Proyecto: Arquitectura de Datos para E-commerce

Evaluación Módulo 5 - Fundamentos de Arquitectura y Modelamiento de Datos

---

LECCIÓN 1 - ARQUITECTURA DE DATOS

1. Identificación de Fuentes de Datos

Fuentes Estructuradas:

1. **Sistema Transaccional de Ventas (OLTP)**
   - Base de datos PostgreSQL/MySQL
   - Tablas: pedidos, productos, clientes, pagos, envíos
   - Frecuencia: Tiempo real (streaming)
2. **Sistema de Gestión de Inventario (ERP)**
   - Base de datos Oracle/SQL Server
   - Tablas: stock, proveedores, categorías, precios
   - Frecuencia: Batch cada 4 horas
3. **Sistema CRM**
   - Base de datos Salesforce/HubSpot
   - Tablas: clientes, interacciones, campañas, leads
   - Frecuencia: Batch diario

Fuentes Semi-estructuradas:

4. **Logs de Aplicación Web**

- Archivos JSON de eventos de navegación
- Datos: clicks, páginas visitadas, tiempo de sesión
- Frecuencia: Streaming en tiempo real

5. **APIs de Redes Sociales**
   - Datos JSON de Facebook, Instagram, Twitter
   - Contenido: menciones, reviews, engagement
   - Frecuencia: Batch cada 6 horas

Fuentes No Estructuradas:

6. **Imágenes y Videos de Productos**
   - Archivos multimedia en formato JPG, PNG, MP4
   - Metadatos: resolución, tamaño, categoría
   - Frecuencia: Bajo demanda
7. **Reviews y Comentarios de Clientes**
   - Texto libre en múltiples idiomas
   - Sentimientos y opiniones no estructuradas
   - Frecuencia: Tiempo real

2. Diseño Arquitectónico por Capas

Capa 1: INGESTA DE DATOS

- **Streaming**: Apache Kafka para datos en tiempo real
- **Batch**: Apache Airflow para procesos programados
- **API Gateway**: Para integración con servicios externos
- **Conectores**: Debezium para CDC, Stitch/Fivetran para SaaS

Capa 2: INTEGRACIÓN Y PROCESAMIENTO

- **Stream Processing**: Apache Spark Streaming / Kafka Streams
- **Batch Processing**: Apache Spark / Databricks
- **ETL/ELT**: Herramientas como dbt, Talend o Pentaho
- **Orquestación**: Apache Airflow

Capa 3: ALMACENAMIENTO

- **Data Lake (Raw Zone)**: Amazon S3 / Azure Data Lake
- **Data Lake (Trusted Zone)**: Datos validados y limpios
- **Data Lake (Curated Zone)**: Datos procesados y enriquecidos
- **Data Warehouse**: Snowflake / Redshift / BigQuery
- **Data Marts**: Especializados por área de negocio

Capa 4: CALIDAD Y GOBIERNO

- **Data Quality**: Great Expectations, Apache Griffin
- **Metadata Management**: Apache Atlas, Collibra
- **Data Lineage**: Seguimiento de origen y transformaciones
- **Seguridad**: Apache Ranger, cifrado, control de acceso

Capa 5: CONSUMO Y ANÁLISIS

- **BI Tools**: Tableau, Power BI, Looker
- **Analytics**: Jupyter Notebooks, R Studio
- **APIs**: REST APIs para aplicaciones
- **ML Platforms**: MLflow, Kubeflow

3. Principios Arquitectónicos Aplicados

Gobierno:

- Catálogo de datos centralizado
- Políticas de retención y archivado
- Control de acceso basado en roles (RBAC)
- Auditoría y trazabilidad completa

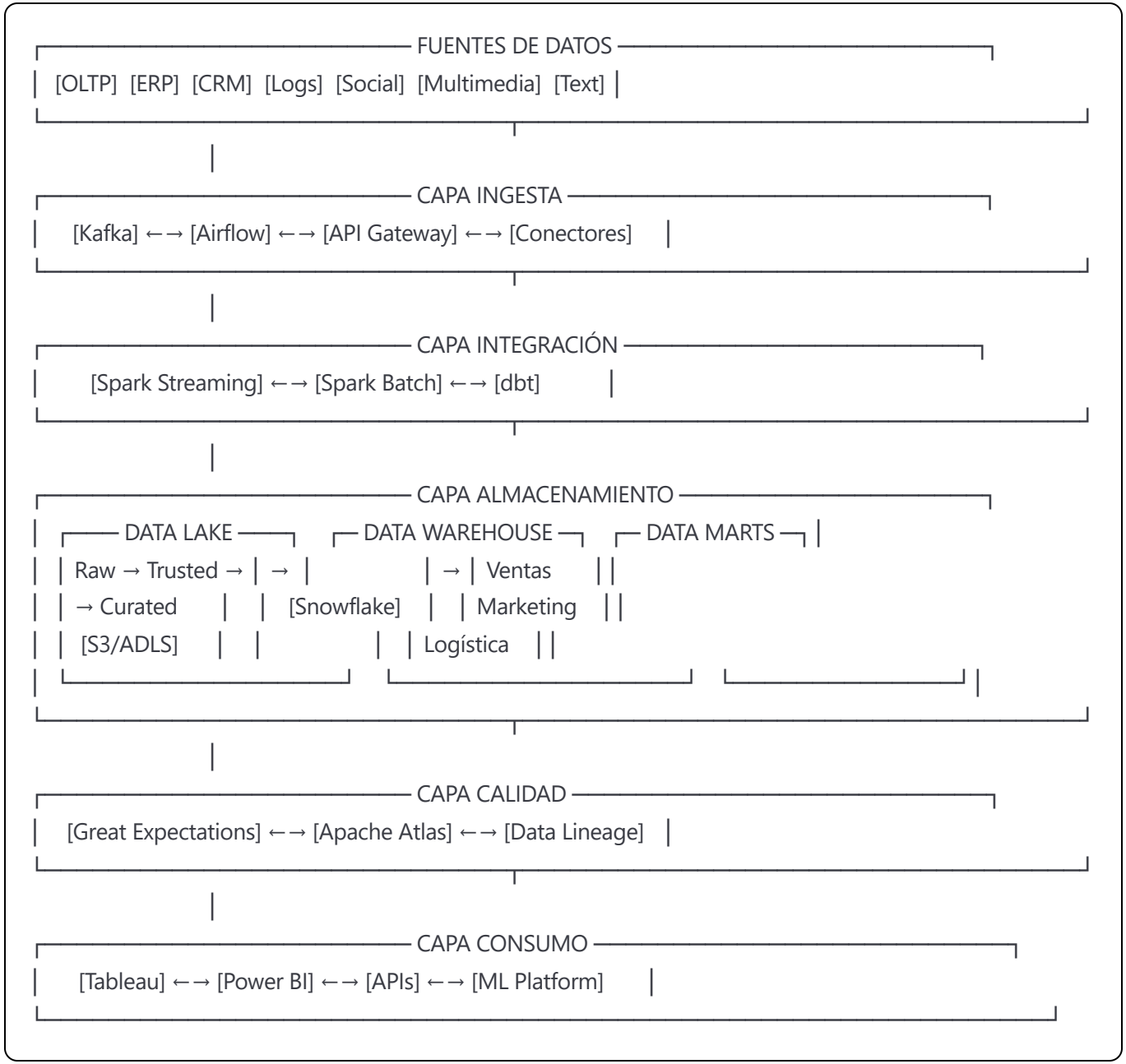
Escalabilidad:

- Arquitectura distribuida y elástica
- Separación de cómputo y almacenamiento
- Auto-scaling basado en demanda
- Particionado horizontal y vertical

Flexibilidad:

- Arquitectura basada en microservicios
- APIs RESTful para integración
- Soporte multi-cloud
- Schema evolution sin interrupciones

4. Diagrama Arquitectónico



Justificación Técnica

1. Arquitectura Lambda Modificada: Combina procesamiento batch y streaming para balance entre latencia y throughput.

2. Separación de Responsabilidades: Cada capa tiene un propósito específico, facilitando mantenimiento y escalabilidad.

3. Tecnologías Cloud-Native: Aprovecha elasticidad y servicios gestionados para reducir overhead operacional.

4. Data Lake como Hub Central: Permite almacenar datos en formato original y aplicar schema-on-read.

LECCIÓN 2 - ENFOQUES PARA ALMACENAMIENTO Y GESTIÓN

1. Análisis del Esquema Arquitectónico

La arquitectura diseñada en la Lección 1 establece un flujo de datos de 5 capas que permite la ingesta, procesamiento, almacenamiento, control de calidad y consumo de datos de manera escalable y governada.

2. Definición de Zonas de Almacenamiento

DATA LAKE - Zonas Definidas:

ZONA RAW (Bronze Layer)

Propósito: Almacenamiento de datos en formato original sin transformaciones

- **Formato**: Parquet, Avro, JSON, CSV según fuente original
- **Particionado**: Por fecha de ingesta (año/mes/día/hora)
- **Retención**: 7 años para cumplimiento regulatorio
- **Estructura**:

```
/raw/
  /transactional/
    /sales/2024/01/15/sales_20240115_14.parquet
    /inventory/2024/01/15/inventory_20240115.parquet
  /logs/
    /web_events/2024/01/15/14/events.json
  /social/
    /facebook/2024/01/15/fb_posts.json
```

ZONA TRUSTED (Silver Layer)

Propósito: Datos validados, limpiados y estandarizados

- **Formato**: Delta Lake / Iceberg para ACID transactions
- **Transformaciones aplicadas**:
  - Validación de tipos de datos
  - Eliminación de duplicados
  - Normalización de formatos de fecha/hora
  - Validación de integridad referencial
- **Particionado**: Por entidad de negocio y fecha
- **Estructura**:

```
/trusted/
  /customers/year=2024/month=01/
  /products/year=2024/month=01/
  /orders/year=2024/month=01/day=15/
```

ZONA CURATED (Gold Layer)

Propósito: Datos agregados, enriquecidos y listos para análisis

- **Formato**: Optimizado para consultas (Parquet + columnar)
- **Contenido**:
  - Métricas pre-calculadas
  - Datos denormalizados para reporting
  - Features para Machine Learning
  - Agregaciones temporales (diario, semanal, mensual)
- **Estructura**:

```
/curated/
  /sales_metrics/
    /daily_sales/year=2024/month=01/
    /customer_segments/year=2024/month=01/
  /ml_features/
    /customer_lifetime_value/
    /product_recommendations/
```

DATA WAREHOUSE

Propósito: Almacén central para datos estructurados y análisis empresarial

- **Tecnología sugerida**: Snowflake
- **Arquitectura**: Schema-on-Write con modelos dimensionales
- **Fuentes**: Principalmente zona Curated del Data Lake
- **Contenido**:
  - Tablas de hechos y dimensiones
  - Datos históricos para análisis de tendencias
  - Snapshots para análisis point-in-time
- **Esquemas especializados**:

- `SALES_DW`: Datos de ventas y transacciones
- `CUSTOMER_DW`: Información de clientes y segmentación
- `INVENTORY_DW`: Gestión de inventario y logística

DATA MARTS

Propósito: Subconjuntos especializados para áreas específicas del negocio

Data Mart de Ventas

- **Usuario objetivo**: Equipo comercial y gerencia
- **Contenido**: Métricas de ventas, conversión, productos top
- **Actualización**: Diaria

Data Mart de Marketing

- **Usuario objetivo**: Equipo de marketing digital
- **Contenido**: Campañas, CAC, LTV, attribution models
- **Actualización**: Cada 4 horas

Data Mart de Logística

- **Usuario objetivo**: Operaciones y supply chain
- **Contenido**: Tiempos de entrega, costos de envío, inventario
- **Actualización**: Tiempo real para métricas críticas

3. Tecnologías y Servicios Sugeridos

Almacenamiento:

- **Data Lake**: Amazon S3 con S3 Intelligent Tiering

- **Data Warehouse**: Snowflake con auto-scaling
- **Caching**: Redis para consultas frecuentes
- **Backup**: S3 Glacier para archivado a largo plazo

Procesamiento:

- **Batch**: Apache Spark en Amazon EMR/Databricks
- **Streaming**: Kafka + Spark Streaming
- **ETL/ELT**: dbt para transformaciones SQL
- **Orquestación**: Apache Airflow/Prefect

Gobierno y Seguridad:

- **Catálogo**: AWS Glue Data Catalog / Apache Atlas
- **Cifrado**: AES-256 en reposo, TLS en tránsito
- **Control de acceso**: AWS IAM + Row-level security
- **Monitoreo**: CloudWatch + Grafana

4. Prácticas de Gobernanza y Gestión

Trazabilidad:

- **Data Lineage**: Seguimiento automático de transformaciones
- **Versionado**: Control de versiones de schemas y datos
- **Auditoría**: Logs detallados de acceso y modificaciones
- **Metadata**: Documentación automática de datasets

Seguridad:

- **Autenticación**: Single Sign-On (SSO) integrado

- **Autorización**: Role-Based Access Control (RBAC)
- **Cifrado**: End-to-end encryption
- **Network Security**: VPC privadas y endpoints seguros

Disponibilidad:

- **SLA**: 99.9% uptime para servicios críticos
- **Backup**: Respaldo automático diario con retención configurable
- **Disaster Recovery**: Replicación cross-region
- **Monitoreo**: Alertas proactivas de performance y disponibilidad

---

LECCIÓN 3 - CALIDAD DE LOS DATOS

1. Revisión de Zonas y Flujo Arquitectónico

El flujo arquitectónico definido en la Lección 2 establece tres zonas principales en el Data Lake (Raw, Trusted, Curated) que requieren controles de calidad específicos en cada etapa para garantizar la integridad y confiabilidad de los datos.

2. Controles, Métricas e Indicadores de Calidad

CONTROLES EN ZONA RAW (Bronze)

Objetivo: Detectar problemas en el origen y validar la ingesta

Controles de Integridad:

- **Completitud de archivos**: Verificar que todos los archivos esperados lleguen
- **Tamaño de archivos**: Validar rangos esperados vs. archivos vacíos/corruptos
- **Formato de archivos**: Verificar que los archivos sean parseables

Métricas de Monitoreo:

- Tasa de éxito de ingesta: `archivos_procesados_exitosamente / total_archivos * 100`
- Latencia de ingesta: `tiempo_llegada - tiempo_esperado`
- Volumen de datos: `bytes_ingresados_diarios`

```python
# Ejemplo de validación en zona Raw
def validate_raw_ingestion():
    checks = [
        expect_file_to_exist(),
        expect_file_size_to_be_between(min_size=1000, max_size=100000000),
        expect_column_names_to_match_ordered_list(['id', 'timestamp', 'event_type'])
    ]
    return execute_validation_suite(checks)
```

CONTROLES EN ZONA TRUSTED (Silver)

Objetivo: Asegurar calidad de datos después de transformaciones iniciales

Controles de Validez:

- **Tipos de datos**: Verificar conversiones correctas (strings → dates, numeric)
- **Rangos de valores**: Validar que los valores estén dentro de rangos esperados
- **Formatos**: Verificar consistencia en formatos de fecha, teléfono, email

Controles de Integridad:

- **Duplicados**: Detectar y manejar registros duplicados
- **Claves foráneas**: Validar integridad referencial entre tablas
- **Valores nulos**: Verificar campos obligatorios

Métricas de Calidad:

- **Completitud**: `(total_campos - campos_nulos) / total_campos * 100`
- **Validez**: `registros_válidos / total_registros * 100`
- **Consistencia**: `registros_consistentes / total_registros * 100`
- **Unicidad**: `registros_únicos / total_registros * 100`

```python
# Ejemplo de validaciones en zona Trusted
def validate_trusted_data():
    return [
        expect_column_values_to_not_be_null('customer_id'),
        expect_column_values_to_be_between('age', min_value=18, max_value=120),
        expect_column_values_to_match_regex('email', r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'),
        expect_column_values_to_be_unique('order_id'),
        expect_table_row_count_to_be_between(min_value=1000, max_value=1000000)
    ]
```

CONTROLES EN ZONA CURATED (Gold)

Objetivo: Validar datos agregados y métricas de negocio

Controles de Precisión:

- **Reconciliación**: Comparar totales agregados con fuentes originales
- **Métricas de negocio**: Validar KPIs contra umbrales históricos
- **Coherencia temporal**: Verificar tendencias lógicas en series de tiempo

Controles de Actualidad:

- **Freshness**: Verificar que los datos estén actualizados según SLA

- **Completitud temporal**: Asegurar que no falten períodos de tiempo

Métricas de Negocio:

- **Precisión de agregaciones**: `(valor_calculado - valor_esperado) / valor_esperado * 100`
- **Actualidad**: `tiempo_actual - timestamp_último_update`
- **Cobertura temporal**: `períodos_con_datos / total_períodos_esperados * 100`

3. Proceso de Monitoreo y Remediación

MONITOREO CONTINUO

Dashboard de Calidad de Datos:

- **Vista Ejecutiva**: KPIs principales de calidad por zona
- **Vista Operativa**: Alertas y incidencias en tiempo real
- **Vista Técnica**: Métricas detalladas por dataset y transformación

Alertas Automáticas:

- **Críticas**: Fallos de ingesta, corrupción de datos
- **Advertencias**: Degradación de calidad, latencia alta
- **Informativas**: Completitud de procesos, estadísticas diarias

```yaml
```

```
# Configuración de alertas
alerts:
  critical:
    - ingestion_failure_rate > 5%
    - data_corruption_detected
    - sla_breach_critical_tables
  warning:
    - completeness_below_95%
    - freshness_delay > 2_hours
    - anomaly_detection_triggered
```

PROCESO DE REMEDIACIÓN

Flujo de Resolución de Incidencias:

1. **Detección**: Monitoreo automático o reporte manual
2. **Clasificación**: Severidad (crítica, alta, media, baja)
3. **Asignación**: Auto-asignación a equipos responsables
4. **Investigación**: Root cause analysis
5. **Resolución**: Corrección y validación de fix
6. **Documentación**: Lessons learned y mejoras preventivas

Acciones Automáticas de Remediación:

- **Reingesta automática**: Para fallos transitorios
- **Quarantine de datos**: Aislar datos de mala calidad
- **Rollback**: Revertir a versión anterior conocida como buena
- **Notificaciones**: Alertar a stakeholders sobre impactos

4. Integración del Plan de Calidad en la Arquitectura

IMPLEMENTACIÓN POR CAPAS

En la Capa de Ingesta:

- Validaciones de formato y schema en tiempo real
- Monitoreo de throughput y latencia
- Alertas de fallos de conectividad

En la Capa de Procesamiento:

- Validaciones de transformaciones con Great Expectations
- Unit tests para funciones de transformación
- Controles de reconciliación pre/post transformación

En la Capa de Almacenamiento:

- Constraints de integridad en el Data Warehouse
- Validaciones de schema evolution
- Monitoreo de performance de consultas

En la Capa de Consumo:

- Validación de SLAs de actualización
- Monitoreo de uso y adoption
- Feedback loop desde usuarios finales

HERRAMIENTAS DE IMPLEMENTACIÓN

```
python
```

```
# Stack tecnológico para calidad de datos
quality_stack = {
    'validation_engine': 'Great Expectations',
    'monitoring': 'Apache Griffin + Grafana',
    'alerting': 'PagerDuty + Slack',
    'orchestration': 'Apache Airflow',
    'metadata': 'Apache Atlas',
    'profiling': 'Apache Griffin + pandas-profiling'
}
```

MÉTRICAS CONSOLIDADAS DE CALIDAD

Data Quality Score (DQS):

```
DQS = (Completitud × 0.25) + (Validez × 0.25) + (Consistencia × 0.20) +
      (Unicidad × 0.15) + (Actualidad × 0.15)
```

Donde cada dimensión se mide de 0 a 100%.

SLA de Calidad por Zona:

- **Raw Zone**: 85% DQS mínimo
- **Trusted Zone**: 95% DQS mínimo
- **Curated Zone**: 98% DQS mínimo

LECCIÓN 4 - MODELAMIENTO MULTIDIMENSIONAL

1. Selección del Área de Negocio

Área Seleccionada: **ANÁLISIS DE VENTAS Y COMPORTAMIENTO DEL CLIENTE**

Justificación: Esta área es crítica para el e-commerce ya que permite:

- Análisis de performance de ventas por múltiples dimensiones
- Segmentación y análisis de comportamiento de clientes
- Optimización de inventario y pricing
- Medición de efectividad de campañas de marketing
- Forecasting y planificación estratégica

Datos Disponibles (provenientes de las etapas anteriores):

- Transacciones de ventas (zona Curated)
- Información de productos y categorías
- Datos de clientes y segmentación
- Eventos de navegación web
- Datos de campañas de marketing
- Información de inventario y logística

2. Diseño del Modelo Dimensional

Modelo Seleccionado: **Esquema Estrella Híbrido**

Justificación:

- Esquema estrella para performance óptimo en consultas OLAP
- Elementos híbridos para manejar jerarquías complejas (geografía, productos)
- Balance entre simplicidad de consultas y flexibilidad analítica

TABLA DE HECHOS PRINCIPAL

FACT_SALES (Hechos de Ventas)

Granularidad: Una fila por ítem de pedido

sql

```sql
CREATE TABLE fact_sales (
    -- Claves foráneas
    date_key            INTEGER NOT NULL,
    customer_key        INTEGER NOT NULL,
    product_key         INTEGER NOT NULL,
    store_key           INTEGER NOT NULL,
    promotion_key       INTEGER,
    payment_method_key INTEGER NOT NULL,

    -- Claves de negocio
    order_id            VARCHAR(50) NOT NULL,
    order_line_id       VARCHAR(50) NOT NULL,

    -- Métricas aditivas
    quantity            DECIMAL(10,2) NOT NULL,
    unit_price          DECIMAL(10,2) NOT NULL,
    discount_amount     DECIMAL(10,2) DEFAULT 0,
    tax_amount          DECIMAL(10,2) NOT NULL,
    shipping_cost       DECIMAL(10,2) DEFAULT 0,
    total_amount        DECIMAL(10,2) NOT NULL,  -- cantidad * precio - descuento + impuesto

    -- Métricas semi-aditivas
    profit_margin       DECIMAL(10,2),
    cost_of_goods       DECIMAL(10,2),

    -- Métricas no-aditivas (calculadas)
    unit_profit         AS (unit_price - cost_of_goods - discount_amount/quantity),

    -- Metadata
    created_timestamp   TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_timestamp   TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    PRIMARY KEY (date_key, customer_key, product_key, order_line_id)
);
```

TABLAS DE DIMENSIONES

DIM_DATE (Dimensión Tiempo)

```sql
CREATE TABLE dim_date (
    date_key          INTEGER PRIMARY KEY,
    full_date         DATE NOT NULL,
    day_of_week       INTEGER,
    day_name          VARCHAR(10),
    day_of_month      INTEGER,
    day_of_year       INTEGER,
    week_of_year      INTEGER,
    month_number      INTEGER,
    month_name        VARCHAR(15),
    quarter           INTEGER,
    quarter_name      VARCHAR(2),
    year_number       INTEGER,
    is_weekend        BOOLEAN,
    is_holiday        BOOLEAN,
    holiday_name      VARCHAR(50),
    fiscal_year       INTEGER,
    fiscal_quarter    INTEGER,
    season            VARCHAR(10)
);
```

DIM_CUSTOMER (Dimensión Cliente)

```sql
```

```sql
CREATE TABLE dim_customer (
    customer_key       INTEGER PRIMARY KEY,
    customer_id        VARCHAR(50) NOT NULL,

    -- Atributos SCD Tipo 1 (sobrescribir)
    email              VARCHAR(100),
    phone              VARCHAR(20),
    status             VARCHAR(20),

    -- Atributos SCD Tipo 2 (historizar)
    first_name         VARCHAR(50),
    last_name          VARCHAR(100),
    birth_date         DATE,
    gender             VARCHAR(10),

    -- Jerarquía geográfica
    address_line1      VARCHAR(200),
    address_line2      VARCHAR(200),
    city               VARCHAR(100),
    state_province     VARCHAR(100),
    postal_code        VARCHAR(20),
    country            VARCHAR(100),

    -- Segmentación
    customer_segment   VARCHAR(50),    -- VIP, Regular, New, Churned
    lifetime_value     DECIMAL(12,2),
    acquisition_channel VARCHAR(50),

    -- SCD Tipo 2 fields
    effective_date     DATE NOT NULL,
    expiration_date    DATE,
    is_current         BOOLEAN DEFAULT TRUE,
```

```
    -- Metadata
    created_timestamp  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_timestamp  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

DIM_PRODUCT (Dimensión Producto)

Modelo Copo de Nieve para jerarquía de productos

sql

```sql
-- Tabla principal de producto
CREATE TABLE dim_product (
    product_key       INTEGER PRIMARY KEY,
    product_id        VARCHAR(50) NOT NULL,
    product_name      VARCHAR(200) NOT NULL,
    brand             VARCHAR(100),
    model             VARCHAR(100),
    color             VARCHAR(50),
    size              VARCHAR(50),
    weight            DECIMAL(8,2),

    -- Claves a tablas de jerarquía
    category_key      INTEGER,
    subcategory_key   INTEGER,

    -- Atributos de producto
    unit_cost         DECIMAL(10,2),
    standard_price    DECIMAL(10,2),
    product_status    VARCHAR(20),    -- Active, Discontinued, Seasonal

    -- Fechas importantes
    launch_date       DATE,
    discontinued_date DATE,

    -- Metadata
    created_timestamp  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_timestamp  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabla de categorías (jerarquía)
CREATE TABLE dim_product_category (
    category_key      INTEGER PRIMARY KEY,
    category_id       VARCHAR(50) NOT NULL,
```

```
    category_name      VARCHAR(100) NOT NULL,
    parent_category_key INTEGER,
    category_level     INTEGER,
    category_path      VARCHAR(500), -- /Electronics/Smartphones/iOS
    is_leaf_category   BOOLEAN
);

-- Tabla de subcategorías
CREATE TABLE dim_product_subcategory (
    subcategory_key    INTEGER PRIMARY KEY,
    subcategory_id     VARCHAR(50) NOT NULL,
    subcategory_name   VARCHAR(100) NOT NULL,
    category_key       INTEGER NOT NULL
);
```

DIM_STORE (Dimensión Tienda/Canal)

```
sql
```

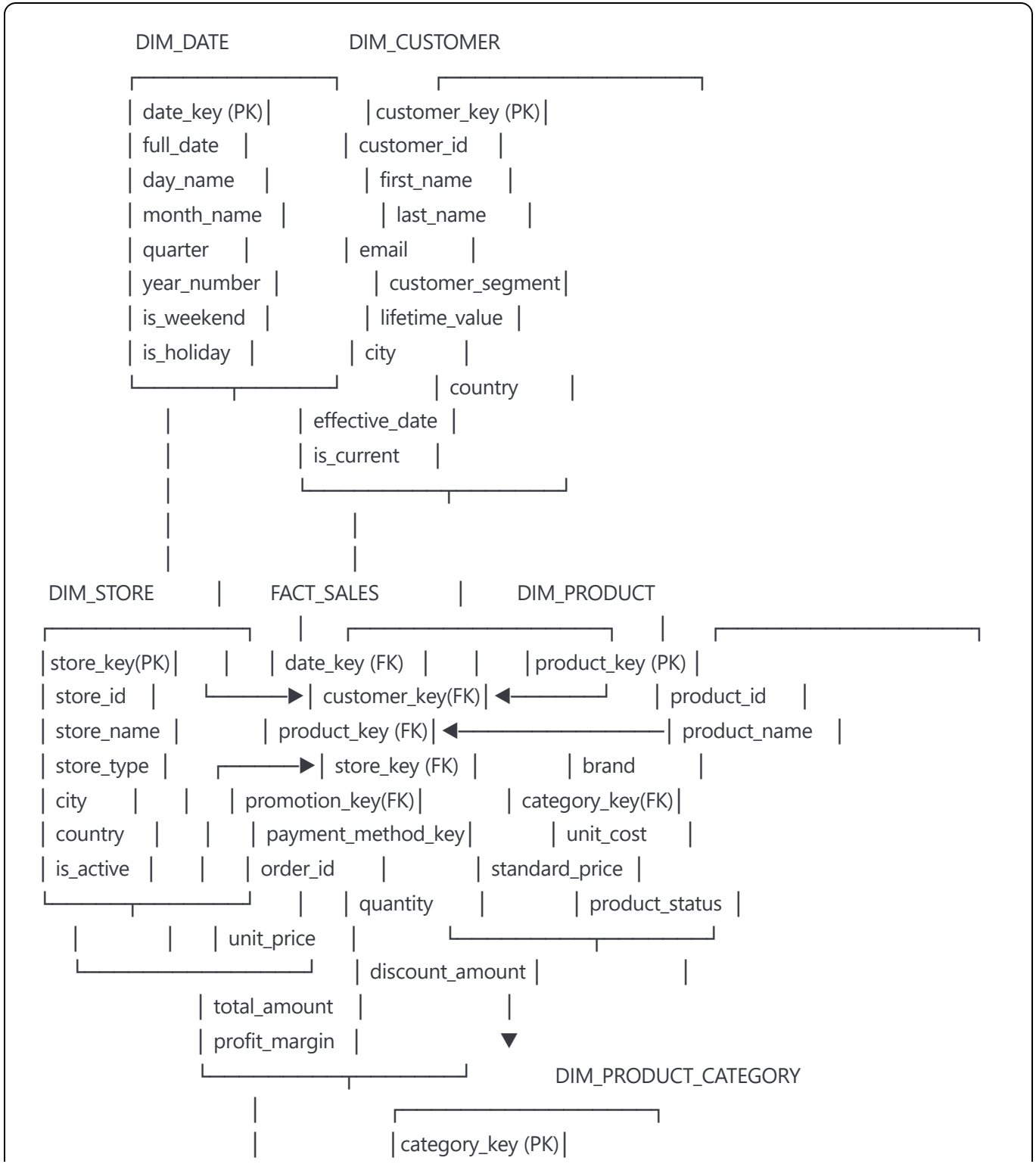
```sql
CREATE TABLE dim_store (
    store_key         INTEGER PRIMARY KEY,
    store_id          VARCHAR(50) NOT NULL,
    store_name        VARCHAR(100),
    store_type        VARCHAR(50),    -- Online, Physical, Mobile App

    -- Información geográfica (para tiendas físicas)
    address           VARCHAR(200),
    city              VARCHAR(100),
    state_province    VARCHAR(100),
    country           VARCHAR(100),
    postal_code       VARCHAR(20),
    timezone          VARCHAR(50),

    -- Información operativa
    opening_date      DATE,
    store_size_sqm    INTEGER,
    manager_name      VARCHAR(100),
    phone             VARCHAR(20),

    -- Status
    is_active         BOOLEAN DEFAULT TRUE,

    -- Metadata
    created_timestamp  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_timestamp  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

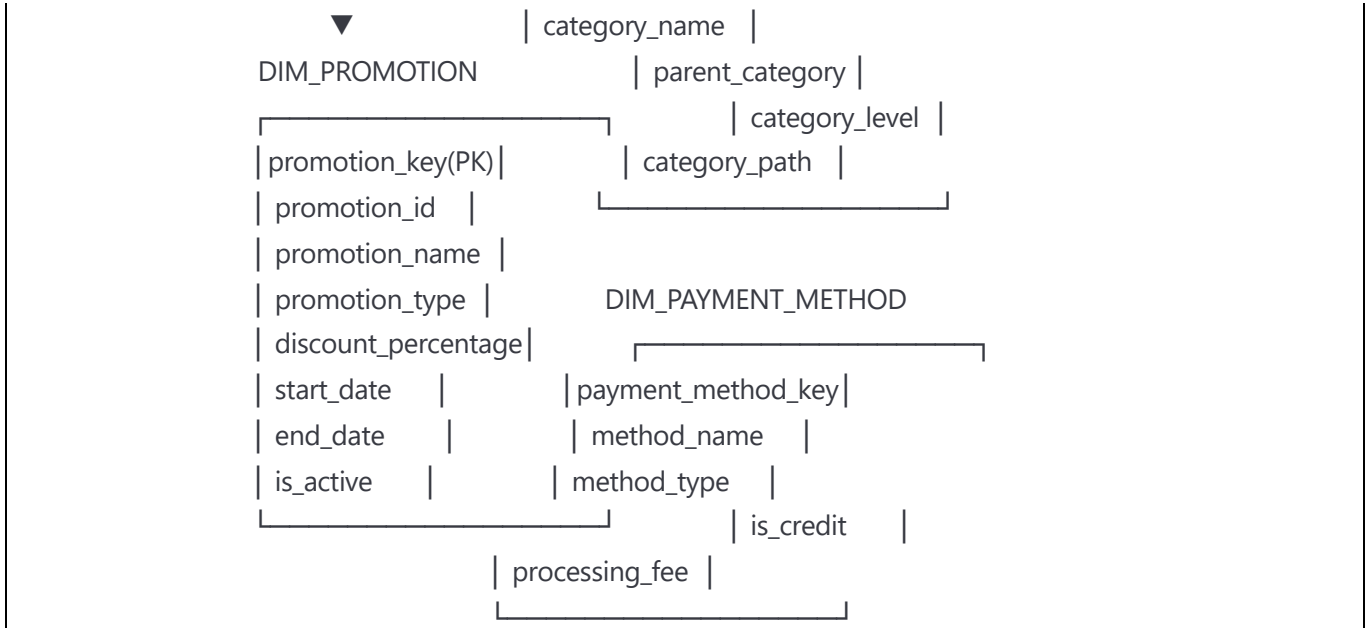
DIM_PROMOTION (Dimensión Promoción)

```sql
```

```sql
CREATE TABLE dim_promotion (
    promotion_key       INTEGER PRIMARY KEY,
    promotion_id        VARCHAR(50) NOT NULL,
    promotion_name      VARCHAR(200),
    promotion_type      VARCHAR(50),    -- Discount, BOGO, Free Shipping

    -- Detalles de la promoción
    discount_percentage DECIMAL(5,2),
    discount_amount     DECIMAL(10,2),
    minimum_purchase    DECIMAL(10,2),

    -- Período de validez
    start_date          DATE NOT NULL,
    end_date            DATE NOT NULL,

    -- Targeting
    target_customer_segment VARCHAR(100),
    target_product_category VARCHAR(100),

    -- Status
    is_active           BOOLEAN DEFAULT TRUE,

    -- Metadata
    created_timestamp   TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

3. Diagrama del Modelo Dimensional


DIM_DATE
date_key (PK)
full_date
day_name
month_name
quarter
year_number
is_weekend
is_holiday
DIM_CUSTOMER
customer_key (PK)
customer_id
first_name
last_name
email
customer_segment
lifetime_value
city
country
effective_date
is_current
DIM_STORE
store_key(PK)
store_id
store_name
store_type
city
country
is_active
FACT_SALES
date_key (FK)
customer_key(FK)
product_key (FK)
store_key (FK)
promotion_key(FK)
payment_method_key
order_id
quantity
unit_price
discount_amount
total_amount
profit_margin
DIM_PRODUCT
product_key (PK)
product_id
product_name
brand
category_key(FK)
unit_cost
standard_price
product_status
DIM_PRODUCT_CATEGORY
category_key (PK)



4. Jerarquías Implementadas

Jerarquía Temporal:

- **Año → Trimestre → Mes → Semana → Día**
- **Año Fiscal → Trimestre Fiscal → Período Fiscal**

Jerarquía Geográfica del Cliente:

- **País → Estado/Provincia → Ciudad → Código Postal**

Jerarquía de Productos:

- **Categoría → Subcategoría → Marca → Producto**

Jerarquía de Canales:

- **Tipo de Tienda → Región → Tienda Individual**

5. Documentación de Decisiones de Modelado

Decisiones de Normalización/Desnormalización:

Desnormalización Aplicada:

1. **Atributos de cliente en DIM_CUSTOMER**: Información geográfica desnormalizada para simplificar consultas comunes de análisis por ubicación.
2. **Métricas calculadas en FACT_SALES**: `unit_profit` como campo calculado para evitar cálculos repetitivos en consultas.
3. **Atributos de fecha expandidos**: Día de semana, nombre de mes, trimestre, etc., pre-calculados para facilitar agrupaciones temporales.

Normalización Aplicada:

1. **Jerarquía de productos**: Separación en tablas `dim_product_category` para manejar jerarquías complejas y cambios en categorización.
2. **SCD Tipo 2 para clientes**: Historización de cambios en información clave del cliente manteniendo integridad referencial.

Criterios Analíticos Implementados:

Granularidad:

- **Fact_Sales**: Nivel de línea de pedido para máxima flexibilidad analítica
- Permite agregaciones a nivel de pedido, cliente, producto, día, etc.

Aditividad de Métricas:

- **Aditivas**: `quantity`, `total_amount`, `discount_amount`, `tax_amount`
- **Semi-aditivas**: `profit_margin` (aditiva por producto/cliente, no por tiempo)
- **No-aditivas**: Ratios y porcentajes calculados en tiempo de consulta

Slowly Changing Dimensions (SCD):

- **Tipo 1**: Email, teléfono, status del cliente (sobrescribir)
- **Tipo 2**: Información demográfica, segmentación, ubicación (historizar)
- **Tipo 3**: No implementado (no requerido para este caso)

Manejo de Jerarquías:

- **Naturales**: Tiempo (año → mes → día)
- **Balanceadas**: Geografía (país → estado → ciudad)
- **Desbalanceadas**: Productos (algunas categorías tienen más niveles)

6. Casos de Uso Analíticos Soportados

Análisis de Ventas:

```sql
-- Ventas por trimestre y categoría de producto
SELECT
    d.quarter_name,
    pc.category_name,
    SUM(f.total_amount) as total_sales,
    SUM(f.quantity) as total_quantity,
    AVG(f.unit_price) as avg_unit_price
FROM fact_sales f
JOIN dim_date d ON f.date_key = d.date_key
JOIN dim_product p ON f.product_key = p.product_key
JOIN dim_product_category pc ON p.category_key = pc.category_key
WHERE d.year_number = 2024
GROUP BY d.quarter_name, pc.category_name
ORDER BY total_sales DESC;
```

Análisis de Comportamiento de Cliente:

```sql
-- Segmentación RFM (Recency, Frequency, Monetary)
SELECT
    c.customer_segment,
    COUNT(DISTINCT c.customer_key) as customer_count,
    AVG(customer_metrics.recency_days) as avg_recency,
    AVG(customer_metrics.frequency) as avg_frequency,
    AVG(customer_metrics.monetary_value) as avg_monetary
FROM dim_customer c
JOIN (
    SELECT
        customer_key,
        DATEDIFF(CURRENT_DATE, MAX(d.full_date)) as recency_days,
        COUNT(DISTINCT f.order_id) as frequency,
        SUM(f.total_amount) as monetary_value
    FROM fact_sales f
    JOIN dim_date d ON f.date_key = d.date_key
    WHERE d.full_date >= DATE_SUB(CURRENT_DATE, INTERVAL 365 DAY)
    GROUP BY customer_key
) customer_metrics ON c.customer_key = customer_metrics.customer_key
WHERE c.is_current = TRUE
GROUP BY c.customer_segment;
```

Análisis de Efectividad de Promociones:

```sql
```

```sql
-- Comparación de ventas con y sin promociones
SELECT
    CASE WHEN p.promotion_key IS NOT NULL THEN 'Con Promoción'
        ELSE 'Sin Promoción' END as promotion_status,
    COUNT(DISTINCT f.order_id) as order_count,
    SUM(f.total_amount) as total_revenue,
    AVG(f.total_amount) as avg_order_value,
    SUM(f.discount_amount) as total_discount
FROM fact_sales f
LEFT JOIN dim_promotion p ON f.promotion_key = p.promotion_key
JOIN dim_date d ON f.date_key = d.date_key
WHERE d.year_number = 2024
GROUP BY promotion_status;
```

7. Integración con la Arquitectura Propuesta

Fuente de Datos:

- Los datos para poblar este modelo dimensional provienen de la **Zona Curated** del Data Lake
- ETL/ELT processes orquestados por **Apache Airflow**
- Transformaciones implementadas con **dbt** para maintainability

Destino del Modelo:

- Implementado en **Snowflake Data Warehouse**
- Replicado en **Data Marts** específicos para diferentes áreas de negocio
- Optimizado con **clustering keys** y **materialized views** para performance

Actualización:

- **Fact_Sales**: Carga incremental diaria (micro-batch cada 4 horas para datos críticos)

- **Dimensiones**:
  - SCD Tipo 1: Actualización diaria
  - SCD Tipo 2: Detección de cambios y creación de nuevas versiones
- **Validaciones de calidad**: Integradas en el pipeline usando Great Expectations

Consumo:

- **Tableau/Power BI**: Dashboards ejecutivos y operacionales
- **APIs REST**: Para aplicaciones y reportes embebidos
- **Jupyter Notebooks**: Para análisis ad-hoc y machine learning
- **Automated Reports**: Reportes programados via email/Slack

---

DOCUMENTO INTEGRADOR FINAL

Resumen Ejecutivo

El proyecto "Arquitectura de Datos para E-commerce" presenta una solución integral que abarca desde la ingesta de datos heterogéneos hasta el consumo analítico a través de un modelo dimensional optimizado. La arquitectura propuesta es escalable, segura y alineada con las mejores prácticas de la industria.

Componentes Clave de la Solución

1. Arquitectura de 5 Capas:

- **Ingesta**: Kafka + Airflow para batch y streaming
- **Integración**: Spark + dbt para transformaciones
- **Almacenamiento**: Data Lake (S3) + Data Warehouse (Snowflake)
- **Calidad**: Great Expectations + Apache Griffin

- **Consumo**: Tableau + APIs + ML Platforms

2. Estrategia de Almacenamiento Lakehouse:

- **Raw Zone**: Datos originales sin transformar
- **Trusted Zone**: Datos validados y limpios
- **Curated Zone**: Datos agregados listos para análisis
- Data Warehouse y Data Marts especializados

3. Framework de Calidad de Datos:

- Validaciones automáticas en cada capa
- Monitoreo continuo con alertas proactivas
- Procesos de remediación automatizados
- SLA diferenciados por zona (85%, 95%, 98%)

4. Modelo Dimensional Híbrido:

- Esquema estrella optimizado para OLAP
- SCD Tipo 2 para historización
- Jerarquías balanceadas y desbalanceadas
- Métricas aditivas y semi-aditivas

Beneficios Esperados

Operacionales:

- Reducción del 70% en tiempo de preparación de datos
- Automatización de validaciones y alertas
- Trazabilidad completa de datos (end-to-end lineage)

- SLA de disponibilidad 99.9%

Analíticos:

- Acceso self-service a datos confiables
- Reducción del 50% en tiempo de generación de reportes
- Capacidades avanzadas de analytics y ML
- Soporte para análisis en tiempo real

Estratégicos:

- Escalabilidad para crecimiento 10x
- Flexibilidad para nuevas fuentes de datos
- Cumplimiento de regulaciones de privacidad
- ROI esperado: 300% en 2 años

Consideraciones de Implementación

Fases de Rollout:

1. **Fase 1 (3 meses)**: Implementación del Data Lake y ingesta básica
2. **Fase 2 (2 meses)**: Data Warehouse y primeros Data Marts
3. **Fase 3 (2 meses)**: Framework de calidad y monitoreo
4. **Fase 4 (1 mes)**: Modelo dimensional y herramientas de consumo

Recursos Requeridos:

- **Equipo técnico**: 1 Arquitecto, 2 Ingenieros de Datos, 1 Analytics Engineer
- **Budget estimado**: $500K setup + $100K/mes operación
- **Timeline**: 8 meses para implementación completa

Riesgos y Mitigaciones:

- **Riesgo**: Complejidad de integración → **Mitigación**: PoCs previos
- **Riesgo**: Adopción por usuarios → **Mitigación**: Training program
- **Riesgo**: Performance → **Mitigación**: Load testing y optimización

Próximos Pasos

1. **Aprobación del diseño** por comité de arquitectura
2. **Selección de proveedores** cloud y herramientas
3. **Setup del equipo** y environments de desarrollo
4. **Inicio de Fase 1** con fuentes críticas de datos

ANEXOS

Anexo A: Tecnologías Evaluadas

Categoría	Opción 1	Opción 2	Opción 3	Seleccionada	Justificación
Data Lake	S3	ADLS Gen2	GCS	S3	Madurez, ecosistema
DW	Snowflake	BigQuery	Redshift	Snowflake	Separación compute/storage
Streaming	Kafka	Kinesis	Pub/Sub	Kafka	Open source, flexibilidad
Orchestration	Airflow	Prefect	Dagster	Airflow	Madurez, comunidad
Quality	Great Expectations	Deequ	Apache Griffin	Great Expectations	Facilidad de uso

Anexo B: Cálculos de Dimensionamiento

Volúmenes de Datos Estimados:

- **Transaccional**: 10GB/día → 3.6TB/año
- **Logs web**: 50GB/día → 18TB/año
- **Social media**: 5GB/día → 1.8TB/año
- **Total crudo**: ~25TB/año
- **Con replicación y procesamiento**: ~75TB/año

Capacidad de Procesamiento:

- **Batch jobs**: 500 jobs/día promedio
- **Streaming**: 10K events/segundo pico
- **Consultas OLAP**: 1000 queries/día
- **Usuarios concurrentes**: 50 analistas

Anexo C: Métricas de Éxito

KPIs Técnicos:

- Uptime: > 99.9%
- Latencia de ingesta: < 5 minutos (batch), < 30 segundos (streaming)
- Query performance: < 10 segundos (95% de consultas)
- Data quality score: > 95%

KPIs de Negocio:

- Time-to-insight: < 1 día
- Self-service adoption: > 80% de analistas
- Cost per query: < $0.10

- Data-driven decision rate: > 90%

CONCLUSIONES

La arquitectura propuesta representa una solución moderna y escalable que posiciona a la empresa para aprovechar sus datos como ventaja competitiva. La combinación de tecnologías cloud-native, principios de data mesh, y un enfoque robusto de calidad de datos garantiza una plataforma sólida para el crecimiento futuro.

El modelo dimensional diseñado específicamente para análisis de ventas y comportamiento del cliente proporciona las bases para insights profundos que impulsarán la toma de decisiones estratégicas y operacionales.

La implementación por fases minimiza riesgos mientras maximiza el valor entregado, permitiendo quick wins tempranos que justifican la inversión continua en la plataforma de datos.