TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

Trabajo Práctico 5:
Desnormalización de Bases de Datos

Nombre: Farias, Gustavo
Comisión: M2025-13
Matrícula: 101662
Repositorio GitHub:
https://github.com/Lucenear/UTN-TUPaD-TPs/tree/main/Bases%20de%20Datos/Bases%20de%20Datos%20I

Desnormalización de Bases de Datos

Objetivos:

- Revisar y validar si un conjunto de tablas está correctamente normalizado.
- Identificar errores de diseño frecuentes y proponer correcciones.
- Comprender cuándo puede ser conveniente desnormalizar una parte del diseño para optimizar consultas o almacenamiento.
- Aplicar criterios de diseño relacional desde una mirada crítica.

Parte 1: Evaluación de diseño normalizado

1) **Análisis de claves primarias y claves foráneas:**

Tabla	PK	FK
clientes	id_cliente	id_localidad -> localidades
ventas	id_venta	id_cliente -> clientes
detalles_venta	id_venta, id_producto	id_venta -> ventas id_producto -> productos
productos	id_producto	-
localidades	id_localidad	-

Considera que las claves primarias y foráneas están bien definidas.

2) **Verificación de formas normales:**

A priori, se estaria cumpliendo con las formas 1FN, 2FN, en donde detalles_venta tendria una clave compuesto por id_venta y id_producto.
Pero en el caso de 3FN, entiendo que habria una violacion en cuanto al atributo ‘total’, ya que se trata de un valor calculado que depende de ‘cantidad’ * ‘precio_unitario’, atributos que se encuentran en la tabla ‘detalles_venta’.

Solución propuesta:

Se deberia eliminar el atributo ‘total’ de la tabla ‘ventas’ y calcular este valor al realizar la consulta misma.
De esta manera, la tabla ‘ventas’ solo contendrá los atributos (id_venta, fecha, id_cliente).

3) **Conclusión:**

El diseño propuesto se acerca bastante a cumplir con la Tercera Forma Normal (3FN). La única irregularidad se presenta en el campo total de la tabla ‘ventas’. Este atributo representa un valor calculado a partir de la información contenida en la tabla ‘detalles_venta’. Al prescindir de este campo redundante, se logrará una normalización total del diseño.

Parte 2: Normalización inversa guiada

1) **Objetivo propuesto**

Reducir la cantidad de JOINs en reportes frecuentes de ventas detalladas, mejorando así la velocidad de consulta y simplificando el acceso a la información para usuarios no técnicos (como gerentes, analistas o sistemas de BI).

2) **Qué tablas decido fusionar o simplificar?**

Propongo fusionar las tablas ‘ventas’ y ‘detalles_venta’ en una única tabla desnormalizada llamada ‘ventas_detalladas’.

Mantenemos las tablas ‘clientes’, ‘productos’ y ‘localidades’ separadas y normalizadas, ya que su información cambia con poca frecuencia y no son el foco de optimización.

3) **Justificación funcional y de rendimiento**

Un escenario frecuente de uso podría ser Mostrar todas las ventas del último mes con: cliente, fecha, producto, cantidad, precio unitario y subtotal.

Por ejemplo, podemos realizar un comparativo de dos (2) consultas de sql en donde se aprecia el esfuerzo y conocimiento tecnico que puede requerir acceder a la informacion que se necesita.

a) **Consulta con modelo normalizado:** Se requieren 4 tablas involucradas, 3 joins y cálculo en tiempo real para armar el atributo ‘total’.

```
SELECT
   c.nombre AS cliente,
   v.fecha,
   p.descripcion AS producto,
   dv.cantidad,
   dv.precio_unitario,
   (dv.cantidad * dv.precio_unitario) AS subtotal
FROM VENTAS v
        JOIN DETALLES_VENTA dv ON v.id_venta = dv.id_venta
        JOIN CLIENTES c ON v.id_cliente = c.id_cliente
        JOIN PRODUCTOS p ON dv.id_producto = p.id_producto
WHERE v.fecha >= ‘2025-01-01’
```

b) **Consulta pos desnormalizacion:** Se requiere 1 tabla involucrada, 0 joins y no se precisa realizar cálculos. Es más rápido y simple de usar para el usuario final.

```
SELECT
   nombre_cliente,
   fecha,
   descripcion_producto,
   cantidad,
   precio_unitario,
```

```
   subtotal
FROM VENTAS_DETALLADAS
WHERE fecha >= ‘2025-01-01’
```

4) **Ventajas y desventajas**

   a) **Ventajas:**
      i) **Rendimiento superior:** Elimina JOINs y cálculos en tiempo real. Ideal para dashboards, BI y reportes ejecutivos.
      ii) **Simplicidad para usuarios:** Consultas accesibles para personal no técnico. Menos errores al escribir SQL.

   b) **Desventajas:**
      i) **Redundancia de datos:** Se repiten nombres de clientes y descripciones de productos en múltiples filas.
      ii) **Mayor consumo de almacenamiento:** La tabla crece más rápido por la duplicación de información.

5) **Propuesta de implementación de tabla desnormalizada ventas_detalladas**

Atributo	Tipo	Descripción	Origen (tabla)
id_venta	int	Identificador de la venta	ventas
fecha	date	Fecha de la venta	ventas
id_cliente	int	FK al cliente	ventas
id_producto	int	FK al producto	detalles_venta
cantidad	int	Cantidad vendida	detalles_venta
precio_unitario	decimal(10,2)	precio del producto	detalles_venta
subtotal	decimal(10,2)	cantidad * precio unitario	calculado
descripcion_producto	varchar(50)	Descripción del producto	productos
nombre_cliente	varchar(50)	Nombre del cliente	clientes

***** Se incluyen descripcion_producto y nombre_cliente para evitar JOINs con ‘productos’ y ‘clientes’ en reportes comunes.

6) **Conclusion**

Esta desnormalizacion forzada e intencional, si bien viola principios de redundancia y dependencias transitivas, se justifica plenamente en beneficio del rendimiento y usabilidad en contextos de lectura intensiva, debido a que se busca priorizar velocidad de consultas por sobre pureza del modelo, sobre todo si no se encuentra alojado en cloud.