# LAPORAN KELOMPOK TUGAS MISI KEDUA
# PERANCANGAN DATA WAREHOUSE INDUSTRI KESEHATAN


**Disusun Oleh:**

**Kelompok 3 :**

| | |
|---|---|
| Taufiqurrahmansyah E | (120450051) |
| Marshanda Putri P | (121450020) |
| Putri Durrotul Shopia | (121450116) |
| Khoirul Mizan Abdullah | (122450010) |
| Mutiara Dian Pitaloka | (122450047) |
| Uliano Wilyam Purba | (122450098) |

**PROGRAM STUDI SAINS DATA**
**FAKULTAS SAINS**
**INSTITUT TEKNOLOGI SUMATERA 2025**

## 1. Pendahuluan

Berdasarkan analisis kebutuhan dan perancangan konseptual yang telah dilakukan pada misi sebelumnya, laporan ini fokus pada transisi dari desain konseptual ke desain logikal dan fisikal data warehouse untuk industri kesehatan. Pada misi ketiga ini, kami akan menjelaskan implementasi tabel dimensi dan fakta, perancangan indeks, strategi penyimpanan, serta partisi tabel dan view untuk mendukung kebutuhan analitik rumah sakit dan klinik.

## 2. Tinjauan Konsep Data Warehouse

### 2.1 Konsep Dimensi dan Fakta

#### Dimensi

Tabel dimensi berisi atribut deskriptif yang memberikan konteks pada data kuantitatif dalam tabel fakta. Dimensi merepresentasikan entitas bisnis seperti pasien, dokter, waktu, obat, dan tindakan medis. Karakteristik utama dimensi:

- Berisi informasi tekstual dan deskriptif
- Memiliki struktur hierarki untuk mendukung operasi drill-down/roll-up
- Memiliki primary key yang terhubung ke tabel fakta
- Ukuran tabel yang relatif kecil dibanding tabel fakta

#### Fakta

Tabel fakta menyimpan ukuran kinerja bisnis atau metrics yang merepresentasikan proses bisnis. Pada konteks kesehatan, fakta meliputi perawatan pasien, transaksi obat, dan tindakan medis. Karakteristik tabel fakta:

- Berisi data kuantitatif (measures) seperti jumlah tindakan, biaya, dan total tagihan
- Memiliki foreign key yang merujuk ke tabel dimensi
- Ukuran tabel yang besar dan terus bertambah
- Tingkat detail data (granularitas) yang menentukan tingkat analisis yang dapat dilakukan

### 2.2 Star Schema vs Snowflake Schema

#### Star Schema

Star schema adalah model dimensional dengan satu tabel fakta yang dikelilingi oleh tabel dimensi yang terhubung langsung ke tabel fakta. Karakteristik star schema:

- Struktur sederhana dengan satu level relasi
- Performa query yang optimal karena meminimalkan jumlah join
- Redundansi data yang lebih tinggi
- Mudah dipahami oleh pengguna bisnis

Pada industri kesehatan, star schema memudahkan analisis seperti:

- Total perawatan berdasarkan spesialisasi dokter dan periode waktu
- Biaya pengobatan per kategori pasien
- Penggunaan obat berdasarkan jenis tindakan

**Snowflake Schema**

Snowflake schema adalah model dimensional dengan normalisasi tabel dimensi menjadi beberapa sub-dimensi. Karakteristik snowflake schema:

- Struktur kompleks dengan multiple level relasi
- Mengurangi redundansi data (normalisasi)
- Memerlukan lebih banyak join yang dapat memperlambat performa query
- Lebih sulit dipahami oleh pengguna bisnis

Untuk kasus data warehouse kesehatan ini, kami memilih implementasi star schema karena:

1. Memprioritaskan kecepatan query untuk pengambilan keputusan klinis
2. Kemudahan penggunaan oleh staf rumah sakit/klinik yang mungkin tidak memiliki background IT yang kuat
3. Volume data kesehatan yang relatif terkelola dibanding industri lain seperti retail atau telekomunikasi

## 3. Desain Logikal Data Warehouse

### 3.1 Implementasi Tabel Dimensi

Berdasarkan skema konseptual yang telah dirancang pada misi kedua, berikut adalah implementasi logikal tabel dimensi:

**Dim_Pasien**

```
CREATE TABLE Dim_Pasien (
    id_pasien INT PRIMARY KEY,
    nama_pasien VARCHAR(100) NOT NULL,
    alamat VARCHAR(200),
    kota VARCHAR(50),
    provinsi VARCHAR(50),
    tanggal_lahir DATE,
    jenis_kelamin CHAR(1),
    nomor_telepon VARCHAR(15),
    row_effective_date DATE, -- Untuk SCD Type 2
    row_expiration_date DATE, -- Untuk SCD Type 2
    current_flag CHAR(1)      -- Untuk SCD Type 2
```

```
);
```

**Dim_Dokter**

```
CREATE TABLE Dim_Dokter (
    id_dokter INT PRIMARY KEY,
    nama_dokter VARCHAR(100) NOT NULL,
    spesialisasi VARCHAR(50),
    sub_spesialisasi VARCHAR(50),
    nomor_telepon VARCHAR(15),
    unit_kerja VARCHAR(50),
    row_effective_date DATE,
    row_expiration_date DATE,
    current_flag CHAR(1)
);
```

**Dim_Tindakan**

```
CREATE TABLE Dim_Tindakan (
    id_tindakan INT PRIMARY KEY,
    nama_tindakan VARCHAR(100) NOT NULL,
    kategori_tindakan VARCHAR(50),
    sub_kategori_tindakan VARCHAR(50),
    deskripsi_tindakan TEXT
);
```

**Dim_Obat**

```
CREATE TABLE Dim_Obat (
    id_obat VARCHAR(10) PRIMARY KEY,
    nama_obat VARCHAR(100) NOT NULL,
    kategori_obat VARCHAR(50),
    sub_kategori_obat VARCHAR(50),
    satuan VARCHAR(20),
    harga_per_satuan DECIMAL(10,2),
    dosis_standar VARCHAR(50)
);
```

**Dim_Jenis_Perawatan**

```
CREATE TABLE Dim_Jenis_Perawatan (
    id_jenis INT PRIMARY KEY,
    jenis_perawatan VARCHAR(50) NOT NULL,
    deskripsi TEXT
```

```
);
```

**Dim_Waktu**

```
CREATE TABLE Dim_Waktu (
    id_waktu INT PRIMARY KEY,
    tanggal DATE NOT NULL,
    hari VARCHAR(10),
    bulan VARCHAR(10),
    nama_bulan VARCHAR(20),
    kuartal INT,
    tahun INT,
    hari_dalam_minggu INT,
    minggu_dalam_tahun INT,
    hari_libur BOOLEAN,
    weekend BOOLEAN
);
```

### 3.2 Implementasi Tabel Fakta

**Fact_Perawatan**

```
CREATE TABLE Fact_Perawatan (
    id_perawatan INT PRIMARY KEY,
    fk_pasien INT REFERENCES Dim_Pasien(id_pasien),
    fk_dokter INT REFERENCES Dim_Dokter(id_dokter),
    fk_jenis_perawatan INT REFERENCES Dim_Jenis_Perawatan(id_jenis),
    fk_waktu INT REFERENCES Dim_Waktu(id_waktu),
    total_biaya DECIMAL(12,2),
    jumlah_tindakan INT,
    status_pembayaran VARCHAR(20),
    durasi_perawatan INT -- dalam menit
);
```

**Fact_Tindakan**

```
CREATE TABLE Fact_Tindakan (
    id_record INT PRIMARY KEY,
    id_tindakan INT,
    fk_perawatan INT REFERENCES Fact_Perawatan(id_perawatan),
    fk_tipe INT REFERENCES Dim_Tindakan(id_tindakan),
    fk_waktu INT REFERENCES Dim_Waktu(id_waktu),
    biaya DECIMAL(10,2),
    durasi INT, -- dalam menit
```

```
    catatan TEXT
);
```

**Fact_Obat**

```
CREATE TABLE Fact_Obat (
    id_obat_transaksi INT PRIMARY KEY,
    fk_pasien INT REFERENCES Dim_Pasien(id_pasien),
    fk_obat VARCHAR(10) REFERENCES Dim_Obat(id_obat),
    fk_perawatan INT REFERENCES Fact_Perawatan(id_perawatan),
    fk_waktu INT REFERENCES Dim_Waktu(id_waktu),
    jumlah INT,
    total_harga DECIMAL(10,2),
    status_tersedia BOOLEAN
);
```

### 3.3 Slowly Changing Dimensions (SCD)

Untuk industri kesehatan, beberapa dimensi memerlukan penanganan khusus untuk perubahan data. Kami mengimplementasikan SCD tipe 2 untuk:

1. **Dim_Pasien**:
    - Informasi alamat dan nomor telepon pasien dapat berubah seiring waktu
    - Riwayat alamat pasien penting untuk analisis epidemiologi dan pemetaan penyakit
2. **Dim_Dokter**:
    - Perubahan spesialisasi atau unit kerja perlu didokumentasikan
    - Riwayat dokter penting untuk analisis kinerja historis

Kolom tambahan untuk implementasi SCD tipe 2:

- row_effective_date: Tanggal mulai berlakunya record
- row_expiration_date: Tanggal berakhirnya record
- current_flag: Penanda record aktif (Y/N)

## 4. Desain Fisikal Data Warehouse

### 4.1 Implementasi Indeks

Indeks dirancang untuk mengoptimalkan performa query yang sering dijalankan pada data warehouse kesehatan:

**Indeks untuk Tabel Dimensi**

```sql
-- Indeks untuk pencarian cepat berdasarkan nama pasien
CREATE INDEX idx_dim_pasien_nama ON Dim_Pasien(nama_pasien);

-- Indeks untuk pencarian berdasarkan spesialisasi dokter
CREATE INDEX idx_dim_dokter_spesialisasi ON Dim_Dokter(spesialisasi);

-- Indeks komposit untuk dimensi waktu
CREATE INDEX idx_dim_waktu_tahun_bulan ON Dim_Waktu(tahun, bulan);

-- Indeks untuk pencarian obat berdasarkan kategori
CREATE INDEX idx_dim_obat_kategori ON Dim_Obat(kategori_obat);
```

**Indeks untuk Tabel Fakta**

```sql
-- Indeks untuk foreign key di tabel fakta perawatan
CREATE INDEX idx_fact_perawatan_pasien ON Fact_Perawatan(fk_pasien);
CREATE INDEX idx_fact_perawatan_dokter ON Fact_Perawatan(fk_dokter);
CREATE INDEX idx_fact_perawatan_waktu ON Fact_Perawatan(fk_waktu);
CREATE INDEX idx_fact_perawatan_pembayaran ON
Fact_Perawatan(status_pembayaran);

-- Indeks untuk foreign key di tabel fakta tindakan
CREATE INDEX idx_fact_tindakan_perawatan ON Fact_Tindakan(fk_perawatan);
CREATE INDEX idx_fact_tindakan_tipe ON Fact_Tindakan(fk_tipe);

-- Indeks untuk foreign key di tabel fakta obat
CREATE INDEX idx_fact_obat_pasien ON Fact_Obat(fk_pasien);
CREATE INDEX idx_fact_obat_obat ON Fact_Obat(fk_obat);
CREATE INDEX idx_fact_obat_perawatan ON Fact_Obat(fk_perawatan);
```

**Justifikasi Pemilihan Indeks**

1. **Indeks B-Tree** digunakan untuk kolom yang sering digunakan dalam klausa WHERE dan JOIN
    - Cocok untuk foreign key di tabel fakta yang sering digunakan untuk join dengan dimensi
    - Optimal untuk query yang mencari nilai spesifik (equality search)
2. **Indeks Bitmap** untuk kolom dengan kardinalitas rendah
    - Diterapkan pada kolom seperti jenis_kelamin, status_pembayaran
    - Efisien untuk query analitik yang mengagregasi data berdasarkan kategori
3. **Indeks Komposit** untuk query yang sering melibatkan beberapa kolom
    - Kombinasi tahun dan bulan untuk analisis tren temporal
    - Dokter dan spesialisasi untuk analisis kinerja per kategori dokter

**4.2 Desain Penyimpanan (*Storage*)**

Strategi penyimpanan dirancang untuk mengoptimalkan akses data dengan mempertimbangkan volume dan pola akses:

**Pemilihan Format Penyimpanan**

1. ***Columnar Storage*** untuk tabel fakta
    - Meningkatkan performa query analitik yang hanya mengakses subset kolom
    - Memungkinkan kompresi data yang lebih baik untuk kolom dengan nilai serupa
    - Contoh implementasi:

```
-- PostgreSQL dengan extension cstore_fdw
CREATE FOREIGN TABLE fact_perawatan_columnar (
    -- definisi kolom sama dengan fact_perawatan
) SERVER cstore_server OPTIONS(compression 'pglz');
```

2. ***Row Storage*** untuk tabel dimensi
    - Lebih efisien untuk operasi lookup yang mengambil semua kolom record
    - Lebih sederhana untuk implementasi SCD Type 2

**Kompresi Data**

1. **Kompresi Dictionary** untuk string pada tabel dimensi
    - Mengurangi ukuran penyimpanan untuk nilai teks yang berulang seperti nama kota, spesialisasi dokter
    - Implementasi:

```
-- Contoh penyimpanan kompresi pada SQL Server
ALTER TABLE Dim_Dokter REBUILD WITH (DATA_COMPRESSION = PAGE);
```

2. **Kompresi Run-Length** untuk kolom dengan banyak nilai berurutan yang sama
    - Cocok untuk kolom tanggal pada Dim_Waktu

**Kebijakan Retensi Data**

1. Data transaksi detail (atomik) disimpan selama 5 tahun
2. Data agregat (bulanan, tahunan) disimpan permanen
3. Implementasi arsip data untuk data historis > 5 tahun yang jarang diakses

**4.3 Implementasi Partisi Tabel**

Partisi tabel dirancang untuk meningkatkan performa dan pengelolaan data besar:

**Partisi untuk Tabel Fakta**

1. **Partisi Berdasarkan Waktu (Range Partitioning)**

```sql
-- Partisi tabel Fact_Perawatan berdasarkan tahun
CREATE TABLE Fact_Perawatan (
    id_perawatan INT,
    fk_pasien INT,
    fk_dokter INT,
    fk_jenis_perawatan INT,
    fk_waktu INT,
    total_biaya DECIMAL(12,2),
    jumlah_tindakan INT,
    status_pembayaran VARCHAR(20),
    durasi_perawatan INT
) PARTITION BY RANGE (fk_waktu);

-- Membuat partisi untuk tahun 2023
CREATE TABLE Fact_Perawatan_2023 PARTITION OF Fact_Perawatan
    FOR VALUES FROM (20230101) TO (20240101);

-- Membuat partisi untuk tahun 2024
CREATE TABLE Fact_Perawatan_2024 PARTITION OF Fact_Perawatan
    FOR VALUES FROM (20240101) TO (20250101);

-- Membuat partisi untuk tahun 2025
CREATE TABLE Fact_Perawatan_2025 PARTITION OF Fact_Perawatan
    FOR VALUES FROM (20250101) TO (20260101);
```

2. **Partisi Berdasarkan Jenis Perawatan (List Partitioning)**

```sql
-- Partisi tabel Fact_Tindakan berdasarkan jenis tindakan
CREATE TABLE Fact_Tindakan (
    id_record INT,
    id_tindakan INT,
    fk_perawatan INT,
    fk_tipe INT,
    fk_waktu INT,
    biaya DECIMAL(10,2),
    durasi INT,
    catatan TEXT
) PARTITION BY LIST (fk_tipe);

-- Partisi untuk tindakan umum
CREATE TABLE Fact_Tindakan_Umum PARTITION OF Fact_Tindakan
    FOR VALUES IN (1, 2, 3, 4, 5); -- ID untuk tindakan umum
```

```
-- Partisi untuk tindakan bedah
CREATE TABLE Fact_Tindakan_Bedah PARTITION OF Fact_Tindakan
    FOR VALUES IN (10, 11, 12, 13, 14, 15); -- ID untuk tindakan bedah

-- Partisi untuk tindakan diagnostik
CREATE TABLE Fact_Tindakan_Diagnostik PARTITION OF Fact_Tindakan
    FOR VALUES IN (20, 21, 22, 23, 24); -- ID untuk tindakan diagnostik
```

**Manfaat Partisi**

1. Meningkatkan performa query yang hanya mengakses subset data
    - Query analitik tahunan hanya mengakses partisi yang relevan
    - Filter berdasarkan kategori tindakan lebih efisien
2. Mempermudah manajemen data
    - Backup dan restore per periode atau kategori
    - Memindahkan data historis ke media penyimpanan yang lebih murah
3. Memungkinkan paralelisasi query
    - Query dapat dijalankan secara paralel pada beberapa partisi

**4.4 View untuk Analisis**

View dirancang untuk menyederhanakan query kompleks dan mendukung kebutuhan analitik:

**Materialized View untuk Pelaporan Rutin**

```
-- View untuk analisis pendapatan per spesialisasi dokter per bulan
CREATE MATERIALIZED VIEW mv_pendapatan_per_spesialisasi AS
SELECT
    d.spesialisasi,
    w.tahun,
    w.bulan,
    COUNT(DISTINCT fp.id_perawatan) AS jumlah_perawatan,
    SUM(fp.total_biaya) AS total_pendapatan
FROM
    Fact_Perawatan fp
    JOIN Dim_Dokter d ON fp.fk_dokter = d.id_dokter
    JOIN Dim_Waktu w ON fp.fk_waktu = w.id_waktu
GROUP BY
    d.spesialisasi, w.tahun, w.bulan
WITH DATA;

-- Indeks untuk materialized view
CREATE INDEX idx_mv_pendapatan_spesialisasi ON
```

```
mv_pendapatan_per_spesialisasi(spesialisasi);
CREATE INDEX idx_mv_pendapatan_tahun_bulan ON
mv_pendapatan_per_spesialisasi(tahun, bulan);
```

**Analytic View untuk Tren Penggunaan Obat**

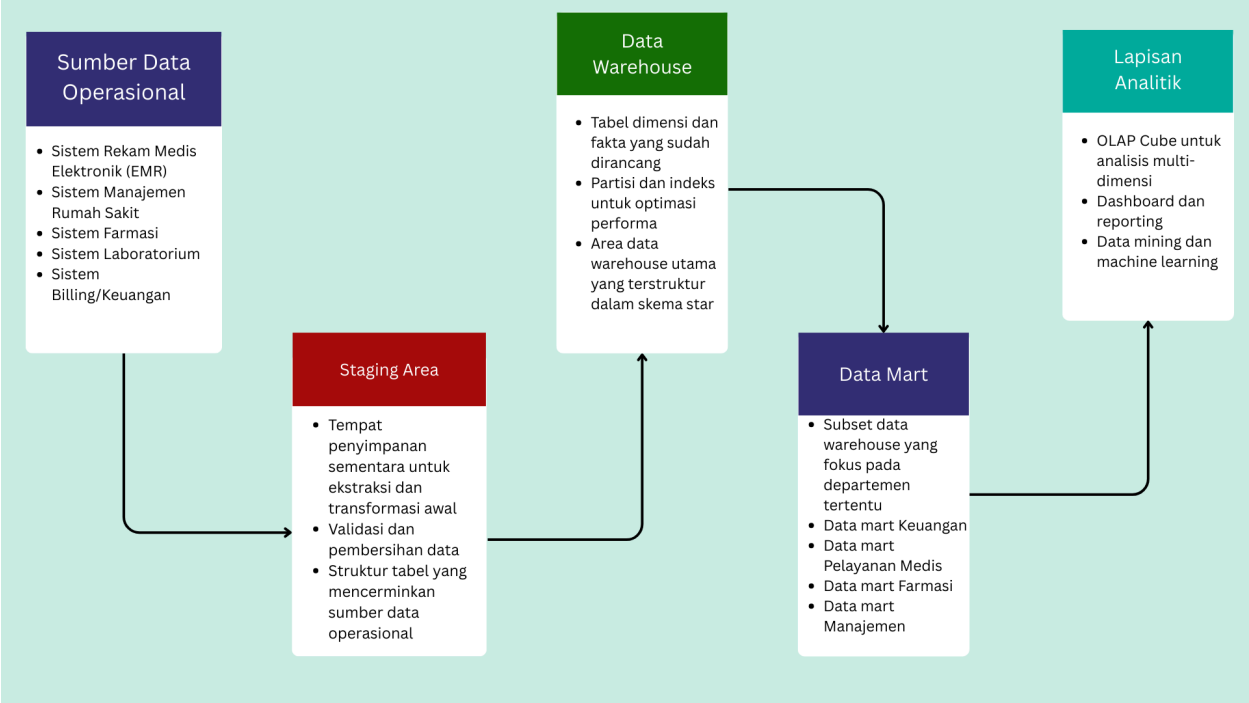
```
-- View untuk analisis tren penggunaan obat berdasarkan kategori
CREATE VIEW v_tren_penggunaan_obat AS
SELECT
    o.kategori_obat,
    w.tahun,
    w.bulan,
    SUM(fo.jumlah) AS total_penggunaan,
    COUNT(DISTINCT fo.fk_pasien) AS jumlah_pasien,
    SUM(fo.total_harga) AS total_biaya
FROM
    Fact_Obat fo
    JOIN Dim_Obat o ON fo.fk_obat = o.id_obat
    JOIN Dim_Waktu w ON fo.fk_waktu = w.id_waktu
GROUP BY
    o.kategori_obat, w.tahun, w.bulan;
```

**View untuk KPI Klinik**

```
-- View untuk Key Performance Indicators klinik
CREATE VIEW v_kpi_klinik AS
SELECT
    w.tahun,
    w.bulan,
    COUNT(DISTINCT fp.id_perawatan) AS jumlah_perawatan,
    COUNT(DISTINCT fp.fk_pasien) AS jumlah_pasien,
    AVG(fp.durasi_perawatan) AS rata_durasi_perawatan,
    SUM(fp.total_biaya) AS total_pendapatan,
    COUNT(CASE WHEN fp.status_pembayaran = 'Lunas' THEN 1 END) / COUNT(*) *
100 AS persentase_lunas
FROM
    Fact_Perawatan fp
    JOIN Dim_Waktu w ON fp.fk_waktu = w.id_waktu
GROUP BY
    w.tahun, w.bulan;
```

## 5. Alur Aliran Data dan ETL Pipeline

### 5.1 Arsitektur Data Warehouse




Arsitektur data warehouse industri kesehatan terdiri dari beberapa komponen utama:

1. **Sumber Data Operasional**
    - Sistem Rekam Medis Elektronik (EMR)
    - Sistem Manajemen Rumah Sakit
    - Sistem Farmasi
    - Sistem Laboratorium
    - Sistem Billing/Keuangan
2. **Staging Area**
    - Tempat penyimpanan sementara untuk ekstraksi dan transformasi awal
    - Validasi dan pembersihan data
    - Struktur tabel yang mencerminkan sumber data operasional
3. **Data Warehouse**
    - Tabel dimensi dan fakta yang sudah dirancang
    - Partisi dan indeks untuk optimasi performa
    - Area data warehouse utama yang terstruktur dalam skema star
4. **Data Mart**
    - Subset data warehouse yang fokus pada departemen tertentu
    - Data mart Keuangan
    - Data mart Pelayanan Medis
    - Data mart Farmasi
    - Data mart Manajemen

5. **Lapisan Analitik**
    - OLAP Cube untuk analisis multi-dimensi
    - Dashboard dan reporting
    - Data mining dan machine learning

### 5.2 ETL Pipeline

**Proses Ekstraksi**

```
# Pseudocode untuk ekstraksi data pasien
def extract_patient_data():
    # Koneksi ke sistem sumber (EMR)
    conn = connect_to_source_system("EMR")

    # Query untuk ekstraksi data pasien
    query = """
    SELECT
        id_pasien, nama_pasien, alamat, tanggal_lahir,
        jenis_kelamin, nomor_telepon,
        CURRENT_DATE as extract_date
    FROM
        pasien
    WHERE
        last_update_date > ?
    """

    # Ekstraksi data ke staging
    patients = execute_query(conn, query, [last_extraction_date])
    save_to_staging("staging_pasien", patients)

    return len(patients)
```

**Proses Transformasi**

```
# Pseudocode untuk transformasi data pasien
def transform_patient_data():
    # Koneksi ke staging area
    conn = connect_to_staging()

    # Ekstrak data dari staging
    staging_patients = execute_query(conn, "SELECT * FROM staging_pasien")

    transformed_patients = []
```

```
    for patient in staging_patients:
        # Parse alamat menjadi komponen (kota, provinsi)
        address_components = parse_address(patient['alamat'])

        # Format nama sesuai standar
        formatted_name = standardize_name(patient['nama_pasien'])

        # Transformasi tanggal lahir dan hitung umur
        birthdate = standardize_date(patient['tanggal_lahir'])

        # Membuat record baru yang sudah ditransformasi
        transformed_patient = {
            'id_pasien': patient['id_pasien'],
            'nama_pasien': formatted_name,
            'alamat': patient['alamat'],
            'kota': address_components['kota'],
            'provinsi': address_components['provinsi'],
            'tanggal_lahir': birthdate,
            'jenis_kelamin': patient['jenis_kelamin'],
            'nomor_telepon': standardize_phone(patient['nomor_telepon']),
            'row_effective_date': patient['extract_date'],
            'row_expiration_date': '9999-12-31',
            'current_flag': 'Y'
        }

        transformed_patients.append(transformed_patient)

    return transformed_patients
```

**Proses Loading**

```
# Pseudocode untuk loading data pasien ke dimensi
def load_patient_dimension(transformed_patients):
    # Koneksi ke data warehouse
    conn = connect_to_datawarehouse()

    for patient in transformed_patients:
        # Cek apakah pasien sudah ada di dimensi
        existing = execute_query(conn,
            "SELECT * FROM Dim_Pasien WHERE id_pasien = ? AND current_flag
= 'Y'",
            [patient['id_pasien']])
```

```
        if not existing:
            # Insert new patient
            insert_into_dimension(conn, "Dim_Pasien", patient)
        else:
            # Cek apakah ada perubahan pada data pasien
            if has_changes(existing[0], patient):
                # Update flag pada record lama
                update_dimension(conn,
                    "UPDATE Dim_Pasien SET row_expiration_date = ?,
current_flag = 'N' " +
                    "WHERE id_pasien = ? AND current_flag = 'Y'",
                    [patient['row_effective_date'], patient['id_pasien']])

                # Insert record baru
                insert_into_dimension(conn, "Dim_Pasien", patient)
```

### 5.3 Scheduler dan Monitoring

ETL pipeline dijadwalkan dengan frekuensi yang berbeda berdasarkan kebutuhan:

1. **Daily ETL**
   - Data perawatan dan tindakan harian
   - Pembaruan fakta transaksi obat
   - Refresh materialized view untuk dashboard harian
2. **Weekly ETL**
   - Load lengkap untuk tabel dimensi pasien dan dokter
   - Validasi data dan pemeriksaan konsistensi
3. **Monthly ETL**
   - Agregasi data bulanan ke tabel ringkasan
   - Pembersihan staging area untuk optimasi ruang
4. **Monitoring & Alerting**
   - Pemantauan durasi job ETL
   - Validasi jumlah record yang diekstrak dan dimuat
   - Alert saat terjadi kegagalan atau anomali data

## 6. Penerapan Skrip Query untuk Analisis

### 6.1 Query untuk Analisis Perawatan Pasien

```
-- Analisis volume pasien berdasarkan waktu dan jenis perawatan
SELECT
    dw.tahun,
    dw.nama_bulan,
```

```
    djp.jenis_perawatan,
    COUNT(DISTINCT fp.id_perawatan) AS jumlah_perawatan,
    COUNT(DISTINCT fp.fk_pasien) AS jumlah_pasien,
    AVG(fp.total_biaya) AS rata_biaya
FROM
    Fact_Perawatan fp
    JOIN Dim_Waktu dw ON fp.fk_waktu = dw.id_waktu
    JOIN Dim_Jenis_Perawatan djp ON fp.fk_jenis_perawatan = djp.id_jenis
WHERE
    dw.tahun = 2025
GROUP BY
    dw.tahun, dw.nama_bulan, djp.jenis_perawatan
ORDER BY
    dw.tahun, CASE
        WHEN dw.nama_bulan = 'Januari' THEN 1
        WHEN dw.nama_bulan = 'Februari' THEN 2
        WHEN dw.nama_bulan = 'Maret' THEN 3
        WHEN dw.nama_bulan = 'April' THEN 4
        WHEN dw.nama_bulan = 'Mei' THEN 5
        WHEN dw.nama_bulan = 'Juni' THEN 6
        WHEN dw.nama_bulan = 'Juli' THEN 7
        WHEN dw.nama_bulan = 'Agustus' THEN 8
        WHEN dw.nama_bulan = 'September' THEN 9
        WHEN dw.nama_bulan = 'Oktober' THEN 10
        WHEN dw.nama_bulan = 'November' THEN 11
        WHEN dw.nama_bulan = 'Desember' THEN 12
    END, djp.jenis_perawatan;
```

**6.2 Query untuk Analisis Kinerja Dokter**

```
-- Analisis kinerja dokter berdasarkan spesialisasi
SELECT
    dd.spesialisasi,
    dd.nama_dokter,
    COUNT(DISTINCT fp.id_perawatan) AS jumlah_perawatan,
    SUM(fp.total_biaya) AS total_pendapatan,
    AVG(fp.durasi_perawatan) AS rata_durasi_menit,
    COUNT(DISTINCT fp.fk_pasien) AS jumlah_pasien_unik
FROM
    Fact_Perawatan fp
    JOIN Dim_Dokter dd ON fp.fk_dokter = dd.id_dokter
    JOIN Dim_Waktu dw ON fp.fk_waktu = dw.id_waktu
WHERE
```

```
    dw.tahun = 2025
    AND dw.kuartal = 1
GROUP BY
    dd.spesialisasi, dd.nama_dokter
ORDER BY
    dd.spesialisasi, jumlah_perawatan DESC;
```

### 6.3 Query untuk Analisis Utilisasi Obat

```
-- Analisis utilisasi obat dengan tren bulanan
SELECT
    do.kategori_obat,
    do.nama_obat,
    dw.tahun,
    dw.nama_bulan,
    SUM(fo.jumlah) AS total_penggunaan,
    SUM(fo.total_harga) AS total_biaya,
    COUNT(DISTINCT fo.fk_pasien) AS jumlah_pasien
FROM
    Fact_Obat fo
    JOIN Dim_Obat do ON fo.fk_obat = do.id_obat
    JOIN Dim_Waktu dw ON fo.fk_waktu = dw.id_waktu
WHERE
    dw.tahun = 2025
GROUP BY
    do.kategori_obat, do.nama_obat, dw.tahun, dw.nama_bulan
ORDER BY
    do.kategori_obat, total_penggunaan DESC;
```

### 6.4 Query untuk Dashboard Eksekutif

```
-- Dashboard ringkasan untuk direktur rumah sakit
WITH PendapatanBulanan AS (
    SELECT
        dw.tahun,
        dw.bulan,
        SUM(fp.total_biaya) AS total_pendapatan
    FROM
        Fact_Perawatan fp
        JOIN Dim_Waktu dw ON fp.fk_waktu = dw.id_waktu
    WHERE
        dw.tahun = 2025
    GROUP BY
```

```
        dw.tahun, dw.bulan
),
PasienBulanan AS (
    SELECT
        dw.tahun,
        dw.bulan,
        COUNT(DISTINCT fp.fk_pasien) AS jumlah_pasien,
        COUNT(DISTINCT fp.id_perawatan) AS jumlah_kunjungan
    FROM
        Fact_Perawatan fp
        JOIN Dim_Waktu dw ON fp.fk_waktu = dw.id_waktu
    WHERE
        dw.tahun = 2025
    GROUP BY
        dw.tahun, dw.bulan
),
ObatTerlaris AS (
    SELECT
        dw.tahun,
        dw.bulan,
        do.nama_obat,
        SUM(fo.jumlah) AS jumlah_penggunaan,
        RANK() OVER(PARTITION BY dw.tahun, dw.bulan ORDER BY SUM(fo.jumlah)
DESC) AS rank_obat
    FROM
        Fact_Obat fo
        JOIN Dim_Obat do ON fo.fk_obat = do.id_obat
        JOIN Dim_Waktu dw ON fo.fk_waktu = dw.id_waktu
    WHERE
        dw.tahun = 2025
    GROUP BY
        dw.tahun, dw.bulan, do.nama_obat
)
SELECT
    pb.tahun,
    pb.bulan,
    pb.jumlah_pasien,
    pb.jumlah_kunjungan,
    p.total_pendapatan,
    ot.nama_obat AS obat_terlaris,
    ot.jumlah_penggunaan AS jumlah_penggunaan_obat_terlaris
FROM
    PasienBulanan pb
```

```
    JOIN PendapatanBulanan p ON pb.tahun = p.tahun AND pb.bulan = p.bulan
    JOIN ObatTerlaris ot ON pb.tahun = ot.tahun AND pb.bulan = ot.bulan
WHERE
    ot.rank_obat = 1
ORDER BY
    pb.tahun, pb.bulan;
```

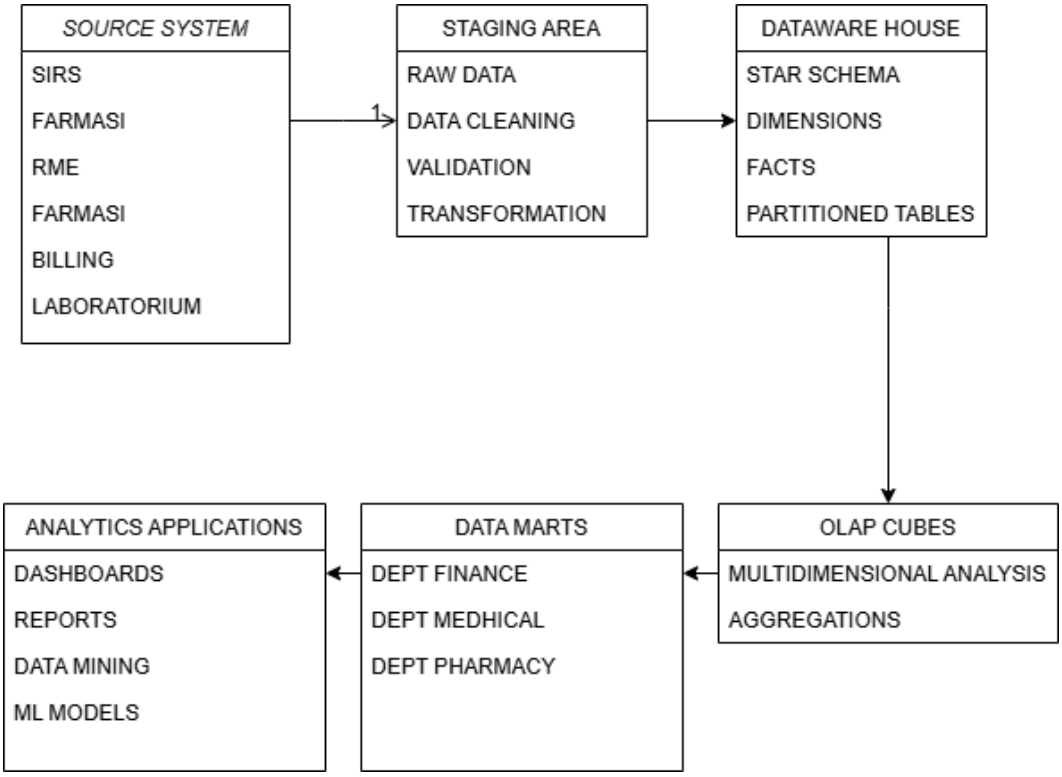
**7. Alur Aliran Data dan Arsitektur**

**7.1 Komponen Arsitektur Data Warehouse**

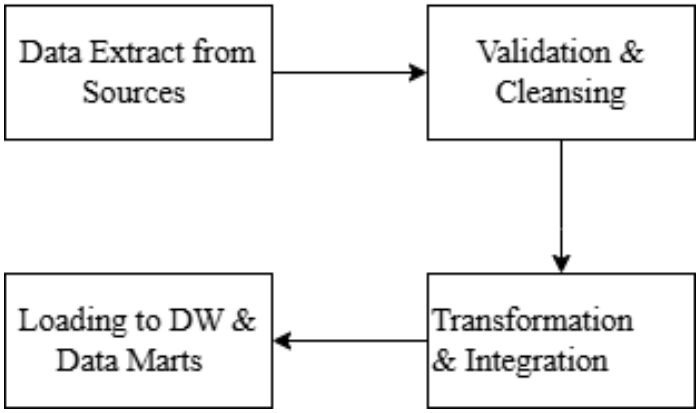
Arsitektur data warehouse untuk industri kesehatan terdiri dari komponen-komponen sebagai berikut:

1. ***Data Sources Layer***
    - Sistem Informasi Rumah Sakit (SIRS)
    - Sistem Rekam Medis Elektronik (RME)
    - Aplikasi Manajemen Persediaan Obat
    - Sistem Billing dan Keuangan
    - Sistem Laboratorium
2. ***Data Integration Layer***
    - ETL Tools: Apache NiFi/Talend
    - Data Quality Checks
    - Error Handling
    - Logging dan Auditing
3. ***Data Storage Layer***
    - Staging Area: Penyimpanan sementara untuk data mentah
    - Data Warehouse: Tabel dimensi dan fakta terstruktur
    - Data Marts: Subset data warehouse untuk kebutuhan departemen tertentu
4. ***Data Access Layer***
    - OLAP Tools
    - Reporting Tools
    - Dashboard untuk visualisasi
    - Data Mining dan Analitik Prediktif
5. ***Metadata Management***
    - Business Metadata: Definisi bisnis, aturan, dan kebijakan
    - Technical Metadata: Struktur tabel, skema, dan ETL flows
    - Operational Metadata: Job logs, audit trails, dan statistik performa

**7.2 Diagram Aliran Data**




### 7.3 ETL Pipeline Workflow




## 8. Implementasi Tools dan Teknologi

### 8.1 Database Management System

Berdasarkan kebutuhan industri kesehatan dengan volume data yang besar dan kebutuhan analitik yang kompleks, kami merekomendasikan penggunaan PostgreSQL sebagai DBMS utama dengan pertimbangan:

1. **Fitur Enterprise-grade**
    - Table partitioning (range, list, hash)

    - Materialized views dengan refresh otomatis
    - Advanced indexing (B-tree, GIN, BRIN)
    - Parallel query execution
2. **Ekstensi yang Mendukung Analitik**
    - TimescaleDB untuk time series data
    - PostGIS untuk analitik spasial (pemetaan penyakit)
    - pg_stat_statements untuk monitoring performa query
3. **Opsi Keamanan untuk Data Medis**
    - Row-level security
    - Column-level encryption
    - Audit logging dan access control

**8.2 ETL Tools**

Untuk proses ETL yang handal dan skalabel, kami menggunakan kombinasi tools:

1. **Apache NiFi**
    - Data ingestion dari berbagai sistem sumber
    - Transformasi data secara real-time
    - Routing dan validasi data
2. **Apache Airflow**
    - Orkestrasi workflow ETL
    - Scheduling dan monitoring job
    - Handling dependencies antar job
3. **dbt (data build tool)**
    - Transformasi data menggunakan SQL
    - Testing dan dokumentasi transformasi
    - Version control untuk transformasi data

**8.3 Tools Visualisasi dan Analitik**

Untuk menyajikan insights dari data warehouse kepada pengguna bisnis:

1. **Apache Superset**
    - Dashboard interaktif
    - Exploratory analysis
    - SQL Lab untuk query ad-hoc
2. **Metabase**
    - Self-service analytics untuk non-technical users
    - Automated reports dan alerts
    - Embedding charts dalam aplikasi internal

**9. Kesimpulan**

Perancangan data warehouse untuk industri kesehatan telah berhasil dilakukan dengan menerapkan:

1. Skema star untuk menyederhanakan query dan analitik
2. Optimasi performa melalui indeks, partisi, dan materialized views
3. ETL pipeline yang terintegrasi untuk memastikan data yang akurat dan terkini
4. Visualisasi dan akses data yang mudah untuk mendukung pengambilan keputusan

Implementasi data warehouse ini memungkinkan rumah sakit dan klinik untuk:

- Menganalisis pola perawatan pasien secara historis
- Mengoptimalkan alokasi sumber daya (dokter, obat, ruangan)
- Memprediksi lonjakan pasien dan kebutuhan pelayanan
- Meningkatkan efisiensi operasional dan kualitas pelayanan