Laporan Struktur Data
Tugas 1 - Membuat ...

Disusun Oleh:
Nama Mahasiswa
NRP
Dosen Pengampu:
Arta Kusuma Hernanda

Departmen Teknik Komputer
Institut Teknologi Sepuluh Nopember
Semester Ganjil 2024

Daftar Isi

Daftar Gambar

Daftar Tabel

Final Project 1

Document Database (MongoDB) dan Key-Value Store (Redis)

As shown in Figure **??**, the logo represents...

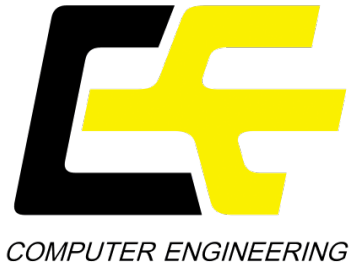

Figure 1.1: Caption describing the logo

1.1 Tujuan

Memahami konsep dan penggunaan document database dan key-value store.

1.2 Dasar Teori

- Konsep NoSQL
- Karakteristik document database dan key-value store
- Use cases untuk MongoDB dan Redis

1.3 Alat dan Bahan

- MongoDB dan Redis
- MongoDB Compass dan Redis CLI/Desktop Manager

1.4 Prosedur

1.4.1 Bagian 1: MongoDB

- Instalasi dan konfigurasi MongoDB
- CRUD operations pada document
- Querying dan indexing di MongoDB

1.4.2 Bagian 2: Redis

- Instalasi dan konfigurasi Redis
- Operasi dasar Redis (SET, GET, DEL, etc.)
- Implementasi caching sederhana
- Penggunaan struktur data Redis (Lists, Sets, Hashes)

1.5 Skenario

1. MongoDB digunakan untuk menyimpan konten blog yang memiliki struktur fleksibel (judul, konten, tag, dll).
2. Redis digunakan untuk menghitung view, menyimpan daftar post populer, dan caching konten.

1.6 Instruksi Praktikum

1. Mulai dengan MongoDB:
    - Implementasikan operasi CRUD untuk post blog.
    - Eksperimen dengan query berbasis tag dan sorting.
2. Lanjutkan dengan Redis:
    - Implementasikan penghitung view menggunakan inkremen.
    - Buat leaderboard post terpopuler dengan sorted set.
    - Implementasikan sistem caching sederhana.

3. Analisis:

    - Bandingkan kecepatan operasi antara MongoDB, Redis, dan RDBMS yang telah dipelajari sebelumnya.
    - Diskusikan skenario di mana masing-masing teknologi paling cocok digunakan.

1.7 Implementasi

1.7.1 Bagian 1: MongoDB - Manajemen Konten Blog

Koneksi ke MongoDB

Code

```
from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')
db = client['blog_db']
posts = db['posts']
```

Membuat Post Baru

Code

```
new_post = {
    "title": "Pengenalan NoSQL",
    "content": "NoSQL adalah jenis database yang...",
    "author": "John Doe",
    "tags": ["database", "nosql", "tutorial"],
    "date": datetime.now()
}
result = posts.insert_one(new_post)
print(f"Post baru dibuat dengan ID: {result.inserted_id}")
```

Mencari Post

Code

```
# Mencari berdasarkan tag
for post in posts.find({"tags": "nosql"}):
    print(f"Judul: {post['title']}")

```

```
# Mencari dan sort berdasarkan tanggal
for post in posts.find().sort("date", -1).limit(5):
    print(f"Judul: {post['title']}, Tanggal: {post['date']}")
```

Update Post

Code

```
posts.update_one(
    {"title": "Pengenalan NoSQL"},
    {"$set": {"content": "NoSQL adalah jenis database yang fleksibel..."}}
)
```

Menghapus Post

Code

```
posts.delete_one({"title": "Pengenalan NoSQL"})
```

1.7.2 Bagian 2: Redis - Caching View Count

Koneksi ke Redis

Code

```
import redis

r = redis.Redis(host='localhost', port=6379, db=0)
```

Increment View Count

Code

```
def increment_view(post_id):
    key = f"post:{post_id}:views"
    return r.incr(key)

# Simulasi view
```

```
post_id = "12345"
views = increment_view(post_id)
print(f"Post {post_id} telah dilihat {views} kali")
```

Get Top Posts

Code

```
def get_top_posts(limit=5):
    # Asumsikan kita menyimpan skor dalam sorted set
    return r.zrevrange("top_posts", 0, limit-1, withscores=True)

# Simulasi update top posts
r.zadd("top_posts", {"post:12345": 10, "post:67890": 15})

top_posts = get_top_posts()
for post, score in top_posts:
    print(f"Post {post.decode()} memiliki {int(score)} views")
```

Caching Konten Post

Code

```
def get_post_content(post_id):
    cache_key = f"post:{post_id}:content"
    content = r.get(cache_key)
    if content:
        return content.decode()
    else:
        # Simulasi mengambil dari MongoDB
        content = "Ini adalah konten post yang diambil dari MongoDB"
        r.setex(cache_key, 3600, content)  # Cache selama 1 jam
        return content

content = get_post_content("12345")
print(f"Konten post: {content}")
```

1.8 Analisis

1.8.1 MongoDB

- Fleksibel untuk menyimpan struktur data yang kompleks (nested documents, arrays).
- Baik untuk data yang sering berubah strukturnya.
- Query yang powerful untuk pencarian dan aggregasi.

1.8.2 Redis

- Sangat cepat untuk operasi read/write sederhana.
- Ideal untuk caching, counting, dan leaderboards.
- Struktur data yang beragam memungkinkan implementasi fitur seperti rate limiting, session management.

1.8.3 Perbandingan dengan RDBMS

- MongoDB lebih fleksibel dalam skema dibanding RDBMS, cocok untuk rapid development.
- Redis jauh lebih cepat untuk operasi sederhana, tapi tidak cocok untuk query kompleks.
- RDBMS lebih baik untuk data yang memerlukan integritas referensial yang ketat.

Final Project 2

Time Series Database Praktikum

As shown in Table **??**, the data illustrates...

Column 1	Column 2	Column 3	Column 4
Row 1, Col 1	Row 1, Col 2	Row 1, Col 3	Row 1, Col 4
Row 2, Col 1	Row 2, Col 2	Row 2, Col 3	Row 2, Col 4
Row 3, Col 1	Row 3, Col 2	Row 3, Col 3	Row 3, Col 4

Table 2.1: Example table with 4 columns and 3 rows

2.1 Tujuan

Mengenalkan konsep dan aplikasi time series database.

2.2 Dasar Teori

- Konsep time series data
- Use cases IoT data

2.3 Alat dan Bahan

- InfluxDB (untuk time series)

2.4 Prosedur

1. Instalasi dan konfigurasi database
2. Ingestion data time series
3. Query dan analisis data time series

2.5 Skenario: Monitoring Suhu Ruangan dengan InfluxDB

Anda akan mengimplementasikan sistem monitoring suhu untuk beberapa ruangan dalam sebuah gedung menggunakan InfluxDB.

2.5.1 Langkah-langkah

Instalasi dan Konfigurasi InfluxDB (10 menit)

- Pastikan InfluxDB terinstal
- Buka InfluxDB CLI atau UI

Membuat Database dan Retention Policy (5 menit)

Code

```
[language=SQL]
CREATE DATABASE room_temp
USE room_temp
CREATE RETENTION POLICY "one_week" ON "room_temp" DURATION 1w REPLICATION 1
    ↪ DEFAULT
```

Ingestion Data (15 menit)

Gunakan Python dengan library `influxdb-client`:

Code

```
[language=Python]
from influxdb_client import InfluxDBClient, Point
from influxdb_client.client.write_api import SYNCHRONOUS
import random
from datetime import datetime, timedelta

client = InfluxDBClient(url="http://localhost:8086", token="your-token",
    ↪ org="your-org")
write_api = client.write_api(write_options=SYNCHRONOUS)

# Simulasi data suhu untuk 3 ruangan selama 24 jam
start_time = datetime.now() - timedelta(days=1)
for i in range(288):  # 5 menit interval selama 24 jam
    for room in ["Room A", "Room B", "Room C"]:
        point = Point("temperature") \
            .tag("room", room) \
            .field("value", random.uniform(20.0, 30.0)) \
```

```
            .time(start_time + timedelta(minutes=5*i))
        write_api.write(bucket="room_temp", record=point)

print("Data ingestion complete")
```

Query dan Analisis (15 menit)

Gunakan InfluxDB Query Language (InfluxQL) atau Flux:

Code

```
[language=SQL]
-- Rata-rata suhu per ruangan dalam 24 jam terakhir
SELECT mean("value") FROM "temperature" WHERE time > now() - 24h GROUP BY "room"

-- Suhu maksimum per ruangan
SELECT max("value") FROM "temperature" GROUP BY "room"

-- Jumlah pengukuran di atas 25C per ruangan
SELECT count("value") FROM "temperature" WHERE "value" > 25 GROUP BY "room"
```

Visualisasi (optional, 10 menit)

- Gunakan Grafana atau library plotting Python untuk memvisualisasikan data suhu dari waktu ke waktu.

2.6 Analisis dan Diskusi (20 menit)

2.6.1 Time Series Database (InfluxDB)

- Kelebihan dalam menangani data temporal dengan volume besar
- Efisiensi query untuk analisis time-based
- Use cases: IoT, monitoring sistem, analisis keuangan

2.6.2 Perbandingan dengan RDBMS tradisional

- Diskusikan bagaimana implementasi serupa akan berbeda/lebih sulit di RDBMS
- Bahas trade-offs antara fleksibilitas RDBMS dan kinerja spesifik dari database khusus ini

2.7 Instruksi untuk Instruktur

- Pastikan InfluxDB tersedia untuk semua mahasiswa
- Sediakan environment Python dengan library yang diperlukan
- Dorong mahasiswa untuk bereksperimen dengan query dan parameter berbeda
- Tekankan pada konsep kunci seperti time-based analysis untuk InfluxDB

Daftar Pustaka

[1] MongoDB, Inc., 2024. MongoDB Documentation. [online] Available at: `https://docs.mongodb.com/` [Accessed August 2024].

[2] Redis Ltd., 2024. Redis Documentation. [online] Available at: `https://redis.io/documentation/` [Accessed August 2024].

[3] InfluxData Inc., 2024. InfluxDB Documentation. [online] Available at: `https://docs.influxdata.com/` [Accessed August 2024].