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

Practical exercise to apply the concepts learned.

Ejercicio: Diseño de esquema dimensional completo para análisis de e-commerce

Análisis de requisitos y diseño conceptual:

```
-- Diseño de Star Schema para plataforma de e-commerce
-- Requisitos identificados:
-- - Análisis de ventas por producto, cliente, tiempo y ubicación
-- - Segmentación por categorías y comportamiento de compra
-- - Métricas: ventas, conversiones, retención, valor de vida del cliente

-- 1. Tabla de Hechos: Ventas transaccionales
CREATE TABLE hechos_ventas (
    id_venta INTEGER PRIMARY KEY,
    id_tiempo INTEGER REFERENCES dim_tiempo(id),
    id_cliente INTEGER REFERENCES dim_cliente(id),
    id_producto INTEGER REFERENCES dim_producto(id),
    id_canal INTEGER REFERENCES dim_canal_adquisicion(id),
    id_geografia INTEGER REFERENCES dim_geografia(id),

    -- Métricas transaccionales
    cantidad INTEGER,
    precio_unitario DECIMAL(10,2),
    descuento_aplicado DECIMAL(10,2),
    costo_envio DECIMAL(10,2),
    impuestos DECIMAL(10,2),

    -- Métricas calculadas (desnormalizadas para performance)
    total_bruto DECIMAL(10,2) GENERATED ALWAYS AS (cantidad * precio_unitario),
    total_neto DECIMAL(10,2) GENERATED ALWAYS AS ((cantidad * precio_unitario) - descuento_aplicado + costo_envio + impuestos),
    margen_contribucion DECIMAL(10,2) GENERATED ALWAYS AS ((cantidad * precio_unitario) - descuento_aplicado - costo_envio) * 0.3,

    -- Flags para segmentación rápida
    primera_compra BOOLEAN,
    compra_recurrente BOOLEAN,
    cliente_vip BOOLEAN
);

-- 2. Dimensión Tiempo: Jerarquía temporal completa
CREATE TABLE dim_tiempo (
    id INTEGER PRIMARY KEY,
    fecha DATE UNIQUE,
    dia INTEGER,
    mes INTEGER,
    nombre_mes VARCHAR(20),
    trimestre INTEGER,
    año INTEGER,
    dia_semana VARCHAR(10),
    numero_semana INTEGER,
    festivo BOOLEAN,
    temporada VARCHAR(20),  -- Primavera, Verano, etc.
    fin_semana BOOLEAN,
    dia_habil BOOLEAN
);

-- 3. Dimensión Cliente: Segmentación completa
CREATE TABLE dim_cliente (
    id INTEGER PRIMARY KEY,
    id_cliente_natural INTEGER,  -- Para SCD Tipo 2
    nombre VARCHAR(100),
    email VARCHAR(100),
    fecha_registro DATE,
    segmento_valor VARCHAR(20),  -- Bronce, Plata, Oro, Platino
    segmento_comportamiento VARCHAR(30),  -- Nuevo, Recurrente, VIP, Inactivo
    edad INTEGER,
    genero VARCHAR(10),
    ciudad VARCHAR(50),
    region VARCHAR(50),
    pais VARCHAR(50),
```

```
    frecuencia_compras_mensual DECIMAL(4,1),
    valor_promedio_compra DECIMAL(10,2),
    ultima_compra DATE,
    activo BOOLEAN
);

-- 4. Dimensión Producto: Jerarquía de catálogo
CREATE TABLE dim_producto (
    id INTEGER PRIMARY KEY,
    sku VARCHAR(20) UNIQUE,
    nombre VARCHAR(100),
    descripcion TEXT,
    id_categoria INTEGER REFERENCES dim_categoria(id),
    id_marca INTEGER REFERENCES dim_marca(id),
    precio_lista DECIMAL(10,2),
    costo DECIMAL(10,2),
    margen DECIMAL(5,2),
    stock_actual INTEGER,
    stock_minimo INTEGER,
    disponible BOOLEAN,
    fecha_lanzamiento DATE,
    temporada VARCHAR(20)
);

-- 5. Dimensión Geografía: Ubicación jerárquica
CREATE TABLE dim_geografia (
    id INTEGER PRIMARY KEY,
    codigo_postal VARCHAR(10),
    ciudad VARCHAR(50),
    provincia VARCHAR(50),
    region VARCHAR(50),
    pais VARCHAR(50),
    zona_horaria VARCHAR(10),
    densidad_poblacional VARCHAR(20)
);

-- 6. Dimensión Canal: Marketing y adquisición
CREATE TABLE dim_canal_adquisicion (
    id INTEGER PRIMARY KEY,
    nombre_canal VARCHAR(50),
    tipo_canal VARCHAR(20),  -- Pago, Orgánico, Social, Email, etc.
    costo_adquisicion DECIMAL(8,2),
    roi_promedio DECIMAL(5,2),
    tasa_conversion DECIMAL(5,2),
    activo BOOLEAN
);

-- 7. Tablas de soporte para jerarquías
CREATE TABLE dim_categoria (
    id INTEGER PRIMARY KEY,
    nombre VARCHAR(50),
    categoria_padre INTEGER REFERENCES dim_categoria(id),  -- Para jerarquía
    nivel INTEGER,  -- 1=Principal, 2=Subcategoria, etc.
    descripcion TEXT
);

CREATE TABLE dim_marca (
    id INTEGER PRIMARY KEY,
    nombre VARCHAR(50),
    pais_origen VARCHAR(50),
    segmento VARCHAR(20),  -- Premium, Medio, Económico
    reputacion DECIMAL(3,1)  -- Puntuación 1-10
);
```

Comparación de consultas: Normalizado vs Dimensional:

```
-- Consulta en esquema NORMALIZADO (complejo, lento)
SELECT
    c.nombre_cliente,
    p.nombre_producto,
    cat.nombre_categoria,
    SUM(v.cantidad * v.precio_unitario) as total_ventas,
    AVG(v.cantidad * v.precio_unitario) as ticket_promedio
FROM ventas v
JOIN clientes c ON v.id_cliente = c.id
JOIN productos p ON v.id_producto = p.id
JOIN categorias cat ON p.id_categoria = cat.id
WHERE v.fecha_venta BETWEEN '2024-01-01' AND '2024-12-31'
GROUP BY c.nombre_cliente, p.nombre_producto, cat.nombre_categoria
ORDER BY total_ventas DESC;

-- Consulta en esquema DIMENSIONAL (simple, rápido)
SELECT
    dc.nombre as cliente,
    dp.nombre as producto,
    dcat.nombre as categoria,
    SUM(hv.total_neto) as total_ventas,
    AVG(hv.total_neto) as ticket_promedio
FROM hechos_ventas hv
JOIN dim_tiempo dt ON hv.id_tiempo = dt.id
JOIN dim_cliente dc ON hv.id_cliente = dc.id
JOIN dim_producto dp ON hv.id_producto = dp.id
JOIN dim_categoria dcat ON dp.id_categoria = dcat.id
```

```
WHERE dt.año = 2024
GROUP BY dc.nombre, dp.nombre, dcat.nombre
ORDER BY total_ventas DESC;
```

Análisis de trade-offs y recomendaciones:

```
-- Ventajas del diseño dimensional:
-- 1. Consultas más simples y legibles
-- 2. Performance superior para agregaciones
-- 3. Optimizado para herramientas BI
-- 4. Fácil de entender para analistas de negocio

-- Desventajas:
-- 1. Mayor redundancia de datos
-- 2. Más complejo mantenimiento de dimensiones
-- 3. Menos flexible para cambios estructurales

-- Recomendaciones de implementación:

-- 1. Indices estratégicos para performance
CREATE INDEX idx_hechos_tiempo ON hechos_ventas(id_tiempo);
CREATE INDEX idx_hechos_cliente ON hechos_ventas(id_cliente);
CREATE INDEX idx_hechos_producto ON hechos_ventas(id_producto);
CREATE INDEX idx_dimensiones_compuestas ON hechos_ventas(id_tiempo, id_cliente, id_producto);

-- 2. Particionamiento por tiempo para datasets grandes
CREATE TABLE hechos_ventas_y2024 PARTITION OF hechos_ventas
    FOR VALUES FROM ('2024-01-01') TO ('2025-01-01');

-- 3. Vistas materializadas para consultas frecuentes
CREATE MATERIALIZED VIEW mv_ventas_mensuales AS
SELECT
    dt.año,
    dt.mes,
    SUM(hv.total_neto) as ventas_total,
    COUNT(DISTINCT hv.id_cliente) as clientes_unicos,
    AVG(hv.total_neto) as ticket_promedio
FROM hechos_ventas hv
JOIN dim_tiempo dt ON hv.id_tiempo = dt.id
GROUP BY dt.año, dt.mes;

-- 4. Constraints para integridad
ALTER TABLE hechos_ventas ADD CONSTRAINT ck_total_neto_positivo
    CHECK (total_neto > 0);
ALTER TABLE dim_cliente ADD CONSTRAINT ck_segmento_valido
    CHECK (segmento_valor IN ('Bronce', 'Plata', 'Oro', 'Platino'));
```

Verificación: Compara cómo el diseño dimensional simplifica consultas analíticas complejas, y explica por qué este esquema está optimizado específicamente para análisis en lugar de operaciones transaccionales.

Requerimientos:

PostgreSQL o MySQL para ejemplos prácticos
DBeaver o pgAdmin para diseño visual
Conocimientos básicos de SQL DDL
Entorno de desarrollo de bases de datos