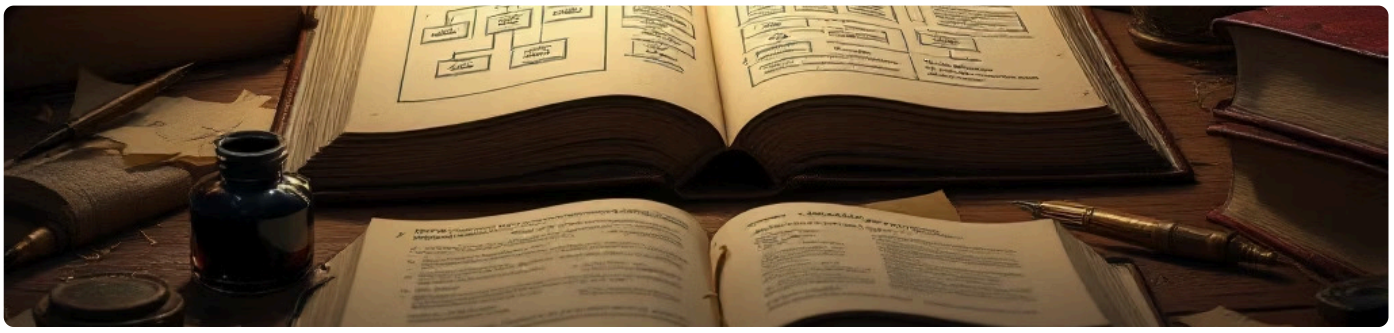
❌ Mito 2

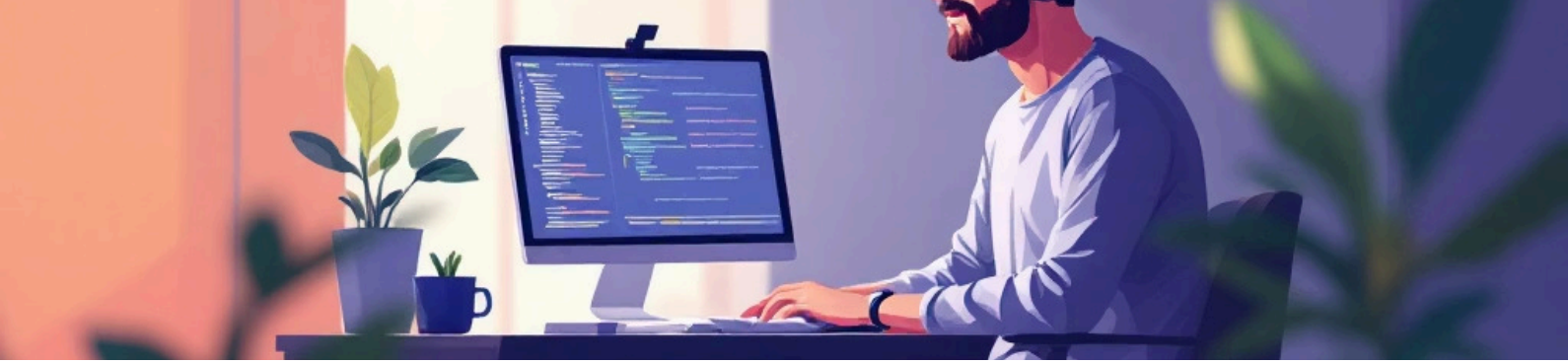
"La BCNF siempre es la mejor opción."

→ **FALSO.** Aunque la BCNF garantiza una estructura más pura, en sistemas de producción puede implicar más uniones (JOINs) en las consultas, lo que ralentiza el rendimiento. Por eso muchas bases se diseñan hasta la 3FN y luego se aplican desnormalizaciones controladas donde se prioriza la eficiencia operativa sin comprometer la integridad.

Resumen final

- **3FN:** elimina dependencias transitivas; todos los atributos dependen solo de la clave primaria.
- **BCNF:** cada determinante es clave candidata; más estricta que la 3FN.
- **Ventaja:** mejora la integridad, evita redundancias y simplifica actualizaciones.
- **Aplicación práctica:** separar tablas cuando un atributo depende de otro no clave.
- **Equilibrio clave:** normalizar lo suficiente para garantizar consistencia, sin sacrificar rendimiento.





Sesión 14 — Otras Formas Normales (4FN, 5FN) y Cuándo Aplicarlas

Hasta la Tercera Forma Normal (3FN) o incluso la Forma de Boyce-Codd (BCNF), una base de datos relacional se considera correctamente diseñada: las dependencias funcionales están controladas y no existen redundancias evidentes. Sin embargo, en entornos empresariales complejos —como sistemas ERP, plataformas de gestión logística o catálogos de productos multinivel— pueden aparecer dependencias más sutiles, imposibles de detectar con las tres primeras formas.

En estos casos entran en juego las formas normales avanzadas: la **Cuarta (4FN)** y la **Quinta (5FN)**.

Cuarta Forma Normal (4FN)

La Cuarta Forma Normal (4FN) se centra en eliminar **dependencias multivaluadas**. Estas ocurren cuando un registro está relacionado con varios conjuntos de valores que son independientes entre sí.

Por ejemplo, un profesor que puede enseñar varios cursos y hablar varios idiomas: los idiomas no dependen de los cursos, ni los cursos de los idiomas. Sin embargo, si ambos aparecen en la misma tabla, se generan combinaciones innecesarias que multiplican los registros sin aportar información nueva.

La 4FN exige que cada tabla describa una sola relación entre las entidades.

Quinta Forma Normal (5FN)

Por su parte, la Quinta Forma Normal (5FN), también conocida como Forma Normal de Proyección-Unión (PJ/NF), busca eliminar redundancias que aparecen cuando los datos pueden reconstruirse a partir de múltiples relaciones parciales.

En otras palabras, evita duplicidades que surgen al recomponer una tabla a partir de sus descomposiciones. Este nivel de normalización es útil cuando los datos provienen de combinaciones complejas entre varias entidades, como (Proveedor, Producto, Distribuidor), donde una misma combinación puede aparecer más de una vez si no se gestionan correctamente las dependencias de unión.

En la práctica, pocas bases de datos operativas llegan hasta la 5FN, porque el esfuerzo de diseño y el coste de procesamiento pueden ser mayores que los beneficios. Sin embargo, en sistemas de alta integridad y consistencia, como los que gestionan transacciones financieras o catálogos globales, su aplicación resulta crucial para evitar redundancias y mantener la coherencia de los datos a largo plazo.

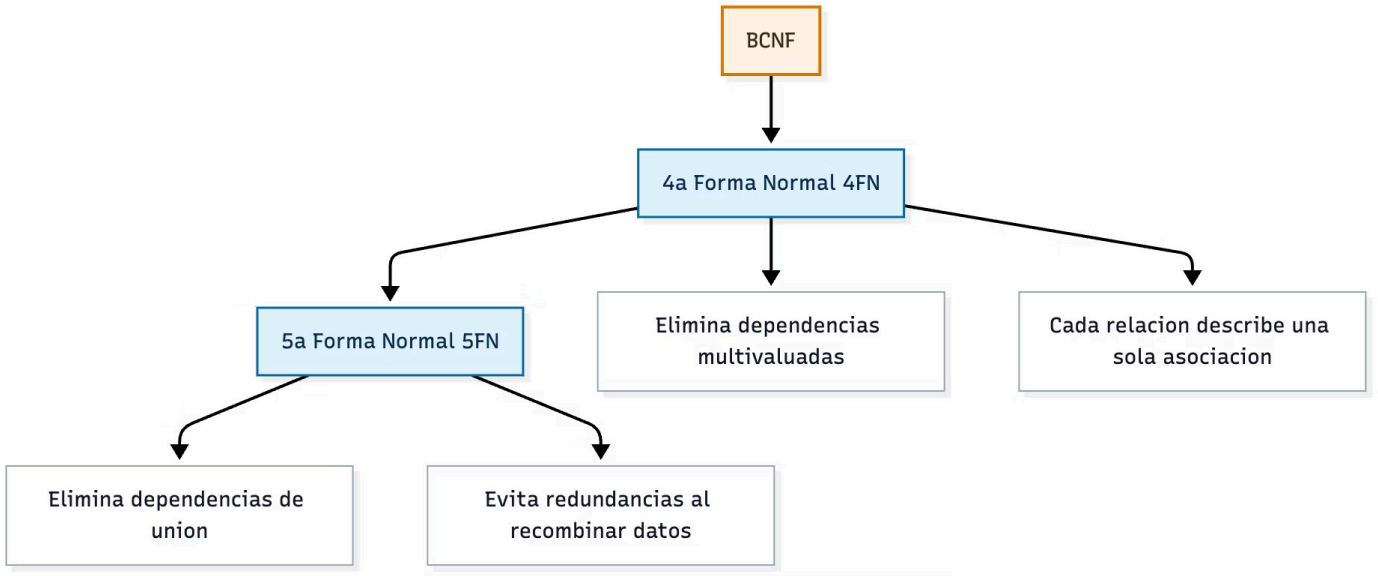
En resumen:

4FN	5FN
elimina dependencias multivaluadas (varios conjuntos independientes).	elimina redundancias derivadas de combinaciones múltiples (dependencias de unión).

Ambas aseguran la pureza del modelo lógico de datos en entornos donde la complejidad relacional es alta.

Esquema Visual

El siguiente esquema muestra la progresión hacia las formas normales avanzadas, destacando cómo se construyen sobre las anteriores y qué problema corrige cada una.



Explicación del esquema:

01

La BCNF es el punto de partida

garantiza que todos los determinantes sean claves candidatas.

02

La 4FN amplía ese control

eliminando las dependencias multivaluadas, asegurando que cada relación represente una única asociación.

03

La 5FN lleva la normalización al máximo nivel

evitando redundancias al unir tablas derivadas de descomposiciones complejas.

Este modelo visual puede leerse como una escalera de depuración: cada nueva forma normal elimina un tipo más específico de dependencia, reduciendo progresivamente el riesgo de duplicidad o inconsistencia en los datos.

Caso de Estudio: Dependencias Multivaluadas en una Empresa de Formación

Contexto

Imagina una empresa de formación que gestiona una base de datos con información sobre los profesores, los cursos que imparten y los idiomas que dominan. La estructura inicial de la tabla es la siguiente:

Profesor	Curso	Idioma
Marta	SQL	Inglés
Marta	Python	Inglés
Marta	SQL	Español
Luis	Redes	Inglés

En este diseño, los campos Curso e Idioma no están relacionados entre sí. Marta puede impartir varios cursos (SQL, Python) y hablar varios idiomas (Inglés, Español), pero estas asociaciones no son dependientes. El hecho de que Marta hable Español no tiene relación directa con el curso que enseña.

Esta situación genera dependencias multivaluadas, violando la 4FN.

Estrategia

Para corregir la dependencia, se dividen los datos en dos tablas independientes, separando las relaciones:

- Profesores_Cursos(Profesor, Curso)

- Profesores_Idiomas(Profesor, Idioma)

Ahora, cada tabla representa una sola asociación y no existe redundancia. Marta aparecerá dos veces en cada tabla, pero nunca se repetirán combinaciones innecesarias de curso e idioma.

Si el sistema creciera y añadiera una tercera entidad, por ejemplo, Centro (lugar donde imparte el curso), podrían aparecer redundancias adicionales. Si los cursos se impartieran en distintos centros y los idiomas también variaran según la localización, el modelo podría requerir pasar a 5FN para evitar duplicidades al recomponer las relaciones entre (Profesor, Curso, Centro).

Resultado

Eliminación de redundancias

cada relación se define con precisión.

Mayor consistencia

se evitan combinaciones artificiales.

Escalabilidad

permite incorporar nuevas entidades sin afectar la integridad de los datos existentes.

En bases reales, este tipo de estructura es común en sistemas educativos, ERP de recursos humanos o plataformas multilingües, donde un mismo registro se relaciona con múltiples dimensiones independientes.

Herramientas y Consejos

Consejos prácticos

Aplica la 4FN solo cuando sea necesario

Si detectas varios conjuntos de atributos independientes en una misma tabla (como cursos e idiomas), es momento de aplicar la Cuarta Forma Normal.

Identifica dependencias multivaluadas mediante ejemplos concretos

Si una combinación de atributos genera más registros sin añadir información, existe una dependencia multivaluada.

Documenta todas las dependencias detectadas

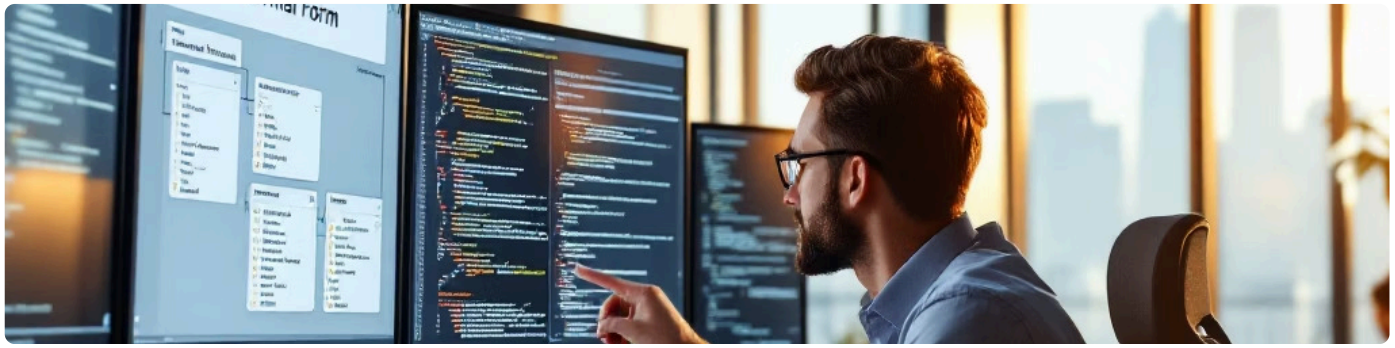
Mantén un registro de las relaciones entre atributos y tablas; esto te permitirá justificar cada paso de normalización.

Verifica con consultas JOIN

Comprueba que, tras descomponer las tablas, los datos puedan reconstruirse correctamente sin pérdida ni duplicación.

Evalúa el costo-beneficio

En bases de datos pequeñas o con poca complejidad, llegar hasta 5FN puede no ser necesario. En cambio, en entornos empresariales distribuidos, es una práctica esencial.



Herramientas recomendadas

ERDPlus

permite crear y analizar diagramas entidad-relación y detectar dependencias multivaluadas visualmente.

MySQL Workbench o DBeaver

útiles para realizar pruebas de normalización y validar integridad mediante consultas SQL.

DBDesigner

facilita el diseño lógico del modelo y la visualización de dependencias entre tablas.

Estas herramientas te permiten pasar de la teoría a la práctica, comprobando en tiempo real cómo afectan las formas normales a la estructura y comportamiento de los datos.

Mitos y Realidades

❌ Mito

"Las formas 4FN y 5FN no se utilizan en proyectos reales."

→ **FALSO.** Aunque no se aplican en la mayoría de las bases pequeñas, son fundamentales en sistemas empresariales complejos, donde múltiples relaciones pueden generar inconsistencias difíciles de detectar. ERP, CRM globales o sistemas de catálogo multinivel dependen de estas formas para garantizar la integridad total de los datos.

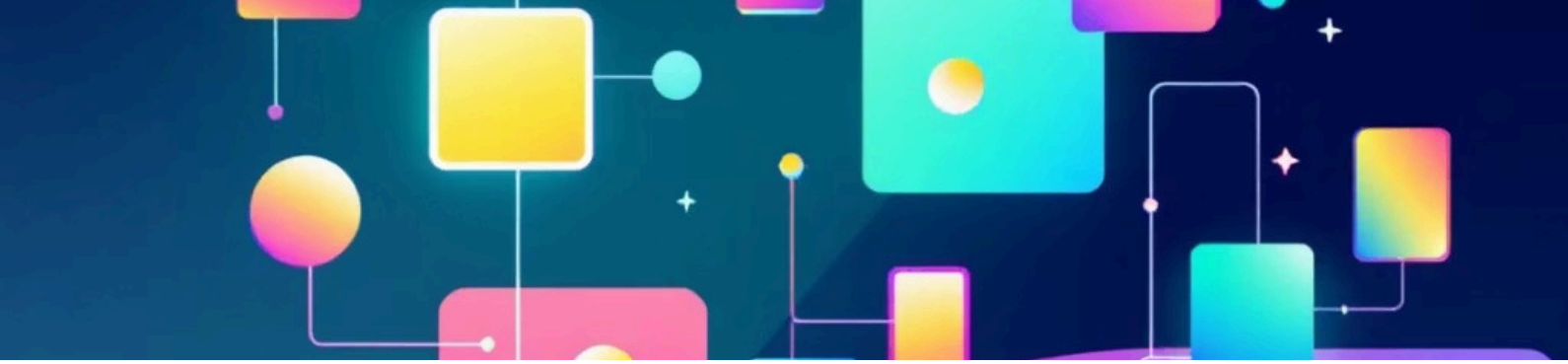
❌ Mito

"Normalizar hasta la 5FN garantiza siempre el mejor rendimiento."

→ **FALSO.** La 5FN mejora la coherencia, pero no siempre el rendimiento. En sistemas de lectura intensiva (como paneles analíticos), una base demasiado fragmentada puede requerir demasiadas operaciones JOIN. Por eso, en producción, a menudo se combinan partes normalizadas (para integridad) con partes desnormalizadas (para rendimiento).

Resumen final

- **4FN:** elimina dependencias multivaluadas.
- **5FN:** elimina redundancias derivadas de dependencias de unión.
- Se aplican en sistemas con datos muy interrelacionados (ERP, catálogos, CRM).
- No siempre son necesarias, pero garantizan integridad total en modelos complejos.
- Cada forma normal mejora la anterior eliminando un tipo más específico de dependencia.



Sesión 15 — Práctica de Normalización Completa sobre un Caso Real

Hasta ahora has aprendido que la normalización es el proceso mediante el cual una base de datos se reorganiza para reducir la redundancia y garantizar la integridad de los datos. Las distintas formas normales (de la Primera a la Quinta) no son solo niveles teóricos, sino pasos concretos que transforman un conjunto de datos desordenados en un modelo estructurado, coherente y escalable.

En la práctica profesional, la normalización se utiliza desde la fase de diseño lógico de un sistema hasta el mantenimiento continuo de una base de datos empresarial. Cuando se aplica correctamente, permite que diferentes equipos —desarrolladores, analistas y responsables de datos— trabajen sobre una base coherente, donde cada tabla cumple una función específica y las relaciones entre entidades están perfectamente definidas.

El objetivo de esta sesión es aplicar de manera integral todas las formas normales (1FN, 2FN, 3FN, BCNF, 4FN y 5FN) sobre un caso real, para que puedas comprender cómo evolucionan los datos a través del proceso de depuración. Más que memorizar definiciones, se trata de que entiendas cómo cada forma corrige un tipo concreto de error estructural.

Las ventajas principales de un modelo normalizado son:

Evita redundancias

un dato se almacena solo una vez, minimizando errores de actualización.

Aumenta la consistencia

cualquier cambio en un registro se propaga de manera controlada.

Mejora la integridad referencial

las claves primarias y foráneas mantienen las relaciones entre tablas.

Optimiza el mantenimiento

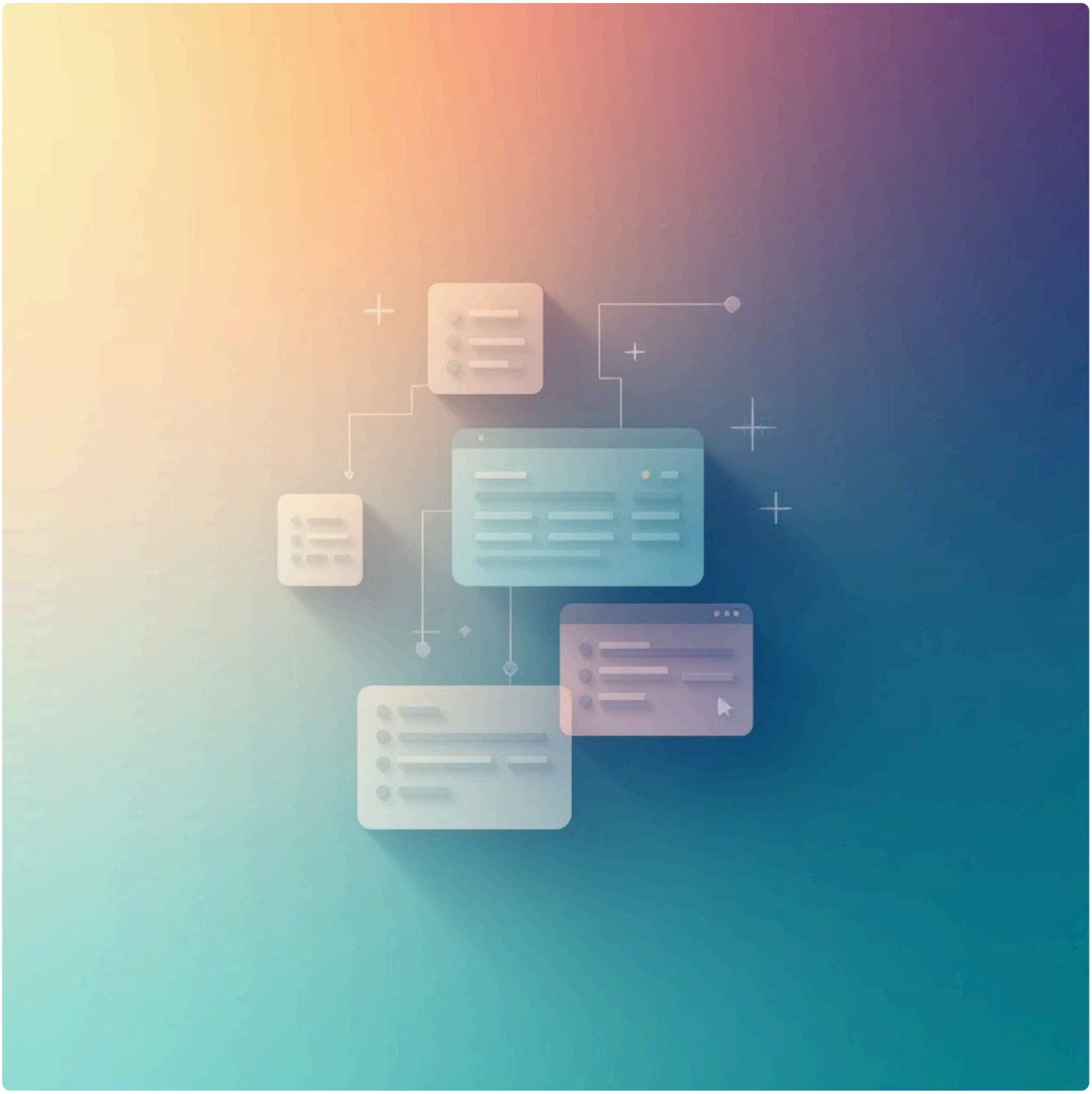
al separar dependencias, se facilita la ampliación del modelo sin afectar otras partes.

Permite escalar el sistema

las bases de datos normalizadas son la base de arquitecturas distribuidas y de alta disponibilidad.

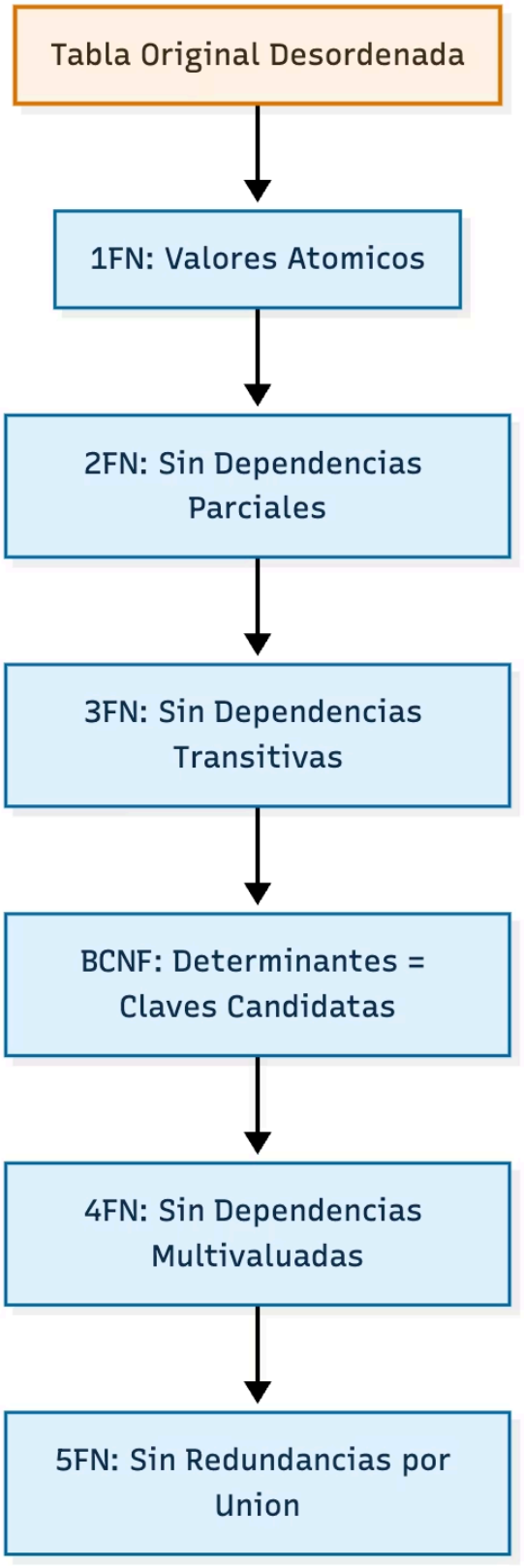
Sin embargo, también es importante reconocer sus límites: una normalización excesiva puede aumentar la complejidad de las consultas, especialmente en entornos de análisis masivo o visualización de datos. En esos casos, se recurre a la desnormalización controlada, un proceso inverso que busca mejorar el rendimiento sin comprometer la integridad.

En esta práctica veremos cómo una base de datos desordenada, con múltiples redundancias, se transforma paso a paso en un modelo perfectamente estructurado.



Esquema Visual: Proceso Completo de Normalización

El siguiente diagrama resume el proceso de normalización aplicado a una tabla desestructurada. Cada paso elimina un tipo distinto de dependencia:



01

1FN

separa los valores múltiples en registros individuales.

02

2FN

garantiza que todos los atributos dependan de la clave completa, no de una parte.

03

3FN

elimina dependencias entre atributos no clave.

04

BCNF

refina el modelo asegurando que solo las claves candidatas determinen otros atributos.

05

4FN

elimina dependencias multivaluadas (relaciones independientes dentro de una misma entidad).

06

5FN

suprime redundancias al combinar relaciones complejas.

Caso de Estudio: Empresa de Eventos

Contexto

Una empresa dedicada a la organización de eventos utiliza una tabla única para registrar todas las operaciones:

EventoID	Cliente	Teléfono	Servicio	Técnico	Equipos
1	Ana	600123	Iluminación	Pedro	Lámpara, Panel
2	Ana	600123	Sonido	Luis	Micrófono
3	Juan	611987	Sonido	Pedro	Micrófono, Altavoz

A simple vista parece funcional, pero presenta múltiples problemas estructurales:

Violación 1FN

Los valores múltiples en la columna "Equipos" violan la Primera Forma Normal (1FN).

Violación 2FN

El Teléfono depende solo del Cliente, no del Evento, lo que rompe la Segunda Forma Normal (2FN).

Violación 3FN

El Técnico depende del Servicio (no directamente del Evento), generando una dependencia transitiva, que viola la Tercera Forma Normal (3FN).

Violación 4FN

Si un servicio tiene varios equipos independientes, aparecen dependencias multivaluadas, lo que rompe la Cuarta Forma Normal (4FN).

Violación 5FN

Al combinar varias tablas (Servicios, Equipos, Técnicos), pueden surgir redundancias de unión, que se corrigen con la Quinta Forma Normal (5FN).

Estrategia de Normalización Paso a Paso

Paso 1: Aplicar 1FN (valores atómicos)

Se separan los valores múltiples en filas independientes:

EventoID	Cliente	Teléfono	Servicio	Técnico	Equipo
1	Ana	600123	Iluminación	Pedro	Lámpara
1	Ana	600123	Iluminación	Pedro	Panel
2	Ana	600123	Sonido	Luis	Micrófono
3	Juan	611987	Sonido	Pedro	Micrófono
3	Juan	611987	Sonido	Pedro	Altavoz

Ahora cada campo contiene un solo valor, cumpliendo la 1FN.

Paso 2: Aplicar 2FN (eliminar dependencias parciales)

"Teléfono" depende únicamente del cliente, no del evento. Creamos una nueva tabla Clientes(ClienteID, Nombre, Teléfono) y referenciamos su clave desde la tabla principal.

Tablas resultantes:

- Clientes(ClienteID, Nombre, Teléfono)
- Eventos(EventoID, ClienteID, Servicio, Técnico, Equipo)

Paso 3: Aplicar 3FN (eliminar dependencias transitivas)

Observamos que el Técnico depende del Servicio (cada servicio tiene asignado un técnico). Por tanto, separamos esta relación:

- Servicios(ServicioID, NombreServicio, TécnicoID)
- Técnicos(TécnicoID, Nombre)

La tabla de eventos queda como: Eventos(EventoID, ClienteID, ServicioID)

Paso 4: Aplicar BCNF (determinantes válidos)

En este modelo, cada determinante es una clave candidata. Por ejemplo, ServicioID determina TécnicoID, y cada ClienteID identifica de forma única a sus eventos. Se cumple la BCNF.

Paso 5: Aplicar 4FN (dependencias multivaluadas)

Un mismo servicio puede requerir varios equipos. Para evitar redundancias, creamos una tabla intermedia:

- Equipos(EquipoID, NombreEquipo)
- Servicios_Equipos(ServicioID, EquipoID)

Así, si un servicio (por ejemplo, Sonido) necesita un "Micrófono" y un "Altavoz", cada combinación se almacena una sola vez, eliminando dependencias multivaluadas.

Paso 6: Aplicar 5FN (dependencias de unión)

Comprobamos que no existan redundancias al recomponer las tablas mediante JOINs. Si cada combinación de Cliente, Servicio y Equipo puede reconstruirse sin duplicar información, la base cumple la 5FN.

Resultado Final

Clientes	ClienteID, Nombre, Teléfono	Registra la información de los clientes.
Eventos	EventoID, ClienteID, ServicioID	Asocia clientes con los servicios contratados.
Servicios	ServicioID, NombreServicio, TécnicoID	Define los tipos de servicio y su responsable.
Técnicos	TécnicoID, Nombre	Almacena los técnicos de la empresa.
Equipos	EquipoID, NombreEquipo	Lista de equipos disponibles.
Servicios_Equipos	ServicioID, EquipoID	Relación muchos-a-muchos entre servicios y equipos.

Con esta estructura, el modelo de datos es coherente, extensible y completamente normalizado hasta la 5FN. Se pueden añadir nuevos técnicos, clientes o equipos sin afectar a otras tablas.

Herramientas y Consejos

Consejos prácticos

Identifica siempre la clave principal antes de normalizar

Una clave bien definida es el punto de partida para detectar dependencias parciales o transitivas.

Usa diagramas de dependencias

Representar las relaciones con flechas te ayudará a visualizar qué atributos dependen de cuáles y evitar errores conceptuales.

Aplica las formas normales de manera secuencial

No intentes saltar directamente a 4FN o 5FN. Cada nivel depende del anterior.

Verifica la reconstrucción con consultas JOIN

Después de dividir las tablas, asegúrate de que la información original pueda recuperarse sin pérdida ni duplicación.

Combina normalización con sentido práctico

En sistemas transaccionales conviene aplicar todas las formas. En entornos analíticos, una desnormalización parcial puede mejorar la velocidad de consulta.

Herramientas recomendadas

Lucidchart o dbdiagram.io

para modelar visualmente las relaciones entre tablas.

MySQL Workbench

permite crear diagramas E-R, definir claves y ejecutar scripts SQL de prueba.

DBeaver

ideal para validar integridad referencial mediante JOINs y analizar la estructura resultante.

ERDPlus

útil para representar dependencias funcionales y verificar el cumplimiento de cada forma normal.

Estas herramientas son de uso habitual en entornos profesionales de diseño de bases de datos y te permitirán practicar con ejemplos reales.

Mitos y Realidades

❌ Mito

"Normalizar siempre mejora el rendimiento del sistema."

→ **FALSO.** La normalización mejora la coherencia y la integridad, pero no necesariamente el rendimiento. En sistemas de análisis o dashboards, donde se prioriza la velocidad de lectura, puede ser útil desnormalizar parcialmente (por ejemplo, almacenando vistas materializadas o tablas resumen).

❌ Mito

"Llegar hasta la 5FN es obligatorio en todos los proyectos."

→ **FALSO.** En la mayoría de las aplicaciones empresariales, normalizar hasta la 3FN o la BCNF es suficiente. Las 4FN y 5FN se aplican solo cuando existen múltiples conjuntos de dependencias independientes o combinaciones complejas entre entidades.

Resumen final

- La normalización elimina redundancias y mantiene la coherencia entre los datos.
- El proceso avanza secuencialmente: 1FN → 2FN → 3FN → BCNF → 4FN → 5FN.
- Cada forma corrige un tipo de dependencia distinto (parcial, transitiva, multivaluada o de unión).
- Un modelo normalizado es más fácil de mantener, más coherente y más escalable.
- En entornos de lectura intensiva puede combinarse con desnormalización controlada para optimizar el rendimiento.