

Data Warehouse para Gestión de Transporte y Logística

Documento Técnico Académico

1. Introducción

1.1 Contexto

Este documento presenta el diseño de un Data Warehouse (DW) para un sistema de gestión de transporte y logística, basado en la metodología de **Ralph Kimball** específicamente en el **Capítulo 12: Transporte** de su libro "The Data Warehouse Toolkit".

1.2 Objetivo

Proporcionar una estructura dimensional que permita el análisis de operaciones de transporte, incluyendo:

- Análisis de rentabilidad por cliente, ruta y tipo de servicio
 - Medición de eficiencia operativa (tiempos de tránsito, retrasos)
 - Evaluación del desempeño de vehículos y rutas
 - Análisis geográfico de origen-destino
 - Tendencias temporales en el negocio de transporte
-

2. Fundamentos Teóricos

2.1 Modelado Dimensional

El **modelado dimensional** es una técnica de diseño de bases de datos optimizada para consultas analíticas y reportes. A diferencia del modelado relacional normalizado (usado en sistemas transaccionales), el modelado dimensional prioriza:

- **Simplicidad:** Esquemas fáciles de entender para usuarios de negocio
- **Performance:** Consultas más rápidas mediante desnormalización controlada
- **Flexibilidad:** Facilita el análisis ad-hoc y drill-down

2.2 Esquema Estrella (Star Schema)

Es el modelo dimensional fundamental que consiste en:

- **Una tabla de hechos central:** Contiene las métricas numéricas del negocio
- **Múltiples tablas de dimensiones:** Proporcionan el contexto descriptivo

Ventajas del esquema estrella:

- Consultas SQL simples con pocos JOINS
- Excelente performance en herramientas de BI
- Fácil comprensión para usuarios finales
- Compatible con técnicas de optimización de bases de datos

2.3 Conceptos Clave de Kimball

2.3.1 Granularidad

La **granularidad** define el nivel de detalle de cada fila en la tabla de hechos. En nuestro caso:

- **Granularidad elegida:** Un registro por envío completado
- **Justificación:** Permite análisis detallado manteniendo volumen manejable

2.3.2 Surrogate Keys

Son claves artificiales (generalmente números enteros secuenciales) que:

- Independizan el DW de los sistemas operacionales
- Facilitan el manejo de Slowly Changing Dimensions
- Mejoran la performance (claves numéricas pequeñas)
- Permiten integración de múltiples fuentes

2.3.3 Slowly Changing Dimensions (SCD)

Técnicas para manejar cambios en dimensiones a lo largo del tiempo:

Tipo 1 - Sobrescritura: Actualiza el valor sin mantener historial

- Ejemplo: Corrección de error tipográfico en nombre

Tipo 2 - Versionado (implementado en nuestro modelo):

- Crea nueva fila cuando cambia el atributo
- Mantiene historial completo
- Campos: `version`, `fecha_inicio_vigencia`, `fecha_fin_vigencia`, `es_actual`
- Ejemplo: Cliente cambia de segmento "Bronze" a "Gold"

Tipo 3 - Campo adicional: Mantiene valor anterior y actual

- Menos común, para cambios ocasionales

2.3.4 Dimensión Junk

Consolida múltiples atributos de baja cardinalidad en una sola dimensión para:

- Reducir el número de dimensiones
- Evitar explosión de combinaciones en la tabla de hechos
- Mejorar performance
- Simplificar el modelo

2.3.5 Dimensión Degenerada

Un atributo dimensional que se almacena directamente en la tabla de hechos:

- No tiene dimensión propia
 - Generalmente números de transacción/orden
 - Ejemplo: `shipment_number` (número de guía de envío)
-

3. Descripción del Modelo

3.1 Arquitectura General

El modelo implementa un **esquema estrella puro** con:

- 1 tabla de hechos (F_Shipment)
- 6 tablas de dimensiones conformadas
- 1 dimensión junk para atributos transaccionales

3.2 Tabla de Hechos: F_Shipment

3.2.1 Propósito

Registra cada envío completado con sus métricas de negocio y referencias a dimensiones.

3.2.2 Estructura

Claves:

- `shipment_key` (PK): Surrogate key, identificador único del DW
- `shipment_number`: Dimensión degenerada, número de guía del sistema operacional
- Foreign Keys: 8 claves foráneas a dimensiones

Métricas Aditivas:

- `ingresos`: Ingresos generados por el envío
- `costo_total`: Costos operativos totales
- `distancia_km`: Distancia recorrida
- `peso_kg`: Peso total de la carga

- `volumen_m3`: Volumen ocupado
- `cantidad_paradas`: Número de paradas realizadas
- `cantidad_items`: Cantidad de ítems transportados

Métricas Semi-Aditivas:

- `tiempo_transito_min`: Tiempo real de tránsito
- `retraso_min`: Minutos de retraso respecto al plan

Métricas Derivadas (calculadas, no almacenadas):

- `margen = ingresos - costo_total`
- `margen_porcentual = (margen / ingresos) * 100`
- `variacion_distancia = distancia_km - distancia_planificada_km`

3.2.3 Justificación de Diseño

- **No se almacena margen**: Se calcula dinámicamente para evitar inconsistencias
- **Múltiples métricas de negocio**: Permiten análisis desde diferentes perspectivas
- **Tiempos en minutos**: Granularidad suficiente para análisis operativo

3.3 Dimensión: D_Fecha

3.3.1 Importancia

La dimensión fecha es **crítica** en todo DW. Es la dimensión más consultada y permite:

- Análisis de tendencias temporales
- Comparaciones período sobre período
- Estacionalidad y patrones cíclicos

3.3.2 Estructura

Atributos de Identificación:

- `fecha_key` (PK): Surrogate key (formato común: YYYYMMDD como integer)
- `fecha_completa`: Fecha real (date type)

Jerarquía Temporal:

- `año`: 2024, 2025
- `trimestre`: 1, 2, 3, 4
- `mes`: 1-12
- `semana`: 1-53 (semana del año)
- `día_mes`: 1-31
- `día_año`: 1-366

Atributos Descriptivos:

- `dia_semana`: Lunes, Martes, ..., Domingo
- `nombre_mes`: Enero, Febrero, ..., Diciembre
- `trimestre_desc`: "Q1 2024", "Q2 2024"
- `temporada`: Verano, Otoño, Invierno, Primavera

Atributos de Negocio:

- `es_fin_semana`: Boolean (TRUE/FALSE)
- `es_festivo`: Boolean (identifica días no laborables)

3.3.3 Uso en el Modelo

La tabla de hechos tiene **DOS referencias** a D_Fecha:

- `fecha_salida_key`: Cuándo inició el envío
- `fecha_llegada_key`: Cuándo se completó el envío

Esto permite análisis como:

- Envíos iniciados por mes vs completados por mes
- Tiempo promedio entre salida y llegada
- Retrasos por día de la semana

3.3.4 Poblamiento

Típicamente se pre-carga con 10-20 años de datos (pasado y futuro).

3.4 Dimensión: D_Cliente (con SCD Tipo 2)

3.4.1 Propósito

Describe los clientes del servicio de transporte, manteniendo historial de cambios.

3.4.2 Estructura

Claves e Identificación:

- `cliente_key` (PK): Surrogate key
- `version`: Número de versión del registro
- `codigo_cliente`: Código de negocio (natural key)

Atributos Descriptivos:

- `nombre`: Razón social o nombre del cliente
- `segmento`: Bronze, Silver, Gold, Platinum

- **tipo_cliente**: Corporativo, PYME, Individual
- **industria**: Retail, Manufactura, Farmacéutica, etc.

Atributos Geográficos:

- **region**: Norte, Sur, Centro, Este, Oeste
- **pais**: Argentina, Brasil, Chile, etc.

Atributos Temporales:

- **fecha_alta**: Cuándo se dio de alta el cliente
- **fecha_inicio_vigencia**: Inicio de validez de esta versión
- **fecha_fin_vigencia**: Fin de validez (NULL si es actual)
- **es_actual**: Boolean (TRUE solo para versión vigente)

3.4.3 Implementación SCD Tipo 2

Escenario de Ejemplo:

Estado Inicial:

cliente_key	version	codigo_cliente	nombre	segmento	es_actual	fecha_inicio	fecha_fin
1001	1	CLI-001	Empresa X	Bronze	TRUE	2023-01-15	NULL

Cambio: Cliente es promovido a segmento Gold el 2024-03-01

Estado Final:

cliente_key	version	codigo_cliente	nombre	segmento	es_actual	fecha_inicio	fecha_fin
1001	1	CLI-001	Empresa X	Bronze	FALSE	2023-01-15	2024-02-29
1002	2	CLI-001	Empresa X	Gold	TRUE	2024-03-01	NULL

Ventajas:

- Los envíos históricos mantienen el segmento que tenía el cliente en ese momento
- Permite análisis "as-was" (como era) y "as-is" (como es)
- Preguntas respondibles: "¿Cuánto facturamos a clientes que ERAN Bronze pero AHORA son Gold?"

3.5 Dimensión: D_Vehiculo (con SCD Tipo 2)

3.5.1 Propósito

Describe la flota de vehículos, con tracking de cambios en el tiempo.

3.5.2 Estructura

Identificación:

- `vehiculo_key` (PK): Surrogate key
- `version`: Versión del registro
- `matricula`: Patente/placa del vehículo

Características del Vehículo:

- `tipo_vehiculo`: Camión, Camioneta, Trailer, Contenedor
- `marca`: Mercedes-Benz, Scania, Volvo
- `modelo`: FH16, Actros, etc.
- `año_fabricacion`: 2018, 2019, 2020

Capacidades:

- `capacidad_kg`: Capacidad máxima de peso
- `capacidad_m3`: Capacidad volumétrica

Atributos Operacionales:

- `estado_vehiculo`: Operativo, Mantenimiento, Fuera de Servicio
- `proveedor_transporte`: Propio, Tercerizado (nombre del proveedor)

Control de Versiones:

- `fecha_inicio_vigencia`
- `fecha_fin_vigencia`
- `es_actual`

3.5.3 Casos de Uso SCD

- Cambio de proveedor (de propio a tercerizado)
- Cambio de estado (operativo → mantenimiento)
- Actualización de capacidades tras modificación del vehículo

3.6 Dimensión: D_Ruta

3.6.1 Propósito

Define las rutas planificadas entre orígenes y destinos.

3.6.2 Estructura

Identificación:

- `ruta_key` (PK)
- `codigo_ruta`: Código de negocio (ej: "RTA-001")
- `nombre_ruta`: "Buenos Aires - Córdoba", "São Paulo - Río"

Características de Planificación:

- `tipo_ruta`: Urbana, Interurbana, Internacional
- `distancia_planificada_km`: Distancia teórica de la ruta
- `tiempo_estimado_min`: Tiempo esperado de tránsito

Atributos de Complejidad:

- `nivel_dificultad`: Fácil, Media, Difícil
- `tiene_peajes`: Boolean
- `cantidad_peajes`: Número de peajes en la ruta

Atributos Geográficos:

- `zona_geografica`: Norte, Sur, Centro, Costa, Montaña

3.6.3 Análisis Posibles

- Rentabilidad por tipo de ruta
 - Rutas con mayor índice de retrasos
 - Comparación distancia planificada vs real
 - Impacto de peajes en costos
-

3.7 Dimensión: D_Lugar

3.7.1 Propósito

Representa ubicaciones geográficas (ciudades, almacenes, puntos de entrega).

3.7.2 Estructura

Identificación:

- `lugar_key` (PK)
- `codigo_postal`: Código postal de la ubicación

Jerarquía Geográfica (para drill-down/up):

Continente

└ Pais

└ Provincia/Estado

└ Ciudad

Atributos:

- **ciudad**: Buenos Aires, Córdoba, São Paulo
- **provincia_estado**: Buenos Aires, São Paulo, etc.
- **país**: Argentina, Brasil, Uruguay
- **region**: LATAM, América del Norte
- **continente**: América del Sur, América del Norte

Atributos Adicionales:

- **latitud / longitud**: Coordenadas GPS
- **zona_horaria**: America/Argentina/Buenos_Aires
- **tipo_ubicacion**: Centro de Distribución, Puerto, Aeropuerto, Cliente

3.7.3 Uso en el Modelo

La tabla de hechos referencia D_Lugar **DOS veces**:

- **origen_key**: Punto de partida del envío
- **destino_key**: Punto de llegada del envío

3.7.4 Análisis Geográficos

- Matriz origen-destino de volúmenes
- Rutas más rentables por ciudad
- Análisis de clusters geográficos
- Mapas de calor de entregas
- Drill-down: País → Provincia → Ciudad

3.8 Dimensión: D_Junk

3.8.1 Concepto

Una **dimensión junk** consolida múltiples atributos de baja cardinalidad que:

- Son principalmente flags o códigos
- Tienen pocas combinaciones posibles
- No justifican dimensiones separadas

3.8.2 Estructura

Identificación:

- **junk_key** (PK)

Atributos de Servicio:

- `tipo_servicio`: Express, Estándar, Económico
- `desc_tipo_servicio`: Descripción detallada
- `prioridad`: Alta, Media, Baja

Atributos de Estado:

- `estado_envio`: Completado, En Tránsito, Cancelado, Con Incidencias
- `desc_estado`: Descripción del estado

Atributos de Carga:

- `tipo_carga`: Paquetería, Paletizado, Granel, Contenedor
- `requiere_refrigeracion`: Boolean
- `es_fragil`: Boolean
- `requiere_seguro`: Boolean

3.8.3 Poblamiento

Se generan todas las combinaciones válidas de atributos:

junk_key | tipo_servicio | estado_envio | prioridad | es_fragil | requiere_refrig.

1	Express	Completado	Alta	TRUE	FALSE
2	Express	Completado	Alta	TRUE	TRUE
3	Express	Completado	Alta	FALSE	FALSE
...					

Cardinalidad Estimada:

- 3 tipos de servicio × 4 estados × 3 prioridades × 2 frágil × 2 refrigeración × 2 seguro × 4 tipos carga
- = ~576 combinaciones (mucho menor que 7 dimensiones separadas)

3.8.4 Ventajas

- Reduce dimensiones de 7 a 1
- Mejora performance (menos JOINS)
- Simplifica el modelo visualmente
- Facilita consultas con múltiples filtros de estos atributos

4. Relaciones y Cardinalidades

4.1 Modelo de Relaciones

Todas las relaciones son **muchos-a-uno** desde la tabla de hechos hacia dimensiones:

F_Shipment (N) —→ (1) D_Fecha [fecha_salida]
F_Shipment (N) —→ (1) D_Fecha [fecha_llegada]
F_Shipment (N) —→ (1) D_Cliente
F_Shipment (N) —→ (1) D_Vehiculo
F_Shipment (N) —→ (1) D_Ruta
F_Shipment (N) —→ (1) D_Lugar [origen]
F_Shipment (N) —→ (1) D_Lugar [destino]
F_Shipment (N) —→ (1) D_Junk

4.2 Interpretación

- Muchos envíos pueden ocurrir en la misma fecha
 - Muchos envíos pueden ser del mismo cliente
 - Muchos envíos pueden usar el mismo vehículo
 - Muchos envíos pueden seguir la misma ruta
 - Muchos envíos pueden tener el mismo origen/destino
 - Muchos envíos pueden tener la misma combinación de atributos junk
-

5. Casos de Uso y Consultas Analíticas

5.1 Análisis de Rentabilidad

Pregunta de Negocio: "¿Cuál es el margen promedio por segmento de cliente en el Q1 2024?"

SQL Conceptual:

```
SELECT
    c.segmento,
    SUM(f.ingresos - f.costo_total) as margen_total,
    AVG(f.ingresos - f.costo_total) as margen_promedio,
    COUNT(*) as cantidad_envios
FROM F_Shipment f
JOIN D_Cliente c ON f.cliente_key = c.cliente_key
JOIN D_Fecha d ON f.fecha_salida_key = d.fecha_key
WHERE d.año = 2024 AND d.trimestre = 1
    AND c.es_actual = TRUE
GROUP BY c.segmento
ORDER BY margen_total DESC;
```

5.2 Análisis de Eficiencia Operativa

Pregunta: "¿Qué rutas tienen mayor índice de retraso promedio?"

SQL Conceptual:

```
SELECT
    r.nombre_ruta,
    r.tipo_ruta,
    AVG(f.retraso_min) as retraso_promedio,
    AVG(f.tiempo_transito_min) as tiempo_transito_promedio,
    AVG(f.distancia_km - r.distancia_planificada_km) as desviacion_distancia
FROM F_Shipment f
JOIN D_Ruta r ON f.ruta_key = r.ruta_key
GROUP BY r.nombre_ruta, r.tipo_ruta
HAVING AVG(f.retraso_min) > 30
ORDER BY retraso_promedio DESC;
```

5.3 Análisis Geográfico

Pregunta: "¿Cuáles son los corredores origen-destino más rentables?"

SQL Conceptual:

```
SELECT
    o.ciudad as ciudad_origen,
    o.provincia_estado as provincia_origen,
    d.ciudad as ciudad_destino,
    d.provincia_estado as provincia_destino,
    COUNT(*) as cantidad_envios,
    SUM(f.ingresos - f.costos_total) as margen_total,
    AVG(f.distancia_km) as distancia_promedio
FROM F_Shipment f
JOIN D_Lugar o ON f.origen_key = o.lugar_key
JOIN D_Lugar d ON f.destino_key = d.lugar_key
GROUP BY o.ciudad, o.provincia_estado, d.ciudad, d.provincia_estado
HAVING COUNT(*) > 100
ORDER BY margen_total DESC
LIMIT 20;
```

5.4 Análisis Temporal con Estacionalidad

Pregunta: "¿Cómo varía el volumen de envíos por día de la semana?"

SQL Conceptual:

```
SELECT
    f.dia_semana,
    COUNT(*) as cantidad_envios,
```

```

SUM(sh.peso_kg) as peso_total,
AVG(sh.ingresos) as ingreso_promedio
FROM F_Shipment sh
JOIN D_Fecha f ON sh.fecha_salida_key = f.fecha_key
WHERE f.año = 2024
GROUP BY f.dia_semana
ORDER BY
CASE f.dia_semana
  WHEN 'Lunes' THEN 1
  WHEN 'Martes' THEN 2
  WHEN 'Miércoles' THEN 3
  WHEN 'Jueves' THEN 4
  WHEN 'Viernes' THEN 5
  WHEN 'Sábado' THEN 6
  WHEN 'Domingo' THEN 7
END;

```

5.5 Análisis de Performance de Vehículos

Pregunta: "¿Qué vehículos tienen mejor eficiencia (ingresos/km recorrido)?"

SQL Conceptual:

```

SELECT
  v.matricula,
  v.tipo_vehiculo,
  v.marca,
  v.proveedor_transporte,
  COUNT(*) as viajes_realizados,
  SUM(f.distancia_km) as km_totales,
  SUM(f.ingresos) as ingresos_totales,
  SUM(f.ingresos) / SUM(f.distancia_km) as ingreso_por_km
FROM F_Shipment f
JOIN D_Vehiculo v ON f.vehiculo_key = v.vehiculo_key
WHERE v.es_actual = TRUE
GROUP BY v.matricula, v.tipo_vehiculo, v.marca, v.proveedor_transporte
HAVING COUNT(*) > 50
ORDER BY ingreso_por_km DESC;

```

5.6 Análisis con Dimensión Junk

Pregunta: "¿Cuál es el costo promedio por tipo de servicio y características de carga?"

SQL Conceptual:

```

SELECT

```

```
j.tipo_servicio,  
j.tipo_carga,  
j.requiere_refrigeracion,  
j.es_fragil,  
COUNT(*) as cantidad_envios,  
AVG(f.costos_total) as costo_promedio,  
AVG(f.ingresos) as ingreso_promedio  
FROM F_Shipment f  
JOIN D_Junk j ON f.junk_key = j.junk_key  
GROUP BY  
j.tipo_servicio,  
j.tipo_carga,  
j.requiere_refrigeracion,  
j.es_fragil  
ORDER BY cantidad_envios DESC;
```

6. Proceso ETL Conceptual

6.1 Extracción (Extract)

Fuentes de Datos:

- Sistema transaccional de órdenes de transporte (OLTP)
- Sistema de gestión de flota
- Sistema de facturación
- Datos externos: festivos, zonas horarias

Frecuencia:

- Diaria (carga nocturna) para datos completados
- Tiempo real/near-real-time para dashboards operativos

6.2 Transformación (Transform)

Dimensiones:

1. **D_Fecha:** Pre-cargada, actualización anual
2. **D_Cliente** (SCD Tipo 2):
 - Comparar con versión actual en DW
 - Si cambió atributo rastreado (ej: segmento):
 - Cerrar registro actual (actualizar `fecha_fin_vigencia`, `es_actual = FALSE`)
 - Insertar nuevo registro con nueva versión

3. **D_Vehiculo** (SCD Tipo 2):
 - Similar a cliente
 - Rastrear cambios en estado, proveedor, capacidades
4. **D_Ruta**: SCD Tipo 1 (sobrescritura)
 - Las rutas no suelen cambiar sus características históricas
5. **D_Lugar**: SCD Tipo 1
 - Datos geográficos estables
6. **D_Junk**:
 - Pre-poblada con todas las combinaciones válidas
 - Lookup para asignar **junk_key** correcto

Tabla de Hechos:

- Obtener surrogate keys de dimensiones mediante lookups
- Calcular métricas derivadas si es necesario
- Validar integridad referencial
- Aplicar reglas de negocio (ej: margen no puede ser negativo en más de X%)

6.3 Carga (Load)

Estrategias:

- **Dimensiones**: Upsert (Update + Insert)
 - Usar merge/upsert para SCD
 - Bulk insert para dimensiones estáticas
- **Hechos**: Insert incremental
 - Solo nuevos registros desde última carga
 - Partition by fecha para mejor performance

Validaciones Post-Carga:

- Conteos de registros source vs target
- Validación de totales (suma de ingresos, costos)
- Verificación de integridad referencial
- Auditoría de surrogate keys no resueltas

7. Consideraciones de Implementación

7.1 Tecnologías Recomendadas

Bases de Datos:

- PostgreSQL (con extensiones para DW)
- Microsoft SQL Server (con columnstore indexes)
- Snowflake / Redshift (cloud data warehouses)
- BigQuery (Google Cloud)

Herramientas ETL:

- Apache Airflow
- Talend
- Microsoft SSIS
- Informatica PowerCenter
- Pentaho Data Integration

Herramientas de BI:

- Tableau
- Power BI
- Looker
- Qlik Sense
- Metabase (open source)

7.2 Índices y Optimización

Tabla de Hechos:

- Índice clustered en `shipment_key`
- Índices no-clustered en foreign keys frecuentemente filtradas
- Particionamiento por `fecha_salida_key` (mensual/trimestral)
- Columnstore index para consultas analíticas (SQL Server)

Dimensiones:

- Índice clustered en surrogate key
- Índices en natural keys (códigos de negocio)
- Índices en atributos frecuentemente filtrados
- Para SCD Tipo 2: índice compuesto en (codigo_natural, es_actual)

7.3 Estimación de Volúmenes

Supuestos para empresa mediana:

- 10,000 envíos/día = 3.6M envíos/año
- 5 años de historia = 18M registros en F_Shipment

Tamaño Estimado:

- F_Shipment: $\sim 150 \text{ bytes/registro} \times 18\text{M} = 2.7 \text{ GB}$

- Dimensiones: < 100 MB combinadas
- Índices: ~30% del tamaño de datos = 1 GB
- **Total:** ~4-5 GB (muy manejable)

7.4 Estrategia de Particionamiento

Tabla de Hechos - Particionamiento por Rango:

-- Ejemplo PostgreSQL

```
CREATE TABLE f_shipment (
```

```
...
```

```
) PARTITION BY RANGE (fecha_salida_key);
```

```
CREATE TABLE f_shipment_2024_q1
```

```
  PARTITION OF f_shipment
```

```
  FOR VALUES FROM (20240101) TO (20240401);
```

```
CREATE TABLE f_shipment_2024_q2
```

```
  PARTITION OF f_shipment
```

```
  FOR VALUES FROM (20240401) TO (20240701);
```

Beneficios:

- Mantenimiento más fácil (drop partition antigua)
- Mejora performance de queries con filtro de fecha
- Backups incrementales más eficientes

7.5 Seguridad y Gobernanza

Control de Acceso:

- Vistas restringidas por rol de usuario
- Row-level security para datos sensibles
- Encriptación de datos en reposo y tránsito

Auditoría:

- Tabla de log de ETL con timestamps, conteos, errores
- Versionado de estructura del DW
- Documentación de cambios en metadatos

Calidad de Datos:

- Validaciones en ETL (nulos, rangos, integridad)
 - Reportes de calidad automáticos
 - Proceso de data profiling periódico
-

8. Evolución y Escalamiento

8.1 Extensiones Futuras Posibles

Nuevas Dimensiones:

- D_Conductor: Información del chofer asignado
- D_Tipo_Incidencia: Catálogo de problemas en tránsito
- D_Condicion_Climatica: Datos meteorológicos
- D_Tipo_Mercancia: Clasificación detallada de productos

Nuevas Tablas de Hechos:

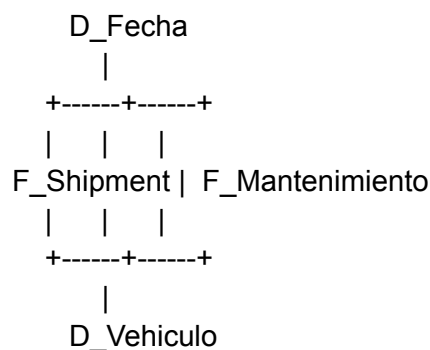
- F_Incidencia: Eventos durante el transporte (granularidad más fina)
- F_Mantenimiento_Vehiculo: Tracking de mantenimientos y reparaciones
- F_Cotizacion: Análisis de conversión de cotizaciones a envíos
- F_Facturacion: Hechos específicos del ciclo de facturación y cobranza

Métricas Adicionales:

- Emisiones de CO2 por envío (sostenibilidad)
- Índice de satisfacción del cliente (NPS)
- Tiempo de carga/descarga
- Consumo de combustible

8.2 Patrón de Constelación (Constellation Schema)

A medida que el DW crece, múltiples tablas de hechos pueden compartir dimensiones conformadas:



Dimensiones Conformadas: Misma estructura y significado en todas las tablas de hechos que las usan.

8.3 Agregados y Cubos OLAP

Para mejorar performance:

Tablas Agregadas Pre-calculadas:







```
-- Agregado mensual por cliente
CREATE TABLE agg_shipment_month_cliente AS
SELECT
    fecha_mes_key,
    cliente_key,
    COUNT(*) as cantidad_envios,
    SUM(ingresos) as ingresos_totales,
    SUM(costo_total) as costos_totales,
    SUM(distancia_km) as distancia_total,
    AVG(retraso_min) as retraso_promedio
FROM F_Shipment f
JOIN D_Fecha d ON f.fecha_salida_key = d.fecha_key
GROUP BY fecha_mes_key, cliente_key;
```

Cubos OLAP multidimensionales:

- Dimensiones: Fecha, Cliente, Ruta, Tipo_Servicio
- Medidas: Ingresos, Costos, Cantidad
- Operaciones: Drill-down, Roll-up, Slice, Dice, Pivot

9. Buenas Prácticas de Kimball Aplicadas

9.1 Principios Fundamentales Implementados

 **Bus Architecture:** Dimensiones conformadas reutilizables
  **Granularidad atómica:** Nivel más bajo de detalle posible
  **Dimensiones desnormalizadas:** Atributos redundantes para simplificar queries
  **Hechos aditivos:** Métricas sumables a través de dimensiones
  **Surrogate keys:** Independencia del sistema operacional
  **SCD para historial:** Tracking de cambios importantes

9.2 Reglas de Oro del Diseño Dimensional

1. **"Carga una vez, consulta muchas veces"**
 - Optimizar para lectura, no para escritura
 - ETL puede ser complejo, pero queries deben ser simples
2. **"El usuario de negocio primero"**
 - Modelo debe ser intuitivo para analistas
 - Nombres de tablas y columnas en lenguaje de negocio
 - Evitar tecnicismos innecesarios
3. **"Dimensiones robustas y descriptivas"**
 - Muchos atributos descriptivos en dimensiones

- Facilitar filtrado y agrupamiento flexible
 - Preferir desnormalización sobre normalización
4. **"Mantener la historia"**
- SCD para cambios importantes
 - Nunca eliminar datos históricos (soft deletes)
 - Permitir análisis "as-was"
5. **"Integridad referencial estricta"**
- Todas las FK deben resolver a dimensiones
 - Usar registros "Unknown" o "Not Applicable" en dimensiones
 - Nunca NULL en foreign keys

9.3 Antipatrones a Evitar

❌ **Snowflaking excesivo**: No normalizar dimensiones sin razón fuerte ❌ **Hechos demasiado agregados**: Perder granularidad atómica ❌ **Demasiadas dimensiones pequeñas**: Usar dimension junk ❌ **Métricas calculables en hechos**: Almacenar solo componentes base ❌ **Cambiar granularidad**: Mantener consistencia en tabla de hechos ❌ **Dimensiones cambiantes sin SCD**: Perder contexto histórico

10. Validación del Modelo

10.1 Checklist de Calidad

Tabla de Hechos:

- ☒ Granularidad claramente definida
- ☒ Solo métricas numéricas (no textos descriptivos)
- ☒ Foreign keys a todas las dimensiones relevantes
- ☒ Surrogate key como PK
- ☒ Métricas mayormente aditivas

Dimensiones:

- ☒ Surrogate key como PK
- ☒ Natural key presente (código de negocio)
- ☒ Atributos descriptivos ricos
- ☒ SCD implementado donde es necesario
- ☒ Jerarquías navegables

General:

- ☒ Esquema estrella puro (no snowflake)
- ☒ Nombres claros en lenguaje de negocio
- ☒ Integridad referencial garantizada

- ☒ Documentación completa

10.2 Pruebas de Validación

Consultas de Prueba:

1. Test de Totalización:

-- Los totales deben cuadrar con sistema source

```
SELECT SUM(ingresos), COUNT(*)  
FROM F_Shipment  
WHERE fecha_salida_key BETWEEN 20240101 AND 20240131;
```

2. Test de Integridad Referencial:

-- No deben haber FK sin resolver

```
SELECT COUNT(*) as huerfanos  
FROM F_Shipment f  
WHERE NOT EXISTS (  
    SELECT 1 FROM D_Cliente c  
    WHERE c.cliente_key = f.cliente_key  
);
```

3. Test de SCD:

-- Cada cliente debe tener exactamente 1 registro actual

```
SELECT codigo_cliente, COUNT(*)  
FROM D_Cliente  
WHERE es_actual = TRUE  
GROUP BY codigo_cliente  
HAVING COUNT(*) > 1;
```

4. Test de Performance:

-- Query típico debe ejecutar < 3 segundos

```
EXPLAIN ANALYZE  
SELECT c.segmento, SUM(f.ingresos)  
FROM F_Shipment f  
JOIN D_Cliente c ON f.cliente_key = c.cliente_key  
JOIN D_Fecha d ON f.fecha_salida_key = d.fecha_key  
WHERE d.año = 2024  
GROUP BY c.segmento;
```

11. Ejemplo Práctico Completo

11.1 Escenario de Negocio

Empresa: "TransLogistics S.A." **Operación:** 500 envíos/día, 5 países de LATAM
Necesidad: Dashboard ejecutivo con KPIs clave

11.2 Datos de Ejemplo

Dimensión D_Cliente:

cliente_key	version	codigo_cliente	nombre	segmento	es_actual
1001	1	CLI-001	Farmacias XYZ	Gold	TRUE
1002	1	CLI-002	Retail ABC	Silver	TRUE
1003	2	CLI-003	E-commerce 123	Platinum	TRUE

Dimensión D_Fecha (muestra):

fecha_key	fecha_completa	año	mes	dia_semana	es_festivo
20240315	2024-03-15	2024	3	Viernes	FALSE
20240316	2024-03-16	2024	3	Sábado	FALSE
20240325	2024-03-25	2024	3	Lunes	TRUE

Tabla de Hechos F_Shipment (muestra):

shipment_key	shipment_number	fecha_salida_key	cliente_key	ingresos	costo_total	distancia_km	retraso_min
5001	SHIP-2024-0001	20240315	1001	1500.00	1100.00	450.5	0
5002	SHIP-2024-0002	20240315	1002	2300.00	1800.00	780.2	30
5003	SHIP-2024-0003	20240316	1003	5000.00	3200.00	1200.0	0

11.3 Query de Dashboard Ejecutivo

```
-- KPIs del mes actual
WITH mes_actual AS (
  SELECT
    COUNT(*) as total_envios,
    SUM(ingresos) as ingresos_totales,
    SUM(costo_total) as costos_totales,
    SUM(ingresos - costo_total) as margen_total,
    AVG(retraso_min) as retraso_promedio,
    SUM(CASE WHEN retraso_min = 0 THEN 1 ELSE 0 END) * 100.0 / COUNT(*) as
pct_on_time
  FROM F_Shipment f
  JOIN D_Fecha d ON f.fecha_salida_key = d.fecha_key
```

```

WHERE d.año = 2024 AND d.mes = 3
),
por_segmento AS (
  SELECT
    c.segmento,
    COUNT(*) as envios,
    SUM(f.ingresos) as ingresos,
    SUM(f.ingresos - f.costos_total) as margen
  FROM F_Shipment f
  JOIN D_Cliente c ON f.cliente_key = c.cliente_key
  JOIN D_Fecha d ON f.fecha_salida_key = d.fecha_key
  WHERE d.año = 2024 AND d.mes = 3 AND c.es_actual = TRUE
  GROUP BY c.segmento
),
top_rutas AS (
  SELECT
    r.nombre_ruta,
    COUNT(*) as envios,
    AVG(f.retraso_min) as retraso_avg
  FROM F_Shipment f
  JOIN D_Ruta r ON f.ruta_key = r.ruta_key
  JOIN D_Fecha d ON f.fecha_salida_key = d.fecha_key
  WHERE d.año = 2024 AND d.mes = 3
  GROUP BY r.nombre_ruta
  ORDER BY envios DESC
  LIMIT 10
)
SELECT * FROM mes_actual
UNION ALL
SELECT * FROM por_segmento
UNION ALL
SELECT * FROM top_rutas;

```

Resultado Dashboard:

=== KPIs GENERALES MARZO 2024 ===

Total Envíos: 15,000

Ingresos: \$22,500,000

Margen: \$6,750,000 (30%)

On-Time Delivery: 87.5%

Retraso Promedio: 18 min

=== POR SEGMENTO ===

Platinum: 3,200 envíos | \$8.5M | Margen 35%

Gold: 5,100 envíos | \$9.2M | Margen 28%

Silver: 4,800 envíos | \$3.8M | Margen 25%

Bronze: 1,900 envíos | \$1.0M | Margen 20%

=== TOP 5 RUTAS ===

1. Buenos Aires - Córdoba: 2,100 envíos | 15 min retraso
 2. São Paulo - Río: 1,850 envíos | 22 min retraso
 3. Santiago - Valparaíso: 1,200 envíos | 8 min retraso
-

12. Glosario de Términos

Aditivo: Métrica que puede sumarse significativamente a través de todas las dimensiones (ej: ingresos, cantidad).

Bus Architecture: Framework de Kimball para construir DW empresarial con dimensiones conformadas compartidas.

Cardinalidad: Número de valores únicos distintos en un atributo o dimensión.

Dimensión Conformada: Dimensión con estructura y contenido idéntico usada en múltiples procesos de negocio.

Dimensión Degenerada: Atributo dimensional que se almacena en la tabla de hechos sin tener su propia tabla de dimensión.

Dimensión Junk: Dimensión que consolida múltiples flags y códigos de baja cardinalidad.

Drill-Down: Navegación desde nivel agregado a nivel más detallado (Año → Trimestre → Mes → Día).

ETL: Extract, Transform, Load - proceso de mover datos desde fuentes al DW.

Fact Table (Tabla de Hechos): Tabla central que contiene métricas de negocio y foreign keys a dimensiones.

Granularidad: Nivel de detalle de cada registro en la tabla de hechos.

Natural Key: Identificador de negocio del sistema operacional (ej: código de cliente).

OLAP: Online Analytical Processing - consultas analíticas complejas.

OLTP: Online Transaction Processing - sistemas transaccionales operacionales.

Roll-Up: Agregación desde nivel detallado a nivel más alto (inverso de drill-down).

Schema Estrella (Star Schema): Modelo dimensional con tabla de hechos central rodeada de dimensiones.

SCD (Slowly Changing Dimension): Técnicas para manejar cambios en dimensiones a lo largo del tiempo.

Semi-Aditivo: Métrica que puede sumarse en algunas dimensiones pero no en todas (ej: saldos, inventarios).

Snowflake Schema: Variante del esquema estrella donde dimensiones están normalizadas (generalmente evitado).

Surrogate Key: Clave artificial (generalmente integer secuencial) usada como PK en DW.

13. Referencias y Bibliografía

13.1 Fuentes Principales

1. **Kimball, Ralph & Ross, Margy. (2013)** *"The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling"* 3rd Edition, Wiley
 - Capítulo 12: Transportation
2. **Kimball, Ralph et al. (2008)** *"The Data Warehouse Lifecycle Toolkit"* 2nd Edition, Wiley
3. **Inmon, William H. (2005)** *"Building the Data Warehouse"* 4th Edition, Wiley

13.2 Recursos Adicionales

Sitios Web:

- Kimball Group: <https://www.kimballgroup.com>
- Design Tips (Kimball Group): Artículos sobre técnicas específicas
- TDWI (The Data Warehousing Institute): <https://tdwi.org>

Artículos Clave:

- "Slowly Changing Dimensions" - Ralph Kimball
- "Dimensional Modeling Techniques" - Kimball Group
- "Handling Late Arriving Facts and Dimensions"

Comunidades:

- Stack Overflow: Tag [data-warehouse]
- Reddit: r/dataengineering, r/BusinessIntelligence
- LinkedIn Groups: Data Warehousing Professionals

13.3 Herramientas Mencionadas

- **PostgreSQL:** <https://www.postgresql.org>
- **Apache Airflow:** <https://airflow.apache.org>

- **Tableau:** <https://www.tableau.com>
 - **Power BI:** <https://powerbi.microsoft.com>
 - **dbt (data build tool):** <https://www.getdbt.com>
-

14. Apéndices

14.1 Script DDL Conceptual

```
-- =====
-- CREACIÓN DE DIMENSIONES
-- =====

-- Dimensión Fecha
CREATE TABLE D_Fecha (
  fecha_key INTEGER PRIMARY KEY,
  fecha_completa DATE NOT NULL,
  año INTEGER NOT NULL,
  trimestre INTEGER NOT NULL,
  mes INTEGER NOT NULL,
  semana INTEGER NOT NULL,
  dia_mes INTEGER NOT NULL,
  dia_año INTEGER NOT NULL,
  dia_semana VARCHAR(20) NOT NULL,
  nombre_mes VARCHAR(20) NOT NULL,
  trimestre_desc VARCHAR(10) NOT NULL,
  es_fin_semana BOOLEAN NOT NULL,
  es_festivo BOOLEAN NOT NULL,
  temporada VARCHAR(20)
);

CREATE INDEX idx_fecha_año_mes ON D_Fecha(año, mes);
CREATE INDEX idx_fecha_completa ON D_Fecha(fecha_completa);

-- Dimensión Cliente (SCD Tipo 2)
CREATE TABLE D_Cliente (
  cliente_key INTEGER PRIMARY KEY,
  version INTEGER NOT NULL,
  codigo_cliente VARCHAR(50) NOT NULL,
  nombre VARCHAR(200) NOT NULL,
  segmento VARCHAR(50),
  tipo_cliente VARCHAR(50),
  region VARCHAR(100),
  pais VARCHAR(100),
  industria VARCHAR(100),
  fecha_alta DATE,
  fecha_inicio_vigencia DATE NOT NULL,
```

```
    fecha_fin_vigencia DATE,  
    es_actual BOOLEAN NOT NULL  
);
```

```
CREATE INDEX idx_cliente_codigo ON D_Cliente(codigo_cliente);  
CREATE INDEX idx_cliente_codigo_actual ON D_Cliente(codigo_cliente, es_actual);
```

-- Dimensión Vehículo (SCD Tipo 2)

```
CREATE TABLE D_Vehiculo (  
    vehiculo_key INTEGER PRIMARY KEY,  
    version INTEGER NOT NULL,  
    matricula VARCHAR(20) NOT NULL,  
    tipo_vehiculo VARCHAR(50),  
    marca VARCHAR(50),  
    modelo VARCHAR(50),  
    año_fabricacion INTEGER,  
    capacidad_kg DECIMAL(10,2),  
    capacidad_m3 DECIMAL(10,2),  
    estado_vehiculo VARCHAR(50),  
    proveedor_transporte VARCHAR(100),  
    fecha_inicio_vigencia DATE NOT NULL,  
    fecha_fin_vigencia DATE,  
    es_actual BOOLEAN NOT NULL  
);
```

```
CREATE INDEX idx_vehiculo_matricula ON D_Vehiculo(matricula);  
CREATE INDEX idx_vehiculo_matricula_actual ON D_Vehiculo(matricula, es_actual);
```

-- Dimensión Ruta

```
CREATE TABLE D_Ruta (  
    ruta_key INTEGER PRIMARY KEY,  
    codigo_ruta VARCHAR(50) NOT NULL,  
    nombre_ruta VARCHAR(200),  
    tipo_ruta VARCHAR(50),  
    distancia_planificada_km DECIMAL(10,2),  
    tiempo_estimado_min INTEGER,  
    nivel_dificultad VARCHAR(20),  
    tiene_peajes BOOLEAN,  
    cantidad_peajes INTEGER,  
    zona_geografica VARCHAR(100)  
);
```

```
CREATE INDEX idx_ruta_codigo ON D_Ruta(codigo_ruta);
```

-- Dimensión Lugar

```
CREATE TABLE D_Lugar (  
    lugar_key INTEGER PRIMARY KEY,  
    codigo_postal VARCHAR(20),
```

```

ciudad VARCHAR(100) NOT NULL,
provincia_estado VARCHAR(100),
pais VARCHAR(100) NOT NULL,
region VARCHAR(100),
continente VARCHAR(50),
latitud DECIMAL(10,6),
longitud DECIMAL(10,6),
zona_horaria VARCHAR(50),
tipo_ubicacion VARCHAR(50)
);

CREATE INDEX idx_lugar_ciudad ON D_Lugar(ciudad);
CREATE INDEX idx_lugar_pais ON D_Lugar(pais);

```

-- Dimensión Junk

```

CREATE TABLE D_Junk (
    junk_key INTEGER PRIMARY KEY,
    tipo_servicio VARCHAR(50),
    desc_tipo_servicio VARCHAR(200),
    estado_envio VARCHAR(50),
    desc_estado VARCHAR(200),
    prioridad VARCHAR(20),
    tipo_carga VARCHAR(50),
    requiere_refrigeracion BOOLEAN,
    es_fragil BOOLEAN,
    requiere_seguro BOOLEAN
);

```

```

-- =====
-- CREACIÓN DE TABLA DE HECHOS
-- =====

```

```

CREATE TABLE F_Shipment (
    shipment_key BIGSERIAL PRIMARY KEY,
    shipment_number VARCHAR(50) NOT NULL,
    fecha_salida_key INTEGER NOT NULL,
    fecha_llegada_key INTEGER NOT NULL,
    cliente_key INTEGER NOT NULL,
    ruta_key INTEGER NOT NULL,
    vehiculo_key INTEGER NOT NULL,
    origen_key INTEGER NOT NULL,
    destino_key INTEGER NOT NULL,
    junk_key INTEGER NOT NULL,
    ingresos DECIMAL(12,2) NOT NULL,
    costo_total DECIMAL(12,2) NOT NULL,
    distancia_km DECIMAL(10,2),
    peso_kg DECIMAL(10,2),
    volumen_m3 DECIMAL(10,3),

```

```

tiempo_transito_min INTEGER,
retraso_min INTEGER,
cantidad_paradas INTEGER,
cantidad_items INTEGER,
-- Foreign Keys
CONSTRAINT fk_fecha_salida FOREIGN KEY (fecha_salida_key)
    REFERENCES D_Fecha(fecha_key),
CONSTRAINT fk_fecha_llegada FOREIGN KEY (fecha_llegada_key)
    REFERENCES D_Fecha(fecha_key),
CONSTRAINT fk_cliente FOREIGN KEY (cliente_key)
    REFERENCES D_Cliente(cliente_key),
CONSTRAINT fk_ruta FOREIGN KEY (ruta_key)
    REFERENCES D_Ruta(ruta_key),
CONSTRAINT fk_vehiculo FOREIGN KEY (vehiculo_key)
    REFERENCES D_Vehiculo(vehiculo_key),
CONSTRAINT fk_origen FOREIGN KEY (origen_key)
    REFERENCES D_Lugar(lugar_key),
CONSTRAINT fk_destino FOREIGN KEY (destino_key)
    REFERENCES D_Lugar(lugar_key),
CONSTRAINT fk_junk FOREIGN KEY (junk_key)
    REFERENCES D_Junk(junk_key)
) PARTITION BY RANGE (fecha_salida_key);

-- Índices en tabla de hechos
CREATE INDEX idx_fact_fecha_salida ON F_Shipment(fecha_salida_key);
CREATE INDEX idx_fact_cliente ON F_Shipment(cliente_key);
CREATE INDEX idx_fact_ruta ON F_Shipment(ruta_key);
CREATE INDEX idx_fact_shipment_num ON F_Shipment(shipment_number);

-- Particiones ejemplo (trimestral)
CREATE TABLE F_Shipment_2024_Q1 PARTITION OF F_Shipment
    FOR VALUES FROM (20240101) TO (20240401);

CREATE TABLE F_Shipment_2024_Q2 PARTITION OF F_Shipment
    FOR VALUES FROM (20240401) TO (20240701);

```

14.2 Ejemplo de Proceso ETL en Pseudocódigo

Pseudocódigo ETL para F_Shipment

```

def etl_shipment_daily():
    """
    Proceso ETL diario para cargar envíos completados
    """

    # 1. EXTRACT
    log("Iniciando extracción...")

```

```

shipments_source = extract_from_oltp("""
SELECT * FROM shipments
WHERE completed_date = YESTERDAY
AND status = 'COMPLETED'
""")

```

2. TRANSFORM

```
log(f"Transformando {len(shipments_source)} registros...")
```

```
for shipment in shipments_source:
```

```
    try:
```

```
        # Obtener surrogate keys
```

```
        fecha_salida_key = lookup_fecha(shipment.departure_date)
```

```
        fecha_llegada_key = lookup_fecha(shipment.arrival_date)
```

```
        # Cliente con SCD Tipo 2
```

```
        cliente_key = lookup_cliente_scd2(
```

```
            shipment.customer_code,
```

```
            shipment.departure_date
```

```
        )
```

```
        # Vehículo con SCD Tipo 2
```

```
        vehiculo_key = lookup_vehiculo_scd2(
```

```
            shipment.vehicle_plate,
```

```
            shipment.departure_date
```

```
        )
```

```
        # Dimensiones simples
```

```
        ruta_key = lookup_ruta(shipment.route_code)
```

```
        origen_key = lookup_lugar(shipment.origin_city)
```

```
        destino_key = lookup_lugar(shipment.destination_city)
```

```
        # Dimensión Junk
```

```
        junk_key = lookup_junk(
```

```
            service_type=shipment.service_type,
```

```
            status=shipment.status,
```

```
            priority=shipment.priority,
```

```
            cargo_type=shipment.cargo_type,
```

```
            needs_refrigeration=shipment.refrigerated,
```

```
            is_fragile=shipment.fragile,
```

```
            needs_insurance=shipment.insured
```

```
        )
```

```
        # Calcular métricas
```

```
        retraso_min = calculate_delay(
```

```
            shipment.actual_arrival,
```

```
            shipment.planned_arrival
```

```
        )
```

```

# Construir registro de hechos
fact_record = {
    'shipment_number': shipment.shipment_id,
    'fecha_salida_key': fecha_salida_key,
    'fecha_llegada_key': fecha_llegada_key,
    'cliente_key': cliente_key,
    'ruta_key': ruta_key,
    'vehiculo_key': vehiculo_key,
    'origen_key': origen_key,
    'destino_key': destino_key,
    'junk_key': junk_key,
    'ingresos': shipment.revenue,
    'costo_total': shipment.total_cost,
    'distancia_km': shipment.distance,
    'peso_kg': shipment.weight,
    'volumen_m3': shipment.volume,
    'tiempo_transito_min': shipment.transit_time,
    'retraso_min': retraso_min,
    'cantidad_paradas': shipment.stops,
    'cantidad_items': shipment.items_count
}

# Validaciones
validate_fact_record(fact_record)

# Agregar a batch
batch.append(fact_record)

except Exception as e:
    log_error(f"Error procesando {shipment.shipment_id}: {e}")
    continue

# 3. LOAD
log(f"Cargando {len(batch)} registros a DW...")
bulk_insert_to_dwh('F_Shipment', batch)

# 4. VALIDACIÓN POST-CARGA
validate_load(
    expected_count=len(shipments_source),
    loaded_count=len(batch),
    sum_revenue=sum([r['ingresos'] for r in batch])
)

log("ETL completado exitosamente")

return {
    'status': 'SUCCESS',

```

```

        'records_processed': len(batch),
        'errors': len(shipments_source) - len(batch)
    }

def lookup_cliente_scd2(customer_code, as_of_date):
    """
    Lookup de cliente considerando SCD Tipo 2
    Retorna el cliente_key vigente en la fecha especificada
    """
    result = query_dwh("""
        SELECT cliente_key
        FROM D_Cliente
        WHERE codigo_cliente = %s
            AND %s BETWEEN fecha_inicio_vigencia AND COALESCE(fecha_fin_vigencia,
'9999-12-31')
    """, [customer_code, as_of_date])

    if not result:
        # Cliente nuevo - insertar
        return insert_new_cliente(customer_code, as_of_date)

    return result[0]['cliente_key']

```

14.3 Checklist de Implementación

Fase 1: Diseño (2-3 semanas)

- [] Reuniones con stakeholders para requerimientos
- [] Identificar procesos de negocio prioritarios
- [] Definir granularidad de tabla de hechos
- [] Diseñar dimensiones conformadas
- [] Documentar modelo dimensional
- [] Revisar modelo con usuarios de negocio
- [] Obtener aprobación formal del diseño

Fase 2: Infraestructura (1-2 semanas)

- [] Aprovisionar servidor/cloud para DW
- [] Instalar y configurar base de datos
- [] Configurar backups automáticos
- [] Establecer políticas de seguridad
- [] Crear esquemas y usuarios
- [] Configurar herramienta ETL

Fase 3: Implementación de Dimensiones (2-3 semanas)

- [] Crear scripts DDL para dimensiones

- ☐ Implementar lógica SCD Tipo 2
- ☐ Desarrollar procesos ETL de dimensiones
- ☐ Poblar dimensión fecha (10 años)
- ☐ Carga inicial de dimensiones desde OLTP
- ☐ Validar calidad de datos en dimensiones
- ☐ Crear índices y optimizaciones

Fase 4: Implementación de Hechos (2-3 semanas)

- ☐ Crear scripts DDL para tabla de hechos
- ☐ Implementar particionamiento
- ☐ Desarrollar proceso ETL de hechos
- ☐ Carga histórica (último año mínimo)
- ☐ Validación de totales vs source
- ☐ Crear índices en tabla de hechos
- ☐ Pruebas de performance

Fase 5: Reportería y BI (3-4 semanas)

- ☐ Conectar herramienta de BI al DW
- ☐ Crear capa semántica (universo/modelo)
- ☐ Desarrollar reportes prioritarios
- ☐ Crear dashboards ejecutivos
- ☐ Configurar alertas automáticas
- ☐ Capacitación a usuarios finales
- ☐ Documentación de usuario

Fase 6: Operación (ongoing)

- ☐ Monitoreo diario de procesos ETL
- ☐ Mantenimiento de particiones
- ☐ Actualizaciones de índices/estadísticas
- ☐ Revisión mensual de performance
- ☐ Gestión de solicitudes de cambios
- ☐ Mejora continua basada en feedback

15. Conclusiones

15.1 Resumen Ejecutivo

Este documento ha presentado un **modelo dimensional completo** para un Data Warehouse de transporte y logística, siguiendo rigurosamente la metodología de **Ralph Kimball**. El modelo implementa:

✓ **Esquema estrella** con 1 tabla de hechos y 6 dimensiones ✓ **SCD Tipo 2** para tracking histórico de clientes y vehículos ✓ **Dimensión Junk** para consolidar atributos

transaccionales  **Dimensión degenerada** para números de envío  **Granularidad atómica** permitiendo análisis flexibles  **Métricas de negocio** completas para análisis de rentabilidad y eficiencia

15.2 Beneficios del Modelo

Para el Negocio:

- Visibilidad 360° de operaciones de transporte
- Identificación de clientes y rutas más rentables
- Medición precisa de eficiencia operativa
- Análisis de tendencias y estacionalidad
- Soporte a decisiones estratégicas basadas en datos

Para TI:

- Arquitectura escalable y mantenible
- ETL predecible y automatizable
- Performance optimizado para analítica
- Independencia de sistemas operacionales
- Facilita integración de nuevas fuentes

Para Usuarios:

- Modelo intuitivo y fácil de consultar
- Flexibilidad para análisis ad-hoc
- Respuestas rápidas a preguntas de negocio
- Capacidad de drill-down/roll-up natural
- Datos históricos confiables

15.3 Lecciones Clave

1. **La simplicidad es poder:** El esquema estrella es simple pero extremadamente poderoso
2. **La granularidad correcta es crítica:** Define el nivel de detalle desde el principio
3. **Las dimensiones ricas facilitan el análisis:** Invierte tiempo en diseñar dimensiones robustas
4. **El historial importa:** SCD Tipo 2 permite análisis temporal sofisticados
5. **La calidad de datos es fundamental:** Validaciones en ETL evitan problemas downstream
6. **Optimizar para lectura:** Desnormalización controlada mejora drásticamente performance
7. **Documentar exhaustivamente:** Facilita mantenimiento y onboarding de nuevos usuarios

15.4 Próximos Pasos Recomendados

Corto Plazo (1-3 meses):

1. Implementar el modelo según DDL proporcionado
2. Desarrollar procesos ETL incrementales
3. Crear dashboards ejecutivos prioritarios
4. Capacitar usuarios clave
5. Establecer métricas de calidad de datos

Mediano Plazo (3-6 meses):

1. Agregar nuevas dimensiones según necesidades
2. Implementar tablas agregadas para performance
3. Desarrollar reportes operacionales detallados
4. Integrar datos de sistemas adicionales
5. Establecer gobierno de datos formal

Largo Plazo (6-12 meses):

1. Expandir a otras áreas de negocio (constelación)
2. Implementar machine learning predictivo
3. Crear data products para clientes
4. Automatización avanzada de ETL
5. Migración a arquitecturas cloud-native

15.5 Consideraciones Finales

Este modelo representa un **punto de partida sólido** para un Data Warehouse de transporte. La metodología de Kimball ha probado su efectividad en miles de implementaciones durante más de 30 años.

Recuerda:

- El modelo evoluciona con el negocio
 - La retroalimentación de usuarios es invaluable
 - La documentación debe mantenerse actualizada
 - La calidad de datos es un viaje, no un destino
 - El éxito se mide por el valor generado al negocio
-

16. Ejercicios Prácticos

16.1 Ejercicio 1: Diseño de Query

Enunciado: La gerencia quiere saber: *"¿Cuál fue el top 10 de rutas más rentables en el último trimestre, considerando solo envíos de clientes Gold y Platinum, y mostrando también el porcentaje de entregas a tiempo?"*

Tu tarea: Escribir la consulta SQL completa.

Pistas:

- Necesitas: F_Shipment, D_Ruta, D_Cliente, D_Fecha
- Calcular: margen = ingresos - costo_total
- Entregas a tiempo: retraso_min = 0
- Último trimestre: usar funciones de fecha o definir rango

<details> <summary>Ver Solución</summary>

```
WITH ultimo_trimestre AS (  
    SELECT MAX(año) as año_max,  
           MAX(trimestre) as trim_max  
    FROM D_Fecha  
)  
,  
rentabilidad_rutas AS (  
    SELECT  
        r.ruta_key,  
        r.codigo_ruta,  
        r.nombre_ruta,  
        COUNT(*) as total_envios,  
        SUM(f.ingresos - f.costo_total) as margen_total,  
        AVG(f.ingresos - f.costo_total) as margen_promedio,  
        SUM(CASE WHEN f.retraso_min = 0 THEN 1 ELSE 0 END) * 100.0 / COUNT(*) as  
pct_on_time,  
        SUM(f.ingresos) as ingresos_totales  
    FROM F_Shipment f  
    JOIN D_Ruta r ON f.ruta_key = r.ruta_key  
    JOIN D_Cliente c ON f.cliente_key = c.cliente_key  
    JOIN D_Fecha d ON f.fecha_salida_key = d.fecha_key  
    CROSS JOIN ultimo_trimestre ut  
    WHERE d.año = ut.año_max  
        AND d.trimestre = ut.trim_max  
        AND c.segmento IN ('Gold', 'Platinum')  
        AND c.es_actual = TRUE  
    GROUP BY r.ruta_key, r.codigo_ruta, r.nombre_ruta  
)  
SELECT  
    codigo_ruta,  
    nombre_ruta,  
    total_envios,  
    ingresos_totales,  
    margen_total,  
    margen_promedio,  
    ROUND(pct_on_time, 2) as pct_entregas_a_tiempo  
FROM rentabilidad_rutas  
ORDER BY margen_total DESC  
LIMIT 10;
```

</details>

16.2 Ejercicio 2: Análisis SCD

Enunciado: El cliente "CLI-001" cambió de segmento Bronze a Gold el 1 de marzo de 2024. Tienes envíos de este cliente en febrero y marzo.

Preguntas:

1. ¿Cuántos registros tendrá D_Cliente para este cliente?
2. ¿Cómo se reflejarán los envíos de febrero vs marzo en la tabla de hechos?
3. Escribe la query para ver ingresos totales "como era" (Bronze) vs "como es" (Gold).

<details> <summary>Ver Solución</summary>

Respuesta 1: 2 registros en D_Cliente:

cliente_key	version	codigo_cliente	segmento	fecha_inicio	fecha_fin	es_actual
1001	1	CLI-001	Bronze	2023-01-01	2024-02-29	FALSE
1002	2	CLI-001	Gold	2024-03-01	NULL	TRUE

Respuesta 2:

- Envíos de febrero apuntan a cliente_key=1001 (Bronze)
- Envíos de marzo apuntan a cliente_key=1002 (Gold)

Respuesta 3:

```
-- Ingresos cuando era Bronze
SELECT
  'Como ERA (Bronze)' as escenario,
  SUM(f.ingresos) as ingresos_totales
FROM F_Shipment f
JOIN D_Cliente c ON f.cliente_key = c.cliente_key
WHERE c.codigo_cliente = 'CLI-001'
      AND c.segmento = 'Bronze'
```

UNION ALL

```
-- Ingresos actuales como Gold
SELECT
  'Como ES (Gold)' as escenario,
  SUM(f.ingresos) as ingresos_totales
FROM F_Shipment f
JOIN D_Cliente c ON f.cliente_key = c.cliente_key
WHERE c.codigo_cliente = 'CLI-001'
      AND c.es_actual = TRUE;
```

</details>

16.3 Ejercicio 3: Diseño de Nueva Dimensión

Enunciado: La empresa quiere empezar a trackear conductores. Cada envío es asignado a un conductor específico.

Tu tarea:

1. Diseña la estructura de la dimensión D_Conductor
2. ¿Qué tipo de SCD usarías y por qué?
3. ¿Qué atributos incluirías?
4. ¿Cómo modificarías F_Shipment?

<details> <summary>Ver Solución</summary>

1. Estructura D_Conductor:

```
CREATE TABLE D_Conductor (  
  conductor_key INTEGER PRIMARY KEY,  
  version INTEGER NOT NULL,  
  codigo_empleado VARCHAR(20) NOT NULL,  
  nombre_completo VARCHAR(200) NOT NULL,  
  licencia_numero VARCHAR(50),  
  licencia_tipo VARCHAR(20), -- A, B, C, D, E  
  licencia_vencimiento DATE,  
  fecha_contratacion DATE,  
  estado_conductor VARCHAR(50), -- Activo, Vacaciones, Licencia, Inactivo  
  categoria VARCHAR(50), -- Junior, Semi-Senior, Senior  
  pais_residencia VARCHAR(100),  
  tiene_certificacion_internacional BOOLEAN,  
  rating_promedio DECIMAL(3,2), -- 1.00 a 5.00  
  fecha_inicio_vigencia DATE NOT NULL,  
  fecha_fin_vigencia DATE,  
  es_actual BOOLEAN NOT NULL  
);
```

2. Tipo de SCD: SCD Tipo 2 - Porque necesitamos saber:

- Qué categoría tenía el conductor cuando hizo el envío
- Cambios en estado (activo → licencia médica)
- Evolución del rating a lo largo del tiempo
- Historial de certificaciones

3. Atributos clave:

- **Identificación:** código empleado, nombre
- **Credenciales:** licencia, tipo, vencimiento
- **Operacionales:** estado, categoría
- **Performance:** rating promedio

- **Compliance:** certificaciones, país
- **SCD:** versión, fechas vigencia, es_actual

4. Modificación F_Shipment:

```
ALTER TABLE F_Shipment
ADD COLUMN conductor_key INTEGER,
ADD CONSTRAINT fk_conductor
    FOREIGN KEY (conductor_key)
    REFERENCES D_Conductor(conductor_key);

CREATE INDEX idx_fact_conductor ON F_Shipment(conductor_key);
```

Análisis posible:

```
-- Performance por conductor
SELECT
    c.nombre_completo,
    c.categoria,
    COUNT(*) as total_viajes,
    AVG(f.retraso_min) as retraso_promedio,
    SUM(f.ingresos - f.costos_total) as margen_generado,
    c.rating_promedio
FROM F_Shipment f
JOIN D_Conductor c ON f.conductor_key = c.conductor_key
WHERE c.es_actual = TRUE
GROUP BY c.conductor_key, c.nombre_completo, c.categoria, c.rating_promedio
ORDER BY margen_generado DESC;
```

</details>

16.4 Ejercicio 4: Dimensión Junk

Enunciado: Actualmente tienes en D_Junk estos atributos:

- tipo_servicio (3 valores)
- prioridad (3 valores)
- es_fragil (2 valores)
- requiere_refrigeracion (2 valores)

Preguntas:

1. ¿Cuántas filas tendrá D_Junk si cargas TODAS las combinaciones?
2. ¿Cuántas dimensiones separadas ahorras usando junk?
3. ¿Por qué no crear una dimensión separada para cada atributo?

<details> <summary>Ver Solución</summary>

Respuesta 1: Combinaciones = $3 \times 3 \times 2 \times 2 = 36$ filas

```
-- Script para poblar D_Junk
INSERT INTO D_Junk (
    junk_key, tipo_servicio, prioridad,
    es_fragil, requiere_refrigeracion
)
SELECT
    ROW_NUMBER() OVER (ORDER BY ts, p, f, r) as junk_key,
    ts as tipo_servicio,
    p as prioridad,
    f as es_fragil,
    r as requiere_refrigeracion
FROM (
    SELECT 'Express' as ts UNION ALL
    SELECT 'Estándar' UNION ALL
    SELECT 'Económico'
) servicios
CROSS JOIN (
    SELECT 'Alta' as p UNION ALL
    SELECT 'Media' UNION ALL
    SELECT 'Baja'
) prioridades
CROSS JOIN (
    SELECT TRUE as f UNION ALL SELECT FALSE
) fragil
CROSS JOIN (
    SELECT TRUE as r UNION ALL SELECT FALSE
) refrigeracion;
```

Respuesta 2: Ahorras 4 dimensiones separadas → Reduces a 1 dimensión junk

Antes:

- D_TipoServicio (3 filas)
- D_Prioridad (3 filas)
- D_Fragil (2 filas)
- D_Refrigeracion (2 filas)
- **Total: 4 JOINS en queries**

Después:

- D_Junk (36 filas)
- **Total: 1 JOIN en queries**

Respuesta 3: ¿Por qué NO dimensiones separadas?

✗ Problemas de dimensiones separadas:

- Más JOINS → Queries más lentas
- Más índices → Mayor uso de memoria
- Modelo más complejo visualmente
- Mayor overhead de mantenimiento
- ETL más complejo (lookups múltiples)

✓ Ventajas de dimensión junk:

- 1 solo JOIN vs 4
- Menos índices
- Modelo más limpio
- ETL simplificado
- 36 filas es manejable (vs millones en hechos)

Regla de oro: Usa dimensión junk cuando:

- Atributos tienen baja cardinalidad (< 10 valores cada uno)
- Combinaciones totales < 1000
- Atributos cambian juntos o son independientes
- No necesitas jerarquías en esos atributos

</details>

17. Preguntas Frecuentes (FAQ)

17.1 Diseño

P: ¿Por qué no normalizar las dimensiones? R: El snowflaking (normalización de dimensiones) complica queries sin beneficio significativo. El espacio en disco es barato; la simplicidad y performance valen más.

P: ¿Cuándo usar SCD Tipo 1 vs Tipo 2? R:

- Tipo 1: Correcciones de errores, atributos que no afectan análisis histórico
- Tipo 2: Cambios que necesitas rastrear (segmento de cliente, precio de producto)

P: ¿Puedo tener una dimensión que sea FK en múltiples columnas de hechos? R: Sí, es común (como D_Fecha para fecha_salida y fecha_llegada, o D_Lugar para origen y destino).

P: ¿Qué hago con valores NULL en el sistema origen? R: Crea registros "Unknown" o "Not Applicable" en dimensiones. Nunca dejes FK NULL en la tabla de hechos.

17.2 ETL

P: ¿Con qué frecuencia debo ejecutar el ETL? R: Depende del negocio:

- Reportes ejecutivos: Diario nocturno es común

- Dashboards operativos: Cada hora o near real-time
- Análisis histórico: Semanal puede ser suficiente

P: ¿Cómo manejo envíos que llegan tarde al DW (late arriving facts)? R: Inserta con las surrogate keys correctas basadas en la fecha del evento, no la fecha de carga. El SCD Tipo 2 ya maneja esto.

P: ¿Debo cargar TODO el histórico desde el inicio? R: Depende:

- Mínimo: 12-24 meses para análisis año sobre año
- Ideal: Todo el histórico disponible (si es manejable)
- Considera: Regulaciones (ej: auditorías requieren 7 años)

17.3 Performance

P: Mi tabla de hechos tiene 100M+ registros y las queries son lentas. ¿Qué hago? R:

1. Particionar por fecha (más importante)
2. Crear índices en FKs consultadas frecuentemente
3. Actualizar estadísticas regularmente
4. Considerar tablas agregadas pre-calculadas
5. Usar columnstore indexes (SQL Server) o equivalentes
6. Revisar que queries aprovechen particionamiento

P: ¿Debo crear índices en todos los campos de dimensiones? R: No. Solo en:

- Primary key (automático)
- Natural keys (códigos de negocio)
- Atributos frecuentemente usados en WHERE/JOIN
- Para SCD Tipo 2: índice compuesto en (natural_key, es_actual)

17.4 Modelado Avanzado

P: ¿Qué es una dimensión role-playing? R: Una dimensión usada múltiples veces con roles diferentes (ej: D_Fecha como fecha_salida y fecha_llegada). Es la misma tabla física con alias en queries.

P: ¿Cuándo uso múltiples tablas de hechos? R: Cuando tienes procesos de negocio con granularidades diferentes:

- F_Shipment (un registro por envío)
- F_Incidencia (un registro por evento durante transporte)
- F_Cotizacion (un registro por cotización generada)

P: ¿Qué es una dimensión outrigger y cuándo usarla? R: Una dimensión referenciada por otra dimensión. Generalmente se evita (viola esquema estrella puro), pero puede ser útil en casos específicos como jerarquías complejas.

17.5 Herramientas

P: ¿Qué herramienta de BI recomiendas? R: Depende:

- **Power BI:** Excelente para ecosistema Microsoft, costo razonable
- **Tableau:** Muy visual, potente, más caro
- **Looker:** Moderno, cloud-native, integra bien con BigQuery
- **Metabase:** Open source, bueno para empezar

P: ¿Necesito una herramienta ETL o puedo usar scripts? R:

- Scripts (Python/SQL): OK para empezar, proyectos pequeños
- Herramienta ETL: Mejor para producción (logging, monitoreo, gestión de errores, scheduling)

P: ¿Cloud o on-premise? R:

- **Cloud (Snowflake, Redshift, BigQuery):** Escalabilidad elástica, bajo mantenimiento, pago por uso
 - **On-premise:** Mayor control, puede ser más barato a gran escala, requerimientos de compliance
-

18. Caso de Estudio Real

18.1 Contexto

Empresa: TransLogistics S.A. **Industria:** Transporte terrestre de carga **Geografía:** 5 países de LATAM **Volumen:** 15,000 envíos/mes **Problema:** Reportes manuales en Excel, falta de visibilidad operativa

18.2 Situación Antes del DW

Dolores:

- Reportes de rentabilidad tomaban 3-5 días
- Datos inconsistentes entre áreas
- Imposible analizar tendencias históricas
- Decisiones basadas en "feeling" no datos
- Clientes insatisfechos sin visibilidad de envíos

Sistemas:

- ERP legacy con data desorganizada
- Sistema de tracking GPS separado
- Excel como "fuente de verdad"
- Sin integración entre sistemas

18.3 Implementación

Duración: 4 meses

Fase 1 (mes 1): Diseño

- Workshops con gerencias de operaciones, comercial, finanzas
- Identificación de KPIs prioritarios
- Diseño del modelo dimensional
- Aprobación del modelo

Fase 2 (mes 2): Construcción

- Implementación de dimensiones
- Carga de 24 meses de histórico
- Desarrollo de ETL incremental
- Testing de integridad

Fase 3 (mes 3): Reportería

- Conexión Power BI
- Desarrollo de 15 dashboards
- Capacitación a usuarios
- Documentación

Fase 4 (mes 4): Producción

- Go-live con monitoreo intensivo
- Ajustes post-lanzamiento
- Feedback loop con usuarios

18.4 Resultados

Impacto Cuantificable:

- 🕒 Reportes: de 3-5 días → **15 minutos**
- 📊 Visibilidad: de histórico inexistente → **2 años de data analizable**
- 💰 Identificación de rutas no rentables → **\$450K ahorro anual**
- 🎯 Mejora OTD (On-Time Delivery): 73% → **89%** en 6 meses
- 👥 Reducción trabajo manual: **80 horas/mes** ahorradas

Insights de Negocio Descubiertos:

1. **Segmento Platinum representa 18% envíos pero 42% margen**
 - Acción: Programa de fidelización enfocado
2. **Rutas cortas (<200km) tenían margen negativo**
 - Acción: Ajuste de tarifas mínimas
3. **Retrasos correlacionan 87% con día de semana (Lunes/Viernes)**
 - Acción: Redistribución de carga a días medios

4. Vehículos tercerizados 15% más caros que propios

- Acción: Plan de inversión en flota propia

5. Corredor São Paulo-Río genera 28% de ingresos

- Acción: Apertura de nuevo centro de distribución

Testimonios:

"Antes tardábamos una semana en cerrar el mes. Ahora tenemos los números en tiempo real." — *CFO, TransLogistics*

"Finalmente podemos tomar decisiones basadas en datos reales, no en intuición." — *Director de Operaciones*

"El ROI del proyecto se pagó en menos de 6 meses solo con la optimización de rutas." — *CIO*

18.5 Lecciones Aprendidas

✓ Éxitos:

- Involucrar usuarios desde día 1
- Empezar con histórico de 2 años (suficiente para análisis estacional)
- Priorizar dimensiones conformadas (reutilizables)
- Capacitación intensiva a usuarios

✗ Desafíos:

- Calidad de datos en ERP legacy requirió limpieza extensiva
- Resistencia al cambio de usuarios acostumbrados a Excel
- Subestimamos tiempo de testing de integridad
- Primeros dashboards demasiado complejos (simplificamos después)

🎓 Aprendizajes:

- "Hazlo simple primero, sofisticado después"
- La calidad de datos es 60% del esfuerzo
- Los usuarios quieren respuestas, no tecnología
- La documentación es crítica (no opcional)
- El mantenimiento continuo es tan importante como la construcción

19. Recursos Adicionales para Profundizar

19.1 Libros Recomendados

Nivel Fundacional:

1. **"The Data Warehouse Toolkit" - Kimball & Ross**
 - Capítulos clave: 1-4 (fundamentos), 12 (transporte), 19-20 (ETL)
2. **"Star Schema: The Complete Reference" - Adamson**
 - Profundiza en patrones de modelado dimensional

Nivel Intermedio: 3. **"The Data Warehouse Lifecycle Toolkit" - Kimball et al.**

- Metodología end-to-end de implementación
- 4. **"Building a Scalable Data Warehouse with Data Vault 2.0" - Linstedt**
 - Enfoque alternativo para staging y flexibilidad

Nivel Avanzado: 5. **"Agile Data Warehouse Design" - Corr & Stagnitto**

- Modelado iterativo y ágil
- 6. **"The Data Warehouse ETL Toolkit" - Kimball & Caserta**
 - Profundización en procesos ETL complejos

19.2 Cursos Online

Coursera:

- "Data Warehousing for Business Intelligence" (University of Colorado)
- "Modern Big Data Analysis with SQL" (Cloudera)

Udemy:

- "The Complete Data Warehousing Course"
- "Microsoft SQL Server Data Warehouse from A-Z"

LinkedIn Learning:

- "Data Warehouse Fundamentals"
- "Power BI: Building Data Models"

19.3 Comunidades y Foros

- **r/dataengineering** (Reddit)
- **r/BusinessIntelligence** (Reddit)
- **Kimball Group Discussion Forum**
- **Stack Overflow** (tags: data-warehouse, dimensional-modeling)
- **LinkedIn Groups:** Data Warehousing Professionals

19.4 Blogs y Sitios Web

- **Kimball Group:** <https://www.kimballgroup.com> (Design Tips son oro)
- **Modern Data Stack:** <https://www.moderndatastack.xyz>
- **dbt Blog:** <https://blog.getdbt.com>
- **Locally Optimistic:** <https://locallyoptimistic.com>

19.5 Herramientas para Practicar

Bases de Datos (Gratis):

- PostgreSQL + extensión TimescaleDB
- SQL Server Developer Edition
- MySQL
- DuckDB (para análisis local)

ETL (Open Source):

- Apache Airflow
- Talend Open Studio
- Pentaho Data Integration
- Singer taps

BI (Gratis/Freemium):

- Metabase
- Apache Superset
- Power BI Desktop (gratis)
- Google Data Studio

Datasets de Práctica:

- Kaggle Datasets (logística, retail)
 - UCI Machine Learning Repository
 - Google Cloud Public Datasets
 - AWS Open Data Registry
-

20. Conclusión Final

Has completado un recorrido exhaustivo por el diseño de un Data Warehouse dimensional para transporte y logística. Este documento te proporciona:

- ✓ **Fundamentos teóricos** sólidos de Kimball
- ✓ **Modelo completo** listo para implementar
- ✓ **Código SQL** funcional
- ✓ **Casos de uso** reales
- ✓ **Ejercicios prácticos** para reforzar
- ✓ **Guías de implementación** paso a paso
- ✓ **Recursos** para profundizar

El Viaje Continúa...

El Data Warehousing es un campo en constante evolución. Nuevas tecnologías (lakehouse, data mesh, real-time analytics) expanden las posibilidades, pero **los fundamentos de Kimball permanecen vigentes**:

- Diseñar para el usuario de negocio
- Mantener la simplicidad

- Priorizar la calidad de datos
- Pensar dimensionalmente
- Preservar la historia

Últimos Consejos

1. **Empieza pequeño, crece incrementalmente:** Un DW funcional con 3 dimensiones es mejor que un diseño perfecto que nunca se termina
2. **Escucha a los usuarios:** Los mejores insights vienen de quienes usan los datos diariamente
3. **Documenta TODO:** Tu yo del futuro (y tus colegas) te lo agradecerán
4. **Automatiza despiadadamente:** ETL manual es error humano esperando suceder
5. **Celebra los éxitos:** Cuando un dashboard ayude a tomar una decisión crítica, es una victoria

Información del Documento

Versión: 1.0 **Fecha:** Octubre 2024 **Autor:** Documentación Técnica - Data Warehouse Transporte **Basado en:** "The Data Warehouse Toolkit" - Ralph Kimball, Capítulo 12

Licencia de Uso: Este documento es material educativo. Puede ser utilizado libremente para fines académicos y de aprendizaje. Para uso comercial o redistribución, se recomienda citar la fuente original.

Agradecimientos: A Ralph Kimball y Margy Ross por su invaluable contribución al campo del Data Warehousing dimensional.

¡Fin del Documento!

¿Tienes preguntas? ¿Quieres profundizar en algún tema específico? El aprendizaje es un viaje continuo. ¡Éxito en tu implementación! 🚀📊
