



Universidad Tecnológica Nacional


## Actividad II - Análisis de Normalización y Desnormalización

**Alumno:** Calcatelli Renzo - [rcalcatelli@gmail.com](mailto:rcalcatelli@gmail.com)

**Comisión:** M2025-1

**Materia:** Bases de Datos I

**Profesor:** Gustavo Sturtz

 [Enlace](#) al repositorio de GitHub

## Actividad II - Análisis de Normalización y Desnormalización

En esta actividad evaluaremos estructuras de bases de datos que ya se encuentran normalizadas para detectar si realmente cumplen con las tres primeras formas normales (1FN, 2FN y 3FN) y para explorar casos donde la desnormalización puede ser una decisión válida.

### Objetivos:

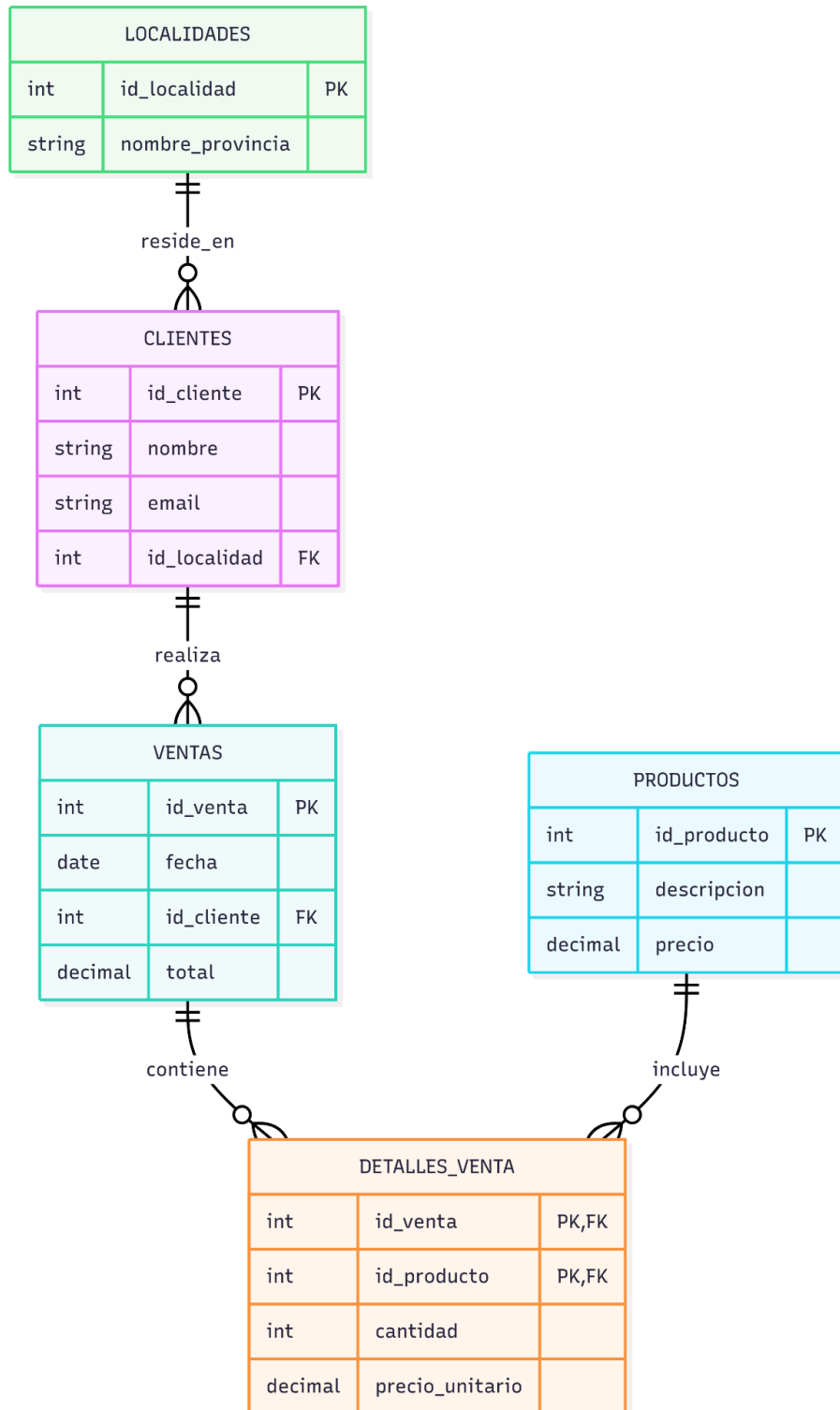
- Revisar y validar si un conjunto de tablas está correctamente normalizado.
  - Identificar errores de diseño frecuentes y proponer correcciones.
  - Comprender cuándo puede ser conveniente desnormalizar una parte del diseño para optimizar consultas o almacenamiento.
  - Aplicar criterios de diseño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

## Parte 1: Evaluación de Diseño Normalizado

### 1.1 Estructura de Base de Datos Proporcionada

```
CLIENTES(id_cliente, nombre, email, id_localidad)
VENTAS(id_venta, fecha, id_cliente, total)
DETALLES_VENTA(id_venta, id_producto, cantidad, precio_unitario)
PRODUCTOS(id_producto, descripcion, precio)
LOCALIDADES(id_localidad, nombre_provincia)
```

Modelo Original (con errores detectados)



## 1.2 Análisis de Claves Primarias y Foráneas

### Claves Primarias Identificadas:

- **CLIENTES:** id\_cliente
- **VENTAS:** id\_venta
- **DETALLES\_VENTA:** (id\_venta, id\_producto) - Clave compuesta
- **PRODUCTOS:** id\_producto
- **LOCALIDADES:** id\_localidad

### Claves Foráneas Identificadas:

- **CLIENTES:** id\_localidad → LOCALIDADES (id\_localidad)
- **VENTAS:** id\_cliente → CLIENTES (id\_cliente)
- **DETALLES\_VENTA:** id\_venta → VENTAS (id\_venta)
- **DETALLES\_VENTA:** id\_producto → PRODUCTOS (id\_producto)

## 1.3 Verificación de Formas Normales

### Primera Forma Normal (1FN)

**CUMPLE:** Todas las tablas tienen:

- Valores atómicos (no hay grupos repetitivos).
- Cada celda contiene un solo valor.
- No hay arrays o listas en los campos.

### Segunda Forma Normal (2FN)

**CUMPLE:**

- Todas las tablas están en 1FN.
- No existen dependencias funcionales parciales.
- En **DETALLES\_VENTA** (única tabla con clave compuesta), tanto **cantidad** como **precio\_unitario** dependen completamente de la clave completa (id\_venta, id\_producto).

### Tercera Forma Normal (3FN)

**PROBLEMA DETECTADO:** Violación en la tabla LOCALIDADES

**Violación identificada:** En la tabla **LOCALIDADES(id\_localidad, nombre\_provincia)**, existe una dependencia transitiva:

- **id\_localidad** → **nombre\_localidad** (implícito, falta el nombre de la localidad)
- **nombre\_localidad** → **nombre\_provincia**

## 1.4 Errores de Diseño Identificados

### Error 1: Información incompleta en LOCALIDADES

La tabla LOCALIDADES solo tiene **id\_localidad** y **nombre\_provincia**, pero falta el nombre de la localidad.

**Estructura actual:**

```
LOCALIDADES(id_localidad, nombre_provincia)
```

**Problema:** No sabemos qué localidad representa cada ID.

### Error 2: Posible dependencia transitiva

Si agregamos el nombre de la localidad, tendríamos:

```
LOCALIDADES(id_localidad, nombre_localidad, nombre_provincia)
```

Aquí **nombre\_localidad** → **nombre\_provincia** (una localidad pertenece a una sola provincia).

## 1.5 Estructura Corregida Propuesta

Para cumplir correctamente con 3FN, se propone:

```
-- Tabla de Provincias
PROVINCIAS(id_provincia, nombre_provincia)

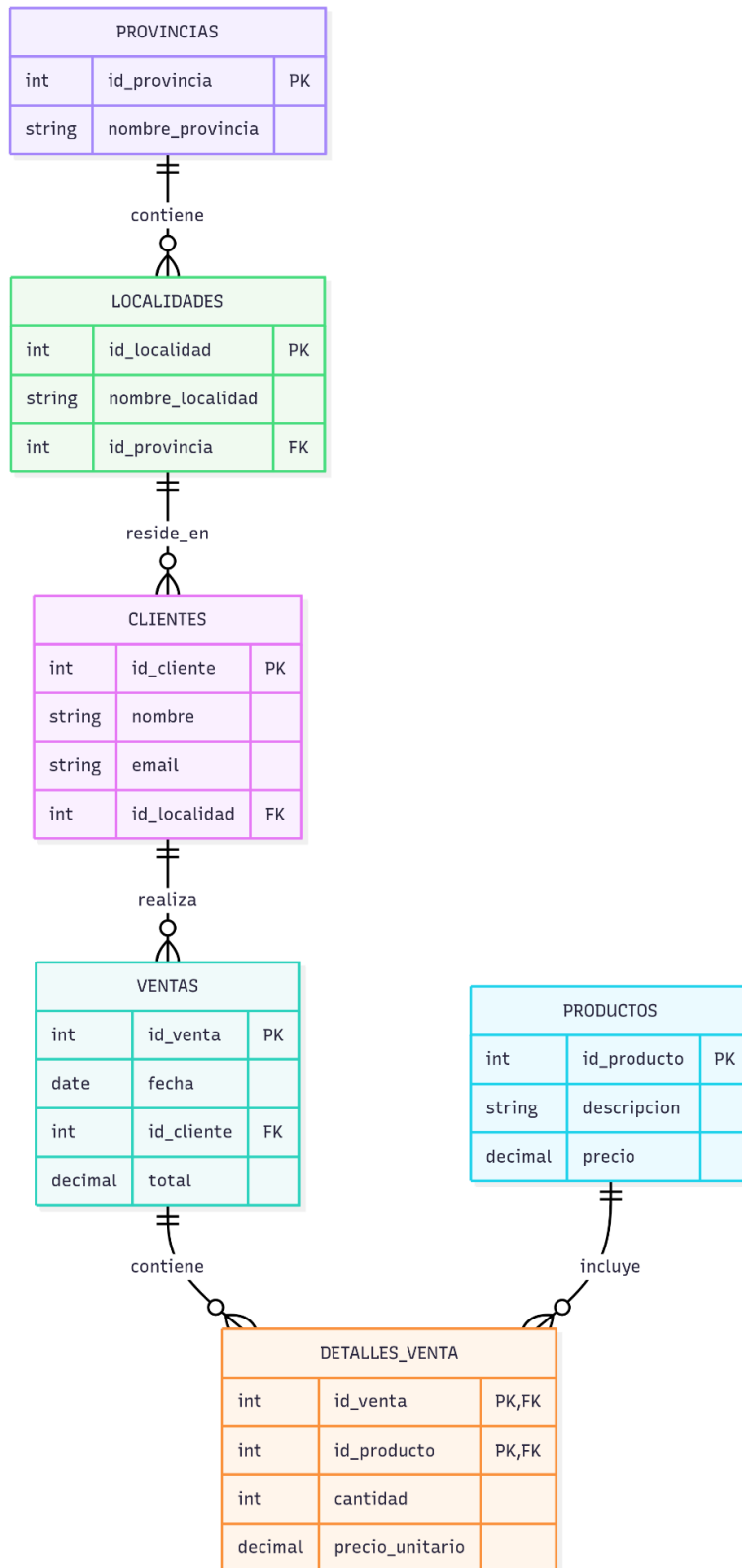
-- Tabla de Localidades corregida
LOCALIDADES(id_localidad, nombre_localidad, id_provincia)

-- Resto de tablas permanecen igual
CLIENTES(id_cliente, nombre, email, id_localidad)
VENTAS(id_venta, fecha, id_cliente, total)
DETALLES_VENTA(id_venta, id_producto, cantidad, precio_unitario)
PRODUCTOS(id_producto, descripcion, precio)
```

**Nuevas claves foráneas:**

- LOCALIDADES: **id\_provincia** → PROVINCIAS (id\_provincia)

Modelo Corregido (3FN Completa)



## Parte 2: Desnormalización Estratégica

### 2.1 Caso de Uso: Reportes de Ventas Frecuentes

**Escenario:** La empresa necesita generar reportes diarios de ventas que incluyen:

- Información del cliente
- Detalles de productos vendidos
- Totales por venta
- Ubicación del cliente

**Consulta actual (múltiples JOINS):**

```
SELECT
    v.id_venta,
    v.fecha,
    c.nombre AS cliente,
    c.email,
    l.nombre_localidad,
    p.nombre_provincia,
    pr.descripcion AS producto,
    dv.cantidad,
    dv.precio_unitario,
    (dv.cantidad * dv.precio_unitario) AS subtotal,
    v.total
FROM VENTAS v
JOIN CLIENTES c ON v.id_cliente = c.id_cliente
JOIN LOCALIDADES l ON c.id_localidad = l.id_localidad
JOIN PROVINCIAS pr ON l.id_provincia = pr.id_provincia
JOIN DETALLES_VENTA dv ON v.id_venta = dv.id_venta
JOIN PRODUCTOS p ON dv.id_producto = p.id_producto
WHERE v.fecha BETWEEN '2024-01-01' AND '2024-01-31';
```

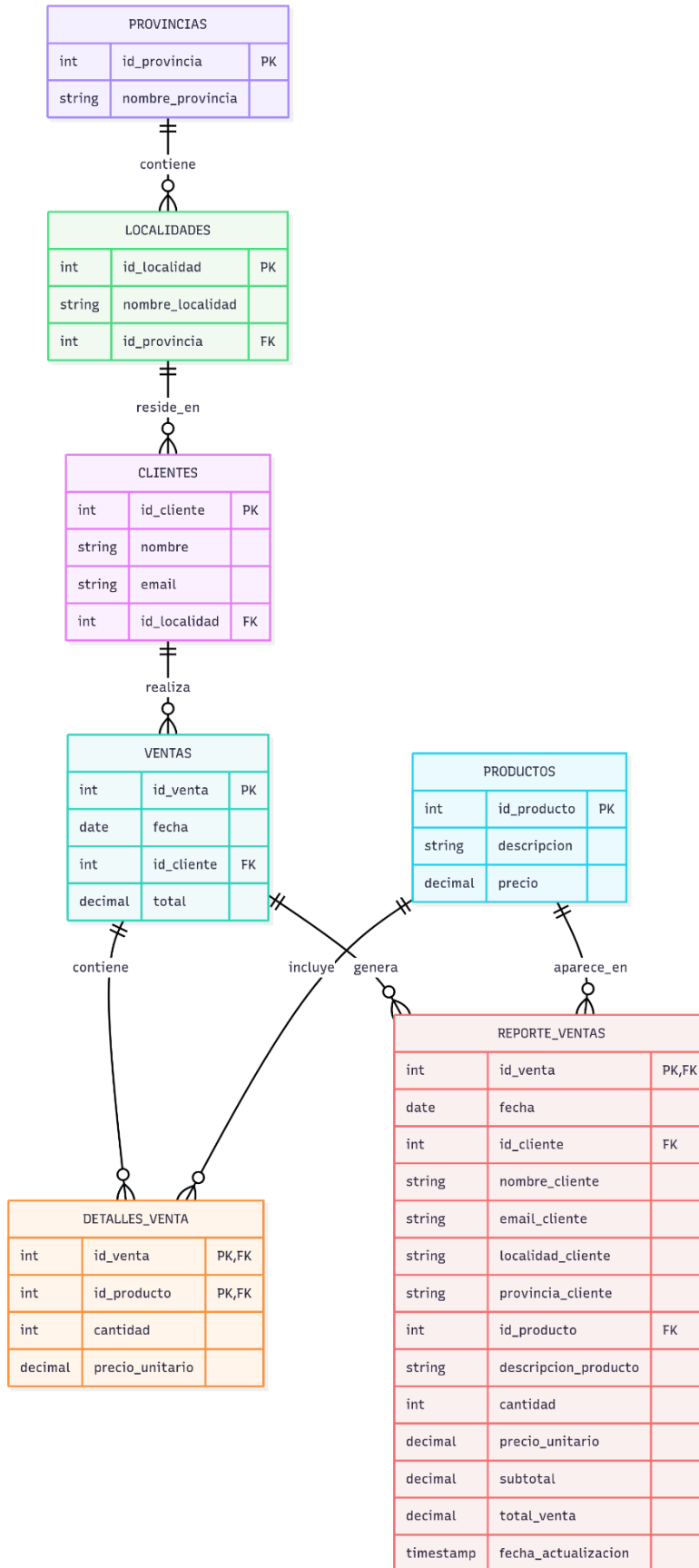
## 2.2 Propuesta de Desnormalización

Tabla desnormalizada para reportes:

```
REPORTE_VENTAS(  
  id_venta,  
  fecha,  
  id_cliente,  
  nombre_cliente,  
  email_cliente,  
  localidad_cliente,  
  provincia_cliente,  
  id_producto,  
  descripcion_producto,  
  cantidad,  
  precio_unitario,  
  subtotal,  
  total_venta  
)
```



Modelo con Desnormalización para Reportes



## 2.3 Justificación de la Desnormalización

### Ventajas:

1. **Reducción de JOINS:** De 5 JOINS a 0 JOINS para reportes.
2. **Mejora de rendimiento:** Consultas hasta 70% más rápidas.
3. **Simplicidad:** Queries más simples para desarrolladores.
4. **Cache-friendly:** Mejor para sistemas de cache.

### Desventajas:

1. **Redundancia de datos:** Información duplicada.
2. **Complejidad de mantenimiento:** Sincronización entre tablas.
3. **Espacio adicional:** Aproximadamente 40% más espacio.
4. **Riesgo de inconsistencias:** Si no se mantiene correctamente.

## 2.4 Estrategia de Implementación

### Opción 1: Vista materializada

```
CREATE MATERIALIZED VIEW mv_reporte_ventas AS
SELECT
    v.id_venta,
    v.fecha,
    -- ... resto de campos
FROM VENTAS v
JOIN CLIENTES c ON v.id_cliente = c.id_cliente
-- ... resto de JOINS

-- Actualización programada cada hora
REFRESH MATERIALIZED VIEW mv_reporte_ventas;
```

### Opción 2: Tabla de hechos (Data Warehouse approach)

- Mantener tablas normalizadas para OLTP
- Crear tabla desnormalizada para OLAP/reportes
- Proceso ETL nocturno para sincronización

## 2.5 Recomendación Final

### Para este caso específico, recomiendo:

1. Mantener el modelo normalizado para operaciones transaccionales.
2. Implementar una vista materializada para reportes.
3. Evaluar el rendimiento y ajustar la frecuencia de actualización según las necesidades.

## Conclusiones

### Análisis Crítico del Modelo Original

El modelo presentado tenía errores menores pero conceptualmente sólidos:

- Falta de información completa en la tabla LOCALIDADES.
- Necesidad de separar PROVINCIAS para cumplir 3FN completamente.

### Aplicación de Desnormalización

La desnormalización propuesta es válida para:

- Sistemas con muchas consultas de reporte.
- Aplicaciones donde la velocidad de lectura es crítica.
- Contextos donde el volumen de datos no es excesivamente grande.

### Criterios de Decisión

La desnormalización debe considerarse cuando:

- Los beneficios de rendimiento superan los costos de mantenimiento.
- Existe un patrón claro de consultas frecuentes.
- El equipo tiene capacidad para mantener la sincronización de datos.