Diseño Avanzado de Esquemas de Bases de Datos - Día 1

Change Status

in-progress 40 min

Learning Objectives

1. Comprender los trade-offs entre normalización y desnormalización para optimización analítica
2. Diseñar esquemas dimensionales apropiados para consultas analíticas complejas
3. Evaluar el impacto de decisiones de diseño en la performance de análisis

Activities and Learning

Task 1: Normalización vs Desnormalización: Trade-offs para Análisis (10 minutos)

La normalización y desnormalización representan filosofías opuestas en diseño de bases de datos, cada una optimizada para diferentes cargas de trabajo. Para análisis de datos, la elección apropiada puede significar la diferencia entre consultas que tardan segundos y las que tardan horas.

Normalización: Integridad Primero

Objetivos: Eliminar redundancia, asegurar consistencia, facilitar mantenimiento.

1NF: Valores atómicos, sin repetición de grupos
2NF: Dependencias funcionales completas
3NF: Dependencias transitivas eliminadas
BCNF: Dependencias más restrictivas

Ventajas para análisis:

Integridad referencial: Datos consistentes y confiables
Flexibilidad: Fácil añadir nuevos tipos de entidades
Mantenimiento: Cambios localizados, sin propagación

Desventajas para análisis:

Joins complejos: Consultas requieren múltiples joins
Performance pobre: Especialmente con agregaciones complejas
Complejidad: Consultas analíticas difíciles de escribir y optimizar
Desnormalización: Performance Primero

Objetivos: Optimizar para consultas analíticas, minimizar joins, maximizar velocidad de lectura.

Estrategias:

Tablas resumen: Precalcular agregaciones comunes
Columnas derivadas: Almacenar cálculos frecuentes
Tablas anchas: Consolidar información relacionada

Ventajas para análisis:

Consultas simples: Menos joins, más rápidas
Performance predecible: Optimizado para patrones conocidos
Simplicidad: Consultas más legibles y mantenibles

Desventajas para análisis:

Redundancia: Almacenamiento duplicado
Inconsistencia: Riesgo de datos contradictorios
Complejidad de actualización: Cambios requieren sincronización múltiple
Estrategia Híbrida: Lo Mejor de Ambos Mundos

Normalización para transaccional: Sistema OLTP normalizado para operaciones diarias. **Desnormalización para analítico**: Data warehouse desnormalizado para reporting y análisis.

```
-- Ejemplo: Sistema híbrido
-- OLTP: Normalizado para transacciones
CREATE TABLE pedidos (
    id SERIAL PRIMARY KEY,
    cliente_id INTEGER REFERENCES clientes(id),
    fecha_pedido DATE,
    estado VARCHAR(20)
);

-- OLAP: Desnormalizado para análisis
CREATE TABLE hechos_ventas (
```

```
    id_venta INTEGER,
    fecha DATE,
    cliente_nombre VARCHAR(100),
    cliente_segmento VARCHAR(20),
    producto_categoria VARCHAR(50),
    cantidad INTEGER,
    precio_unitario DECIMAL,
    total_venta DECIMAL,
    -- Columnas derivadas para análisis rápido
    mes INTEGER,
    trimestre INTEGER,
    año INTEGER,
    margen_beneficio DECIMAL
);
```

Task 2: Modelado Dimensional: Star Schema y Snowflake Schema (10 minutos)

Los esquemas dimensionales están específicamente diseñados para análisis, sacrificando normalización por performance en consultas complejas.

Star Schema: Simplicidad y Performance

Estructura: Una tabla de hechos central rodeada de tablas de dimensiones.

Características:

Tabla de hechos: Métricas numéricas, claves foráneas a dimensiones
Tablas de dimensión: Atributos descriptivos, jerarquías naturales
Relaciones: Una dimensión conecta a hechos con una relación many-to-one

Ventajas:

Consultas intuitivas: "Ventas por producto, cliente y tiempo"
Performance: Menos joins que esquemas normalizados
Agregaciones eficientes: Optimizado para GROUP BY complejos

Ejemplo práctico:

```
-- Star Schema para análisis de ventas
CREATE TABLE hechos_ventas (
    id_venta SERIAL PRIMARY KEY,
    id_tiempo INTEGER REFERENCES dim_tiempo(id),
    id_cliente INTEGER REFERENCES dim_cliente(id),
    id_producto INTEGER REFERENCES dim_producto(id),
    id_tienda INTEGER REFERENCES dim_tienda(id),
    cantidad INTEGER,
    precio_unitario DECIMAL,
    descuento DECIMAL,
    total_venta DECIMAL
);

CREATE TABLE dim_tiempo (
    id SERIAL PRIMARY KEY,
    fecha DATE,
    dia INTEGER,
    mes INTEGER,
    trimestre INTEGER,
    año INTEGER,
    dia_semana VARCHAR(10),
    festivo BOOLEAN
);

CREATE TABLE dim_cliente (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    segmento VARCHAR(20),
    region VARCHAR(50),
    edad INTEGER,
    genero VARCHAR(10)
);
```

Snowflake Schema: Mayor Normalización

Estructura: Dimensiones parcialmente normalizadas, formando estructura de "copo de nieve".

Características:

Dimensiones jerárquicas: Subdimensiones conectadas
Mayor normalización: Reduce redundancia en dimensiones grandes
Flexibilidad: Más fácil mantener jerarquías complejas

Trade-offs:

Ventaja: Menos redundancia, más fácil mantenimiento
Desventaja: Más joins, consultas más complejas

```
-- Snowflake Schema
CREATE TABLE dim_geografia (
    id SERIAL PRIMARY KEY,
    ciudad VARCHAR(50),
```

```
    id_region INTEGER REFERENCES dim_region(id)
);

CREATE TABLE dim_region (
    id SERIAL PRIMARY KEY,
    nombre_region VARCHAR(50),
    id_pais INTEGER REFERENCES dim_pais(id)
);

-- Comparado con Star Schema donde todo está en una tabla
CREATE TABLE dim_cliente_star (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    ciudad VARCHAR(50),
    region VARCHAR(50),
    pais VARCHAR(50),
    segmento VARCHAR(20)
);
```

Task 3: Diseño de Esquemas para Consultas Analíticas Complejas (10 minutos)

El diseño del esquema debe anticipar los patrones de consulta más comunes y críticos para el negocio.

Identificación de Requisitos Analíticos

Preguntas clave:

¿Qué métricas se analizan frecuentemente?
¿Qué dimensiones se usan para segmentar?
¿Qué jerarquías son importantes (tiempo, geografía, producto)?
¿Qué nivel de granularidad se necesita?

Patrones comunes:

Análisis temporal: Tendencias, comparaciones período sobre período
Segmentación geográfica: Por región, ciudad, tienda
Análisis de producto: Categorías, subcategorías, marcas
Segmentación de cliente: Demografía, comportamiento, valor
Estrategias de Diseño Específicas

Granularidad apropiada: Nivel de detalle que balancea utilidad y performance.

Columnas calculadas: Precalcular métricas derivadas comunes.

```
-- Columnas derivadas en tabla de hechos
CREATE TABLE hechos_ventas (
    -- Claves dimensionales
    id_tiempo INTEGER,
    id_cliente INTEGER,
    id_producto INTEGER,

    -- Métricas base
    cantidad INTEGER,
    precio_unitario DECIMAL,

    -- Métricas derivadas (calculadas en ETL)
    total_bruto DECIMAL GENERATED ALWAYS AS (cantidad * precio_unitario),
    descuento_aplicado DECIMAL,
    total_neto DECIMAL GENERATED ALWAYS AS ((cantidad * precio_unitario) - descuento_aplicado),
    margen_beneficio DECIMAL GENERATED ALWAYS AS (total_neto * 0.25)  -- 25% margen
);
```

Índices estratégicos: Planificar indexación desde el diseño del esquema.