

IF4070 Representasi Pengetahuan dan Penalaran  
**Graf Pengetahuan dan Sistem RAG dengan Domain Naruto**  
**Laporan Tugas Proyek 2**

Disusun untuk memenuhi tugas mata kuliah Representasi Pengetahuan dan Penalaran pada Semester 1 (satu) Tahun Akademik 2025/2026



Disusun oleh:

**Rizqika Mulia Pratama (13522126)**  
**Samy Muhammad Haikal (13522151)**  
**Rafif Ardhinto Ichwantoro (13522159)**

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**  
**2025**

# DAFTAR ISI

<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB I</b>	<b>3</b>
1.1 Pemanfaatan	3
1.2 Deskripsi label dan simpul	3
1.3 Keterkaitan dengan Ontologi	4
<b>BAB II</b>	<b>6</b>
<b>SISTEM RAG</b>	<b>6</b>
2.1 Modul Koneksi dan Repositori (database)	6
2.2 Pembangunan Konteks	7
2.3 LLM	8
2.4 Antarmuka Pengguna (GUI)	8
2.5 Pemanfaatan Graf Pengetahuan	8
<b>BAB III</b>	<b>10</b>
<b>PEMBAHASAN</b>	<b>10</b>
3.1. Hasil Implementasi Sistem RAG	10
3.2. Perbandingan Sistem RAG & Berbasis Pengetahuan	15
3.3. Keterbatasan Graf Pengetahuan dan RAG	15
<b>BAB IV</b>	<b>17</b>
<b>KESIMPULAN</b>	<b>17</b>
<b>BAB V</b>	<b>18</b>
6.1. Sumber Pustaka	18
6.2. Lampiran	18

# BAB I

## GRAF PENGETAHUAN

### 1.1 Pemanfaatan

Pada tugas ini, graf pengetahuan berfungsi sebagai basis pengetahuan utama yang menjadi fondasi dari sistem Retrieval-Augmented Generation (RAG) yang dikembangkan. Seluruh proses pencarian konteks, relevansi informasi, hingga penyusunan jawaban oleh model bergantung pada kualitas graf pengetahuan tersebut.

Graf pengetahuan yang tersusun kemudian menjadi inti dari mekanisme retrieval dalam sistem RAG. Ketika pengguna memberikan pertanyaan tertentu, sistem melakukan pencarian konteks melalui query Neo4j untuk menemukan entitas, hubungan, atau atribut yang relevan dengan masukan tersebut. Informasi yang berhasil ditemukan lalu digabungkan (augmented) dengan masukan pengguna sehingga model bahasa dapat menghasilkan jawaban yang lebih tepat, berbasis fakta, serta mengurangi kecenderungan halusinasi.

### 1.2 Deskripsi label dan simpul

Berikut adalah label-label dan simpul yang ada pada graf pengetahuan yang dibuat. Kelas pada ontologi rdf akan dipetakan menjadi label, sementara itu individu akan dipetakan menjadi simpul, selain itu object properties akan dipetakan menjadi hubungan antar simpul pada graf pengetahuan.

Label	Keterangan	Contoh Simpul (Individuals)
Shinobi	Kelas utama untuk karakter ninja. Merupakan domain untuk banyak properti seperti hasElement, isFrom, dan masterJutsu.	Madara_Uchiha, Kakashi_Hatake, RockLee, Han, Juzo_Biwa
Village	Kelas induk yang merepresentasikan desa asal atau afiliasi geografis seorang Shinobi.	Konohagakure, Kirigakure, Iwagakure, Sunagakure
Big_Village	Sub-kelas dari Village. Merepresentasikan Lima Desa Besar Ninja.	Konohagakure, Kirigakure, Iwagakure, Sunagakure
Small_Village	Sub-kelas dari Village. Merepresentasikan desa-desa yang lebih kecil.	Otogakure, Amegakure
Clan	Kelas yang merepresentasikan garis keturunan atau keluarga ninja.	Akimichi, Uchiha, Senju, Hozuki, Nara
Bijuu	Kelas untuk Siluman Berekor (Tailed Beasts). Memiliki properti data tailNumber.	Saiken, Kokuo, Son_Goku
Jinchuuriki	Kelas ekuivalen yang didefinisikan sebagai Shinobi yang menjadi wadah Bijuu (memiliki relasi isJinchuurikiOf).	Han, Roshi, Naruto

Label	Keterangan	Contoh Simpul (Individuals)
Organizations	Kelas untuk kelompok atau organisasi tempat Shinobi bernaung di luar desa.	Akatsuki, Seven_Swordmans_of_The_Mist, Taka
AkatsukiMember	Shinobi yang merupakan anggota Akatsuki	Hidan, Kisame_Hoshigaki , Itachi_Uchiha
UchihaClanMember	Kelas untuk Shinobi yang anggota klan Uchiha. Secara otomatis mencakup karakter dengan relasi isClanMemberOf ke Uchiha.	Madara_Uchiha, Sasuke_Uchiha
Jutsu	Kelas induk untuk semua teknik ninja.	Kage_Bunshin_no_Jutsu, Mugen_Tsukoyomi
Ninjutsu	Sub-kelas dari Jutsu yang berfokus pada teknik ninja berbasis chakra.	Kage_Bunshin_no_Jutsu
Taijutsu	Sub-kelas dari Jutsu yang berfokus pada teknik fisik/bela diri.	Hakkeshō_Kaiten, Drunken_Fist
Genjutsu	Sub-kelas dari Jutsu yang berfokus pada teknik ilusi.	Mugen_Tsukoyomi, Izanagi, Izanami, Tsukoyomi, Kotoamatsukami
KekkeiGenkai	Sub-kelas dari Jutsu yang merupakan warisan genetik.	Hakkeshō_Kaiten, Mokuton
Genin	Sub-kelas dari Rank , tingkatan ninja paling dasar.	Naruto_Uzumaki, Konohamaru
Chuunin	Sub-kelas dari Rank, tingkatan ninja menengah.	RockLee, Tenten
Jounin	Sub-kelas dari Rank, tingkatan ninja elit/atas.	Kakashi_Hatake, Han, Roshi, Juzo_Biwa
Kage	Sub-kelas dari Rank, pemimpin desa.	Hashirama, Naruto, Gaara
Nukenin	status untuk ninja yang hilang atau dalam pelarian.	Hidan, Juzo_Biwa
Sennin	Sub-kelas dari Shinobi, merujuk pada Petapa.	Jiraiya, Naruto_Uzumaki, Jugo
Elemental	Kelas yang merepresentasikan elemen dasar chakra.	Katon, Suiton, Doton, Futon, Raiton, Mokuton

### 1.3 Keterkaitan dengan Ontologi

Ontologi dari rdf digunakan sebagai dasar pembuatan graf pengetahuan pada tugas ini. Ontologi (naruto.rdf) bertindak sebagai skema yang mengatur struktur logika dari Graf Pengetahuan yang tersimpan di Neo4j. Keterkaitannya adalah sebagai berikut:

Setiap kelas dalam ontologi berfungsi sebagai definisi kategori konsep, dan masing-masing dipetakan secara langsung menjadi label simpul di Neo4j. Dengan cara ini, struktur konseptual yang ada pada RDF dapat diwujudkan kembali sebagai struktur graf yang

eksplisit memastikan bahwa semua individu (seperti Madara\_Uchiha) akan memiliki label :Shinobi di dalam graf.

Sementara itu, owl:ObjectProperty di dalam ontologi menentukan jenis hubungan antar entitas, dan masing-masing diterjemahkan menjadi relasi (edge) pada Neo4j. Contohnya, properti isClanMemberOf menyatakan hubungan antara seorang Shinobi dengan sebuah Clan, sehingga pada graf akan muncul relasi (:Shinobi)-[:IS\_CLAN\_MEMBER\_OF]->(:Clan). Dengan pemetaan ini, struktur hubungan yang ada dalam RDF dapat ditranslasikan sebagai jaringan relasi dalam graf pengetahuan.

## BAB II

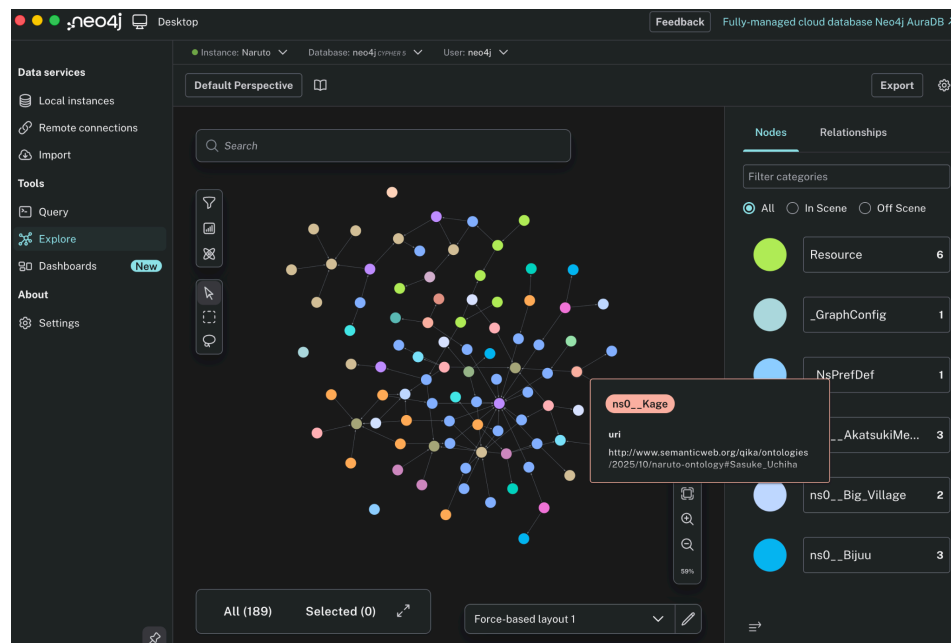
# SISTEM RAG

Arsitektur sistem dirancang untuk mengintegrasikan tiga komponen utama yaitu basis pengetahuan berbasis graf (Neo4j), proses temu balik (retrieval), dan mesin generatif (LLM) ke dalam satu pipeline yang stabil. Desain keseluruhan mengikuti *Repository Pattern* untuk memastikan pemisahan yang jelas antara lapisan akses data, logika bisnis, dan antarmuka pengguna.

Tujuan utama arsitektur ini adalah menghasilkan sistem *retrieval-augmented generation* (RAG) yang mampu:

1. Mengambil informasi relevan dari graf pengetahuan,
2. Menyusun konteks linguistik terstruktur, dan
3. Memberikan jawaban yang informatif dan kontekstual melalui model bahasa.

### 2.1 Modul Koneksi dan Repositori (database)



Modul ini merupakan fondasi dari seluruh proses retrieval, bertugas menyediakan akses stabil dan efisien ke database Neo4j. Kelas `Neo4jConnection` di `connection.py` diimplementasikan menggunakan pola Singleton agar hanya ada satu driver aktif selama aplikasi berjalan. Pendekatan ini memberikan dua keuntungan utama:

- Efisiensi sumber daya: driver Neo4j tidak dibuat berulang-ulang.
- Stabilitas operasi: menghindari risiko *connection overflow* dan kebocoran memori.

Setiap pemanggilan koneksi akan merujuk pada instance yang sama, sehingga pengelolaan sesi *Cypher* menjadi lebih terkontrol.

### Eksekusi Query Cypher

File repository.py bertanggung jawab menjalankan perintah Cypher secara atomik. Setiap fungsi dalam repository ditulis untuk menyelesaikan satu tugas spesifik, misalnya:

- **Get\_shinobi\_info**

Mengambil informasi lengkap mengenai karakter, termasuk relasi desa, tingkat kekuatan, afiliasi, dan elemen. Kueri ini dirancang dalam satu round-trip ke Neo4j dan memanfaatkan OPTIONAL MATCH untuk menangani data hilang, collect() untuk menggabungkan beberapa relasi ke satu array, dan penggabungan properti dan label ke satu record.

Hal ini memastikan sistem retrieval dapat memperoleh konteks yang kaya tanpa perlu memanggil database berulang kali.

### **Penanganan Query Traversal Kompleks**

Beberapa konsep dalam ontologi seperti Dojutsu, Kekkei Genkai, atau teknik turunan tidak selalu memiliki pola relasi rapi karena kelas-kelas tersebut tidak memiliki individu sendiri dan hanya terhubung lewat relasi owl:someValuesFrom. Sementara itu, neo4j hanya akan menggunakan kelas sebagai label dari individu yang ada, karena tidak ada individu yang bertipe kelas 'Eye', 'Dojutsu', atau 'Kekkei\_Genkai' maka neo4j tidak akan membuat label untuk kelas tersebut. Untuk itu, repository menggunakan teknik:

```
cypher

EXISTS { MATCH (...) }
COALESCE()
```

yang memungkinkan sistem menangani variasi struktur ontologi secara fleksibel, terutama saat melakukan pencarian jenis mata (Dojutsu) atau teknik.

## **2.2 Pembangunan Konteks**

Modul ini bertugas menerjemahkan pertanyaan pengguna menjadi struktur pencarian yang lebih formal dan mengumpulkan hasilnya menjadi teks deskriptif yang siap diberikan kepada LLM.

### **Normalisasi dan Entity Linking**

Fungsi normalize\_entity melakukan proses entity linking sederhana. Karena pengguna sering menggunakan istilah tidak baku, sistem menyediakan kamus pemetaan:

```
mapping = {
    "naruto": "Naruto_Uzumaki", "sasuke": "Sasuke_Uchiha", "kakashi": "Kakashi_Hatake",
    "jiraiya": "Jiraiya", "sakura": "Sakura_Haruno", "itachi": "Itachi_Uchiha",
    "madara": "Madara_Uchiha", "obito": "Obito_Uchiha", "tobi": "Tobi",
    "nagato": "Nagato", "pain": "Nagato", "konan": "Konan", "kisame": "Kisame",
    "kakuzu": "Kakuzu", "hidan": "Hidan", "deidara": "Deidara", "sasori": "Sasori",
    "zetsu": "Zetsu", "orochimaru": "Orochimaru", "tsunade": "Tsunade",
    "minato": "Minato_Namikaze", "hashirama": "Hashirama", "tobirama": "Tobirama",
    "hiruzen": "Hiruzen_Sarutobi", "gaara": "Gaara", "killer_bee": "Killer_Bee",
    "bee": "Killer_Bee", "guy": "Might_Guy", "lee": "RockLee", "neji": "Neji_Hyuuga",
    "hinata": "Hinata_Hyuuga", "shikamaru": "Shikamaru", "choji": "Choji", "ino": "Ino",
    "tenten": "Tenten", "konohamaru": "Konohamaru_Sarutobi", "shisui": "Shisui_Uchiha",
    "zabuza": "Zabuza", "haku": "Haku", "yagura": "Yagura", "utakata": "Utakata",
    "han": "Han", "roshi": "Roshi", "yugito": "Yugito", "fuu": "Fuu",
    "chojuro": "Chojuro", "mangetsu": "Mangetsu_Hozuki", "fuguki": "Fuguki",
    "konoha": "Konohagakure", "suna": "Sunagakure", "kiri": "Kirigakure",
}
```

Normalisasi ini sangat penting karena entitas dalam Neo4j menggunakan nama formal dari ontologi RDF, bukan istilah sehari-hari. Proses ini meningkatkan tingkat keberhasilan pencarian dan mengurangi jawaban “entitas tidak ditemukan”.

### Penyusunan Konteks

Fungsi `build_context` menggabungkan hasil dari berbagai fungsi repositori yaitu:

- Informasi dasar entitas
- Hubungan dengan karakter lain
- Afiliasi desa
- Tipe kekuatan (chakra nature, kekkei genkai, dojutsu, dsb.)
- Fakta tambahan dari ontologi RDF

Seluruh informasi disatukan menjadi paragraf rapi seperti:

“Naruto Uzumaki adalah Shinobi dari Konohagakure, memiliki chakra elemen angin, dan merupakan Jinchuriki Ekor Sembilan...”

Konteks ini kemudian menjadi prompt augmentasi bagi LLM untuk menghasilkan jawaban yang lebih akurat, berbasis data, dan tidak halusinatif.

## 2.3 LLM

LLM yang digunakan pada tugas ini adalah *Qwen/Qwen2.5-1.5B-Instruct* dari HuggingFace. Model ini dipilih karena ukurannya yang relatif kecil sehingga dapat dijalankan pada perangkat dengan spesifikasi menengah, serta memiliki performa yang baik dalam memproses instruksi berbahasa Indonesia. Selain itu, model ini mendukung *quantization*, memungkinkan proses inferensi berjalan tanpa memerlukan GPU berkapasitas besar. Penggunaan model lokal juga memberikan sejumlah keuntungan penting, seperti independensi dari koneksi internet, tidak adanya batasan kuota API, serta keamanan data yang lebih terjamin karena seluruh proses dilakukan secara lokal.

Agar performanya optimal di berbagai platform, modul `llm_engine.py` dilengkapi mekanisme deteksi perangkat keras secara otomatis. Pada perangkat macOS dengan chip Apple Silicon, model akan berjalan menggunakan backend mps (Metal); pada Windows atau Linux



yang memiliki GPU Nvidia, sistem akan memanfaatkan cuda; dan ketika tidak ada GPU yang tersedia, model secara otomatis menjalankan inferensi melalui cpu.

#### **2.4 Antarmuka Pengguna (GUI)**

Antarmuka dibangun menggunakan Streamlit, dipilih karena mendukung pembuatan web app interaktif dengan sedikit kode. Agar pengalaman lebih natural, jawaban LLM ditampilkan token demi token menggunakan TextIteratorStreamer. Teknik ini memberikan efek "mengetik langsung di layar", mirip antarmuka ChatGPT.

##### **Flow Interaksi**

1. Pengguna memasukkan pertanyaan
2. Pertanyaan dinormalisasi
3. Entitas dicari di Neo4j
4. Konteks dibangun
5. Token jawaban ditampilkan real-time

Desain ini menghasilkan pengalaman tanya-jawab yang mengalir, interaktif, dan informatif.

## BAB III

### PEMBAHASAN

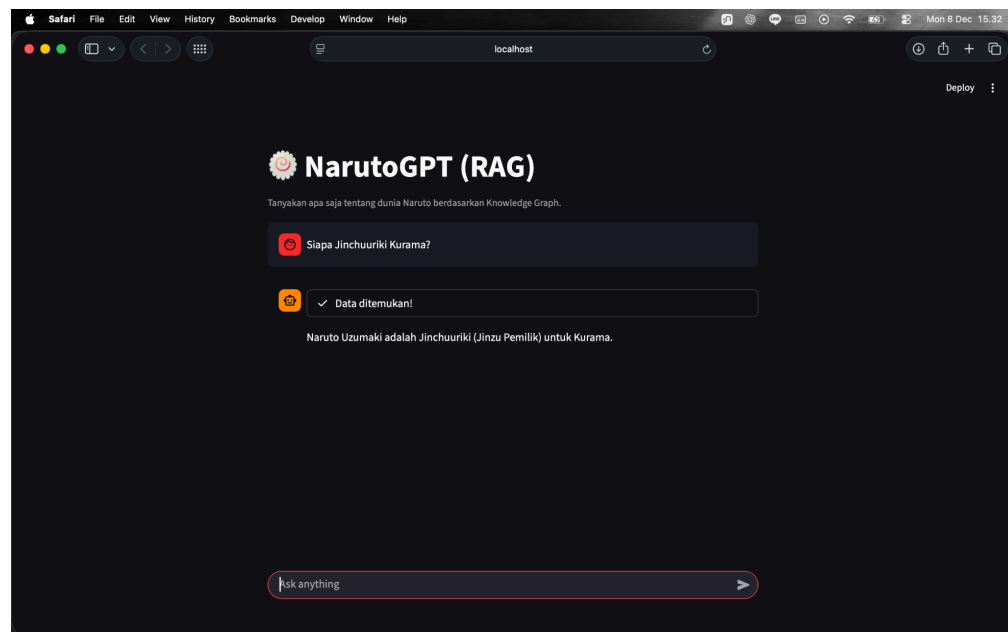
#### 3.1. Hasil Implementasi Sistem RAG

Sistem RAG yang dikembangkan menggunakan berhasil mengintegrasikan basis pengetahuan berupa graf pengetahuan (neo4j dump) ke dalam proses prompting LLM. Sistem ini mampu menjawab berbagai pertanyaan yang berbentuk bahasa alami mulai dari pencarian fakta sederhana hingga inferensi logika yang kompleks. Berikut adalah contoh-contoh inferensi menggunakan sistem RAG yang dibuat:

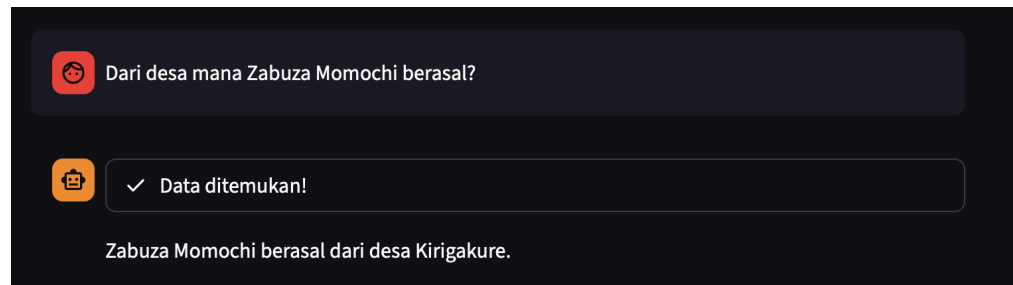
##### A. Pertanyaan Faktual Langsung

Pertanyaan-pertanyaan ini bertujuan untuk menguji apakah sistem dapat menangani pertanyaan faktual langsung terkait domain yang ada dalam basis pengetahuan..

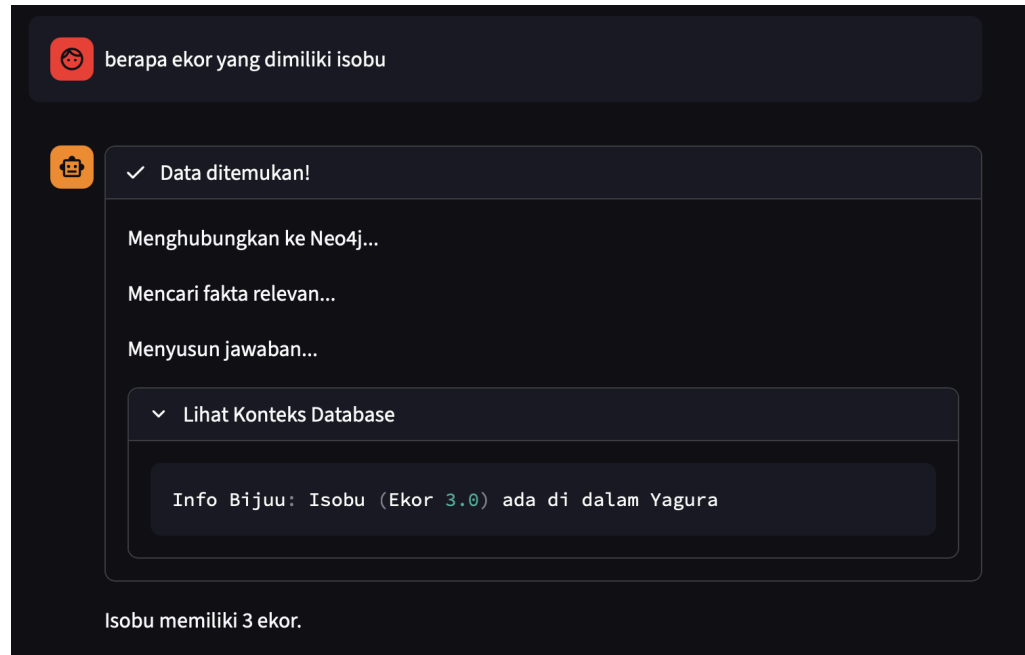
##### 1. Siapa Jinchuuriki Kurama?



##### 2. Dari desa mana Zabuza Momochi berasal?



##### 3. Berapa ekor yang dimiliki Isobu?





Sistem dapat menjawab berbagai pertanyaan-pertanyaan tersebut dengan akurat sesuai dengan basis pengetahuan.

#### B. Pertanyaan Himpunan (List)

Pertanyaan yang menargetkan pengambilan sekumpulan simpul (satu daftar) yang memiliki relasi yang sama ke satu simpul pusat. Pertanyaan ini menguji kemampuan many-to-one retrieval.

4. Siapa saja anggota Seven Swordsmen of The Mist?

 Siapa saja anggota Seven Swordsmen of The Mist?


 ✓ Data ditemukan!


Seven Swordsmen of the Mist adalah kelompok ninja terkenal yang terdiri dari beberapa tokoh penting dalam cerita Naruto. Anggotanya termasuk:

1. Ringo
2. Juzo Biwa
3. Mangetsu Hozuki
4. Raiga Kurosuki
5. Jinpachi Munashi
6. Zabuza
7. Chojuro
8. Kushimaru Kuriarare
9. Jinin Akebino
10. Kisame
11. Fuguki

Semua anggota ini merupakan anggota utama dari kelompok tersebut dan sering dipakai sebagai bahan pertempuran dalam permainan Naruto.

5. Sebutkan anggota klan Uchiha yang ada di database.

 Sebutkan anggota klan Uchiha yang ada.

 ✓ Data ditemukan!

Menghubungkan ke Neo4j...

Mencari fakta relevan...

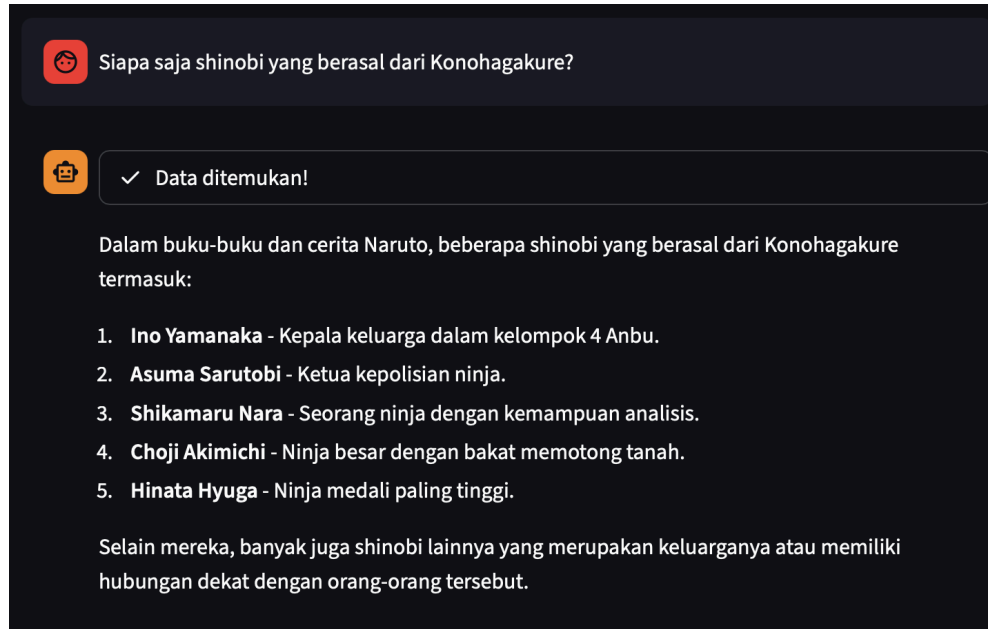
Menyusun jawaban...

▼ Lihat Konteks Database

Anggota Klan Uchiha: Madara\_Uchiha, Sasuke\_Uchiha, Itachi\_Uchiha, O

Anggota klan Uchiha yang ada adalah Madara Uchiha, Sasuke Uchiha, Itachi Uchiha, dan Obito Uchiha.

6. Siapa ninja yang berasal dari Iwagakure?

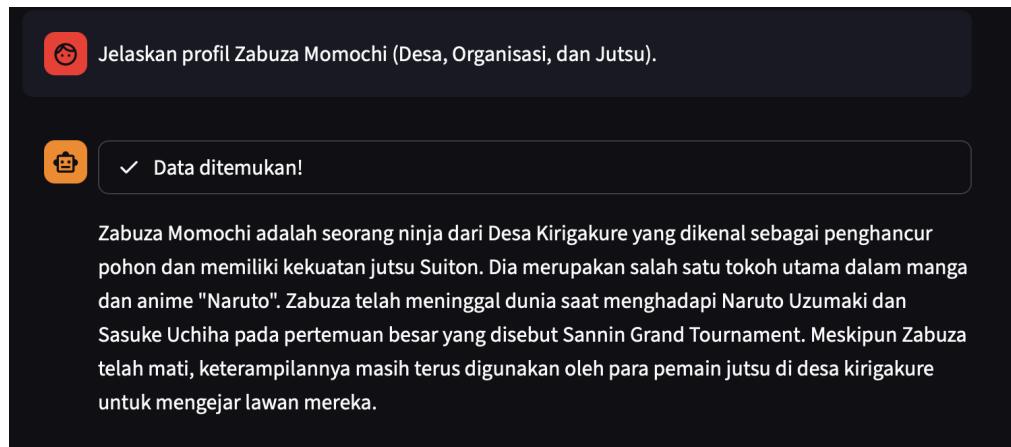


Sistem dapat menjawab berbagai pertanyaan-pertanyaan tersebut dengan akurat sesuai dengan basis pengetahuan.

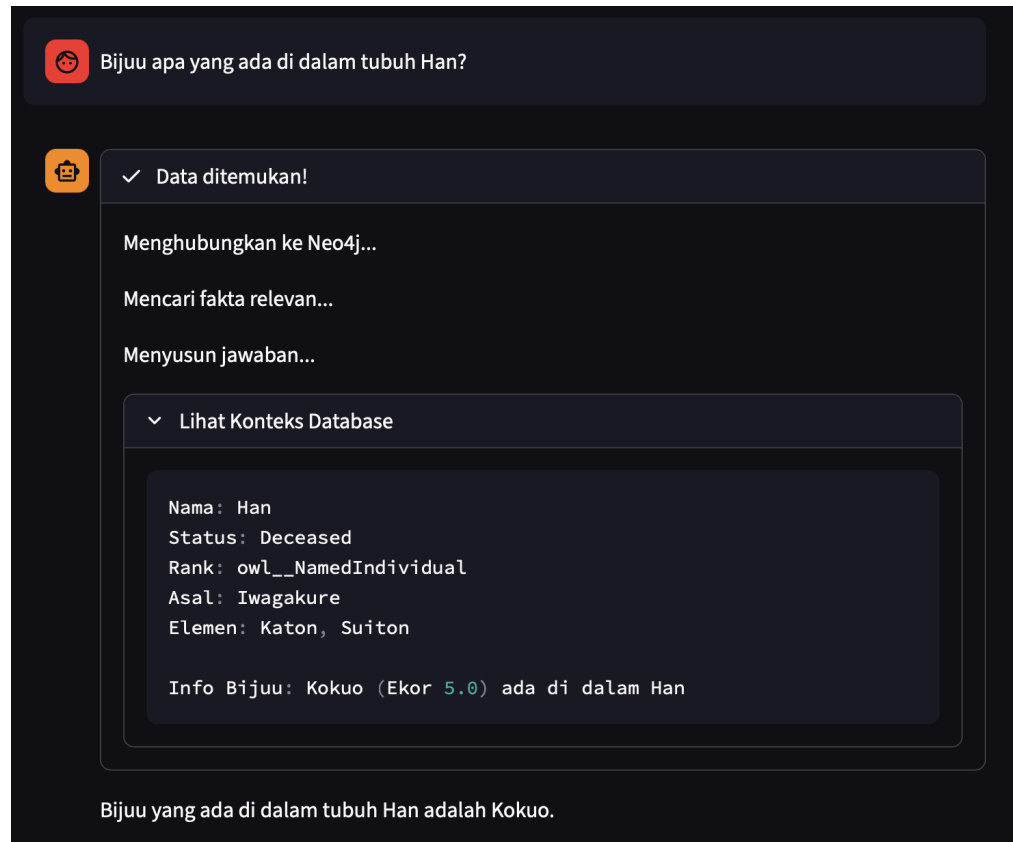
### C. Pertanyaan Kompleks

Pertanyaan-pertanyaan yang melibatkan agregasi banyak tipe relasi secara bersamaan untuk membangun profil lengkap. Pertanyaan-pertanyaan ini bertujuan untuk menguji kemampuan sistem dalam menyusun subgraph yang rumit menjadi satu konteks yang dapat dipahami LLM.

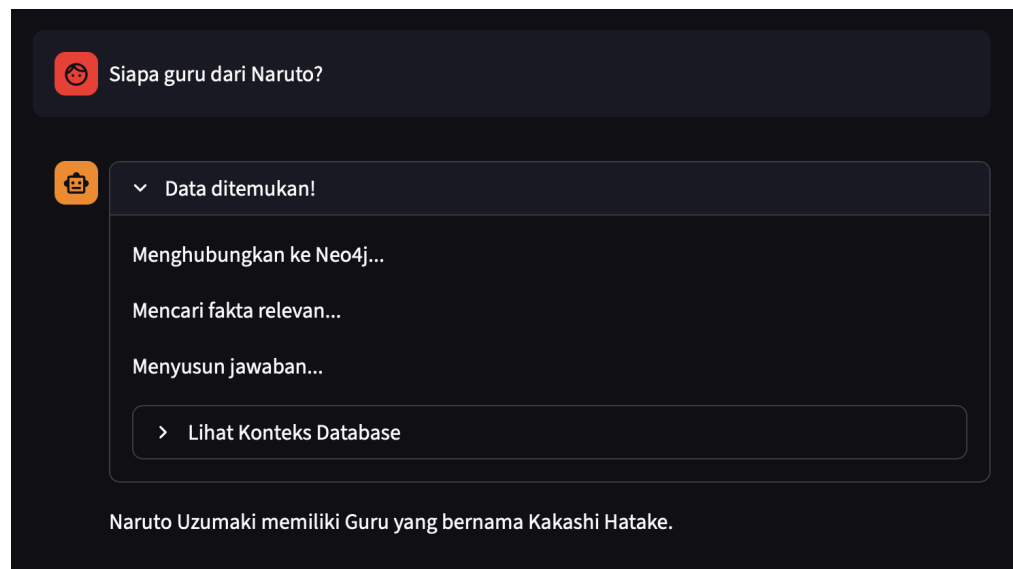
7. Jelaskan profil Zabuza Momochi (Desa, Organisasi, dan Jutsu).



8. Bijuu apa yang ada di dalam tubuh Han?



9. Siapa guru dari Naruto?



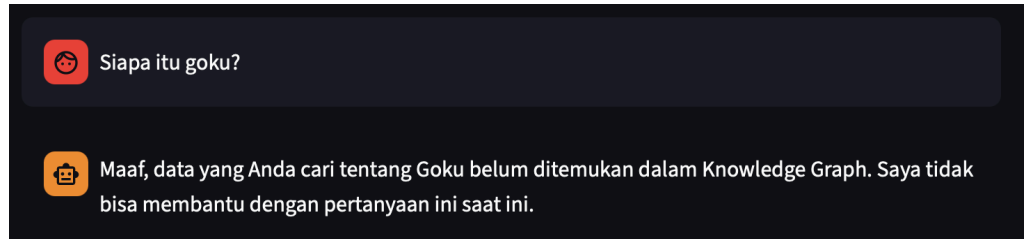
Sistem dapat menjawab berbagai pertanyaan-pertanyaan tersebut dengan akurat sesuai dengan basis pengetahuan.

D. Pertanyaan di luar domain

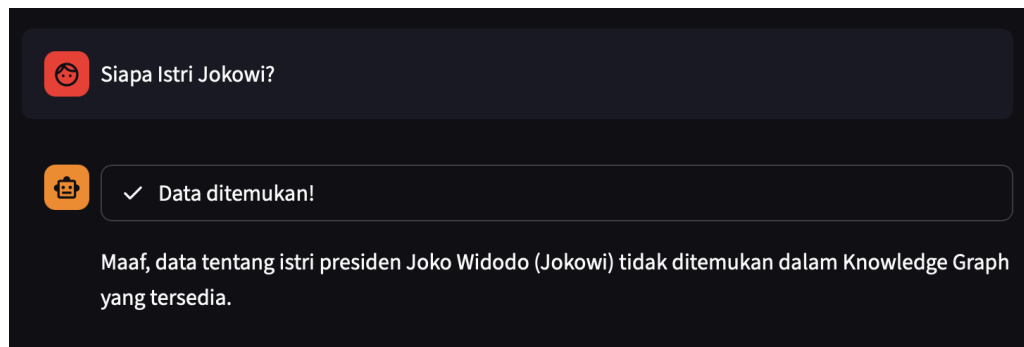
Pertanyaan-pertanyaan ini bertujuan untuk menguji apakah sistem benar-benar merujuk pada basis pengetahuan yang ada. Jika pertanyaan yang ditanyakan di luar dari

domain yang disediakan sistem harusnya akan menjawab bahwa ia tidak memiliki informasi yang diperlukan untuk menjawab pertanyaan tersebut dan tidak mencoba menjawab pertanyaan dengan konteks terbatas yang akan menyebabkan halusinasi.

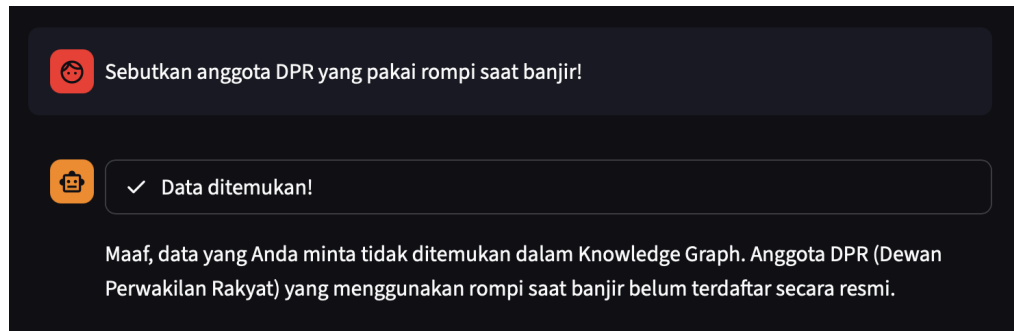
10. Siapa itu goku?



11. Siapa Istri Jokowi?



12. Sebutkan pokemon bertipe air!



Sistem tidak berhalusinasi dan memberikan jawaban yang tidak akurat jika pertanyaan yang ditanyakan tidak ada di basis pengetahuan yang ada.

## 3.2. Perbandingan Sistem RAG & Berbasis Pengetahuan

### 3.2.1 Sistem Berbasis Pengetahuan (Traditional/Logic-Based)

Sistem ini mengandalkan pemrosesan data terstruktur (Ontologi RDF) dengan mesin inferensi logis (seperti SWI-Prolog atau reasoner OWL). Pengetahuan diperoleh dan dianalisis melalui aturan logika yang terdefinisi.

#### Kelebihan:

- **Integritas Data:** Jawaban 100% terikat pada aturan dan fakta yang dikurasi, tanpa ambiguitas.

- **Validasi Ontologi:** Efektif untuk memvalidasi konsistensi struktur ontologi.

**Kekurangan:**

- **Kaku:** Output berupa boolean atau daftar variabel, tidak dalam Bahasa Alami.
- **Input Kaku:** Membutuhkan query formal (misalnya, Prolog atau SPARQL) atau lapisan NLP yang kompleks.
- **Skalabilitas Inferensi:** Performa reasoner tradisional (Prolog) cenderung menurun seiring bertambahnya volume fakta yang sangat besar.

### 3.2.2 Sistem RAG (Retrieval-Augmented Generation)

Sistem ini menggabungkan kemampuan penyimpanan data terstruktur (Graf Pengetahuan Neo4j) dengan model generatif (LLM) untuk sintesis jawaban. Prosesnya melibatkan pencarian data faktual (retrieval) dari Neo4j sebelum LLM menghasilkan output.

**Kelebihan:**

- **Output Bahasa Alami:** Mampu menghasilkan narasi yang koheren dan mudah dipahami oleh pengguna awam.
- **Pengetahuan Dinamis:** Pengetahuan diperbarui secara real-time dengan hanya memperbarui database Graf Pengetahuan (Neo4j) tanpa melatih ulang LLM.
- **Fleksibilitas Input:** Dapat menangani query input dalam Bahasa Alami.

**Kekurangan:**

- **Latensi Lebih Tinggi:** Terdapat penambahan waktu pemrosesan untuk langkah retrieval (eksekusi Cypher ke Neo4j).
- **Ketergantungan Kualitas LLM:** Akurasi sintesis jawaban sangat bergantung pada kemampuan LLM dalam menafsirkan konteks yang diberikan dari Graf Pengetahuan.

### 3.3. Keterbatasan Graf Pengetahuan dan RAG

Meskipun sistem berfungsi dengan baik, terdapat beberapa keterbatasan yang ditemukan selama pengembangan:

1. **Terbatas pada Explicit Knowledge:** Sistem hanya bisa menjawab apa yang tertulis secara eksplisit di graf. Pertanyaan analitik implisit seperti "Apakah Madara lebih kuat dari Naruto?" tidak dapat dijawab secara objektif karena graf hanya menyimpan fakta (seperti `hasElement`, `hasJutsu`), bukan nilai kekuatan tempur (`power level`) atau logika pertarungan yang memungkinkan simulasi perbandingan.
2. **Ketergantungan pada Entity Linking:** Sistem sangat bergantung pada kecocokan nama entitas. Jika pengguna mengetik "Si Petapa Genit" alih-alih "Jiraiya", sistem RAG gagal mengambil konteks karena tidak ada mekanisme fuzzy matching atau kamus sinonim yang komprehensif di dalam pipeline saat ini.



## **BAB IV**

### **KESIMPULAN**

Pengembangan sistem Retrieval-Augmented Generation (RAG) dengan domain Naruto pada tugas proyek ini menunjukkan bagaimana Graf Pengetahuan yang dibentuk dari ontologi, logika deskripsi, dan implementasi pada basis data Neo4j dapat dikombinasikan dengan Large Language Model (LLM) untuk membangun sebuah sistem tanya jawab yang akurat dan koheren.

Dengan memanfaatkan struktur Graf Pengetahuan, seluruh pengetahuan dasar dunia Naruto dapat diakses secara fleksibel melalui query Cypher yang efisien.

Integrasi Graf Pengetahuan dengan LLM melalui mekanisme RAG menjadikan sistem mampu melakukan penalaran faktual terikat (grounded reasoning). Hasil implementasi menunjukkan bahwa sistem mampu menjawab query dari yang paling sederhana hingga kompleks. Ini membuktikan bahwa Graf Pengetahuan berfungsi sebagai fondasi representasi pengetahuan yang kuat, sementara LLM berperan sebagai mesin sintesis yang mengonversi hasil retrieval faktual menjadi Bahasa Indonesia yang luwes dan mudah dipahami.

Meskipun demikian, beberapa keterbatasan tetap ditemukan antara lain:

Keterbatasan Entity Linking: Sistem sangat bergantung pada kesamaan nama entitas. Pertanyaan dengan typo atau nama panggilan yang belum didefinisikan secara eksplisit di kamus entitas dapat menyebabkan kegagalan retrieval.

Analisis Logika yang Terbatas: Sistem hanya mampu mengambil fakta yang ada. Pertanyaan analitik yang membutuhkan operasi agregasi atau perbandingan kompleks (misal: "Siapa ninja terkuat di desa A?") belum dapat dijawab tanpa implementasi query Cypher yang lebih kompleks.

Namun, keterbatasan tersebut tidak mengurangi keberhasilan utama proyek ini, yaitu menunjukkan bahwa kombinasi Graf Pengetahuan (Neo4j) dan LLM (RAG) dapat menghasilkan sistem yang akurat, konsisten, dan ekspresif untuk domain fiksi yang kompleks seperti dunia Naruto.

## BAB V

### DAFTAR PUSTAKA

#### 6.1. Sumber Pustaka

**Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.** (2020). [online]  
Available at: <https://arxiv.org/pdf/2003.02320> [Accessed 1 Dec. 2025].

**Neo4j.What is a Knowledge Graph?** [online] Available at:  
<https://neo4j.com/blog/knowledge-graph/what-is-knowledge-graph/> [Accessed 1 Dec. 2025].

**Neo4j.Neo4j Documentation.** [online] Available at: <https://neo4j.com/docs/> [Accessed 1 Dec. 2025].

#### 6.2. Lampiran

1. Github: <https://github.com/rizqikapratamaa/naruto-graph-rag>
2. Matriks kontribusi

NIM	Tugas
13522126	Ontology Design, RAG Logic & System Architecture..
13522151	Ontology Design, Documentation & Testing.
13522159	Ontology Design, Graph Engineering & Database.