# Capítulo 5: Normalización de bases de datos

La normalización es un proceso en el cual se aplican reglas para organizar los datos de una base de datos de manera más eficiente. El objetivo de la normalización es reducir la redundancia de datos, mejorar la integridad de los datos y evitar errores en la manipulación de la información. La normalización se divide en diferentes niveles, cada uno de los cuales se enfoca en un aspecto específico de la organización de los datos.

Existen 6 formas normales. Las primeras 3 (1FN, 2FN y 3FN), fueron desarrolladas por Edgard Codd y son las más importantes, ya que aseguran la integridad de nuestra base de datos, las que debemos aplicar sí o sí.

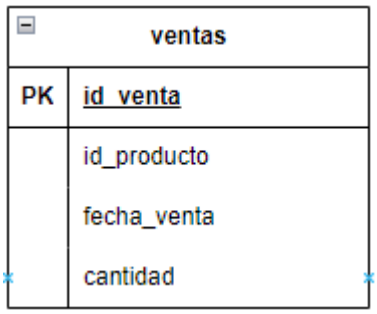
Luego surgieron algunas otras como la Forma Normal de Boyce-Codd (FNBC), 4FN, y 5FN, las cuales menciono por si alguien desea profundizar, pero no las trataremos en este apunte.

## Primera forma normal

La primera forma normal (1FN) es el primer nivel de normalización. Para cumplir con esta forma normal, cada tabla debe tener una **clave primaria única** y **no se deben permitir valores repetidos** en las columnas. Además, cada columna debe tener **un solo valor en cada fila** (que los atributos sean atómicos).

La 1FN es importante porque garantiza que cada tabla tenga una estructura bien definida y que cada fila tenga **información única y no redundante**. Además, permite que las tablas sean unidas y relacionadas correctamente mediante claves primarias y foráneas.

Ejemplo práctico:

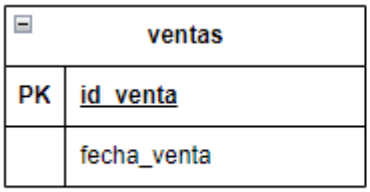


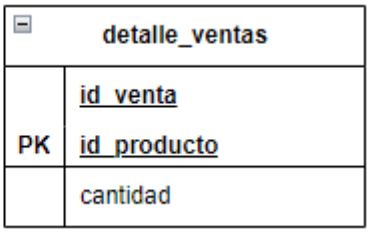

Supongamos que tenemos una tabla llamada "ventas" con las columnas id_venta, id_producto, fecha_venta y cantidad como se muestra en la imagen.

Esta tabla no cumple con la 1FN ya que la columna "id_producto" puede contener múltiples valores en la misma fila (dado porque cada venta generalmente se

compone de varios productos) , lo que dificulta la búsqueda y el análisis de los datos.

Para corregir esto, podemos crear una nueva tabla llamada "detalle_ventas" con las columnas "id_venta", "id_producto" y "cantidad". De esta forma, cada fila de la tabla "detalle_ventas" tendría un solo producto asociado a ella y cumpliría con la 1FN.




Por cada venta, vamos a tener tantos detalles de venta como productos diferentes hayamos vendido en esa venta.

Por ejemplo, si tenemos una venta con el ID "1", la fecha "2022-03-21" y los productos "Camiseta" y "Pantalón" con cantidades vendidas de 2 y 3 unidades respectivamente, en la tabla "Detalle de ventas" tendríamos dos filas: una con el ID de venta "1", el ID de producto "1" (correspondiente a la camiseta) y la cantidad vendida "2", y otra con el mismo ID de venta "1", el ID de producto "2" (correspondiente al pantalón) y la cantidad vendida "3".

| **ventas** | | **detalle_ventas** | | |
|---|---|---|---|---|
| id_venta | fecha_venta | id_venta | id_producto | cantidad |
| 1 | 2022-03-21 | 1 | 1 | 2 |
| | | 1 | 2 | 3 |

En este caso para no complejizar el problema por ser el primer contacto con este tema, hemos dejado de lado la tabla producto que seguramente tendrá lo siguiente.

| **productos** | | | |
|---|---|---|---|
| id_producto | descripcion | stock | precio |
| 1 | Camiseta | 5 | 100 |
| 2 | Pantalon | 3 | 200 |

De esta manera, la información de cada venta estaría organizada de manera más eficiente y se evitaría la redundancia de datos. Además, al tener cada producto en su propia fila, es más fácil realizar consultas y análisis de datos específicos sobre cada producto.

## Segunda forma normal

La segunda forma normal (2FN) establece que cada atributo no clave de una tabla debe **depender funcionalmente de la clave primaria completa**. Esto significa que si una tabla tiene múltiples claves primarias, cada atributo de la tabla debe depender completamente de todas las claves primarias y no de una parte de ellas.

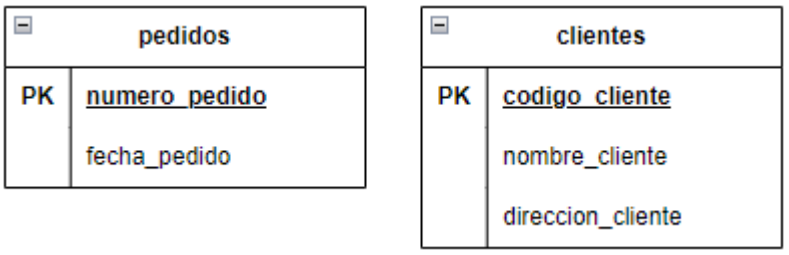
Para entender esto mejor, imaginemos una tabla llamada "Pedidos" con los siguientes atributos:

| pedidos | |
|---|---|
| PK | numero_pedido |
| PK | fecha_pedido |
| | codigo_cliente |
| | nombre_cliente |
| | direccion_cliente |
| | codigo_producto |
| | cantidad_producto |
| | precio_unitario |

La clave primaria de esta tabla está compuesta por dos atributos: "numero_pedido" y "fecha_pedido". Sin embargo, el atributo "codigo_cliente" solo depende de "numero_pedido" y no de "fecha_pedido". En este caso, la tabla no cumple con la segunda forma normal.

Para solucionar este problema, debemos dividir la tabla en dos: una tabla llamada "Pedidos" con los atributos "Número de pedido" y "Fecha de pedido", y otra tabla llamada "Clientes" con los atributos "Código de cliente", "Nombre del cliente" y

"Dirección del cliente". De esta forma, la tabla "Pedidos" solo contendrá información relacionada con los pedidos, y la tabla "Clientes" sólo contendrá información relacionada con los clientes.




De esta manera estaríamos cumpliendo con la segunda forma normal. De todas maneras, se habrán dado cuenta que hay algunos campos como la cantidad de producto, precio y código de producto, que los hemos eliminado y eso no es del todo correcto. Ello lo deberíamos poner en una tercer tabla, que se denomine detalle_pedidos. Por otra parte algunos de ustedes notarán que la tabla pedidos y clientes no se relacionan, y teniendo en cuenta que un pedido pertenece sí o sí a un cliente, debemos relacionarlo de alguna forma. Eso se hace agregando el campo codigo_cliente, en la tabla pedidos, que será clave foránea de codigo_cliente de la tabla clientes. De todas maneras no se preocupen si no lo entienden, porque para llegar a ese nivel tenemos que ver primero la tercera forma normal. Pero no quería dejar de aclararlo en este pequeño párrafo porque puede prestarse a la confusión de que esto es correcto, y si bien lo es, el día que resolvamos un ejercicio debemos aplicarle además la 3 FN.

Es importante destacar que la segunda forma normal **solo se aplica a tablas que ya cumplen con la primera forma normal**. Además, la aplicación de la segunda forma normal puede generar una mayor cantidad de tablas, lo que puede hacer que la base de datos sea más compleja y difícil de manejar. Sin embargo, **garantiza una mayor integridad y consistencia de los datos.**

## Tercera forma normal

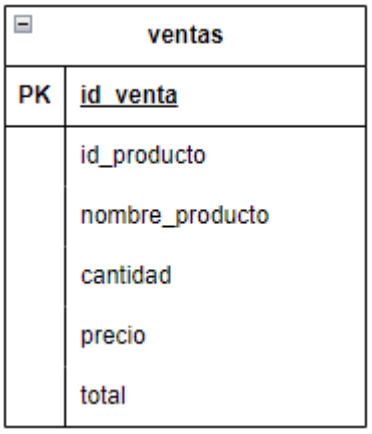
La tercera forma normal (3FN) se enfoca en la eliminación de las dependencias transitivas. Una dependencia transitiva ocurre cuando un atributo depende de otro atributo que no es la clave primaria. En otras palabras, si tenemos tres atributos A, B

y C, y A determina B, y B determina C, entonces C es dependiente de A a través de B.

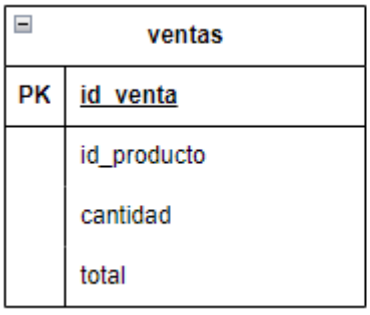
Para aplicar la tercera forma normal, debemos asegurarnos de que **cada atributo no clave de una tabla dependa solo de la clave primaria de la tabla o de otros atributos clave**. Si un atributo depende de otro atributo no clave, debemos separarlo en una tabla diferente.

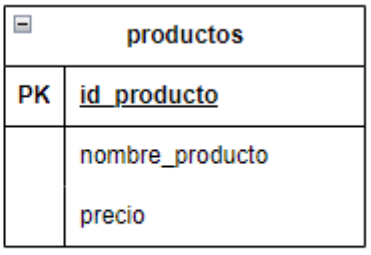
Veamos un ejemplo para entender mejor esta forma normal:

Supongamos que tenemos la siguiente tabla "ventas" con la siguiente estructura:




En esta tabla, la clave primaria es la columna "id_venta". Sin embargo, podemos observar una dependencia transitiva entre las columnas "nombre_producto" y "cantidad" con respecto a la clave primaria. Es decir, el nombre del producto y la cantidad de productos vendidos dependen del "id_producto", no de la clave primaria "id_venta". Por lo tanto, para aplicar la tercera forma normal, debemos separar esta información en una nueva tabla llamada "Productos":




De esta manera, cada tabla tiene su propia clave primaria y la información está organizada de manera más eficiente y sin dependencias transitivas. Para simplificar el caso vamos a hacer de cuenta que por cada venta se puede vender un solo producto. Si quisiéramos contemplar que en cada venta se pueda vender más de un

producto, ya deberíamos separar en otra tabla, que sea detalle_ventas, porque sino por cada venta estaríamos obligados a poner varios id_producto en una misma venta y dejaríamos de cumplir con la 1 FN.

**La tercera forma normal se enfoca en eliminar las dependencias transitivas y garantizar que cada atributo no clave de una tabla dependa solo de la clave primaria de la tabla o de otros atributos clave**. De esta manera, se logra una mejor organización y eficiencia en la base de datos.