Normalización vs Denormalización

Guía práctica para sistemas de Reportes y Business Intelligence

Criterios Clave para Decidir

1. Frecuencia de Actualización

- **Normalizar:** Datos que cambian frecuentemente
- **Denormalizar:** Datos estáticos o batch

2. Tamaño y Volumen

- **Normalizar:** Millones de registros repetitivos
- **Denormalizar:** Volumen moderado, prioridad en velocidad

3. Complejidad de Consultas

- **Normalizar:** Consultas simples y enfocadas
- **Denormalizar:** Múltiples JOINs complejos

4. Integridad Referencial

- **Normalizar:** Integridad crítica, evitar anomalías
- **Denormalizar:** Datos históricos inmutables

5. Patrón de Acceso

- **Normalizar:** Acceso a entidades individuales
- **Denormalizar:** Agregaciones de múltiples tablas

Ejemplos Típicos en Sistemas de Reportes

✓ Tablas que SE NORMALIZAN

Dimensiones (Esquema Estrella)

```
-- DIM_CLIENTE (normalizada) CREATE TABLE
dim_cliente ( cliente_id INT PRIMARY KEY, nombre
VARCHAR(100), email VARCHAR(100), tipo_cliente_id
INT );
```

Razón: Dimensiones pequeñas, estables, reutilizables. La normalización mantiene consistencia.

Catálogos de Referencia

```
CREATE TABLE cat_paises ( pais_id INT PRIMARY KEY,
nombre_pais VARCHAR(100), codigo_iso CHAR(3) );
```

Razón: Tablas pequeñas compartidas. Evita inconsistencias.

✗ Tablas DENORMALIZADAS

Tablas de Hechos con Atributos

```
-- FACT_VENTAS (denormalizada) CREATE TABLE
fact_ventas ( venta_id BIGINT PRIMARY KEY,
fecha_venta DATE, -- Atributos denormalizados
cliente_nombre VARCHAR(100), cliente_region
VARCHAR(50), producto_nombre VARCHAR(100), cantidad
INT, total_venta DECIMAL(12,2) );
```

Razón: Evitar JOINs mejora rendimiento dramáticamente en analítica.

Agregados Precalculados

```
CREATE TABLE agg_ventas_mensuales ( año INT, mes
INT, region VARCHAR(50), total_ventas DECIMAL(15,2),
clientes_unicos INT );
```

Razón: Calculados una vez, aceleran reportes complejos.

Tabla de Decisión Rápida

Característica	Normalizar	Denormalizar
Tamaño tabla	< 1 millón registros	> 10 millones registros
Actualizaciones	Frecuentes (tiempo real)	Batch diario/semanal
JOINs típicos	1-2 tablas	5+ tablas
Tiempo respuesta	Segundos aceptables	Subsegundos requeridos
Usuarios concurrentes	< 50	> 100
Tipo de análisis	Drill-down detallado	Agregaciones y tendencias

Recomendaciones para Balancear

Arquitectura de Capas Separadas

OLTP (Normalizado) → ETL → **Data Warehouse (Híbrido)** → **Data Marts (Denormalizado)**

→ **Capa Transaccional:** Totalmente normalizada (3NF) para integridad

→ **Data Warehouse:** Esquema estrella con dimensiones normalizadas y hechos denormalizados

→ **Data Marts:** Altamente denormalizados para departamentos específicos

Estrategia del "Mejor de Ambos Mundos"

```
-- Versión normalizada para integridad CREATE TABLE ventas_detalle ( venta_id BIGINT PRIMARY KEY, cliente_id INT,
producto_id INT, cantidad INT ); -- Vista materializada denormalizada para reportes CREATE MATERIALIZED VIEW
mv_ventas_reporting AS SELECT v.venta_id, c.cliente_nombre, c.segmento, p.producto_nombre, v.cantidad FROM
ventas_detalle v JOIN dim_cliente c ON v.cliente_id = c.cliente_id JOIN dim_producto p ON v.producto_id =
p.producto_id;
```

Reglas de Oro para Sistemas de Reportes

- → **SIEMPRE denormaliza:** Datos históricos inmutables, consultas con 3+ JOINs, dashboards en tiempo real
- → **SIEMPRE normaliza:** Datos que cambian frecuentemente, integridad crítica, capa de staging
- → **Considera híbridos:** Cuando tienes recursos para vistas materializadas y refrescos periódicos
- → **Mide y ajusta:** Monitorea tiempos de consulta y ajusta basado en patrones reales
- → **Documenta decisiones:** Mantén claridad sobre estructura y propósito de cada tabla