LAPORAN MISI KETIGA PERGUDANGAN DATA
ENERGI & UTILITAS

Disusun oleh :

1. Muhammad Zaki Abdillah 121450008
2. Deva Anjani Khayyuninafsyah 122450014
3. Patricia Leondrea Diajeng Putri 122450050
4. Syadza Puspadari Azhar 122450072
5. Dea Mutia Risani 122450099
6. Amalia Melani Putri 122450122

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2025

1. Design and Implement Dimension Tables

Dalam struktur data warehouse perusahaan energi seperti RubicoNergi, tabel dimensi berperan sebagai penyedia konteks untuk data fakta terukur yang merekam aktivitas bisnis. Perancangan tabel dimensi bertujuan untuk memudahkan pengguna data seperti analis, manajemen, atau sistem pelaporan—dalam mengeksplorasi dan menganalisis informasi dari berbagai perspektif bisnis, misalnya periode waktu, profil pelanggan, geografi lokasi, atau parameter operasional. Dengan demikian, proses pengambilan keputusan dan analisis kinerja bisnis dapat dilakukan secara lebih efisien dan terarah. Selain tujuan tersebut, pembangunan data warehouse juga dirancang agar setiap dimensi dilengkapi atribut deskriptif yang menjawab pertanyaan mendasar seperti:

a. Siapa yang terlibat (misalnya, segmentasi pelanggan),
b. Kapan suatu aktivitas terjadi (periode/waktu),
c. Di mana lokasi kejadian (area distribusi),
d. Kondisi yang melatarbelakangi suatu peristiwa (misalnya, cuaca saat konsumsi energi tercatat).

Berikut penjelasan untuk setiap dimensi yang ada :

1.1 Dimensi Waktu (WaktuDIM)

a. Primary Key: Time_ID
b. Atribut:
    1) Full_DateTime (Timestamp lengkap)
    2) Date (Tanggal yyyy-mm-dd)
    3) Time (Jam dan menit)
    4) Year
    5) Quarter
    6) Month
    7) Day
    8) Hour
    9) Minute
    10) Weekday_Name (Senin – Minggu)
    11) Is_Weekend (Ya/Tidak)
c. Hierarki: Minute → Hour → Day → Month → Quarter → Year

Berfungsi untuk analisis tren penggunaan energi harian, musiman, mingguan.

1.2 Dimensi Pelanggan (PelangganDIM)

a. Primary Key: Household_ID
b. Atribut:
    1) Customer_Name
    2) Customer_Type (Rumah Tangga / Komersial)
    3) Meter_Type (Smart / Manual)
    4) Region_Type (3S / Non-3S)
    5) City_District
    6) Province
c. Hierarki: Pelanggan → Kota/Kabupaten → Provinsi → Region_Type

Berfungsi untuk segmentasi pelanggan berdasarkan lokasi dan kategori.

1.3 **Dimensi Geospasial (GeoDIM)**

a. Primary Key: Geo_ID
b. Atribut:
    1) Latitude, Longitude
    2) Village (Desa)
    3) Subdistrict (Kecamatan)
    4) District (Kabupaten/Kota)
    5) Province
    6) Urban_Rural (Perkotaan / Pedesaan)
    7) Region_Type (3S / Non-3S)
c. Hierarki: Koordinat → Desa → Kecamatan → Kabupaten/Kota → Provinsi

Berfungsi untuk analisis spasial dan visualisasi peta distribusi konsumsi energi.

1.4 Dimensi Perangkat (PerangkatDIM)

a. Primary Key: Segment_ID
b. Atribut:
    1) Segment_Name (Sub-metering_1, _2, _3)
    2) Device_Category (HVAC, Laundry, Kitchen)
    3) Typical_Location (Dapur, Garasi, Ruang Tamu)
c. Hierarki: Submetering → Kategori Perangkat → Lokasi

Berfungsi untuk menganalisis konsumsi berdasarkan fungsi ruang dalam rumah.

1.5 Dimensi Gangguan (GangguanDIM)

a. Primary Key: Gangguan_ID
b. Atribut:
    1) Gangguan_Type (Pemadaman, Fluktuasi Tegangan)
    2) Cause_Code (Bencana Alam, Pemeliharaan)
    3) Resolution_Method (Otomatis, Manual)
c. Hierarki: Jenis Gangguan → Penyebab → Metode Penanganan

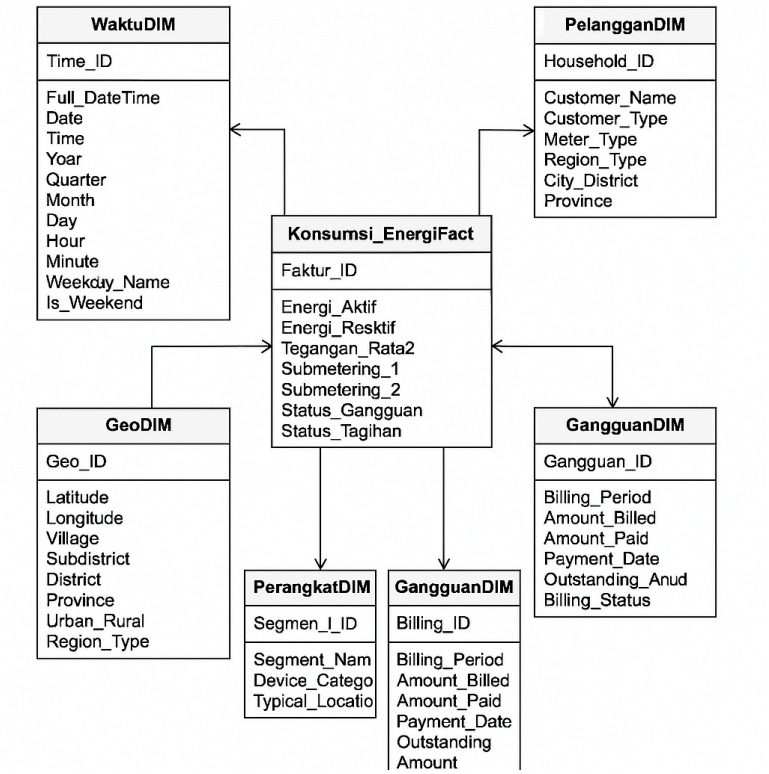
Berfungsi untuk menganalisis keandalan jaringan dan pola gangguan.

1.6 Dimensi Tagihan (TagihanDIM)

a. Primary Key: Billing_ID
b. Atribut:
    1) Billing_Period (Bulan-Tahun)
    2) Amount_Billed
    3) Amount_Paid
    4) Payment_Date
    5) Payment_Method (Tunai, Transfer, Kartu)
    6) Outstanding_Amount
    7) Billing_Status (Lunas, Tunggak, Belum Bayar)
c. Hierarki: Periode Tagihan → Status Pembayaran

Berfungsi untuk Menganalisis perilaku pembayaran dan proyeksi arus kas.

Desain Tabel Dimensi :



Gambar 1. Tabel Dimensi

2. Design and Implement Fact Tables

2.1 Identifikasi Entitas dan Pembentukan Atribut Tabel Fakta

Tabel fakta utama yang dibangun dalam sistem pergudangan data RubicoNergi adalah Konsumsi_EnergiFact. Tabel ini menyimpan data hasil transformasi dari berbagai sumber dan menjadi pusat integrasi analitik lintas domain: konsumsi energi, status gangguan, dan informasi tagihan pelanggan. Terdapat 7 entitas yang ada pada tabel fakta yaitu :

Energi_aktif : Total energi aktif (kWh) per waktu rekam
Energi_reaktif : Total energi reaktif (kWh)
Tegangan_rata2 : Tegangan rata-rata (Volt) per periode waktu
Sub_metering_1 : Energi dari submetering 1 (dapur)
Sub_metering_2 : Energi dari submetering 2 (laundry)
Status_gangguan : Status gangguan saat itu
Status_tagihan : Status pembayaran

Berikut atribut yang terdapat pada tabel fakta yang dibuat dalam bentuk tabel:

Tabel 1. Atribut Tabel

Kolom	Tipe Data	Keterangan
id_waktu	INT	Foreign Key ke dimensi waktu
id_pelanggan	INT	Foreign Key ke dimensi pelanggan

id_geo	INT	Foreign Key ke dimensi geospasial
id_perangkat	INT	Foreign Key ke dimensi perangkat/submetering
id_gangguan	INT	Foreign Key ke dimensi gangguan
id_tagihan	INT	Foreign Key ke dimensi tagihan
energi_aktif	FLOAT	Total energi aktif (kWh) per waktu rekam
energi_reaktif	FLOAT	Energi reaktif total dalam satuan kWh
tegangan_rata2	FLOAT	Tegangan rata-rata harian dalam volt
sub_metering_1	FLOAT	Energi dari submetering 1 (dapur)
sub_metering_1	FLOAT	Energi dari submetering 2 (laundry)
status_gangguan	VARCHAR	Status gangguan saat itu
status_tagihan	VARCHAR	Status pembayaran seperti “Lunas”, “Tunggakan”, “Belum Bayar”

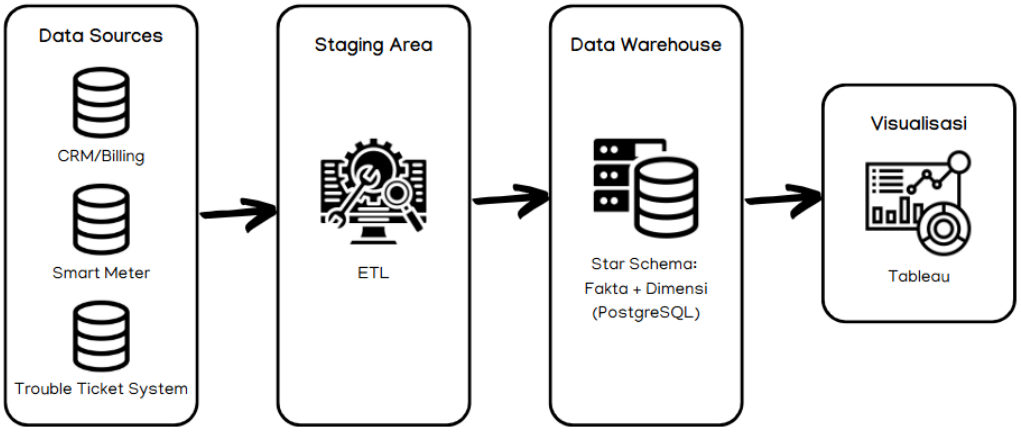
2.2 Penentuan Alur, Arsitektur dan Teknologi yang Digunakan

Alur pembangunan gudang data RubicoNergi (RBN) dimulai dari integrasi data mentah yang berasal dari berbagai sistem internal dan eksternal. Sumber utama berasal dari dataset Household Electric Power Consumption yang tersedia di Kaggle dalam format CSV (.txt). Selain itu, data tambahan seperti informasi pelanggan, data spasial wilayah (termasuk status 3S), laporan gangguan, status tagihan, serta data perangkat sub-metering diambil dari sistem internal RBN seperti CRM, GIS, sistem billing, dan trouble ticket system. Data tersebut kemudian diekstrak menjadi bentuk DataFrame dan disimpan sementara ke dalam bentuk .csv untuk selanjutnya diproses dalam tahap ETL. Setelah proses ETL selesai, data disimpan terlebih dahulu ke Staging Area sebagai tempat penampungan sementara, sebelum akhirnya dimuat ke dalam sistem data warehouse.

Dari sisi arsitektur, sistem dirancang dengan pendekatan Star Schema, di mana tabel fakta KonsumsiEnergiFact menjadi pusat yang terhubung ke enam tabel dimensi utama, yaitu WaktuDIM, PelangganDIM, GeoDIM, PerangkatDIM, GangguanDIM, dan TagihanDIM. Struktur ini mendukung analisis multidimensi dan memungkinkan eksplorasi data berdasarkan hierarki waktu, wilayah geografis, jenis perangkat, dan status pelanggan.

Untuk implementasi basis data, digunakan sistem manajemen basis data relasional PostgreSQL sebagai platform data warehouse. PostgreSQL dipilih karena sifatnya open-source, stabil, dan mendukung struktur relasional dengan skala besar. Di

tahap akhir, proses visualisasi dan pelaporan dilakukan menggunakan Tableau Public, yang berfungsi untuk menyajikan data dalam bentuk dashboard interaktif serta mendukung analisis tren konsumsi, deteksi wilayah rawan gangguan, dan pemantauan kepatuhan pembayaran pelanggan. Dengan integrasi arsitektur ini, RBN dapat mengambil keputusan berbasis data secara cepat, efisien, dan akurat.



Gambar 2. Arsitektur Data Warehouse

2.3 Proses ETL dalam Mendapatkan Data Untuk Tabel Fakta

Proses ETL (Extract, Transform, Load) untuk mendapatkan data bagi tabel fakta Konsumsi_EnergiFact pada sistem pergudangan data RubicoNergi melibatkan beberapa tahapan terstruktur. Berikut adalah detail proses ETL yang dilakukan:

1) Tahapan Ekstraksi (Extract)

   Proses dalam ekstraksi, karena pada dataset dari kaggle awalnya berformat database SQLite maka dikoneksikan terlebih dahulu menggunakan pustaka SQLite3 pada bahasa pemrograman Python. Berikut kode untuk proses Ekstraksi:

```
import sqlite3
conn = sqlite3.connect('/content/database.sqlite')
cursor = conn.cursor()
cursor.execute("PRAGMA table_info(match)")
columns = cursor.fetchall()
# Hanya ambil nama kolom
column_names = [col[1] for col in columns]
print("Columns:", column_names)
cursor.execute("SELECT * FROM match")
rows = cursor.fetchall()
for row in rows:
    print(row)
conn.close()
```

2) Tahapan Transformasi (Transform)

Transformasi digunakan untuk menyempurnakan data yang nantinya akan dimasukkan ke dalam staging area yang terdiri dari membersihkan data yang kosong dan validasi format yang salah (standarisasi format tanggal menjadi integer), lalu ada enhancement yakni menambah kolom hasil pertandingan dan pisahkan kolom date menjadi year, month dan day, lalu lakukan normalisasi untuk pembuatan dimensi yang dibutuhkan.

```
import pandas as pd
import numpy as np
from datetime import datetime

# Membaca data dari SQLite ke DataFrame
df = pd.read_sql_query("SELECT * FROM
household_power_consumption", conn)
conn.close()

# Membersihkan data yang kosong
df.replace('?', np.nan, inplace=True)
df['Global_active_power'] = pd.to_numeric(df['Global_active_power'],
errors='coerce')
df['Global_reactive_power'] = pd.to_numeric(df['Global_reactive_power'],
errors='coerce')
df['Voltage'] = pd.to_numeric(df['Voltage'], errors='coerce')
df['Global_intensity'] = pd.to_numeric(df['Global_intensity'],
errors='coerce')
df['Sub_metering_1'] = pd.to_numeric(df['Sub_metering_1'],
errors='coerce')
df['Sub_metering_2'] = pd.to_numeric(df['Sub_metering_2'],
errors='coerce')

# Mengisi nilai yang kosong dengan interpolasi linear
df['Global_active_power'].interpolate(method='linear', inplace=True)
df['Global_reactive_power'].interpolate(method='linear', inplace=True)
df['Voltage'].interpolate(method='linear', inplace=True)
df['Global_intensity'].interpolate(method='linear', inplace=True)
df['Sub_metering_1'].fillna(0, inplace=True)
df['Sub_metering_2'].fillna(0, inplace=True)

# Standarisasi format tanggal dan waktu
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y')
df['Time'] = pd.to_datetime(df['Time'], format='%H:%M:%S').dt.time
df['DateTime'] = pd.to_datetime(df['Date'].astype(str) + ' ' +
df['Time'].astype(str))

# Enhancement - Pisahkan kolom date menjadi year, month, day
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
```

```
df['Hour'] = pd.to_datetime(df['Time'], format='%H:%M:%S').dt.hour

# Menambah kolom hasil perhitungan
df['energi_aktif'] = df['Global_active_power'] * 1000 / 60  # konversi ke
kWh per menit
df['energi_reaktif'] = df['Global_reactive_power'] * 1000 / 60  # konversi ke
kVARh per menit
df['tegangan_rata2'] = df['Voltage']

# Menentukan status gangguan berdasarkan tegangan
conditions = [
    (df['Voltage'] < 220 * 0.9),
    (df['Voltage'] > 220 * 1.1)
]
choices = ['Under Voltage', 'Over Voltage']
df['status_gangguan'] = np.select(conditions, choices, default='Normal')

# Simulasi status tagihan - dalam implementasi nyata akan berasal dari
sistem billing
status_options = ['Lunas', 'Tunggakan', 'Belum Bayar']
np.random.seed(42)  # untuk reproduktifitas
df['status_tagihan'] = np.random.choice(status_options, size=len(df))

# Normalisasi untuk tabel dimensi
# Membuat kunci pengganti (surrogate key) untuk dimensi waktu
df['id_waktu'] = df['Year'].astype(str) + df['Month'].astype(str).str.zfill(2) +
df['Day'].astype(str).str.zfill(2)
df['id_waktu'] = pd.to_numeric(df['id_waktu'])

# Simpan hasil transformasi ke staging area
df.to_csv('/content/staging_energy_data.csv', index=False)

# Buat dataframe terpisah untuk tabel dimensi
waktu_dim = df[['id_waktu', 'Year', 'Month', 'Day']].drop_duplicates()
```

3) Load

Load bertujuan untuk memindahkan data dalam staging area ke data warehouse untuk dianalisis dan diolah. Berikut kode untuk load khusus tabel fakta:

```
import psycopg2
from sqlalchemy import create_engine

# Koneksi ke PostgreSQL (data warehouse)
engine =
create_engine('postgresql://username:password@localhost:5432/rubico_dw'
)

# Load tabel dimensi terlebih dahulu
```

```
# 1. Waktu Dimensi
waktu_dim = df[['id_waktu', 'Year', 'Month', 'Day']].drop_duplicates()
waktu_dim.columns = ['id_waktu', 'tahun', 'bulan', 'hari']
waktu_dim.to_sql('WaktuDIM', engine, if_exists='append', index=False)

# 2. Dimensi Pelanggan (data dummy untuk contoh)
pelanggan_data = pd.DataFrame({
    'id_pelanggan': range(1, 11),  # 10 pelanggan sampel
    'nama': ['Customer ' + str(i) for i in range(1, 11)],
    'alamat': ['Address ' + str(i) for i in range(1, 11)],
    'segment': np.random.choice(['Residential', 'Commercial', 'Industrial'],
size=10)
})
pelanggan_data.to_sql('PelangganDIM', engine, if_exists='append',
index=False)

# 3. Dimensi Geospasial (data dummy untuk contoh)
geo_data = pd.DataFrame({
    'id_geo': range(1, 6),  # 5 lokasi sampel
    'region': ['North', 'South', 'East', 'West', 'Central'],
    'city': ['City A', 'City B', 'City C', 'City D', 'City E'],
    'status_3s': np.random.choice(['Normal', 'Sulit', 'Sangat Sulit', 'Super
Sulit'], size=5)
})
geo_data.to_sql('GeoDIM', engine, if_exists='append', index=False)

# 4. Dimensi Perangkat
perangkat_data = pd.DataFrame({
    'id_perangkat': [1, 2, 3],
    'tipe': ['Main', 'Kitchen', 'Laundry'],
    'model': ['Main Meter', 'Sub Meter 1', 'Sub Meter 2'],
    'tahun_instalasi': [2006, 2006, 2006]
})
perangkat_data.to_sql('PerangkatDIM', engine, if_exists='append',
index=False)

# 5. Dimensi Gangguan
gangguan_data = pd.DataFrame({
    'id_gangguan': [1, 2, 3],
    'status_gangguan': ['Normal', 'Under Voltage', 'Over Voltage'],
    'deskripsi': ['Tegangan normal', 'Tegangan di bawah ambang normal',
'Tegangan di atas ambang normal']
})
gangguan_data.to_sql('GangguanDIM', engine, if_exists='append',
index=False)

# 6. Dimensi Tagihan
tagihan_data = pd.DataFrame({
    'id_tagihan': [1, 2, 3],
```

```
    'status_tagihan': ['Lunas', 'Tunggakan', 'Belum Bayar'],
    'deskripsi': ['Pembayaran sudah lunas', 'Terdapat tunggakan pembayaran',
'Belum melakukan pembayaran']
})
tagihan_data.to_sql('TagihanDIM', engine, if_exists='append', index=False)

# Siapkan data untuk tabel fakta
# Simulasikan distribusi data ke pelanggan, lokasi, dan perangkat
np.random.seed(42)
df['id_pelanggan'] = np.random.choice(pelanggan_data['id_pelanggan'],
size=len(df))
df['id_geo'] = np.random.choice(geo_data['id_geo'], size=len(df))

# Buat mapping untuk id perangkat berdasarkan submetering
conditions = [
    (df['Sub_metering_1'] > 0) & (df['Sub_metering_2'] == 0),
    (df['Sub_metering_1'] == 0) & (df['Sub_metering_2'] > 0),
    (df['Sub_metering_1'] > 0) & (df['Sub_metering_2'] > 0)
]
choices = [2, 3, 2]  # id_perangkat 2 untuk dapur, 3 untuk laundry, default 2
jika keduanya aktif
df['id_perangkat'] = np.select(conditions, choices, default=1)

# Mapping status gangguan dan tagihan ke id dimensi
gangguan_mapping = dict(zip(gangguan_data['status_gangguan'],
gangguan_data['id_gangguan']))
df['id_gangguan'] = df['status_gangguan'].map(gangguan_mapping)

tagihan_mapping = dict(zip(tagihan_data['status_tagihan'],
tagihan_data['id_tagihan']))
df['id_tagihan'] = df['status_tagihan'].map(tagihan_mapping)

# Siapkan tabel fakta
fact_columns = [
    'id_waktu', 'id_pelanggan', 'id_geo', 'id_perangkat', 'id_gangguan',
'id_tagihan',
    'energi_aktif', 'energi_reaktif', 'tegangan_rata2', 'Sub_metering_1',
'Sub_metering_2',
    'status_gangguan', 'status_tagihan'
]
fact_table = df[fact_columns]

# Load tabel fakta ke data warehouse
fact_table.to_sql('Konsumsi_EnergiFact', engine, if_exists='append',
index=False)

print("Proses ETL selesai dengan sukses. Data berhasil dimuat ke dalam
data warehouse.")
```

3. Design and Implement Indexes

Dalam perencanaan Data Warehouse untuk perusahaan RubicoNergi (RBN), penerapan indeks akan sangat berguna dalam meningkatkan efisiensi kueri, terutama dalam menangani data energi yang bervolume besar dan kompleksitas hubungan antar entitas.

3.1 Jenis-Jenis Indeks yang Digunakan

a. Indeks Clustered

Menerapkan indeks pada kolom 'Time_ID' di tabel fakta 'Konsumsi_EnergiFact'. Indeks ini dipilih karena waktu pencatatan merupakan dimensi sentral dalam analisis tren konsumsi. Dengan indeks ini, proses pengurutan dan pencarian data berdasarkan waktu dapat dilakukan dengan cepat.

```
CREATE CLUSTERED INDEX idx_time_id
ON Konsumsi_EnergiFact (Time_ID);
```

b. Indeks Non-Clustered

Index Non-Clustered diterapkan pada kolom 'Household_ID', 'Geo_ID', dan 'Billing_ID'. Kolom-kolom ini digunakan dalam proses filter, 'join' tabel, dan agregasi yang membuat pengindeksan terhadap kolom-kolom ini cepat terhadap analitik berbasis pelanggan, wilayah, dan tagihan.

```
CREATE NONCLUSTERED INDEX idx_household_id
ON Konsumsi_EnergiFact (Household_ID);

CREATE NONCLUSTERED INDEX idx_geo_id
ON Konsumsi_EnergiFact (Geo_ID);

CREATE NONCLUSTERED INDEX idx_billing_id
ON Konsumsi_EnergiFact (Billing_ID);
```

3.2 Strategi Pemilihan Indeks

Pemilihan indeks berdasar kepada frekuensi akses query dan analisis kebutuhan bisnis. Kolom yang paling sering digunakan dalam kondisi 'WHERE', 'JOIN', dan agregasi seperti pencatatan waktu, pelanggan, wilayah, dan status pembayaran menjadi prioritas utama saat pengindeksan. Hal ini dilakukan agar dapat meminimalisir waktu pencarian dan meningkatkan performa laporan periodik ataupun spasial.

3.3 Pengujian Kinerja

Setelah indeks diimplementasikan, dilakukan uji menggunakan perintah 'EXPLAIN' untuk membandingkan waktu eksekusi kueri sebelum dan sesudah indeks diterapkan. Pengujian ini dilakukan untuk memastikan keberadaan indeks benar-benar berkontribusi dalam mempercepat proses pencarian.

3.4 Pemeliharaan Indeks

Pemeliharaan indeks dilakukan agar tidak mengalami degradasi performa akibat fragmentasi. Pemeliharaan dilakukan secara berkala dengan fungsi 'REBUILD' atau 'REORGANIZE', khususnya setelah proses ETL atau pembaruan data.

```
ALTER INDEX idx_time_id
ON Konsumsi_EnergiFact REBUILD;
```

4. Design Storage

RubicoNergi (RBN) memerlukan data warehouse untuk mengintegrasikan data konsumsi listrik per menit, data spasial, gangguan, pelanggan, dan tagihan. Mengingat karakteristik data yang besar dan kompleks, diperlukan desain storage yang mampu mendukung performa kueri analitis, efisiensi storage, serta skalabilitas.

4.1. Hardware Considerations

Tabel 2. Hardware

Komponen	Jenis Hardware	Kegunaan
Disk	SSD dengan RAID 10	Untuk kecepatan I/O dan redundancy.
CPU	Minimal 16 core	Mendukung parallel query dan data processing.
Memory	128 GB RAM	Agar dapat memproses query besar secara in-memori.

Karena data konsumsi listrik per menit selama 4 tahun mengakibatkan volume data besar, heavy read query, memerlukan high I/O & processing power. Implementasi hardware ini mendukung beban kerja analitik dari konsumsi listrik per menit, yang menghasilkan data dalam jumlah besar.

4.2. Database File Layout

Tabel 3. Database File Layout

Database	File Layout
Data Files (MDF)	SSD 1 (dedicated).
Transaction Logs (LDF)	SSD 2 (terpisah).
TempDB	SSD 3, dioptimalkan dengan multiple data files.

Contoh implementasi konfigurasi TempDB:

```
CREATE TABLE Konsumsi_EnergiFact (
  EnergiFact_ID BIGINT IDENTITY(1,1),
  Time_ID INT,
  Household_ID INT,
  Geo_ID INT,
  Energi_Aktif FLOAT,
  Submeter1 FLOAT,
```

```
  Submeter2 FLOAT,
  Submeter3 FLOAT
)
WITH (CLUSTERED COLUMNSTORE INDEX);
```

Memisahkan file untuk mencegah I/O contention antara data, log, dan tempdb.

4.3. Storage Models

Tabel 4. Model Penyimpanan

Table	Storage Models
Konsumsi_EnergiFact	Clustered Columnstore Index
WaktuDIM PelangganDIM GeoDIM PerangkatDIM GangguanDIM TagihanDIM	Rowstore Clustered Index

Contoh implementasi tabel fakta dengan Clustered Columnstore Index:

```
CREATE TABLE Konsumsi_EnergiFact
(
  EnergiFact_ID BIGINT IDENTITY(1,1) PRIMARY KEY,
  Time_ID INT,
  Household_ID INT,
  Geo_ID INT,
  Energi_Aktif FLOAT,
  Energi_Reaktif FLOAT,
  Tegangan_Rata2 FLOAT,
  Submeter1 FLOAT,
  Submeter2 FLOAT,
  Submeter3 FLOAT
)
WITH (CLUSTERED COLUMNSTORE INDEX, DATA_COMPRESSION = PAGE);
```

Columnstore sangat efisien untuk agregasi & analisis data besar (OLAP), sedangkan dimensi lookup kecil tetap efisien dengan Rowstore.

4.4. Data Compression

Tabel 5. Kompresi Data

Jenis	Tabel
Page Compression	PelangganDIM, GeoDIM, TagihanDIM
Row Compression	Tabel kecil, seperti PerangkatDIM, GangguanDIM
Columnstore Compression	Otomatis di Konsumsi_EnergiFact

Contoh implementasi untuk mengaktifkan Page Compression yang mana dapat mengurangi kebutuhan storage & meningkatkan cache efficiency tanpa mengorbankan performa query:

```
ALTER TABLE PelangganDIM REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = PAGE);
```

4.5. Partitioning Strategy

Tabel 6. Strategi Partisi

Tabel	Strategi
Konsumsi_EnergiFact	Dipartisi berdasarkan Year & Month.
TagihanDIM	Dipartisi per Billing_Period.

Partisi membantu manajemen data historis & percepat query ke data terbaru (sliding window). Contoh implementasi pembuatan Partition Function & Scheme:

```
CREATE PARTITION FUNCTION pf_YearMonth (INT)
AS RANGE LEFT FOR VALUES (2022, 2023, 2024, 2025);
CREATE PARTITION SCHEME ps_YearMonth
AS PARTITION pf_YearMonth ALL TO ([PRIMARY]);
CREATE TABLE Konsumsi_EnergiFact
(
  EnergiFact_ID BIGINT IDENTITY(1,1),
  Year INT,
  Month INT,
  Energi_Aktif FLOAT,
  -- kolom lainnya
)
ON ps_YearMonth(Year);
```

4.6. Data Distribution

Bila diimplementasikan di cloud (Azure Synapse), data fakta akan didistribusikan secara hash atau round robin ke compute nodes. Distribusi memastikan workload merata antar compute node → scaling performance. Contoh implementasi sinapsis:

```
CREATE TABLE Konsumsi_EnergiFact
WITH (DISTRIBUTION = HASH(Household_ID), CLUSTERED COLUMNSTORE
INDEX)
AS SELECT * FROM Staging_Konsumsi;
```

4.7. Maintenance & Performance

Tabel 7. Maintaining

Jenis	Alasan
Index Rebuild/Reorganize	a. Dapat menyelesaikan masalah Fragmentasi Index yang

	membuat query lambat dan I/O boros. b. Tabel fakta (ratusan juta baris) → fragmentasi >30% → Rebuild lebih efektif. c. Rebuild akan membangun ulang index dari nol → urutan data jadi optimal lagi. d. Bisa dilakukan ONLINE (non-disruptive) di edisi Enterprise.
Update Statistics	a. Dapat menyelesaikan masalah statistik kolom out-of-date yang membuat query plan buruk. b. Agar Query Optimizer punya informasi paling akurat tentang distribusi data terkini.

Contoh implementasi rebuild index dan update stats:

```
ALTER INDEX ALL ON Konsumsi_EnergiFact REBUILD WITH (ONLINE = ON);
UPDATE STATISTICS Konsumsi_EnergiFact;
```

4.8. Monitoring & Optimization

Untuk pemantauan dan optimisasi, digunakan Dynamic Management Views (DMV) yang merupakan fitur bawaan SQL Server untuk memantau kesehatan & performa database secara real-time. DMV menyediakan data tentang aktivitas query (paling lambat, paling sering dipakai, paling boros I/O); index usage & fragmentasi; I/O statistics per database & per query; serta CPU usage, waits, bottlenecks. DMV dipilih karena RBN mengelola data konsumsi listrik per menit, query analitik akan berat & kompleks untuk mengetahui bagian mana yang jadi bottleneck: apakah di disk, index, join query, atau concurrency TempDB. Contoh implementasi untuk mengecek I/O intensive queries:

```
SELECT TOP 10
  qs.execution_count,
  qs.total_logical_reads,
  qs.total_worker_time,
  qs.creation_time,
  SUBSTRING(qt.text, (qs.statement_start_offset/2)+1,
  ((CASE qs.statement_end_offset
     WHEN -1 THEN DATALENGTH(qt.text)
     ELSE qs.statement_end_offset END
  - qs.statement_start_offset)/2)+1) AS query_text
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
ORDER BY qs.total_logical_reads DESC;
```

5. Design and Implement Partitioned Tables and Views

Dalam ekosistem gudang data modern seperti pada implementasi RubicoNergi: Empowering Energy Through Data, penggunaan partisi tabel dan view yang terindeks menjadi elemen krusial dalam meningkatkan kinerja sistem, skalabilitas, serta efisiensi pengelolaan data historis. Pendekatan ini dirancang untuk menangani pertumbuhan volume data konsumsi energi yang terus meningkat seiring waktu—khususnya data granular seperti penggunaan energi per menit—serta mempercepat proses kueri analitik yang berbasis waktu, wilayah geografis, maupun segmentasi pelanggan tertentu.

5.1. Pemilihan Struktur Partisi

a. Partisi Horizontal (Row Partitioning)

- Objek : Tabel fakta Konsumsi_EnergiFact.
- Kunci Partisi : Kolom YearMonth (format YYYYMM), berfungsi untuk menggabungkan tahun dan bulan.
- Manfaat:
  - Partition Pruning, digunakan untuk partisi relevan yang dipindai, mempercepat kueri analitik periodik.
  - Archiving & Purging, digunakan untuk data lama yang dapat dipindah atau dihapus partisinya tanpa memengaruhi data terkini.
  - Isolasi Indeks, merupakan *rebuild* atau *reorganize* indeks hanya pada partisi aktif, menghemat waktu pemeliharaan.

```
CREATE PARTITION FUNCTION pf_YearMonth (INT)
AS RANGE LEFT FOR VALUES (2022, 2023, 2024, 2025);

CREATE PARTITION SCHEME ps_YearMonth
AS PARTITION pf_YearMonth ALL TO (
    [Staging_FG], [DW_Fakta], [DW_Fakta], [DW_Fakta],
   [DW_Fakta], [DW_Fakta], [DW_Fakta], [DW_Fakta]
);

CREATE TABLE Konsumsi_EnergiFact
(
  EnergiFact_ID BIGINT IDENTITY(1,1) PRIMARY KEY,
  Time_ID INT,
  Household_ID INT,
  Geo_ID INT,
  Energi_Aktif FLOAT,
  Energi_Reaktif FLOAT,
  Tegangan_Rata2 FLOAT,
  Submeter1 FLOAT,
  Submeter2 FLOAT,
  Submeter3 FLOAT
)
ON ps_YearMonth(Year);
```

b. Strategi Pemeliharaan Partisi

- SPLIT, digunakan untuk menambahkan partisi baru setiap awal periode (bulan/tahun) secara otomatis.

- MERGE, digunakan untuk menggabungkan partisi lama (mis. data > 2 tahun) untuk menyederhanakan manajemen.
- SWITCH, digunakan untuk memindahkan partisi antar filegroup (staging ↔ production) untuk fast load dan data archiving tanpa downtime.

c. Automasi dan Orkestrasi

- Stored Procedure usp_ManagePartitions memeriksa dan menambah partisi baru setiap awal bulan.
- SQL Agent Job menjadwalkan eksekusi prosedur tersebut sehingga partisi selalu terkini tanpa intervensi manual.

5.2. Perancangan Indexed Views

a. Tujuan : Mempercepat agregasi kompleks—misalnya rekap total energi per provinsi/tahun atau jumlah gangguan per wilayah.

b. Zone : Gold Zone (Semantic Layer) untuk laporan dan dashboard.

```
CREATE VIEW vw_Energi_Prov_Tahun
    WITH SCHEMABINDING AS
SELECT
    g.Province,
    f. YearMonth/100 AS Year,
    SUM(f.Energi_Aktif) AS Total_Energi
FROM dbo.Konsumsi_EnergiFact fs
JOIN dbo.GeoDIM g
ON f.Geo_ID = g.Geo_ID
GROUP BY g.Province, f.YearMonth / 100;

CREATE VIEW CLUSTERED INDEX IDX_vwEnergiProvTh
   ON vw_Energi_Prov_Tahun (Province, Year);
```

5.3. Strategi Penyimpanan Data

Layer	Filegroup / File Layour	Partisi & Kompresi
Bronze	Staging_FG (SSD)	Data mentah, tanpa partisi, kompresi minimal
Silver	DW_Fakta & PRIMARY (SSD terpisah)	Tabel fakta/dimensi sudah dipartisi, kompresi PAGE/columnstore
Gold	PRIMARY	Indexed views dioptimasi untuk semantik, materialized & partisi

5.4. Optimasi dan Monitoring

a. Partisi Horizontal (Row Partitioning)

Menggunakan ‘SET STATISTICS IO ON’ untuk memastikan hanya partisi yang relevan diakses.

b. Rebuild & Update Statistik

```
ALTER INDEX ALL ON Konsumsi_EnergiFact REBUILD PARTITION = ALL;
UPDATE STATISTICS vw_Energi_Prov_Tahun WITH FULLSCAN;
```

c. Monitoring

- DMV ‘sys.dm_db_partition_stats’ dan ‘sys.indexes’ untuk memantau fragmentasi dan ukuran partisi.
- Extended Events atau Policy-Based Management untuk notifikasi fragmentasi tinggi, kegagalan job, atau lonjakan growth file data.