

# TUGAS BIG DATA ARSITEKTUR DAN INFRASTRUKTUR

## Pertemuan 3

### *Data Source*

---

Nama	: Lukas Febrian Laufra
Kelas/Nim	: TI22J/20220040076
Dosen Pengajar	: Bapak Ir. Somantri, S.T, M.Kom
Pengerjaan Matkul	: Minggu, 13 Oktober 2024.

---

### Soal

1. Buatlah makalah mengenai perbedaan Data Struktur, Data Tidak Terstruktur dan Semi Terstruktur beserta contoh dan implementasinya, kemudian jelaskan juga mengenai Batch Processing dan Stream Processing, beserta contoh dan implementasinya.

### Jawaban

#### **Struktur Data dan Metode Pemrosesan: Analisis Komparatif dan Implementasi**

##### Abstrak

Makalah ini mengeksplorasi perbedaan antara data terstruktur, tidak terstruktur, dan semi-terstruktur, serta membandingkan metode batch processing dan stream processing. Melalui analisis mendalam dan contoh implementasi, kami bertujuan untuk memberikan pemahaman komprehensif tentang konsep-konsep ini dan relevansinya dalam pengelolaan dan analisis data modern.

##### 1. Pendahuluan

Dalam era big data, pemahaman tentang berbagai jenis struktur data dan metode pemrosesannya menjadi semakin penting. Makalah ini akan membahas tiga kategori utama struktur data: terstruktur, tidak terstruktur, dan semi-terstruktur, serta dua pendekatan utama dalam pemrosesan data: batch processing dan stream processing.

##### 2. Struktur Data

## 2.1 Data Terstruktur

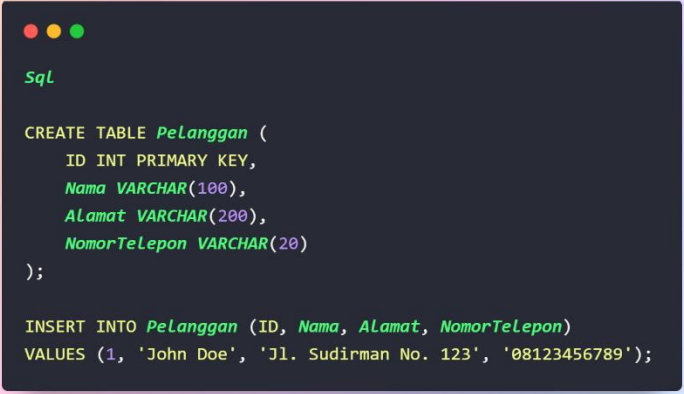
Data terstruktur mengacu pada informasi yang telah diorganisir ke dalam format yang telah ditentukan sebelumnya, biasanya dalam bentuk tabel dengan baris dan kolom yang jelas. Karakteristik utama data terstruktur meliputi:

- Format yang konsisten dan dapat diprediksi
- Mudah dicari dan dianalisis menggunakan metode query tradisional
- Sering disimpan dalam database relasional

Contoh:

Database pelanggan dengan kolom seperti ID, Nama, Alamat, dan Nomor Telepon.

Implementasi:



```
SQL

CREATE TABLE Pelanggan (
  ID INT PRIMARY KEY,
  Nama VARCHAR(100),
  Alamat VARCHAR(200),
  NomorTelepon VARCHAR(20)
);

INSERT INTO Pelanggan (ID, Nama, Alamat, NomorTelepon)
VALUES (1, 'John Doe', 'Jl. Sudirman No. 123', '08123456789');
```

## 2.2 Data Tidak Terstruktur

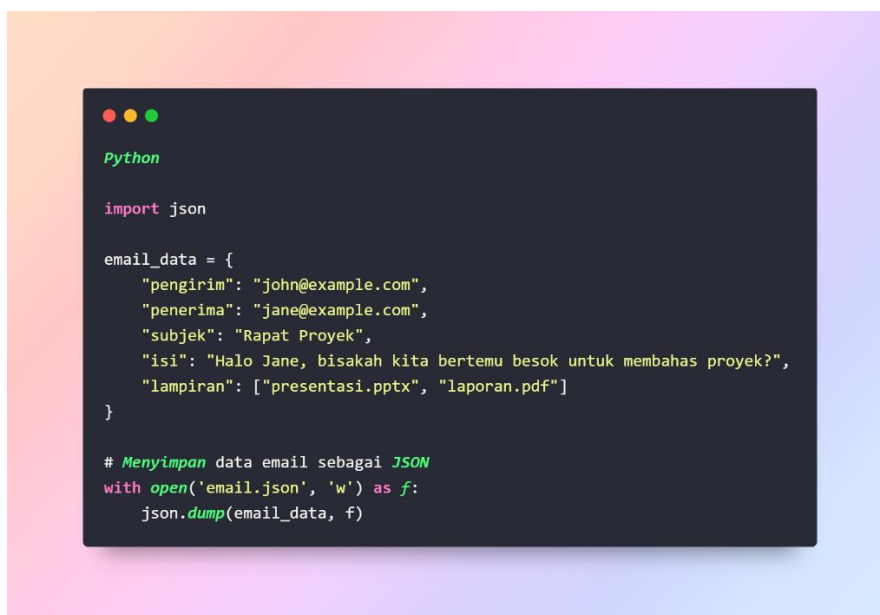
Data tidak terstruktur adalah informasi yang tidak memiliki model atau organisasi yang telah ditentukan sebelumnya. Karakteristiknya meliputi:

- Format yang tidak konsisten atau tidak dapat diprediksi
- Sulit untuk dianalisis menggunakan metode query tradisional
- Sering disimpan dalam sistem penyimpanan objek atau dokumen

Contoh:

Email, dokumen teks, file audio, atau video.

Implementasi:



## 2.3 Data Semi-Terstruktur

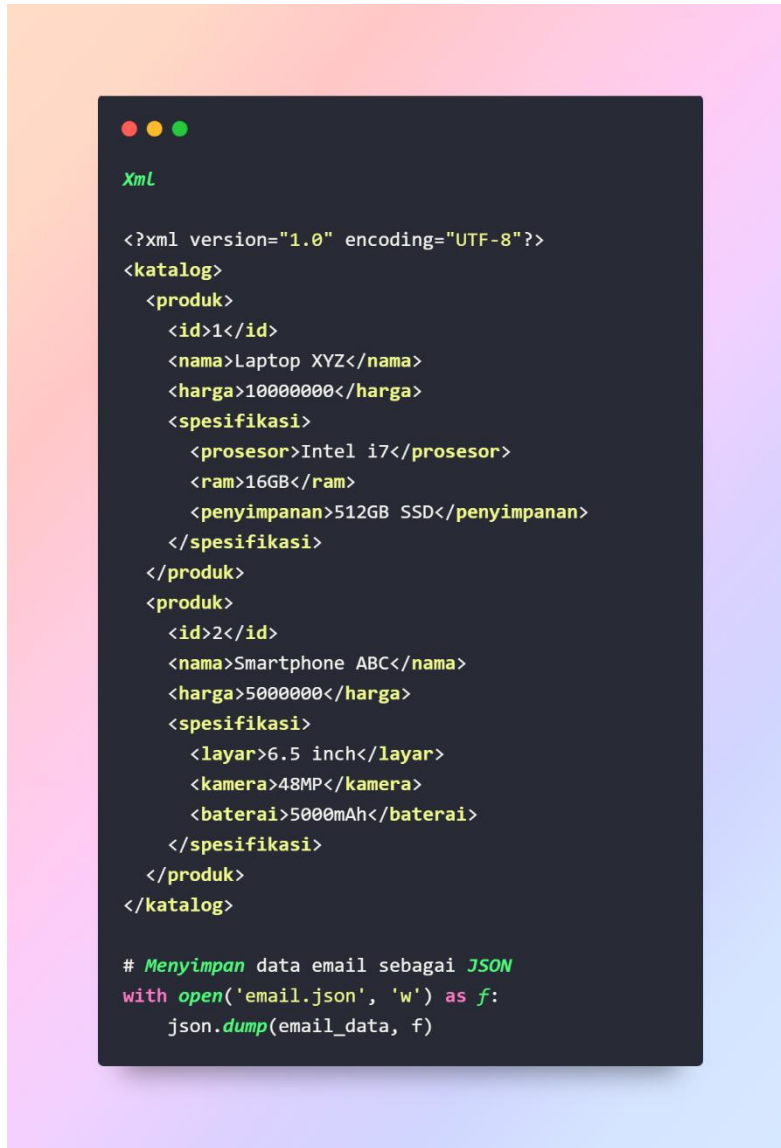
Data semi-terstruktur adalah bentuk hybrid yang memiliki beberapa karakteristik data terstruktur, namun tidak sepenuhnya mengikuti struktur formal tabel relasional. Karakteristiknya meliputi:

- Memiliki beberapa struktur, tetapi lebih fleksibel daripada data terstruktur
- Biasanya self-describing (memiliki metadata terintegrasi)
- Sering disimpan dalam format seperti XML atau JSON

Contoh:

Dokumen XML atau JSON yang menggambarkan produk dalam katalog online.

Implementasi:



```
Xml

<?xml version="1.0" encoding="UTF-8"?>
<katalog>
  <produk>
    <id>1</id>
    <nama>Laptop XYZ</nama>
    <harga>10000000</harga>
    <spesifikasi>
      <prosesor>Intel i7</prosesor>
      <ram>16GB</ram>
      <penyimpanan>512GB SSD</penyimpanan>
    </spesifikasi>
  </produk>
  <produk>
    <id>2</id>
    <nama>Smartphone ABC</nama>
    <harga>5000000</harga>
    <spesifikasi>
      <layar>6.5 inch</layar>
      <kamera>48MP</kamera>
      <baterai>5000mAh</baterai>
    </spesifikasi>
  </produk>
</katalog>

# Menyimpan data email sebagai JSON
with open('email.json', 'w') as f:
    json.dump(email_data, f)
```

### 3. Metode Pemrosesan Data

#### 3.1 Batch Processing

Batch processing adalah metode pemrosesan data di mana sejumlah besar data dikumpulkan selama periode waktu tertentu dan kemudian diproses secara bersamaan dalam satu "batch". Karakteristik utamanya meliputi:

- Pemrosesan data dalam jumlah besar secara periodik
- Biasanya dilakukan pada waktu yang telah dijadwalkan
- Cocok untuk analisis historis dan laporan komprehensif

Contoh:

Pemrosesan transaksi harian untuk laporan keuangan bulanan.

Implementasi:

```
Python

import pandas as pd
from datetime import datetime

def process_daily_transactions(date):
    # Membaca file transaksi harian
    df = pd.read_csv(f'transactions_{date}.csv')

    # Melakukan agregasi
    daily_summary = df.groupby('product_category').agg({
        'sales_amount': 'sum',
        'transaction_id': 'count'
    }).reset_index()

    daily_summary.columns = ['Category', 'Total Sales', 'Number of Transactions']

    # Menyimpan hasil
    daily_summary.to_csv(f'daily_summary_{date}.csv', index=False)

# Menjalankan proses untuk setiap hari dalam sebulan
for day in range(1, 32):
    date = datetime(2024, 1, day).strftime('%Y-%m-%d')
    process_daily_transactions(date)

print("Batch processing selesai.")
```

### 3.2 Stream Processing

Stream processing adalah metode pemrosesan data di mana data diproses secara real-time saat diterima. Karakteristik utamanya meliputi:

- Pemrosesan data secara kontinyu dan real-time
- Cocok untuk aplikasi yang membutuhkan respons cepat
- Biasanya digunakan dalam sistem pemantauan dan deteksi anomali

Contoh:

Sistem deteksi penipuan untuk transaksi kartu kredit.

Implementasi:

```
Python

from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

# Inisialisasi Spark Session
spark = SparkSession.builder.appName("FraudDetection").getOrCreate()

# Definiskan schema untuk stream data
schema = StructType([
    StructField("transaction_id", StringType()),
    StructField("amount", DoubleType()),
    StructField("timestamp", TimestampType()),
    StructField("location", StringType())
])

# Baca stream data
transactions = spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "transactions") \
    .load() \
    .select(from_json(col("value").cast("string"), schema).alias("data")) \
    .select("data.*")

# Deteksi transaksi mencurigakan (contoh: jumlah > 10000)
suspicious_transactions = transactions.filter(transactions.amount > 10000)

# Tulis hasil ke console (dalam praktik nyata, bisa ditulis ke database atau sistem alert)
query = suspicious_transactions.writeStream \
    .outputMode("append") \
    .format("console") \
    .start()

query.awaitTermination()
```

#### 4. Kesimpulan

Pemahaman mendalam tentang struktur data dan metode pemrosesannya sangat penting dalam era big data. Data terstruktur, tidak terstruktur, dan semi-

terstruktur masing-masing memiliki karakteristik dan use case yang unik. Demikian pula, batch processing dan stream processing menawarkan pendekatan yang berbeda untuk menangani data, masing-masing dengan kelebihan dan tantangannya sendiri.

Pemilihan struktur data dan metode pemrosesan yang tepat tergantung pada kebutuhan spesifik dari aplikasi atau sistem yang sedang dikembangkan. Faktor-faktor seperti volume data, kecepatan pemrosesan yang dibutuhkan, kompleksitas analisis, dan kebutuhan real-time semua berperan dalam pengambilan keputusan ini.

Dengan perkembangan teknologi yang terus berlanjut, kita dapat mengharapkan evolusi lebih lanjut dalam cara kita menyimpan, mengelola, dan memproses data. Oleh karena itu, penting bagi para profesional di bidang data dan teknologi informasi untuk terus memperbarui pengetahuan mereka dan beradaptasi dengan perkembangan terbaru dalam pengelolaan dan analisis data.

## Referensi

1. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson.
2. Marz, N., & Warren, J. (2015). *Big Data: Principles and Best Practices of Scalable Real-time Data Systems*. Manning Publications.
3. Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media.
4. Spark Documentation. (2024). *Structured Streaming Programming Guide*. Apache Spark. <https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html>
5. PostgreSQL Global Development Group. (2024). *PostgreSQL Documentation*. <https://www.postgresql.org/docs/>